

Manual Completo do Debian



GNU / LINUX

AI
ED

Wellington Pinto de Oliveira

APRESENTAÇÃO

Este conteúdo foi desenvolvido para auxiliar nas práticas acadêmicas em disciplinas de Sistemas Operacionais com GNU/Linux e Redes de Computadores, é um texto que reúne teoria e a prática, ou seja, respalda teoria de livros consagrados com a prática do autor.

Toda a prática pode ser validada automaticamente com o uso de um agente inteligente que auxilia na resolução dos erros implementados pelo aprendiz, sem a necessidade de intervenção de um instrutor neste primeiro momento, mas é claro, se nem com isso o aprendiz conseguir então é necessário a intervenção do educador.

O material também conta com vídeos explicativos, o objetivo dos vídeos é proporcionar estudos realizados por autodidatas ou aprendizes em tempo vago.

Todo o material está fundamentado em uma distribuição GNU/Linux Debian 12 de 64 bits por ser uma distribuição GNU/Linux conhecida e ser ótima para iniciantes. No primeiro capítulo o aprendiz terá acesso aos links e todo o ambiente é gratuito.

Procuro reforçar neste conteúdo todo o requisito para as provas de certificação Linux LPI e ainda iniciar o administrador Linux nas linguagens C/C++ e Python para resolver problemas por programação, e ainda uma boa explanação sobre Scripts Bash.

O material é gratuito e pode ser utilizado por qualquer instituição de ensino ou curso, sem ônus financeiros. Estamos abertos para idéias no e-mail wellingtont.aied@gmail.com.

Sempre baixe a versão mais atual que está no GitHub:

<https://github.com/naoimportaweb/book/tree/main/Manual%20Completo%20do%20Debian%20GNU-Linux>

SUMÁRIO

APRESENTAÇÃO	1
SUMÁRIO	2
1 Considerações e Ambiente	15
1.1 Tecnologias utilizadas e Download de arquivos	15
1.2 Instalando o Oracle VM VirtualBox	15
1.3 Criando uma Virtual Machine	16
1.4 Instalando o Debian 12 GNU/Linux	18
1.5 Instalando e configurando o SUDO	34
1.6 Validação de práticas por meio de automação	37
1.7 Criando uma exportação ou importando uma VM	41
1.8 Virtual Networking no VirtualBox	45
1.8.1 Adaptador em NAT	50
1.8.2 Modo Bridge	51
1.8.3 Rede Interna	51
1.9 Editor de texto nano	52
1.10 Recomendações e divisão dos capítulos	53
2 Sistema Operacional Linux e Distribuições	55
2.1 Arquitetura de um computador	55
2.1.1 North and Southbridge	56
2.1.2 Acesso direto à memória	56
2.1.3 Memória de Acesso Aleatório (RAM)	57
2.1.4 Arquiteturas de CPU	57
2.2 Sistema Operacional	59
2.2.1 Gerenciamento de processo	59
2.2.2 Gerenciamento de memória	60
2.2.4 Sistema de I/O e Sistema de Arquivos	61
2.3 Um Kernel e muitas Distribuições	62
3 Virtualização	64
3.1 Introdução a Virtualização	64
3.3 Hypervisors	65
3.4 Tipos de virtualização	66
3.4.1 Virtualização de desktop	66
3.4.2 Virtualização de rede	66
3.4.3 Virtualização de armazenamento	67
3.4.4 Virtualização de dados	67
3.4.5 Virtualização de aplicativos	68
3.4.6 Virtualização de CPU	68
3.4.7 Virtualização de GPU	68
3.4.8 Virtualização Linux	69
3.4.9 Virtualização de nuvem	69
3.6 Práticas da aula	69
3.6.1 Prática 98a917ce 01: Criando o ambiente da aula	69

4 Sistema de arquivos no Debian GNU/Linux	71
4.1 Virtual File System	71
4.2 Diretórios padrões de um GNU/Linux Debian	75
4.3 Conceito de i-node	79
4.4 Tipos de arquivos	82
4.4.1 Arquivos regulares	83
4.4.2 Diretórios	87
4.4.3 Arquivos especiais de bloco e de carácter	89
4.5 Informações sobre arquivos (file e stat)	91
4.6 Navegando no sistema de Arquivos (cd, pwd, ls)	93
4.7 Copiando e movendo arquivos (mv e cp)	95
4.8 Lendo arquivos (cat, tac, more, sed, tail, split)	98
4.9 Editores de texto (nano, vim e vi)	101
4.10 Obtendo informações sobre o arquivo (file, stat, cksum, cmp)	105
4.11 Criando e removendo diretórios e arquivos (touch, rm, mkdir, rmdir)	107
4.12 Permissão de acesso e dono (chmod, chown, chgrp, umask)	109
4.13 SetUID, SetGID e Sticky Bits	114
4.13.1 Bit SetUID	115
4.13.3 Bit Sticky	116
4.14 Link simbólico e Hard Link (ln e ln -s)	116
4.14.1 Hard Link	117
4.14.2 Symbolic links	118
4.15 Práticas da aula de Sistema de Arquivos	119
4.15.1 Prática 8e9c361c 01: Criando diretórios e arquivos no GNU/Linux	119
4.15.2 Prática 8e9c361c 02: Copiando arquivo, removendo arquivo e movendo arquivo	120
4.15.3 Prática 8e9c361c 03: Criar links	121
4.15.4 Prática 8e9c361c 04: Diretório e arquivos com permissão especial	122
5 Usuários GNU/Linux	124
5.2 Multusuário e Multitarefas	124
5.3 Usuários do GNU/Linux	125
5.4 O arquivo /etc/passwd	126
5.5 O arquivo /etc/shadow	129
5.6 Arquivo /etc/group	130
5.7 Arquivo /etc/nsswitch.conf	131
5.8 Adicionando usuários no GNU/Linux com adduser e useradd	131
5.9 Criando grupos com adduser e addgroup	135
5.10 Deletando grupos e usuários com deluser e delgroup	135
5.11 Editando um usuário com usermod	136
5.11.1 Adicionar um usuário a um grupo complementar	137
5.11.2 Alterar o default group do usuário	137
5.11.3 Alteração das informações do usuário	138
5.11.4 Alterando um diretório inicial do usuário	138
5.11.5 Mudando o Shell Padrão do Usuário	138

5.11.6 Alterando um UID de usuário	138
5.11.7 Alterar um nome de usuário	139
5.11.8 Definir uma data de expiração do usuário	139
5.11.9 Bloqueio e desbloqueio de uma conta de usuário	139
5.12 Alterando a senha de um usuário	140
5.12.1 Consultando dados sobre o password de um usuário	140
5.12.2 Altere sua senha de usuário	141
5.12.3 Alterar a senha de outro usuário	141
5.12.4 Forçar o usuário a alterar a senha no próximo login	141
5.13.5 Removendo a senha de um usuário	142
5.13.6 Bloqueando e Desbloqueando o password de usuários	142
5.13 Obtendo informações de usuários com finger	142
5.14 Comando sudo, arquivo sudoers e grupo sudo	143
5.15 Práticas do capítulo usuários GNU/Linux	144
5.15.1 Prática fcb0b800 01: Criando usuários e grupos	144
5.15.2 Prática fcb0b800 02: Editando usuários	145
5.15.3 Prática fcb0b800 03: Alterar os arquivos passwd e shadow	146
5.15.4 Prática fcb0b800 04: Usuário tomcat para o serviço tomcat	147
6 Formatando e montando um dispositivo de bloco	148
6.1 Adicionando um novo disco no VirtualBox	148
6.2 Dispositivo especial de bloco	150
6.4 Tabela de Partição	152
6.4.1 Master Boot Record	152
6.4.2 EFI system partition, UEFI e GPT	154
6.5 Tipos de formatação	154
6.5.1 Padrão ext2	156
6.5.2 Padrão ext3	157
6.5.3 Padrão ext4	158
6.7 Praticando leitura de blocos (falta)	158
6.8 Acesso direto aos arquivos e Journaling	159
6.9 Listando dispositivos especiais de bloco com lsblk	159
6.10 Obtendo informações sobre dispositivos de Bloco com blkid	161
6.11 Arquivos partitions, mounts e mtab	162
6.12 Criando partições em discos com fdisk	163
6.12.1 Listar Partições no GNU/Linux	163
6.12.2 Criando Tabela de Partição	164
6.12 Formatando uma partição para uso com mkfs	166
6.13 Montando e desmontando um dispositivo no Sistema de Arquivos com mount e umount	167
6.14 O arquivo /etc/fstab	168
6.15 Prática do capítulo de Formatação	171
6.15.1 Prática 8b65b431 01: Adicionando um novo disco	171
6.15.2 Prática 8b65b431 02: Adicionando um cdrom	172
7 Gerenciamento de Processos no GNU/Linux	173

7.1 Ciclo de Vida do Processo	175
7.2 Bloco de Controle de Processo (PCB)	177
7.1 Compiladores	179
7.1.1 Tipos de compiladores	179
7.1.2 Compilando Versus Interpretando	179
7.1.3 Criando um executável com GCC	180
7.1.3.1 O GCC – GNU Compiler Collection	181
7.1.3.2 Compilação no GCC	182
7.1.3.3 Assembler do Código no gcc	183
7.1.3.4 Linker do código com as bibliotecas	183
7.2 Locando arquivos headers, bibliotecas e comandos	185
7.2.1 Criando um arquivo .h	186
7.3 A implementação de threads no Linux	188
7.3.1 Threads de kernel	188
7.3.2 Implementando uma thread em C++	189
7.4 Criando e gerenciando processos filho em C++	190
7.4.1 Função fork()	190
7.4.2 Aguardando finalização de procedimento filho com wait() e waitpid()	192
7.4.3 Executando comandos de Shell com system() popen() e exec()	196
7.5 Sinais POSIX	198
7.5.1 Enviando um sinal para um processo com o comando kill	201
7.5.2 Tratando sinais em processos	202
7.6 Escalonamento (Scheduling)	206
7.6.2 Policy	208
7.6.3 Prioridade de Processo	209
7.6.4 Timeslice (Quantum)	210
7.6.5 Preempção de processo	211
7.6.6 A política de agendamento em ação	211
7.6.7 O Algoritmo do Schedule	212
7.6.8 Runqueues	212
7.6.9 Matrizes de prioridade	214
7.6.10 Recalculando timeslice	215
7.6.11 Schedule	216
7.6.12 Calculando a Prioridade e o timeslice	216
7.6.13 Dormir e acordar	218
7.6.14 Preempção e troca de contexto	220
7.6.15 Preempção do usuário	220
7.6.16 Preempção de kernel	221
7.6.17 Tempo real (real-time)	222
7.6.18 Chamadas do sistema relacionadas ao agendador	222
7.7 Comunicação Inter-Processo	223
7.8 Exibindo dados de processos com top e htop	224
7.9 Exibindo dados de processos com ps e pstree	227
7.10 Comando free	229

7.11 Trabalhos em background com comando bg e fg	230
7.12 Alterando a prioridade de um processo com nice e renice	231
7.13 Sensors	232
7.14 Práticas do capítulo	232
7.14.1 Prática prc0001 01: Listar processos python	233
8 Questões de memória, Linux e C/C++	234
8.1 Endereços de Memória em Linux	236
8.2 Como é tratada a memória em programas C e C++	240
8.3 Segmentação de áreas de memória	242
8.4 Address Space Layout Randomization (ASLR) e bit NX	244
9 Configurando interface de Redes em Linux	252
9.1 Faixas de IPs reservados para LAN	252
9.1.1 Definindo uma configuração para a LAN	253
9.2 Definição dos nomes das Interfaces de Rede	253
9.2.1 Exibindo dados de interfaces de rede PCI com lspci	254
9.2.2 Exibindo interfaces de rede USB com lsusb	255
9.3 Comando ip	255
9.3.1 Obtendo informações de interfaces de rede	257
9.3.2 Alterando/Atribuindo endereços para uma interface	257
9.3.3 Criando e removendo rotas	258
9.3.4 Ligando e desligando uma interface de rede	260
9.4 Comando ifconfig	260
9.5 Arquivo /etc/network/interfaces	261
9.6 Domain Name System com resolver, host, nslookup, dig, e whois	262
9.7 Arquivo /etc/hosts	268
9.8 Regras de roteamento do Linux	269
9.9 Realizando testes com ping, traceroute, tracepath e mtr	270
9.10 Comando Netstat e ss	273
9.11 Endereço MAC Address e o comando arp	277
9.11.1 Origem do MAC Address	278
9.11.2 Endereço Universal e Local	279
9.11.3 Comunicação Unicast, Multicast e Broadcast	279
9.11.4 Tradução de IP e MAC Address com ARP e RARP	281
9.12 Realizando download com cURL ou WGET	282
9.13 Práticas do capítulo	286
9.13.1 Prática 0002 checkpoint03: Configurando o arquivo de Interfaces	286
9.13.2 Prática 0002 checkpoint04: Configurando por comandos	287
9.13.3 Prática 0002 checkpoint05: Baixando arquivos com wget	288
10 Instalação de Programas e pacotes	289
10.1 O processo de desenvolvimento	289
10.2 Comando make	289
10.2.1 Arquivo Makefile	290
10.3 Instalação de baixo nível	293
10.3.1 Criando um pacote .deb com dpkg-deb	294

10.3.2 Instalando um pacote com dpkg -i	296
10.3.3 Obtendo informações de pacotes instalados	296
10.3.4 Removendo pacotes com dpkg	297
10.4.5 Log de operações do dpkg	297
10.4.6 Procurar nos metadados do pacote	298
10.4 Instalação de alto nível	298
10.4.1 Arquivo de repositórios sources.list	299
10.4.2 Adicionando PPA com add-apt-repository	300
10.4.3 Listando todos os pacotes de um repositório	302
10.4.4 Instalando e reinstalando um pacote com APT	304
10.4.5 Atualizando pacotes e realizando upgrade com APT	304
10.4.7 Removendo pacotes com AP	305
10.4.8 Limpando cache do comando APT	306
10.5 Práticas do capítulo	306
10.5.1 Prática 092900 01: Criando e instalando um pacote	306
10.5.2 Prática 092900 02: Removendo um pacote com dpkg	307
10.5.4 Prática 092900 03: Instalando o pacote net-tools com apt	308
11 Shell Script e Bash	309
11.2 Como criar um script no Linux	309
11.2.1 Variável de ambiente PATH	310
11.3 O Interpretador /bin/bash	310
11.4 Como criar um script	310
11.5 Shell Script BASH /bin/bash	312
11.5.1 Declarações Globais	313
11.5.2 Parando um Script	314
11.5.3 Descritores Input, Output e Error no Linux	314
11.5.3.1 Redirecionando o output	315
11.5.3.2 Redirecionando o input	315
11.5.3.3 Redirecionando error	315
11.5.3.4 Redirecionando o output e o error	316
11.5.3.5 Anexando em arquivos existentes	316
11.5.3.6 Suprimindo o output ou error	316
11.5.4 Uso de comandos no script	316
11.5.5 Variáveis	317
11.5.6 Variáveis Predefinidas	319
11.5.7 Exports	320
11.5.7 Arrays	321
11.5.8 Expressões e test	323
11.5.9 Comando if	326
11.5.10 Comando case	327
11.5.11 Comando while	328
11.5.12 Comando until	329
11.5.13 Comando for	330
11.6 Práticas do capítulo	331

11.6.1 Prática 8ab001 01: Imprimindo na tela	331
11.6.2 Prática 8ab001 02: Uma variável	331
11.6.3 Prática 8ab001 03: Imprimindo mas com python2	332
11.6.4 Prática 8ab001 04: Script que valida par ou ímpar com if	332
11.6.5 Prática 8ab001 05: Script que valida se um número é maior que 10	333
12 Python para Sistemas Operacionais	335
12.1 Listando versões de Python no Linux	335
12.2 Python em modo interativo	336
12.3 Python em scripts	337
12.4 Importando módulos Python	337
12.5 Módulo OS, SYS e SHUTIL	340
12.5.1 Dados do Sistema Operacional e do Environment	340
12.5.2 Comandos clássicos GNU/Linux e Windows no python	343
12.5.3 Trabalhando com PATH	344
12.5.4 Manipulando arquivos	347
12.6 Módulo subprocess	348
12.6.1 subprocess.run()	348
12.6.2 subprocess.Popen()	350
12.5 Scripts profissionais em Python	352
12.5.1 Trabalhando com parâmetros	352
12.5.2 Output profissional	357
12.6 Práticas do capítulo	361
12.6.1 Prática py0001 checkpoint01: Preparando o ambiente	361
12.6.2 Prática py0001 checkpoint02: Imprimindo o usuário corrente do Environment	362
12.6.3 Prática py0001 checkpoint03: Criando uma hash em MD5 do primeiro argumento	362
12.6.4 Prática py0001 checkpoint04: Listando usuários com shell /bin/bash	363
12.6.5 Prática py0001 checkpoint05: Persistir um arquivo	363
13 Adicionando um serviço na inicialização do GNU/Linux (atuando...)	365
13.2 Passos executados pela BIOS ou UEFI	365
13.3 MBR e GPT	366
13.4 Bootloaders	368
13.4.1 GNU GRand Unified Bootloader (GRUB)	368
13.4.2 Segurança do GRUB	371
14 Agendamento de tarefas com cron, crontab e at no GNU/Linux (finalizado)	372
14.2 Daemon cron e comando crontab	372
14.2.1 Daemon cron	372
14.2.1 Comando crontab	373
14.2.2 Formato de entradas cronjob na crontab de usuário	375
14.2.3 A crontab do sistema	376
14.2.4 Execuções de comandos	377
14.2.5 Permissão de acesso ao crontab	377
14.3 Comando at	377

14.3.1 Criando um agendamento com at	377
14.3.2 Especificando o tempo de execução	379
14.3.3 Listagem de tarefas pendentes	379
14.3.4 Removendo trabalhos pendentes	380
14.3.5 Restringindo usuários	380
14.4 Práticas do capítulo	380
14.4.1 Prática 4d4f6ae 01: Instalação do comando apt e um agendamento	380
14.4.2 Prática d7a527b 01: Um script de backup	380
15 Roteador, Firewall e Gateway	382
15.1 MAC ADDRESS, IP e Port Number	382
15.2 Protocolo Ethernet, IP, TCP e UDP	386
15.3 Router, Gateway e Firewall	388
15.1.2 Segmentação da LAN	388
15.3.1 Router	389
15.3.2 Firewall	390
15.3.3 Gateway	392
15.4 Projetos Netfilter	392
15.4.1 Pacote Iptables	394
15.4.1.1 A tabela NAT e seu uso	396
15.4.1.2 A tabela Filter e seu uso	398
15.4.1.3 Política padrão	403
15.4.1.4 Limpando as regras em vigor	403
15.4.1.5 Regras baseadas em flags TCP	403
15.4.1.6 Flow de tabelas e chains Iptables	404
15.4.2 Pacote Nftables	405
15.4.2.1 Criando tabelas e chains	407
15.5 Segmentando uma rede com Router	408
15.5.1 Configuração lógica	411
15.6 Serviço de Internet com Gateway	414
15.6.1 Configurando o ambiente virtual e a rede virtual no VirtualBox	414
15.6.2 Configurando as Interfaces de rede	418
15.6.3 Configurando o servidor Gateway	421
15.6.3.1 Utilizando Iptables	422
15.6.3.2 Utilizando nftables	423
15.6.3.3 Executando script na inicialização do GNU/Linux	425
15.6.4 Realizando os testes com ICMP	426
15.7 Regras de filtro e redirecionamento em um Firewall	427
15.7.1 Configuração lógica	429
15.7.2 Criando as regras	433
15.7.3 Script de inicialização	433
15.7.4 Testando a conexão	434
15.8 Práticas do capítulo	434
15.8.1 Prática acb1b800 01: Configurando interfaces de rede do Gateway	435
15.8.2 Prática acb1b801 02: Ativando o serviço de Gateway	436

15.8.3 Prática acb1b800 03: Configurando a interface de rede do Cliente	436
16 Openssh-server	438
16.2 Introdução OpenSSH e Protocolo SSH	438
16.2 Instalando e configurando o openssh-server	445
16.3 Instalando o serviço SSH	446
16.4 Acessando o terminal por uso de cliente SSH	447
16.4 Transferindo arquivos	449
16.5 Biblioteca libssh	451
16.5 Práticas do capítulo	454
16.5.1 Prática 57671beb 01: Configurando o serviço Openssh-server	454
17 Serviço WEB com Tomcat e Java	456
17.2 Criando a regra nftables no FIREWALL	456
17.3 Configuração de rede da máquina TOMCAT	457
17.2 Instalando o Java no Linux Terminal	457
17.3 Criando um ambiente seguro para seu container	458
17.4 Instalar o Tomcat	459
17.5 Configurando as permissões para o serviço	461
17.6 Adicionando o Tomcat na inicialização do Linux	462
17.7 Configurar a interface de gerenciamento da Web do Tomcat	464
17.8 Acessando a interface da Web	467
18 Serviço WEB com Apache e PHP (finalizado)	468
18.2 Instalando e configurando o Apache2	468
18.3. Instalando certificado digital	470
18.4 Correções de erros	472
18.5 Testando o PHP Instalado	472
18.6 Prática do capítulo	474
18.6.1 Prática bn13la21 01: Configurando o serviço WEB com Apache e PHP	474
19 Serviço DHCP	475
19.1 No VirtualBox	477
19.2 Instalando e Configurando o DHCP Server	478
19.2.1 Configurando o IP do servidor DHCP	478
19.2.2 Instalando o servidor DHCP	480
19.2.3 Arquivos de Configuração do Serviço DHCP	481
19.2.4 Fixando um endereço para um Host	484
19.3 Configurando Debian Cliente	484
19.3 Práticas do capítulo	486
19.5.1 Prática 57t90021 01: Validando o serviço DHCP	486
19.5.2 Prática 57t90021 02: Cliente DHCP	486
20 Serviço DNS com Bind9	488
20.1 Programa Bind9	490
20.2 Configurando o nome das máquinas	491
20.3 Instalando recursos em ns1 e ns2	492
20.4 Configurando Vincular ao Modo IPv4 em ns1 e ns2	492
20.5 Configurando o servidor DNS primário em ns1	493

20.5.1 Configurando o Arquivo de Opções	493
20.5.2 Configurando o Arquivo Local	494
20.5.3 Criação do arquivo de zona de encaminhamento	495
20.5.4 Criando o (s) arquivo (s) de zona reversa	496
20.5.5 Verificando a sintaxe de configuração do BIND	497
20.6 Configurando o servidor DNS secundário ns2	498
20.7 Cliente Debian host1	500
20.8 Práticas do capítulo de DNS com GNU/Linux	501
20.8.1 Prática 1ae1b810 01: Validando a configuração do servidor ns1	501
20.8.2 Prática 1ae1b810 01: Validando a configuração do servidor ns2	502
20.8.3 Prática 1ae1b810 03: Validando a configuração no host1	503
21 Banco de Dados	504
22 Serviço NTP	505
22.1 Recursos do capítulo	507
22.2 Instalação do servidor NTP (máquina NtpServer)	507
22.3 Configuração do servidor NTP	507
22.3.1 Brasil br.pool.ntp.org	508
22.3.2 Restringindo acesso ao NTP Server local	508
22.4 Monitorando	509
22.5 Configuração do Cliente	510
22.5.1 Criando um script de inicialização	511
22.6 Práticas do capítulo	512
22.6.1 Prática i0721ax 01: Configurando o serviço NTP	512
22.6.2 Prática i0721ax 02: Configurando o cliente	513
23 Samba (finalizado)	515
23.1 Recursos do capítulo	516
23.2 Configurando o servidor Samba	516
23.3 Acessando o recurso	520
23.3.1 Acessando com o cliente gráfico no MAC OS	520
23.3.2 Acessando com o cliente gráfico no Ubuntu GNU/Linux	522
23.3.3 Acessando com o cliente gráfico no Windows	523
23.4 Práticas do capítulo	524
23.4.1 Prática 927a3ik 01: Configurando o serviço SMB	525
23.4.2 Prática 927a3ik 02: Criando um diretório e acessando	525
24 Proxy Squid-cache	527
24.1 Recursos do capítulo	527
24.2 Cenário	527
24.3 Instalando o squid	531
24.4 Configurando a permissão de acesso ao Proxy	531
24.4.1 Permissão para todos que podem acessar o servidor	532
24.4.2 Permitindo acesso apenas para uma determinada rede/subrede	532
24.5 Ativando cache	533
24.6 Alterando a porta serviço proxy	534
24.7 Habilitando comunicação por portas para clientes	534

24.8 Controles de acesso no Squid com ACL	535
24.8.1 Bloqueando sites	536
24.8.2 Criando uma lista de sites permitidos	539
24.9 Configurando Proxy + DHCP server	540
24.10 Configurando Clientes por Proxy Transparente para porta 80	541
24.11 SquidGuard e listas de negação	543
24.12 Validando e recarregando a configuração	547
24.13 Configurando uma máquina cliente na abordagem 1	547
24.13.1 GNU/Linux Ubuntu Gráfico	548
24.13.2 Microsoft Windows 10	549
24.13.3 GNU/Linux Debian Terminal	550
24.14 Configurando uma máquina cliente na abordagem 2	551
24.15 Configurando uma máquina cliente na abordagem 3	551
24.16 Práticas do capítulo	552
24.16.1 Prática 30x498c 01: Configurando o Squid na Abordagem 1	552
24.16.2 Prática 30x498c 02: Negando redes sociais com Abordagem 1	552
24.16.3 Prática 4a2f8c 01: Configurando o Squid para atender 192.168.200.0/24 com Abordagem 1	553
24.16.4 Prática 4a2f8c 02: Bloqueando sites Pornográficos com Abordagem 1	554
24.16.5 Prática ccd6973c 01: Configurando Squid + DHCP com Abordagem 2	554
25 Proxy Reverso Nginx (finalizado)	555
25.1 Recursos do capítulo	557
25.2 Prática de Nginx	557
25.3 Instalando e configurando o servidor Apache2	557
25.4 Instalando o Nginx	557
25.4.1 Sites available e sites enabled	559
25.4.2 Balanceamento de Carga	560
25.5 Configurando o hosts da máquina real	560
25.5.1 Configurando o /etc/hosts no GNU/Linux e MAC OS	561
25.5.2 Configurando o C:\windows\system32\drivers\etc\hosts no Microsoft Windows	561
25.6 Práticas do capítulo	563
25.6.1 Prática 09a378b 01: Configurando o Gateway + Nginx	563
25.6.2 Prática 09a378b 02: Configurando o serviço Apache	564
26 Serviço de mail com Postfix (finalizado)	566
26.1 Recursos do capítulo	566
26.2 Instalação e Configuração Postfix	566
26.3 Testando o serviço com TELNET	568
26.4 Enviando e-mail com PHP	569
26.5 Práticas do capítulo	571
26.5.1 Prática ptf0001 01: Configurando o servidor de e-mail postfix	571
26.5.2 Prática ptf0001 02: Enviar um e-mail com telnet	572
26.5.3 Prática ptf0001 03: Enviando e-mail com PHP	572
27 Serviço FTP (finalizado)	574

27.1 Recursos do capítulo	574
27.2 O protocolo FTP	574
27.2.1 Comunicação em modo passivo	574
27.2.2 Comunicação em modo ativo	575
27.3 Preparando o ambiente	576
27.4 Instalando o serviço FTP	576
27.5 Configuração do serviço	577
27.6 Acessando o serviço FTP	578
27.6.1 Cliente FTP gráfico FILEZILLA	578
27.6.2 Cliente FTP Command Line	579
27.7 Configurando um servidor no modo Passivo	580
27.8 Uso de certificado ssl	580
27.9 Análise da comunicação FTP	581
27.9.1 Conexão ativa	581
27.9.2 Conexão passiva	584
27.9.3 Uso de certificados	584
27.10 Práticas do capítulo	585
27.10.1 Prática ftp0001 01: Configurando um serviço FTP	585
28 OpenVPN	586
28.1 Recursos do capítulo	587
28.2 O ambiente	587
28.3 Easy-rsa	592
28.4 Criando certificados para uso no OpenVPN	593
28.5 Criando certificado por Cliente	597
28.6 Configurando o servidor Openvpn	600
28.7 Criando um ambiente na Intranet	603
28.8 Utilizando em um cliente VPN	603
29 Versionadores de código	606
29.1. Versionamento de código	606
29.2 Subversion	607
29.3 Git	608
30 Servidor de Virtualização	608
31 Serviço de controle de Domínio OpenLDAP (atuando)	616
31.1 X.500	616
31.1.1 Object Entry	616
31.2 LDAP	618
31.2.1 Directory Service e Entry	618
32 Ansible (atuando)	624
LAB I Gateway completo com IPCOP (falta formatar)	625
LAB II Enterprise Service Bus (Finalizado)	645
1 Recursos do capítulo	646
2 Preparando o ambiente	646
3 Instalando e configurando o Gateway + Iptables	647
4 Instalando o WSO2	650

4.1 Instalando o Java 8	651
4.2 Instalando o WSO2	652
4.4 Segurança do serviço	654
4.5 Gerar um script de inicialização	656
4.6 Utilizando o painel do Enterprise Service Bus	657
4.6.1 Acessando a interface de manutenção	657
5 Instalando o Apache um provedor de dados	658
6 Configurando um serviço no ESB	659
7 Testando o serviço	663
8 Práticas do capítulo	664
8.1 Prática esb001 01: Configurando o Gateway	664
8.2 Prática esb001 02: Configurando o Enterprise Service Bus	665
8.3 Prática esb001 03: Configurando o serviço WEB com Apache e PHP	666
Correções de falhas	669
fx00001 Conectividade com o cliente	669
fx00002 -> configuração do server gateway	669
fx00003 - 2 placas de rede	669
fx00004 - iniciar serviço gateway.service	669
Referências	670
9.6.3 Prática 8ab001 03: Imprimindo mas com python2	670
15.9 Monitorando portas com nmap	671
15.10 UFW (falta)	673
4.15 Linux Professional Institute Certification	675
5.16 Linux Professional Institute Certification	678
6.16 Linux Professional Institute Certification	678
7.16 Linux Professional Institute Certification	680
7.16 Linux Professional Institute Certification	682
10.7 Linux Professional Institute Certification	684
E. PATHRC	685

1 Considerações e Ambiente

Todo o ambiente é gratuito e todo o conteúdo deste livro está associado às tecnologias descritas neste tópico, qualquer outra tecnologia adicionada pelo aprendiz é de inteira responsabilidade destes, os links se encontram no próximo subtópico de acordo com a listagem, repare na nota de rodapé.



[Considerações sobre o ambiente Linux para o curso - Parte 1 CURSO LINUX](#)

O computador do aprendiz deverá ter a capacidade de virtualização de sistemas operacionais 64 bits, geralmente os computadores já vem configurados para tal, mas se não for capaz, o aprendiz deve com as especificações de seu computador obter os manuais de configuração da BIOS e realizar a configuração manualmente.

1.1 Tecnologias utilizadas e Download de arquivos

Para iniciar, o aprendiz deve realizar o download dos seguintes artefatos:

- **VirtualBox**¹ para virtualizar as práticas, nunca realize as práticas diretamente no Sistema Operacional que utiliza diariamente;
- **Debian versão 12 64bits Terminal**² (debian-12.X.X-amd64-netinst.iso) para operar as práticas que são todas em GNU/Linux, mantenha esta versão para manter a coerência com as práticas;
- **Putty**³ e **WinSCP**⁴ para acessar remotamente as máquinas virtuais caso queira usar recurso CTRL+C e CTRL+V;

1.2 Instalando o Oracle VM VirtualBox

Se o aluno realizar as práticas em seu Sistema Operacional de uso diário, ou seja, se já é usuário GNU/Linux e queria fazer diretamente, este encontrará inúmeros problemas, pois as práticas afetam o Sistema Operacional. Para isso utilizamos uma camada de abstração, ou seja, uma camada de virtualização, será utilizado para fins de padronização o VirtualBox da Oracle.

Após realizar o download de acordo com o tópico anterior, realize um clique duplo sobre o arquivo de instalação se utiliza um Microsoft Windows ou execute o instalador se já usa GNU/Linux como Sistema Operacional. O processo de instalação do VirtualBox é fácil, pois segue um padrão de instalação **Next > Next > Finish**, sem complexidade tecnológica envolvida na instalação padrão. A única observação que deve ser feita é que precisa ser um

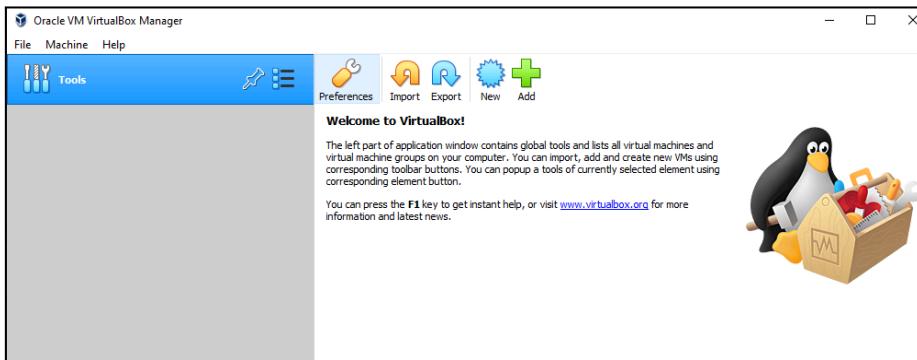
¹ O download pode ser realizado utilizando o link: <https://www.virtualbox.org/wiki/Downloads>

² O download pode ser realizado utilizando o link:
<https://cdimage.debian.org/cdimage/archive/12.0.0/amd64/iso-cd/> ou
<https://www.debian.org/releases/bookworm/debian-installer/>

³ O download pode ser realizado utilizando o link: <https://www.putty.org/>

⁴ O download pode ser realizado utilizando o link: <https://winscp.net/eng/download.php>

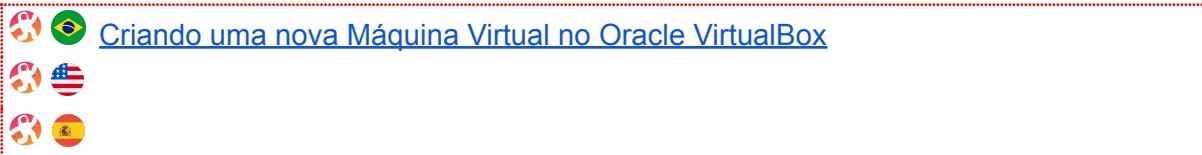
administrador da máquina para instalar, mas isso é trivial para este curso. Na figura abaixo temos o VirtualBox devidamente instalado e sendo executado.



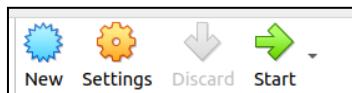
Agora que possui o virtualizador, o aprendiz deve criar sua própria Virtual Machine fazendo uma instalação do Debian 12 GNU/Linux.

1.3 Criando uma Virtual Machine

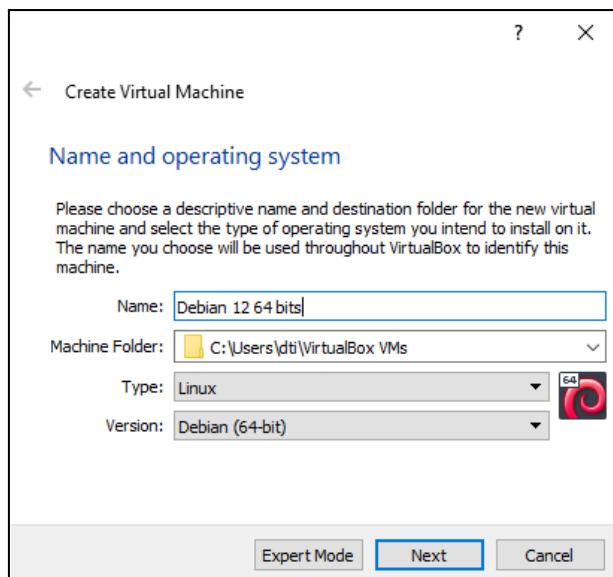
Agora o aprendiz deve obter a imagem virtual que será utilizada para criar em cada prática as máquinas virtuais, neste material será adotado um Debian 12 de 64 bits. O leitor pode obter a imagem do Sistema Operacional e fazer sua própria instalação, mas deve estar atento a versão, pois a plataforma GNU/Linux sofre constantes evoluções e naturalmente arquivos podem deixar de existir ou serem alterados.



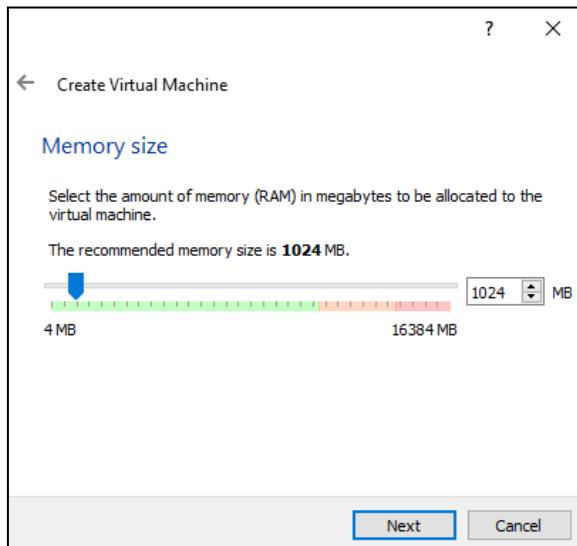
No menu principal clique em **Machine > New** ou pressione o borrão azul conforme figura abaixo.



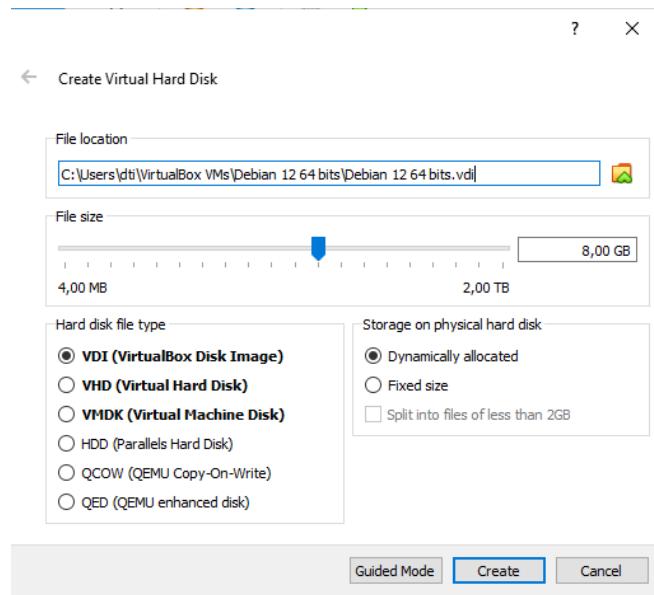
Dê um nome **Debian 12**, e selecione **Type Linux** e **Version Debian (64-bit)** conforme figura abaixo. Isso é necessário pois o VirtualBox irá criar um ambiente propício para um Debian de 64 bits.



Clique em **Next >**, no próximo passo informe que precisará de 1024 MB de memória, isso é suficiente para um GNU/Linux Debian 12, embora possa-se trabalhar com menos memória na máquina virtual, uma redução pode ocasionar maior lentidão nas práticas.



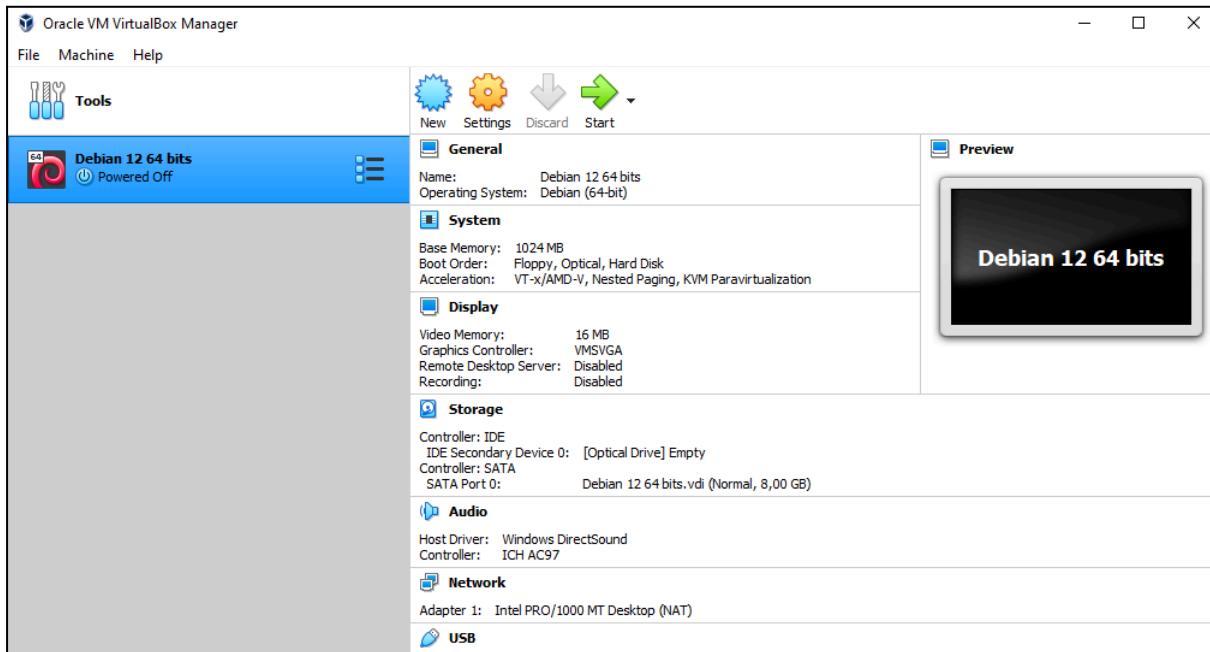
Na próxima etapa, será criado um arquivo .vdi, este arquivo é um disco rígido (HD) virtual e será nele que a instalação será feita. Pode-se utilizar de 8 até 18 GB, mas fique tranquilo que ele vai criar um arquivo pequeno, este arquivo se expande com o passar do tempo, mas nossas práticas são curtas e as máquinas virtuais criadas são apagadas.



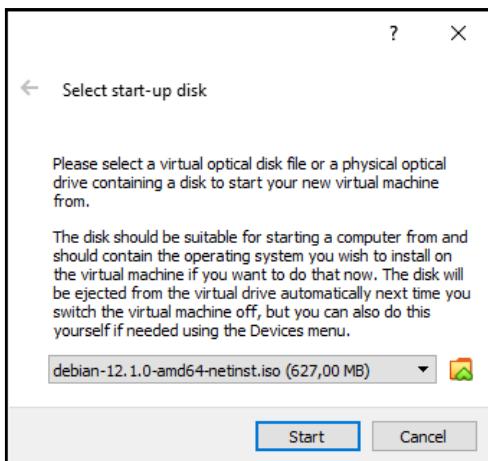
Neste ponto temos uma máquina, com placa mãe, processador, disco, placa de vídeo, entre outros periféricos, mas não temos ainda um GNU/Linux instalado.

1.4 Instalando o Debian 12 GNU/Linux

Não temos o Debian GNU/Linux instalado mas temos a máquina, então selecione a máquina virtual recém criada e clique no botão **Start** conforme figura abaixo.



Selecione a ISO apropriada que foi obtida no início deste capítulo diretamente do site do fornecedor, e inicie o processo de instalação do GNU/Linux selecionando a imagem no formato **ISO**.



Escolhendo a imagem ISO para instalação

Para localizar o arquivo .iso com a distribuição Debian, pressione no símbolo de folder⁵ na imagem acima, será aberto um explorer que poderá localizar o arquivo de instalação. Caso queira acompanhar por vídeo, o link abaixo está disponível para visualização, mas saiba que toda a sequência está descrita neste texto.

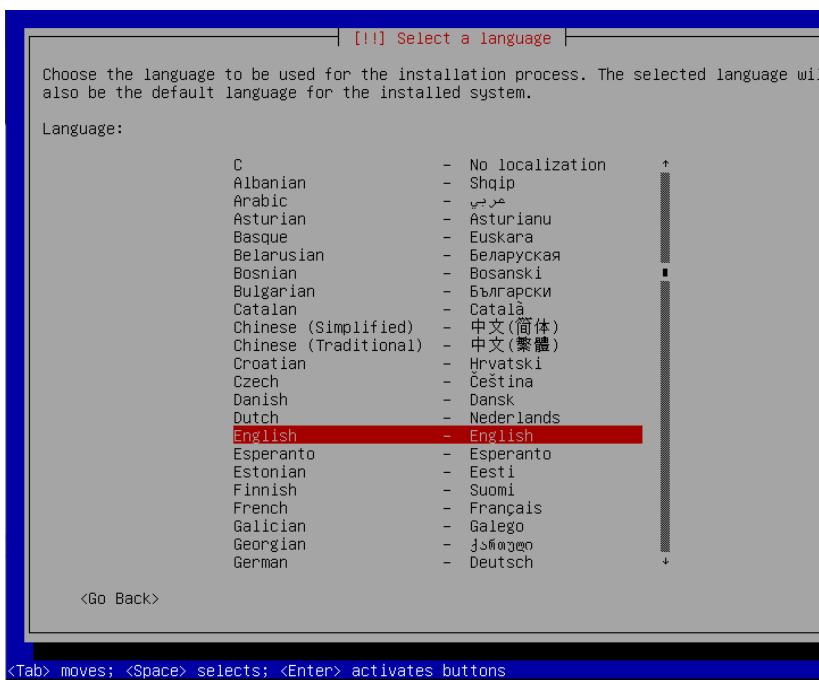


O processo de instalação é simples, e o recomendado para este curso é o **Debian sem interface gráfica**, ou seja, de forma pura somente o terminal pois este treinamento o capacitará a atuar em manutenção de servidores GNU/Linux que são estritamente terminais.

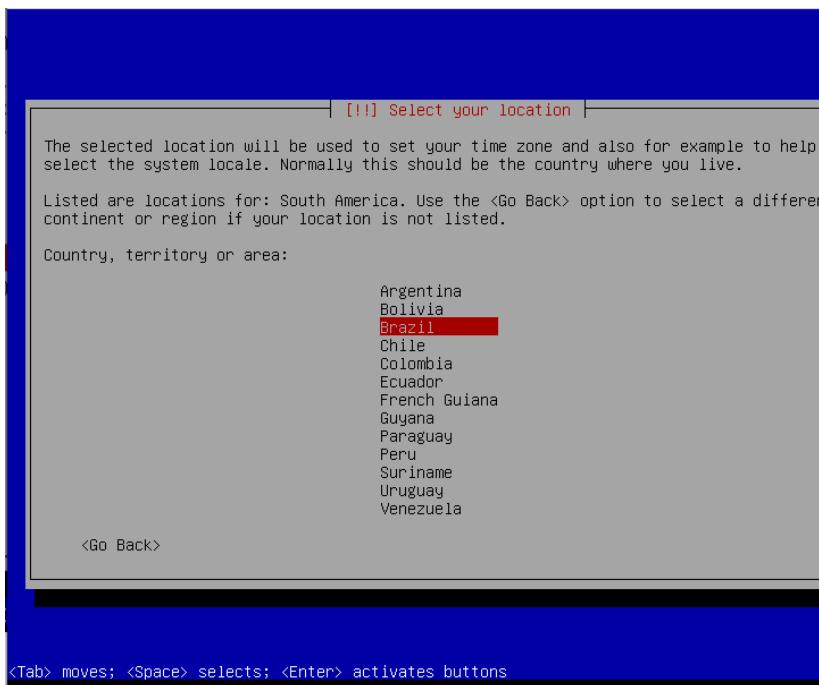
⁵ No mundo Linux não existe a expressão “pasta”, usamos “diretório”, mas como o VirtualBox usa uma imagem de folder, usei esta expressão;



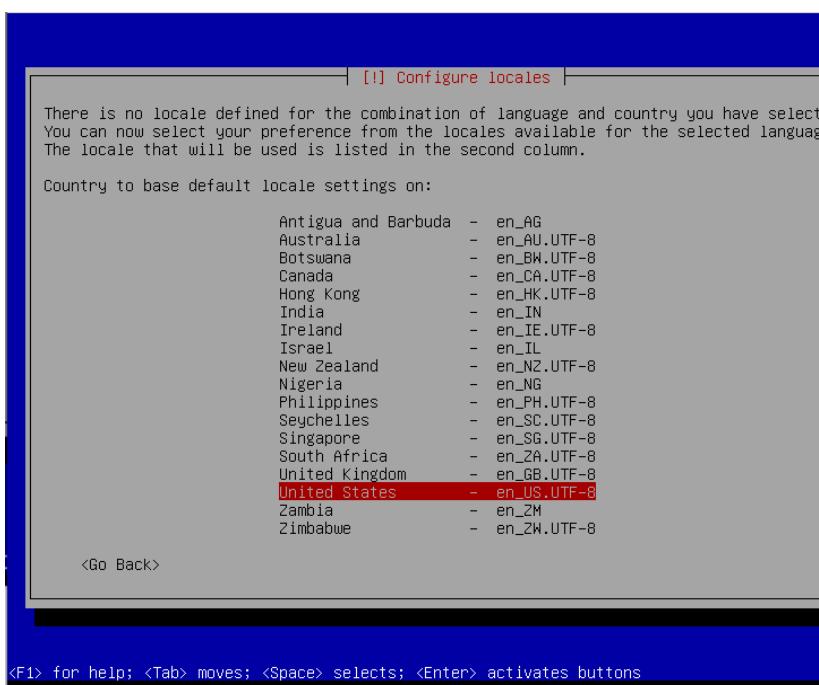
Durante a inicialização o primeiro passo é escolher o modo não gráfico para evitar o uso desnecessário de memória.



O GNU/Linux foi desenvolvido e testado em Inglês, então evite escolher Português do Brasil no próximo passo, saiba que uma coisa é a linguagem do Sistema Operacional, outra coisa é a Localização do seu computador.

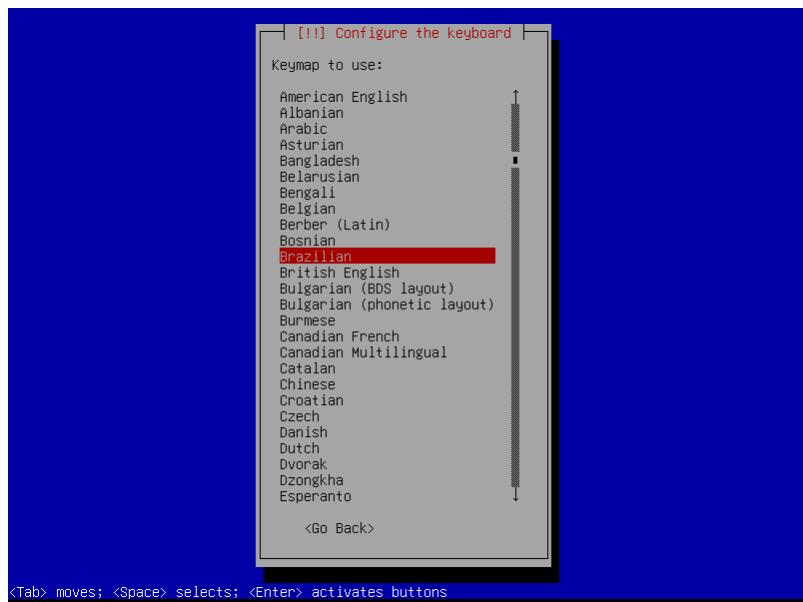


Agora no próximo passo vamos localizar a nossa máquina no Brasil, para isso escolha **Other > South America > Brazil** conforme figura.

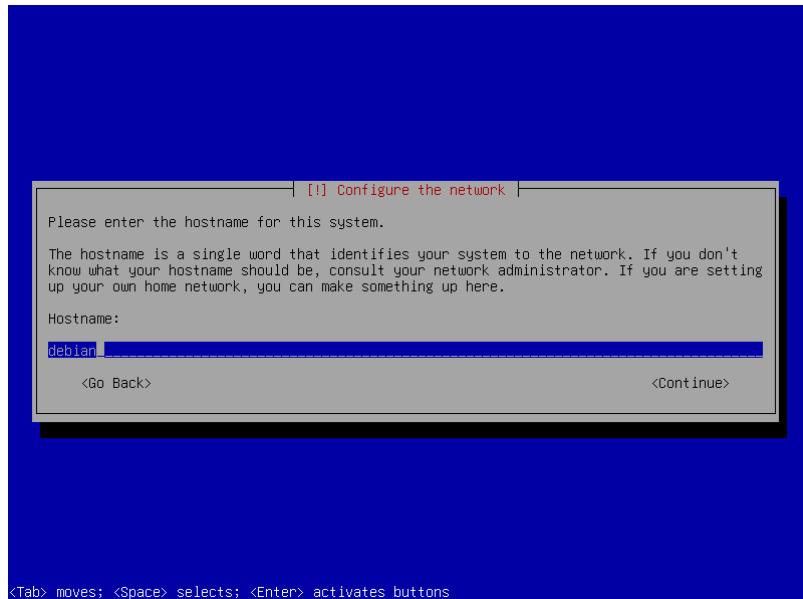


O próximo passo é escolher o Charset do sistema operacional, tanto o Python quanto o Mysql agradecem o uso de UTF-8 e o padrão de charset **English**.

Saiba que todo o sistema GNU/Linux foi programado e testado em UTF-8 e em **English**, o mais sensato é manter tais configurações.

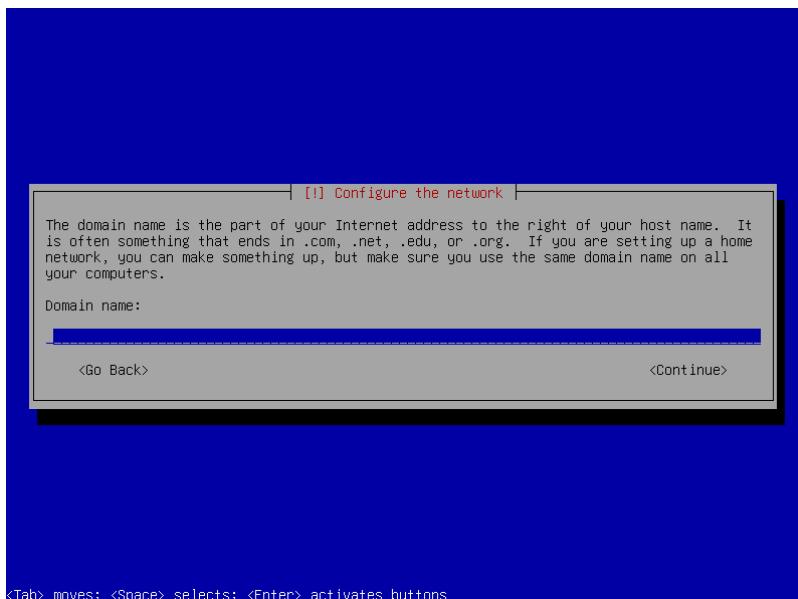


Embora o nosso GNU/Linux esteja em UTF-8 e padrão English, nosso teclado deverá ser o ABNT2 para se adaptar aos hardwares comercializados no Brasil, para isso selecione **Brazilian**.



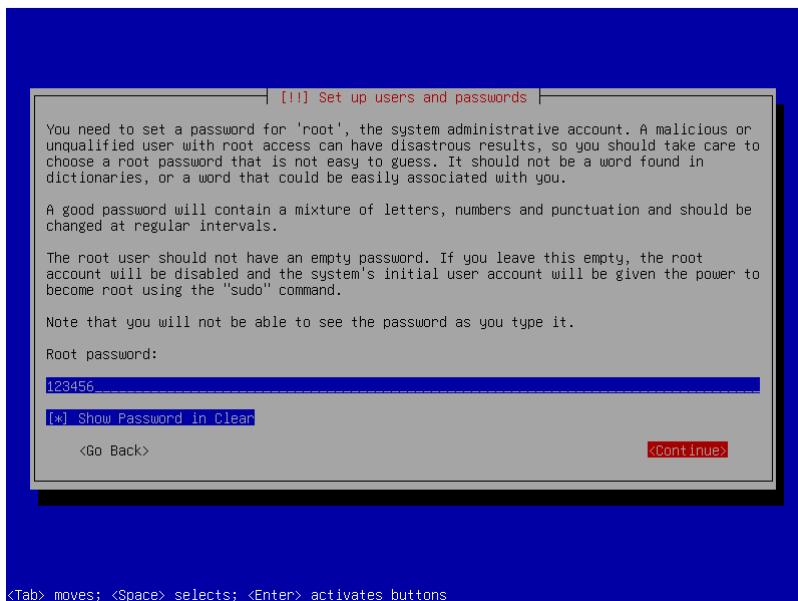
No próximo passo deve-se definir qual o nome do host (computador) que será utilizado, deve-se definir nomes únicos na rede para garantir que quando este host for mapeado pelo servidor DNS não tenhamos ambiguidade de máquinas na rede.

Mas para essas aulas, mantenha o nome **debian** para seu computador, para ser coerente com as práticas.



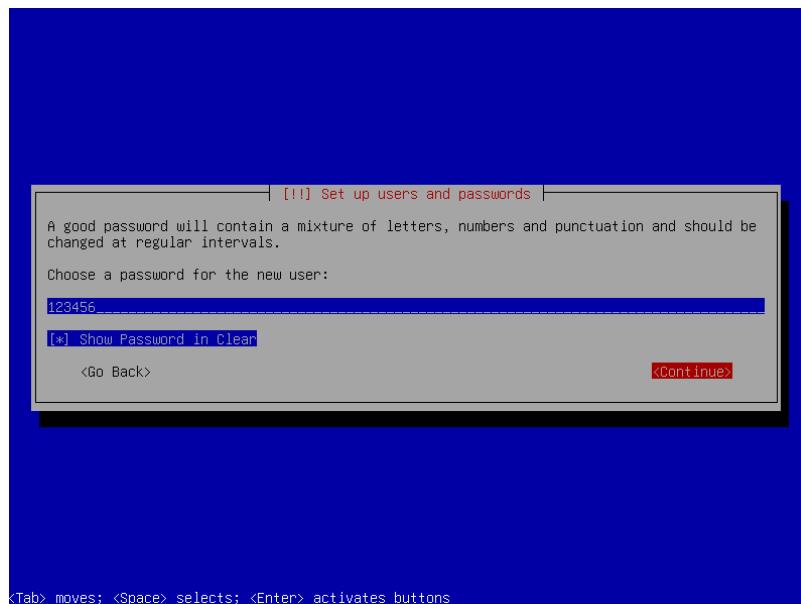
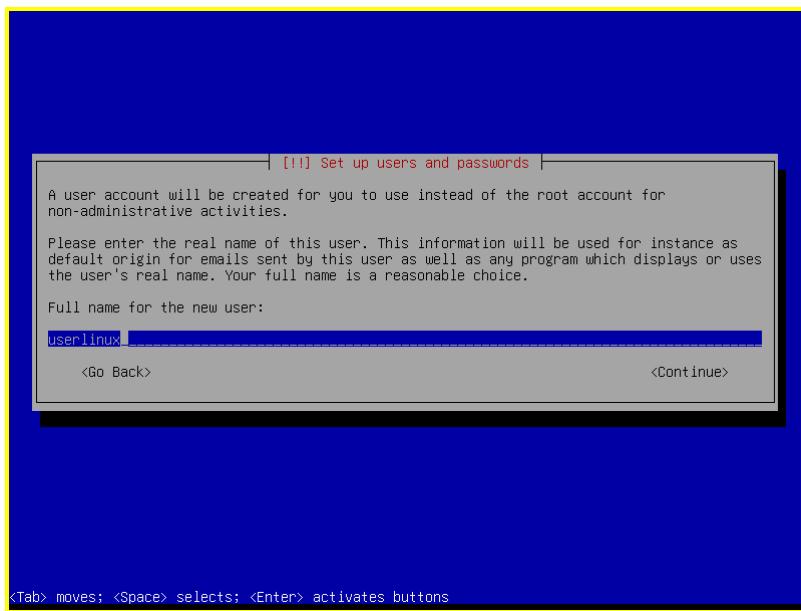
Será solicitado um domínio, ou seja, se a máquina está sob controle de um agente LDAP, exemplo: Microsoft Active Directory e OpenLDAP.

Mantenha vazio o campo e continue, afinal estamos atuando sem o contexto de um controlador de domínio.



Vamos padronizar a senha como **123456** para todo o livro, naturalmente não é uma senha que se utilize na produção.

Durante todo o texto vou me referir a esta senha: **123456**.



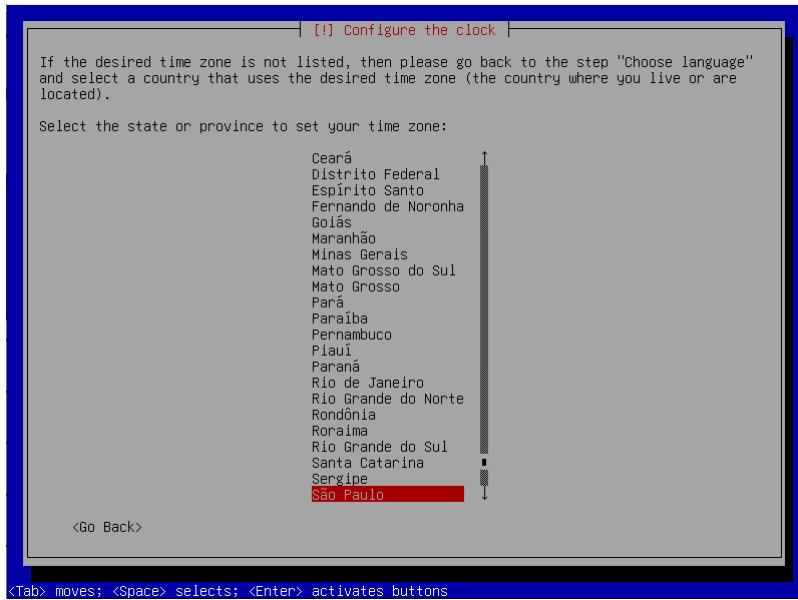
O ideal é não utilizar a conta do ROOT, o ideal é ter uma conta de usuário secundário e usar esta conta específica, então nos GNU/Linux modernos criamos um segundo usuário.

Após informar a senha do usuário ROOT, então o instalador deverá solicitar um nome de usuário, informe que quer criar o usuário chamado⁶: **userlinux**

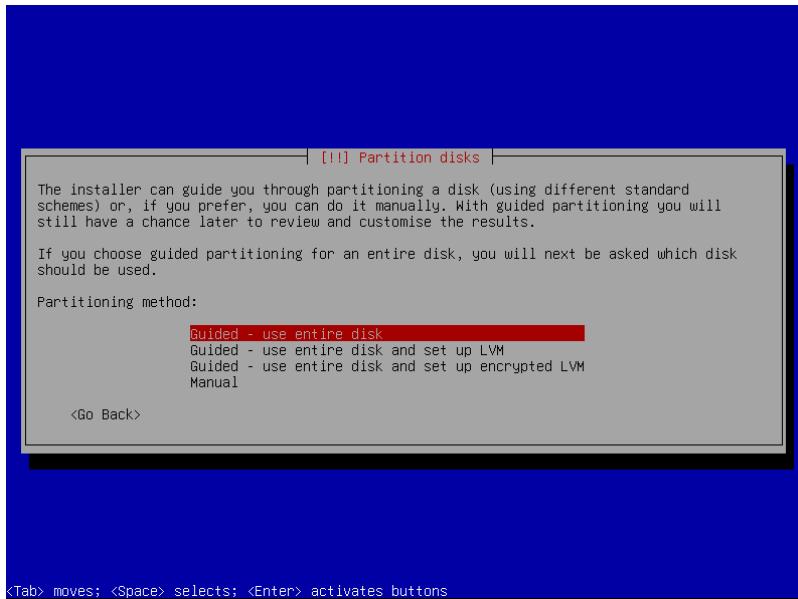
Vamos padronizar a senha como **123456** para todo o livro, naturalmente não é uma senha que se utilize na produção.

Durante todo o texto vou me referir a esta senha: **123456**.

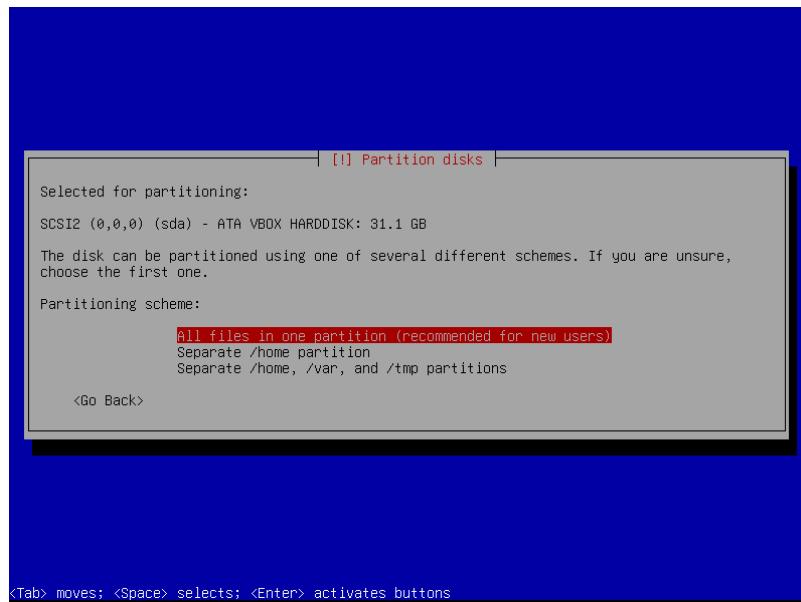
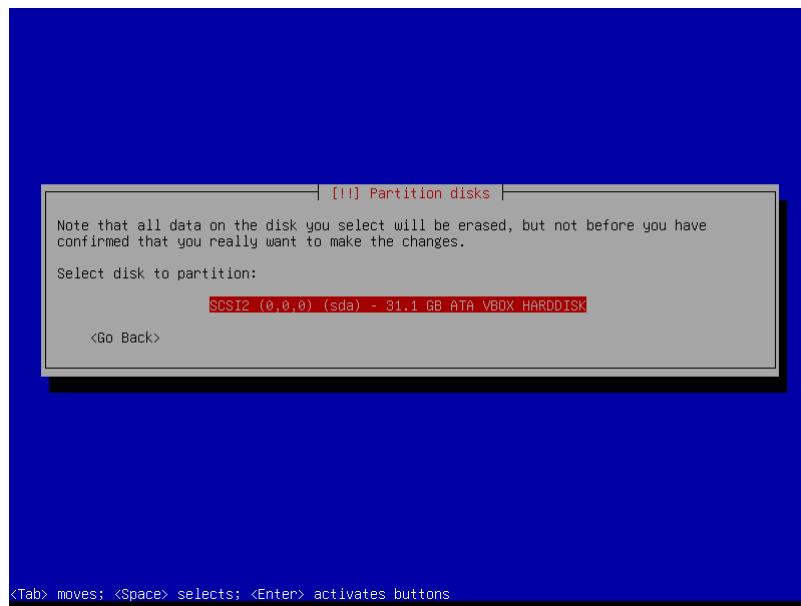
⁶ Todas as práticas do material estão direcionadas ao GNU/Linux Debian 12 e para o usuário **usuario**.



Precisamos localizar o servidor em uma Timezone, para isso escolha o estado que está morando, isso vai garantir o horário correto do seu GNU/Linux, o que é fundamental para criptografias e funcionamento de aplicativos.



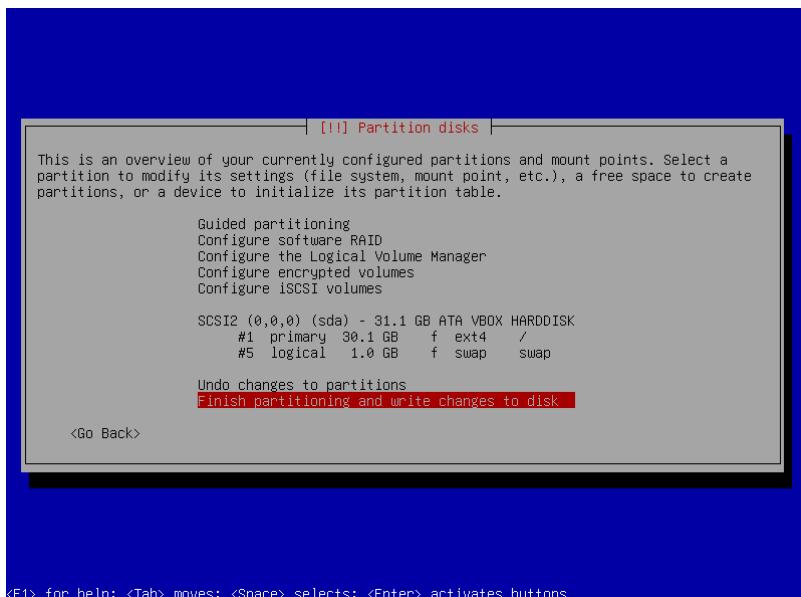
A escolha da forma em que vamos formatar o disco deverá ser o modo automático e todo o disco conforme figura abaixo, após o aluno completar parte do curso será demonstrado como realizar a formatação por meio de fdisk e a devida explanação sobre formatação lógica.



Informe o dispositivo de IO (disco) que irá utilizar, como essa Virtual Machine só possui 1 disco, é natural que terá só 1 opção.

No GNU/Linux existem várias formas de organizar os principais diretórios, o autor gosta de utilizar /home separado dos demais diretórios para poder criptografar os diretórios dos usuários.

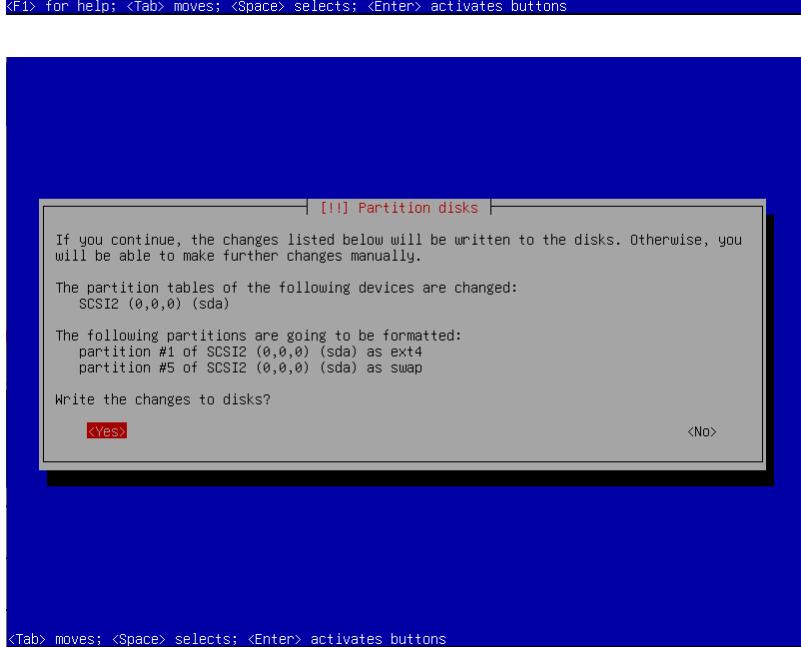
Mas aqui vamos seguir pela simplicidade para melhor aprendizado, escolha a primeira opção conforme figura.



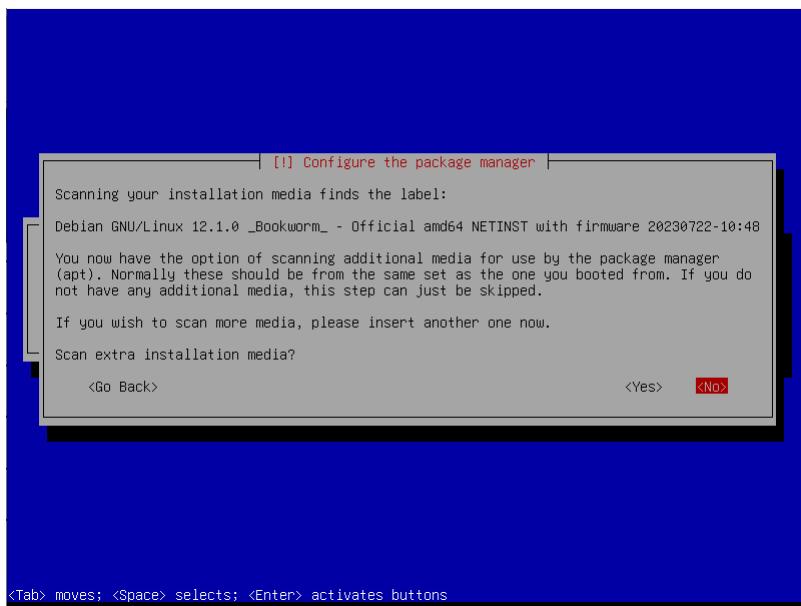
Confirme com a finalização e a escrita do disco, neste momento tenha ciência que será criado as seguintes partições

Raiz: /

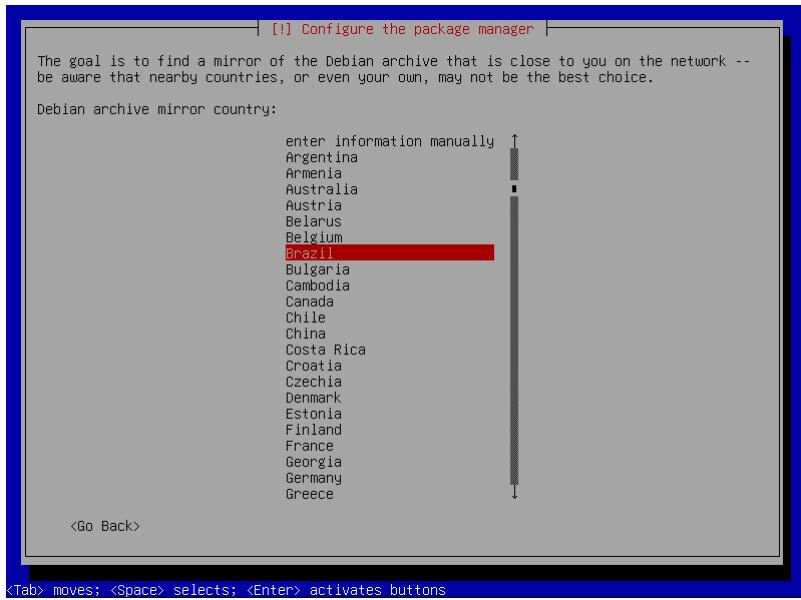
Swap: swap



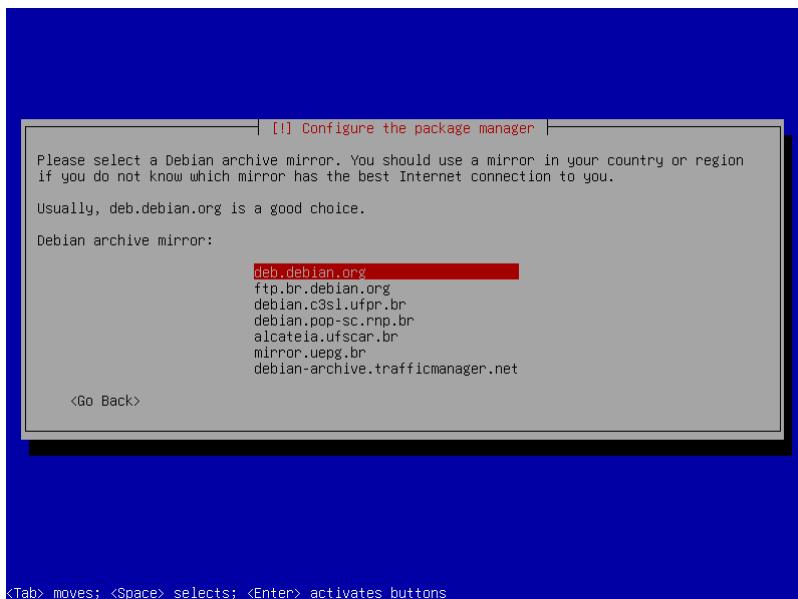
Confirme com **YES**, pois pelo risco o GNU/Linux deverá iniciar esta tela com NO.



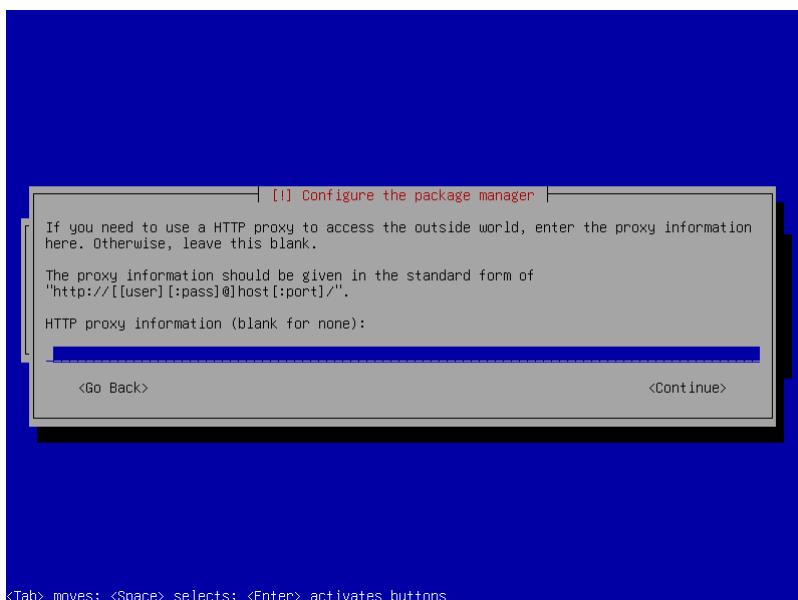
Como foi dito, o DVD ou ISO de instalação não contém todos os dados e arquivos para o seu Debian GNU/Linux, então escolha a opção **No**.



Selecione seu país, para que o GNU/Linux se conecte com o servidor de pacotes apropriado, nosso caso **Brazil**. No seu caso, vai depender da localização que está neste momento.

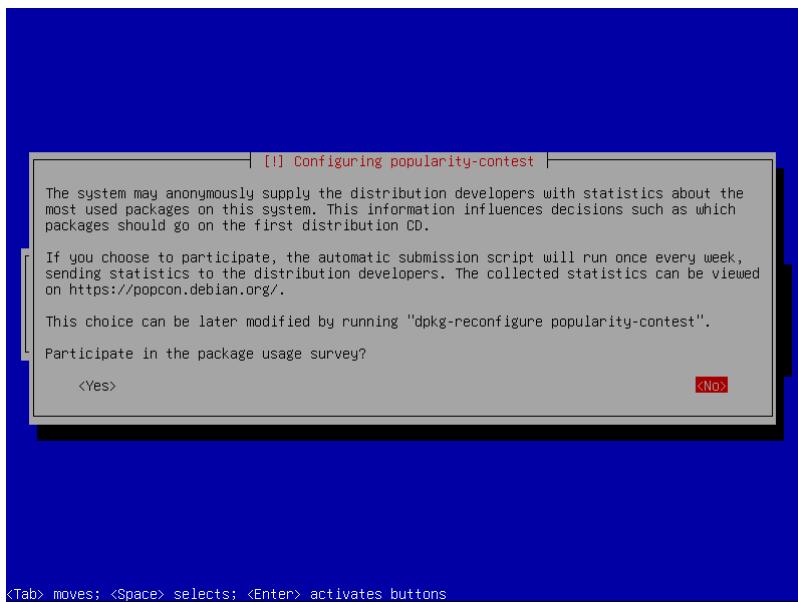


No **Brazil** temos muitas opções de **Mirror**, o melhor que se pode utilizar é o principal **deb.debian.org**



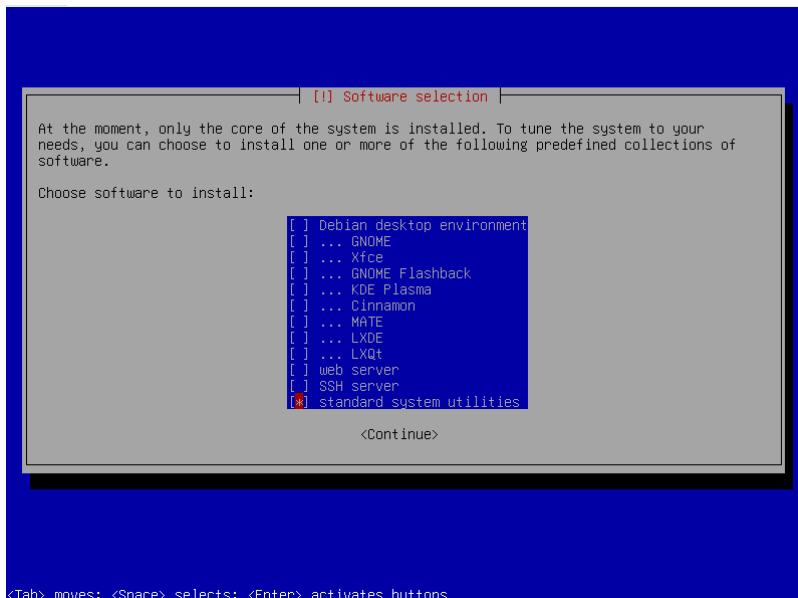
Uma casa ou uma empresa comum não possui um proxy por não necessitar, mas se sua instituição de ensino possui, então informe nesta tela.

Como não temos proxy, deixe o campo vazio e continue.



Evite enviar estatísticas, pois isso pode ser um possível problema de vulnerabilidade, mesmo que o GNU/Linux seja de graça e agradecemos por isso, temos que entender que é perigoso.

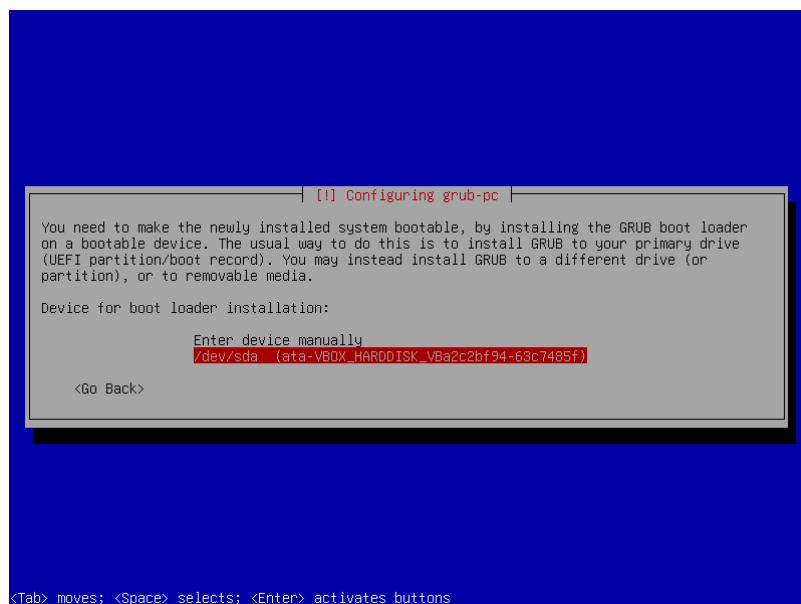
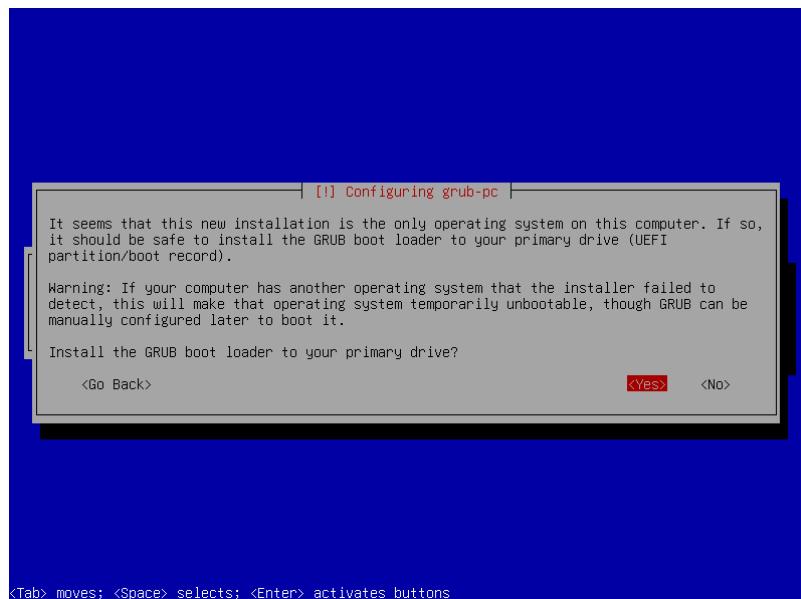
ATENÇÃO: O próximo passo requer muita atenção, leia antes de avançar.



Essa interface é muito importante, aqui definimos o que vamos ou não vamos utilizar (instalar).

Para nossas práticas vamos precisar somente do **Standard System Utilities**. Continue.

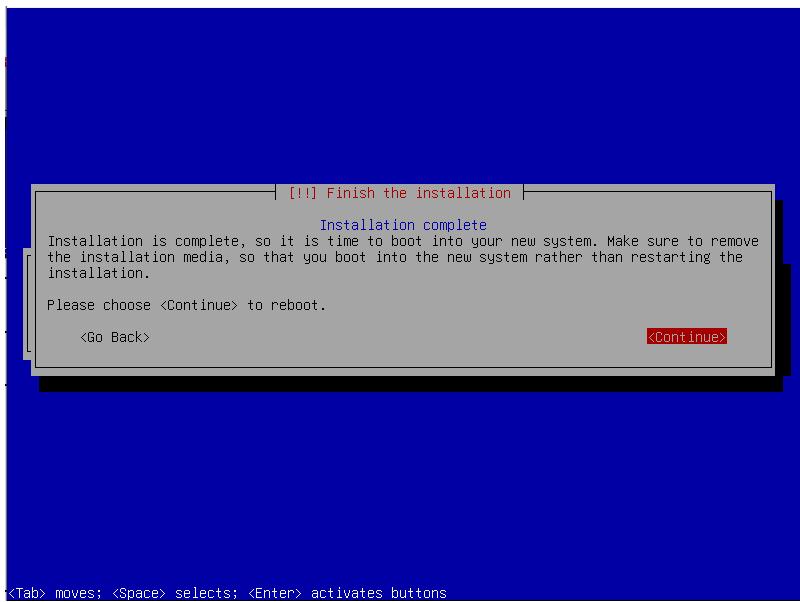
Atenção: Não pressione ENTER antes de desmarcar as opções padrões, utilize os cursores para cima e para baixo para navegar na listagem, e barra de espaço para marcar ou desmarcar.



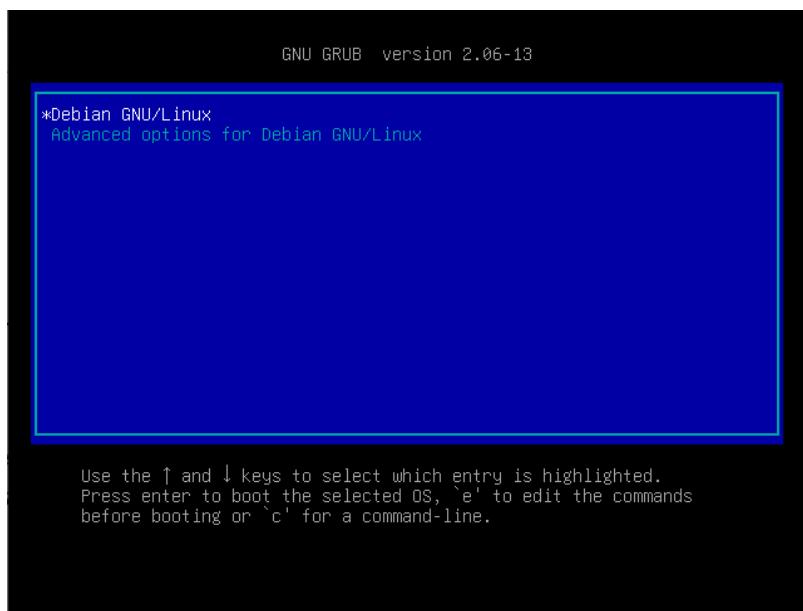
Sim, vamos utilizar o **GRUB** para carregar a imagem do sistema operacional **GNU/Linux**.

GRUB será abordado no capítulo de Inicialização e será discutido em detalhes lá.

E escolha o sistema de arquivo recém criado com a imagem do **GNU/Linux** selecionando a opção **sda**.



Successo, continue para fechar o instalador. Se estiver em um Hardware real (não estiver usando a Virtualização), remova o CD ou o Pendrive com a ISO.



A tela com a opção do Debian GNU/Linux selecionada, espere alguns segundos ou pressione **Enter** para iniciar o arranque do sistema.

```
debian GNU/Linux 12 debian tty1
debian login: _
```

O Debian GNU/Linux deverá iniciar o tty1 e exigirá usuário e senha para entrar.

```
Debian GNU/Linux 12 debian tty1
debian login: usuario
Password: _
```

Informe **userlinux** para o login e a senha **123456**

Repare que ao digitar 123456 os caracteres não aparecem, isso é feito por motivo de segurança, afinal sempre tem um espertinho bisbilhotando senhas digitadas.

```
Debian GNU/Linux 12 debian tty1
debian login: usuario
Password:
Linux debian 6.1.0-11-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.38-4 (2023-08-08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
usuario@debian:~$ _
```

Estando certo o usuário e a senha informados, o Debian GNU/Linux iniciará a interface do usuário baseado no shell default deste usuário.

Reparem que será aberto um banner com detalhes.

Seu Debian está pronto para ser utilizado, mas precisamos de passar a prática correta, ou seja, prática de segurança e bom uso do sistema. Para isso vamos realizar algumas configurações.

Repare nas imagens acima uma palavra chamada TTY, o Debian cria 7 TTY que você pode abrir. Ao alternar de TTY você pode manter dois ou mais prompts de comandos ativos, exemplo: Você pode ter no TTY 1 (Ctrl + Alt + F1)⁷ um editor de scripts como nano, vi ou vim, já no segundo TTY (Ctrl + Alt + F2) você pode ter um terminal só para executar o script. No Debian temos os seguintes TTY:

TTY 1: Um terminal e é acessível por Ctrl + Alt + F1;

TTY 2: Um terminal e é acessível por Ctrl + Alt + F2;

TTY 3: Um terminal e é acessível por Ctrl + Alt + F3;

TTY 4: Um terminal e é acessível por Ctrl + Alt + F4;

TTY 5: Um terminal e é acessível por Ctrl + Alt + F5;

TTY 6: Um terminal e é acessível por Ctrl + Alt + F6;

TTY 7: Se estiver em um Debian Gráfico é este o TTY com a sua interface gráfica, é acessível por Ctrl + Alt + F7;

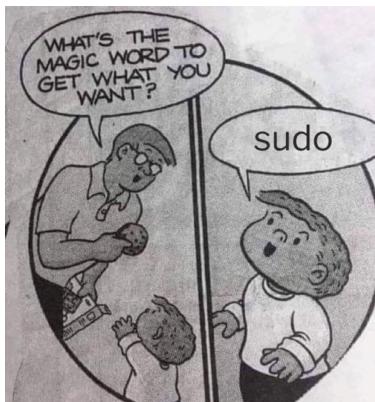
Mais duas informações devem obter agora, a primeira diz respeito ao layout do terminal. Veja nas imagens acima que após logon é exibido no terminal `userlinux@debian`. No caso

⁷ Mas lembre-se que se estiver em uma virtualização por VirtualBox deve-se utilizar o Ctrl da esquerda;

do Debian sem nenhuma alteração o username do usuário aparece antes do @ (arroba) e o nome da máquina aparece após o @ (arroba), desta forma quando está em terminais remoto saberá distinguir aonde você está. Teremos um capítulo que vai explicar como abrir terminais remotos.

A segunda observação é o símbolo (\$) nas imagens acima, isso significa que está como usuário comum e sem permissão de superusuário, isso garantirá que não faça besteiras tão graves. Já se for para conta de superusuário verá que o \$ será trocado por (#). Pessoas com bom senso não usam a conta privilegiada (#) e passam 99% de seu tempo na conta comum. Vamos ter um capítulo para discutir contas de usuário e acesso. Eu como professor reprovo alunos que usam a conta de super usuários nas práticas que não exigem, ou seja, 99% das práticas.

1.5 Instalando e configurando o SUDO



Embora o processo de instalação de pacotes será detalhado no capítulo de [Instalação de Pacotes](#) o sudo se faz necessário inicialmente por motivo de segurança, a idéia é usar a conta root só para fazer a primeira configuração e instalar o sudo, logo após, recomenda-se remover a senha do usuário root para não ser alvo de ataques.



O conceito sobre o comando **su** é necessário para prova de certificação **LPIC-1**

Debian 12 64 bits [Executando] - Oracle VM VirtualBox

Arquivo Máquina Visualizar Entrada Dispositivos Ajuda

```
userlinux@debian:~$  
userlinux@debian:~$ su root
```

No primeiro passo troque a sessão de **userlinux** para uma nova sessão com a conta **root**.

Para isso utilize o comando **su**, informando como parâmetro a conta **root**.

Debian 12 64 bits [Executando] - Oracle VM VirtualBox

Arquivo Máquina Visualizar Entrada Dispositivos Ajuda

```
userlinux@debian:~$  
userlinux@debian:~$ su root  
Password: _
```

Informa o password do usuário **root**, que é **123456** conforme processo de instalação.

O vídeo do processo pode ser visto no vídeo abaixo.



[Instalando e configurando o comando SUDO para práticas do curso Linux](#)



Para fazer a instalação, vamos utilizar um instalador de alto nível, no caso do Debian GNU/Linux utiliza-se o **apt** ou **apt-get**. Esse processo será detalhado no capítulo de [Instalação de Pacotes](#), mas vamos utilizar aqui para instalar o sudo. Então digite o comando.

```
1. apt install sudo -y
```

Veja na imagem abaixo, que deve estar na conta de super usuário.

```
root@debian:/home/usuario#
root@debian:/home/usuario# apt install sudo -y
```

Na imagem abaixo o resultado da instalação é detalhado no output, repare que não retornou erros, saiba que no mundo GNU/Linux ler a saída dos comandos e entender é uma ação fundamental.

Comando que será executado

```
root@debian:/home/usuario#
root@debian:/home/usuario# apt install sudo -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  sudo
  0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
  Need to get 1,889 kB of archives.
  After this operation, 6,199 kB of additional disk space will be used.
  Get:1 http://deb.debian.org/debian bookworm/main amd64 sudo amd64 1.9.13p3-1+deb12u1 [1,889 kB]
  Fetched 1,889 kB in 0s (4,474 kB/s)
  Selecting previously unselected package sudo.
  (Reading database ... 32796 files and directories currently installed.)
  Preparing to unpack .../sudo_1.9.13p3-1+deb12u1_amd64.deb ...
  Unpacking sudo (1.9.13p3-1+deb12u1) ...
  Setting up sudo (1.9.13p3-1+deb12u1) ...
  Processing triggers for man-db (2.11.2-2) ...
  Processing triggers for libc-bin (2.36-9+deb12u1) ...
root@debian:/home/usuario#
```

Saída do processo de instalação

O comando sudo precisa ser configurado, e existem duas formas de fazer isso. Neste ponto do material será utilizado o modo mais simples que é por edição do arquivo **/etc/sudoers**, o processo é simples, com o editor de texto no modo terminal, vamos abrir o arquivo e adicionar 1 linha. Digite o comando abaixo no terminal.

```
1. nano /etc/sudoers
```

Como é seu primeiro contato com o editor, esteja atento se está no terminal ou no editor. O editor possui no topo uma barra cinza com o caminho do arquivo (ver figura abaixo) e na

base inferior da tela comandos. Adicione a linha que vai habilitar o acesso ao comando sudo para o usuário **userlinux**.

```
1. userlinux    ALL=(ALL:ALL)  ALL
```

No capítulo de usuários tais atributos serão discutidos com mais ênfase.

```

GNU nano 7.2
/etc/sudoers *

# This fixes CVE-2005-4890 and possibly breaks some versions of kdesu
# (#1011624, https://bugs.kde.org/show_bug.cgi?id=452532)
Defaults    use_pty

# This preserves proxy settings from user environments of root
# equivalent users (group sudo)
#Defaults: %sudo env_keep += "http_proxy https_proxy ftp_proxy all_proxy no_proxy"

# This allows running arbitrary commands, but so does ALL, and it means
# different sudoers have their choice of editor respected.
#Defaults: %sudo env_keep += "EDITOR"

# Completely harmless preservation of a user preference.
#Defaults: %sudo env_keep += "GREP_COLOR"

# While you shouldn't normally run git as root, you need to with etckeeper
#Defaults: %sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_"

# Per-user preferences; root won't have sensible values for them.
#Defaults: %sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"

# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults: %sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"

# Ditto for GPG agent
#Defaults: %sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL)  ALL
userlinux ALL=(ALL:ALL)  ALL

# Allow members of group sudo to execute any command_
%sudo   ALL=(ALL:ALL)  ALL

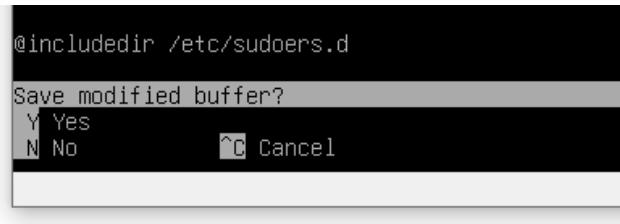
# See sudoers(5) for more information on "@include" directives:

```

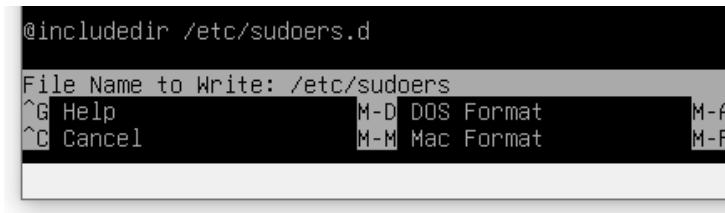
Onde:

1. Editor de texto nano;
2. Path do arquivo que está sendo editado;
3. Linha adicionada;
4. Pressione Ctrl+X (Ctrl da esquerda) para salvar.

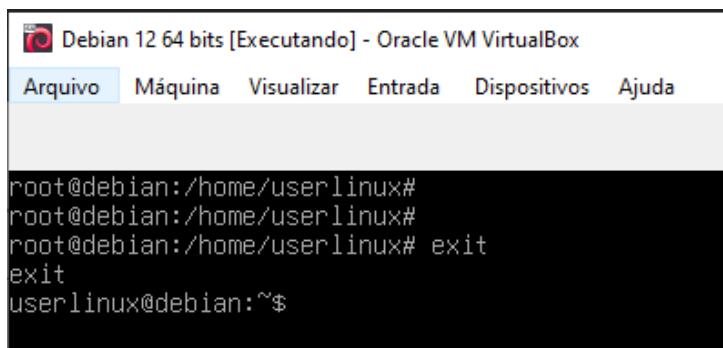
Para salvar pressione Ctrl+x (repare que por estar em uma VirtualBox você deve utilizar o Left Ctrl). O sistema vai informar (conforme figura abaixo) que o arquivo foi alterado, então você deve confirmar com **y** no caso da instalação estar em **English**.



Após responder **y** de **yes**, o sistema deverá confirmar o **path** do arquivo, que por padrão é o **path** inicial, simplesmente pressione **Enter** para continuar.



Feito isso o sistema retornará para o Terminal no qual pode-se executar agora comandos, e naturalmente será **exit** para finalizar a sessão do usuário **root** no qual não será mais acessado por motivo de segurança, deve-se evitar ao máximo o uso desta conta.



Agora seu Debian GNU/Linux está pronto para ser utilizado com segurança.

1.6 Validação de práticas por meio de automação

Como na educação a distância mediada pela Internet o modelo de transferência de conhecimento em grande parte é assíncrona, faz-se referência até ao Tanenbaum neste momento, a validação das práticas devem ser realizadas por intermédio de um agente dotado de validadores que são capazes de explicar ao aluno o ponto em que está errando.

Aied.com.br desenvolveu um conjunto de ferramentas que validam as práticas dos alunos realizadas neste material, e naturalmente abrangendo os erros clássicos observados nos últimos anos, trata-se de um agente que envia dados para um servidor de validação, e por este motivo é importante que **só seja instalado dentro de uma Máquina Virtual** e para fins educacionais e nunca instalado em um servidor ou em um Desktop pessoal.



[Validação de práticas por meio de Robô e suporte](#)



O primeiro passo é realizar o download do script de instalação que está disponível no site, mas isso deve ser feito dentro da máquina virtual que é terminal, para isso vamos utilizar o comando **wget** que será descrito no capítulo de [Interfaces de Rede](#). O script está escrito em Python e pode ser adotado, recomenda-se que sempre analise os scripts antes de executar.

```
1. wget -O /tmp/install.py http://www.aied.com.br/linux/download/install_v2.py
```

Onde em 1:

wget é o comando que faz uma requisição HTTP e obtém o arquivo de instalação;
 -O parâmetro que informa onde será salvo o arquivo bem como o nome;

```
Debian 12 64 bits [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda

userlinux@debian:~$ wget -O /tmp/install.py http://www.aied.com.br/linux/download/install_v2.py
--2024-08-09 17:27:08-- http://www.aied.com.br/linux/download/install_v2.py
Resolving www.aied.com.br (www.aied.com.br)... 212.1.209.207, 2a02:4780:1:793:0:3848:1b21:2
Connecting to www.aied.com.br (www.aied.com.br)|212.1.209.207|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1225 (1.2K) [text/plain]
Saving to: '/tmp/install.py'

/tmp/install.py
2024-08-09 17:27:08 (99.5 MB/s) - '/tmp/install.py' saved [1225/1225]
```

Onde:

1. Comando wget que realiza o download do arquivo de instalação;
2. Código 200 do protocolo http, indica sucesso;
3. Path do arquivo salvo no sistema de arquivos.

Sempre confirme o status da execução, pois um arquivo 404 pode ser baixado se a URL for descrita de forma errada, e naturalmente levar ao erro no próximo passo. Então agora que conhece o script, execute utilizando o Python:

```
1. sudo apt update -y
2. sudo apt install libjsoncpp-dev -y
3. sudo python3 /tmp/install.py
```

O processo de instalação deverá trazer vários pacotes clássicos do GNU/Linux, com o GCC, libCurl, entre outros pacotes do repositório oficial. O único arquivo que não está no repositório oficial está no site www.aied.com.br que vai ser instalado em **/etc/aied/**, este diretório pode ser auditado.

```
Connecting to www.aied.com.br (www.aied.com.br)|212.1.209.207|:80... connected
HTTP request sent, awaiting response... 200 OK
Length: 562454 (549K) [application/gzip]
Saving to: '/tmp/aied.tar.gz'

/tmp/aied.tar.gz          100%[=====] 2024-08-09 17:32:16 (704 KB/s) - '/tmp/aied.tar.gz' saved [562454/562454]

./dist/
./dist/aied_64
./dist/data/
./dist/data/en-us.json
./dist/data/config.json
userlinux@debian:~$
```

Um link será criado em **/bin/aied** apontando para **/etc/aied/aied_64**. Este link se faz necessário para que se utilize o novo comando **aied** de qualquer ponto do sistema de arquivos, então **aied_64** fará parte da lista de comandos do seu Debian GNU/Linux.



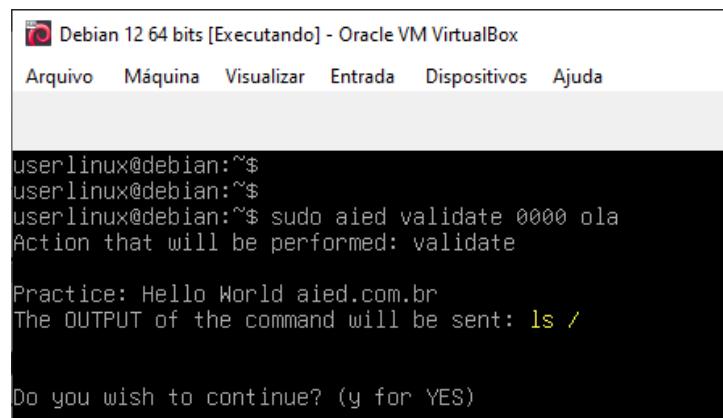
Para testar o ambiente é simples, digite o comando:

```
sudo aied validar 0000 ola
```

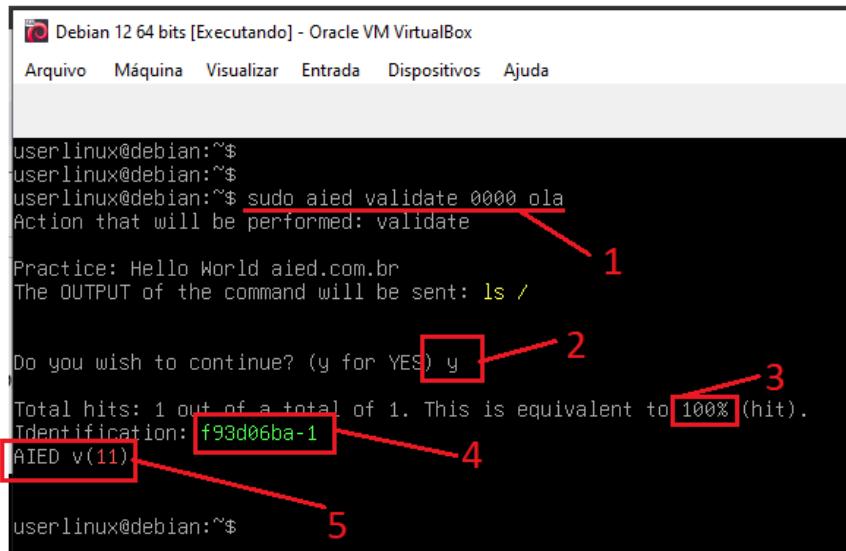
Onde:

aied é o programa escrito em Linguagem C obtido com o auxílio do instalador;
validar é a ação que o programa deverá executar;
0000 ola é o script utilizado para validação pela ferramenta, é o clássico *hello world*, estes dados serão informados durante as práticas e em destaque.

Respeitando o aluno e mesmo sabendo que este script está sendo executado em uma Máquina Virtual isolada é informado os comandos que serão executados pelo script e que o resultado da saída destes comandos serão enviados para o servidor de validação aied.com.br.



Se for aceito então o procedimento executa as operações citadas, que no exemplo acima é apenas um comando e envia para validação. Na figura acima, após ser executado o comando é enviado então para o servidor de validação que fez a validação e no item 1.1 confirmou que está tudo OK ao utilizar a cor VERDE.



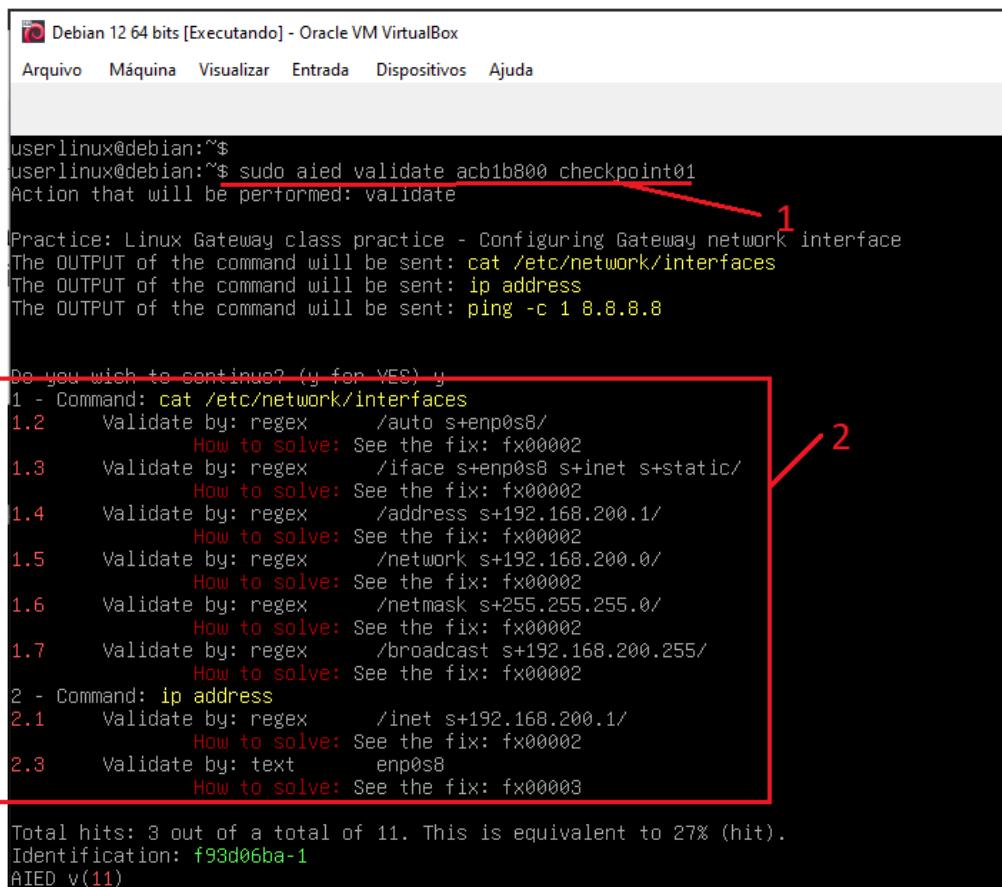
```
Debian 12 64 bits [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda

userlinux@debian:~$ sudo aied validate 0000 ola
Action that will be performed: validate
1
Practice: Hello World aied.com.br
The OUTPUT of the command will be sent: ls /
Do you wish to continue? (y for YES) y
2
Total hits: 1 out of a total of 1. This is equivalent to 100% (hit).
Identification: f93d06ba-1
3
4
AIED v(11)
5
userlinux@debian:~$
```

Onde:

1. Comando aied, com o código da prática 0000 e checkpoint ola;
2. Aceite do aluno, confirmando que aceita enviar dados para www.aied.com.br;
3. Total de acertos do aluno;
4. Código único do computador, cada instalação gera um código único;
5. Versão do sistema aied.

Algumas práticas serão muito complexas, este mecanismo poderá ajudar o aprendiz a localizar erros difíceis de serem localizados pois executam uma série de testes, como no exemplo abaixo.



```

Debian 12 64 bits [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda

userlinux@debian:~$ sudo aied validate acb1b800 checkpoint01
Action that will be performed: validate 1

|Practice: Linux Gateway class practice - Configuring Gateway network interface
|The OUTPUT of the command will be sent: cat /etc/network/interfaces
|The OUTPUT of the command will be sent: ip address
|The OUTPUT of the command will be sent: ping -c 1 8.8.8.8

Do you wish to continue? (y for YES) y
1 - Command: cat /etc/network/interfaces
1.2   Validate by: regex    /auto s+enp0s8/
      How to solve: See the fix: fx00002
1.3   Validate by: regex    /iface s+enp0s8 s+inet s+static/
      How to solve: See the fix: fx00002
1.4   Validate by: regex    /address s+192.168.200.1/
      How to solve: See the fix: fx00002
1.5   Validate by: regex    /network s+192.168.200.0/
      How to solve: See the fix: fx00002
1.6   Validate by: regex    /netmask s+255.255.255.0/
      How to solve: See the fix: fx00002
1.7   Validate by: regex    /broadcast s+192.168.200.255/
      How to solve: See the fix: fx00002
2 - Command: ip address
2.1   Validate by: regex    /inet s+192.168.200.1/
      How to solve: See the fix: fx00002
2.3   Validate by: text    enp0s8
      How to solve: See the fix: fx00003

Total hits: 3 out of a total of 11. This is equivalent to 27% (hit).
Identification: f93d06ba-1
AIED v(11)

```

Onde:

1. A prática que está sendo validada;
2. Falhas do aluno, e repare que é informado um fix para correção, no final deste livro todos estes códigos de FX devidamente abordados para que o aluno acerte a prática sem a necessidade de acionar o professor.

A ideia é que AIED reduza o esforço do professor para que este tenha mais tempo para atuar na gestão e na evolução do curso.

1.7 Criando uma exportação ou importando uma VM

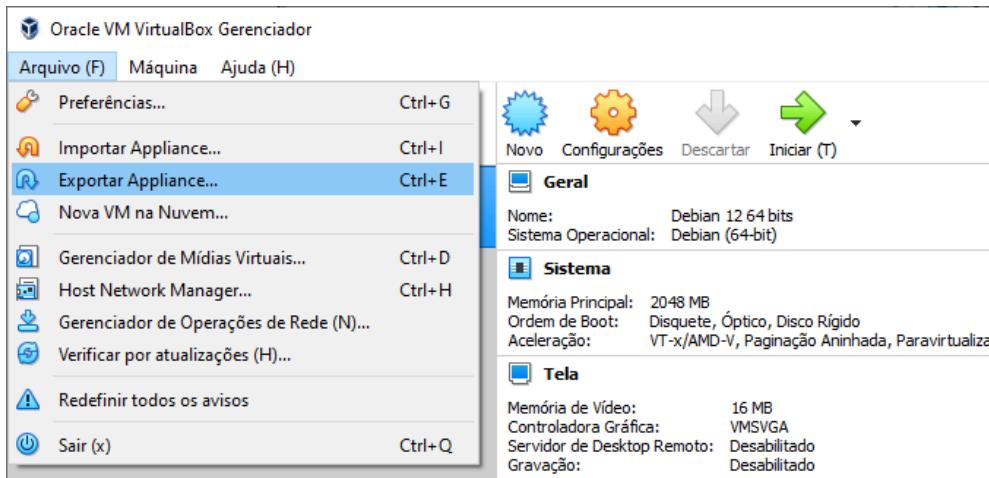
Durante as aulas o aluno irá modificar muito o Debian GNU/Linux na Máquina Virtual, então neste ponto o aluno deve parar e sair do Linux executando o comando de saída conforme listagem abaixo.

1. sudo poweroff

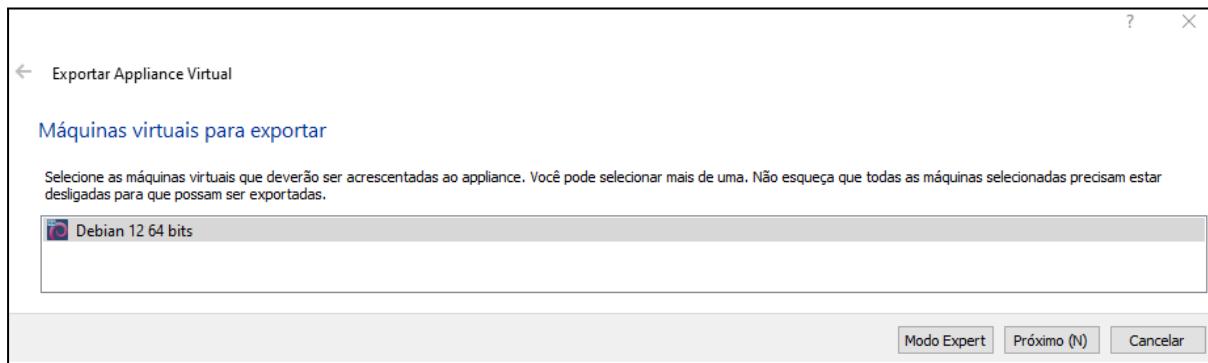
Caso tenha dificuldades, também pode recorrer ao vídeo abaixo.



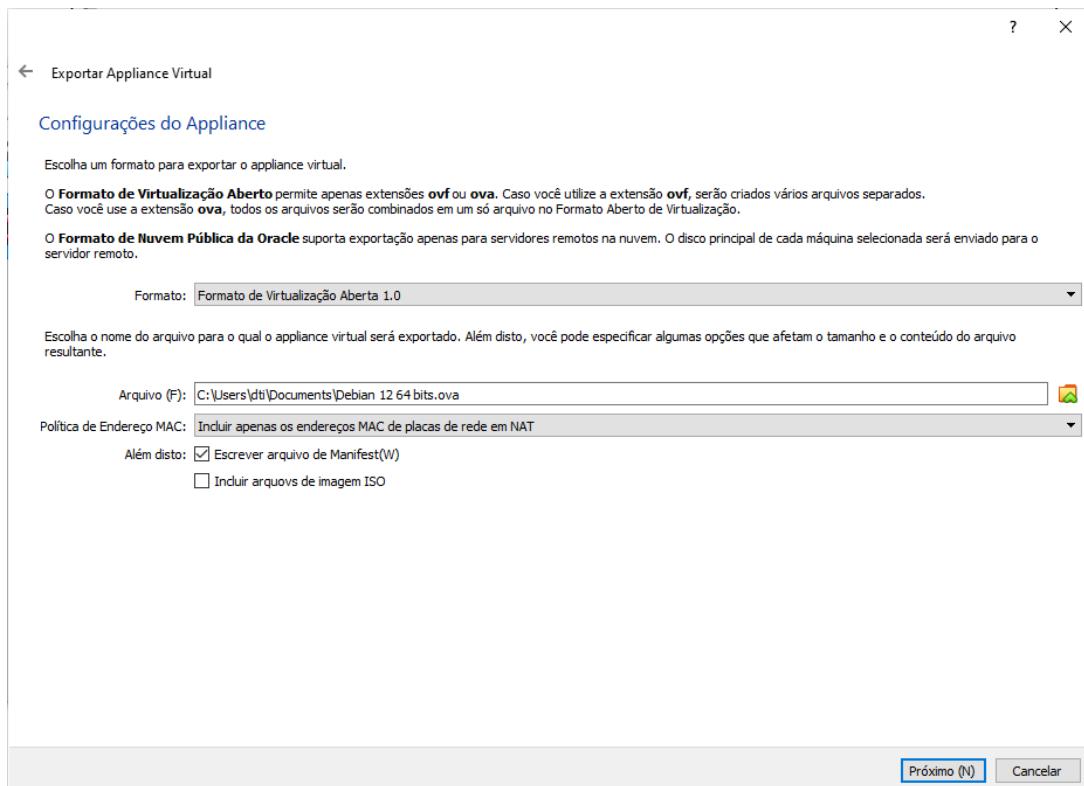
Não se deve desligar o GNU/Linux por VirtualBox (fechando a janela) pois é a mesma coisa que puxar o cabo de energia de seu computador, deve-se executar o comando acima. Com a Máquina Virtual desligada, no VirtualBox selecione a opção **Exportar Appliance...** no menu **Arquivo** conforme figura abaixo.



Escolha a máquina virtual recém criada com nosso Debian GNU/Linux e avance com o botão próximo.

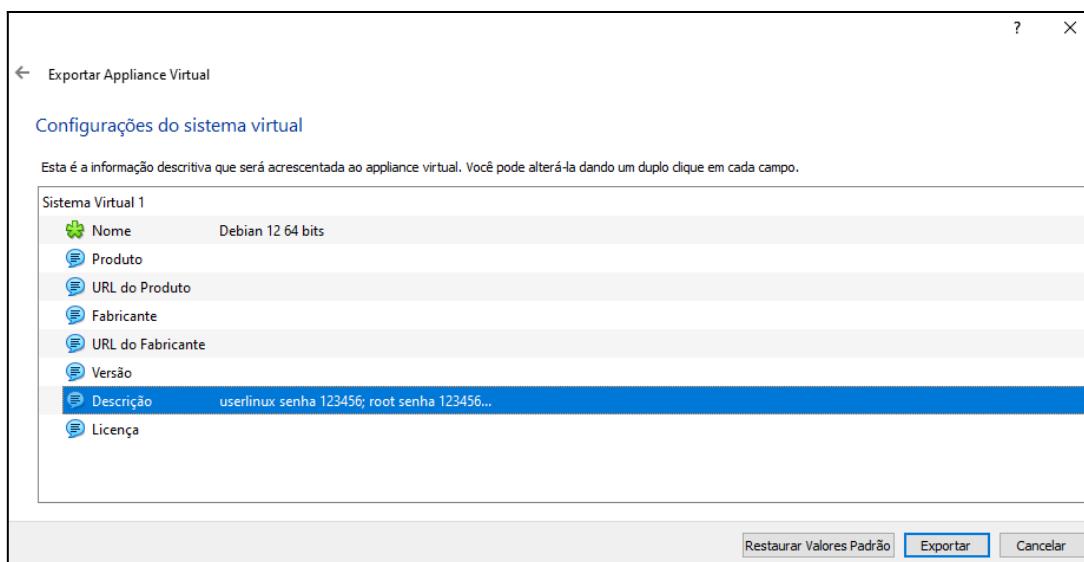


No próximo passo, vamos selecionar o formato do arquivo, utilize o formato aberto 1.0 para ser compatível com outros virtualizadores. Agora é só avançar com o botão Próximo.

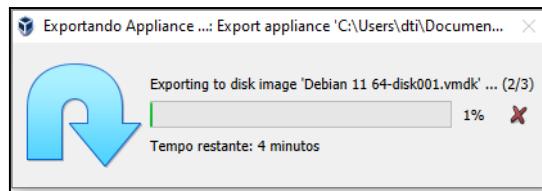


Definindo formato do arquivo

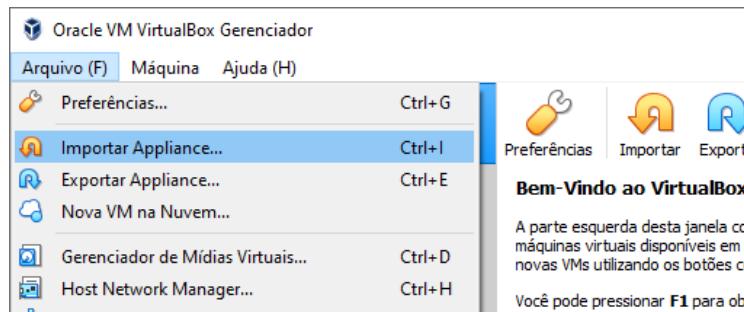
Recomenda-se colocar na descrição o usuário e a senha para que no futuro se lembre e não tenha que refazer tal instalação.



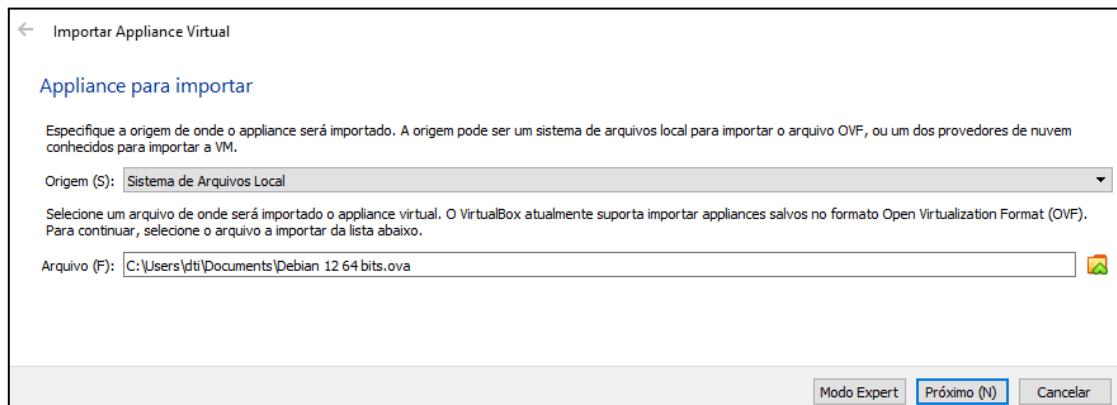
Agora é o momento de aguardar, o processo vai depender de 3 a 10 minutos para um Debian GNU/Linux deste tamanho.



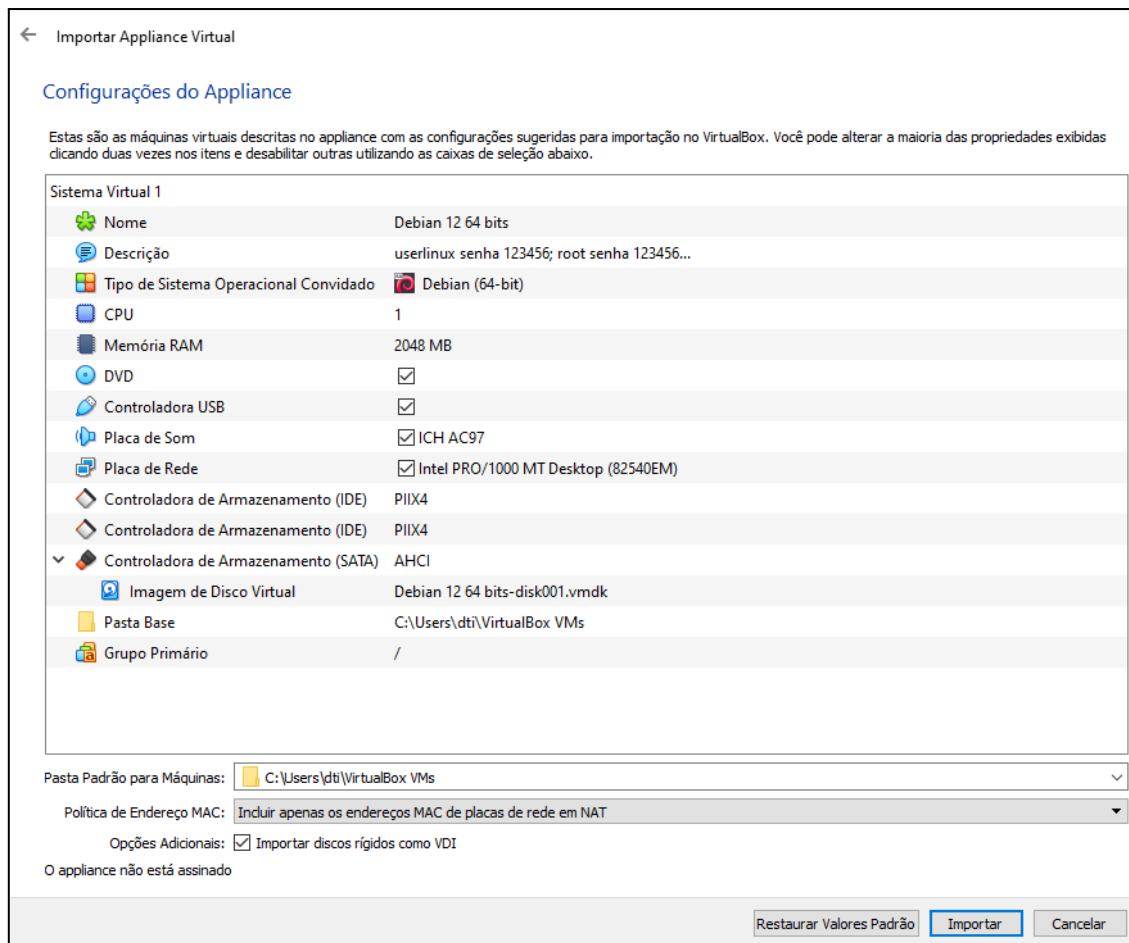
No final o aluno terá um arquivo com extensão **.ova**, guarde este arquivo e sempre que precisar de uma nova máquina virtual não precisa realizar toda a instalação realizada até então, basta importar. Para exemplificar como se importa vamos importar uma máquina virtual para que entenda o processo de importação por completo. Comece indo na opção **Importar Appliance...** conforme figura abaixo.



No próximo passo, selecione o arquivo **.ova** que foi criado no processo de exportação, conforme exemplo na figura abaixo.



Recomendo sempre trocar o Nome, pois o Nome da Virtual Machine pois o VirtualBox irá criar um diretório com o nome da Virtual Machine.



Agora clique no botão **Importar** e aguarde o processo. É recomendado que sempre que iniciar um capítulo faça uma importação nova.



1.8 Virtual Networking no VirtualBox

Sendo simplista neste momento, posso dizer que em um cenário real temos inúmeras formas de se arranjar os equipamentos e serviços, quero deixar bem abstrato neste momento pois de prática em prática o aluno pegará a proposta da topologia e da rede.



Nem toda a prática terá a mesma configuração de rede, e a medida que o aluno avança ele vai entendendo. O VirtualBox nos dá todos os cenários que precisamos, ou seja, além de virtualizar o Sistema Operacional podemos virtualizar uma rede pequena para o estudo e o

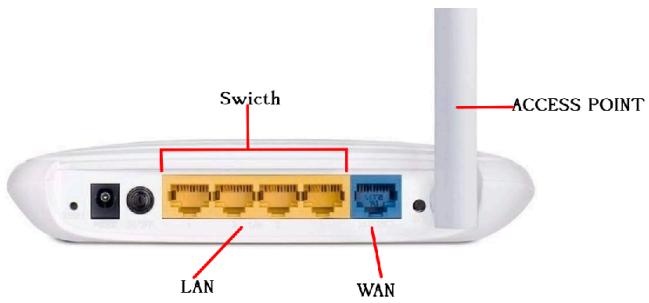
aprendizado. Costumo indicar depois deste conteúdo que o aluno estude sobre GNS3, mas ele precisa de ter maturidade para isso, e este livro o dará.

Neste ponto do livro não posso exigir do aluno que ele saiba tudo sobre redes, é intuído deste livro o ensinar isso também, então vamos começar a descrever a rede do próprio aluno. A começar pela nomenclatura⁸, conforme tabela abaixo.

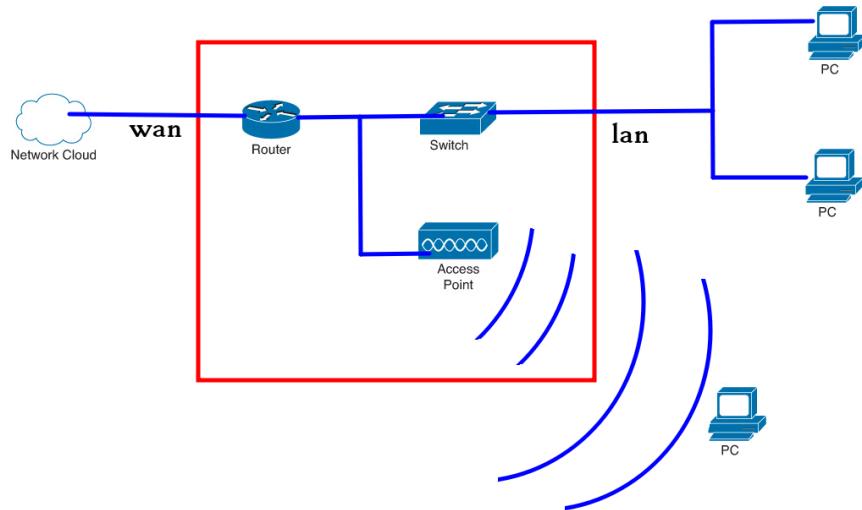
Imagen	Nome	Função
 Router	Router	O Router tem inúmeras funções em uma rede de computadores, neste livro será utilizado como segmentador de redes e gateway.
 Switch	Switch	Os equipamentos finais (Desktop, mobile, Impressora) são ligados nos dispositivos de acesso ao meio, no qual o Switch faz parte.
 Access Point	Access Point	Equipamento de acesso ao meio com tecnologia 802.11 e operando no meio físico com radiofrequência.
 Firewall	Firewall	Firewall é uma função muito específica do Router, é tão específica que neste livro será tratada com uma imagem diferente.
 Desktop	Desktop	O Desktop é um dispositivo final de usuário, utilizado pelas pessoas com o intuito de consumir serviços da rede.
 Server	Server	O Server é um dispositivo final de usuário, utilizado pelas pessoas para provar serviços na rede.
 Database	Database	É um serviço da rede, um serviço de armazenamento de dados, o que é comum para desenvolvedores.
 Server Web	Server Web	Um servidor específico projetado para servir protocolo HTTP para browsers, geralmente com um Container WEB.
 Network Cloud	Network	Será utilizada para descrever uma rede mais abrangente, geralmente obtida por parte de um contrato com a operadora.

Devo supor que o aluno está em uma rede doméstica, então devo supor que este possui um Router comum, que na verdade é uma caixa plástica conforme figura abaixo, que possui a função de Modem, Router, Switch e Access Point.

⁸ Ícones padrões definidos pela CISCO e descritos pela OREILLY na url:
<https://www.oreilly.com/library/view/interconnecting-cisco-network/9780133410235/pref04.html>



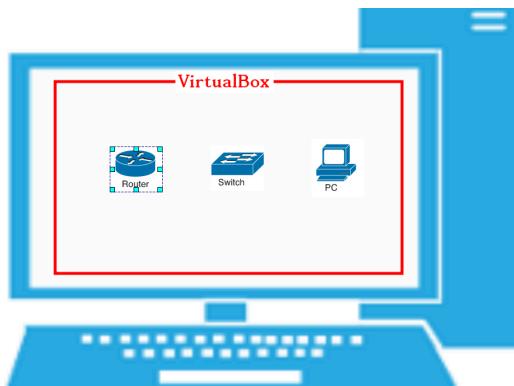
Em um ambiente empresarial tais funções são separadas em equipamentos independentes para atingir maior performance. Na figura abaixo a caixa vermelha representa o equipamento da figura acima.



Não temos nenhum controle sobre a rede WAN que é cedida pela empresa de Telecomunicação, mas na LAN temos controle e podemos operar normalmente. Este é um esquema de uma rede doméstica. Estas redes domésticas estão configuradas para assumir alguns destes IPs:

Tipo	Início	Fim	Máscara
A	10.0.0.0	10.255.255.255	255.0.0.0
B	172.16.0.0	172.31.255.255	255.255.0.0
C	192.168.0.0	192.168.255.255	255.255.255.0

Se o aluno utiliza o **Microsoft Windows** pode digitar o comando **ipconfig** que irá verificar tal configuração, e se o usuário já é um usuário **GNU/Linux** pode digitar o comando **ip address** e também será capaz de verificar isso. Como geralmente os equipamentos residenciais operam entre 192.168.0 até 192.168.5 neste livro vamos trabalhar com o endereço 192.168.200 dentro da rede interna do VirtualBox. Mas antes de saber sobre isso o aluno tem que entender aonde o VirtualBox está na topologia acima. Na verdade o VirtualBox é um conjunto de aplicações que permitem a virtualização de um hardware e então é possível instalar outro sistema operacional sobre essa virtualização, então na topologia acima, pode estar em qualquer equipamento Desktop (PC).



No VirtualBox podemos simular uma rede pequena, pois geralmente o computador do aluno possui pouca memória, o GNS3 requer muita memória. Cada máquina virtual possui no máximo 4 adaptadores de rede, um adaptador de rede é uma placa de rede virtual. Cada um dos adaptadores de rede pode ser configurado separadamente para operar em um dos seguintes modos:

Not attached: Neste modo, o Oracle VM VirtualBox informa ao convidado que uma placa de rede está presente, mas que não há conexão. É como se nenhum cabo Ethernet estivesse conectado. Usando este modo, é possível “puxar” o cabo Ethernet virtual e interromper a conexão, o que pode ser útil para informar a um sistema operacional convidado que nenhuma rede conexão está disponível e impõe uma reconfiguração;

Network Address Translation (NAT): Se tudo o que você deseja é navegar na Web, baixar arquivos, e visualizar e-mail dentro da máquina virtual, então este modo padrão deve ser suficiente para você, e você pode pular o restante desta seção. Não requer conhecimento nenhum para usar;

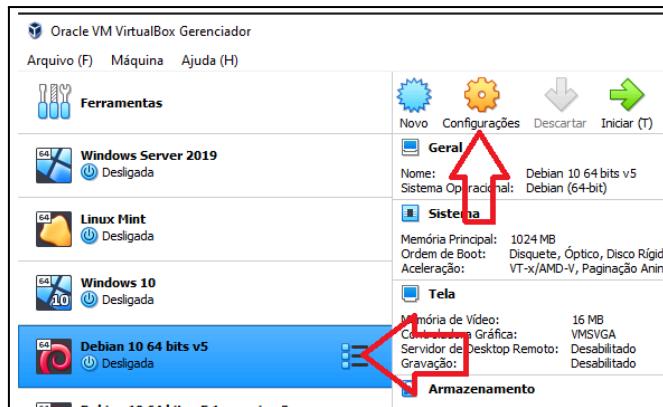
NAT Network: Uma rede NAT é um tipo de rede interna que permite conexões de saída;

Bridged networking: Isto é para necessidades de rede mais avançadas, como simulação de rede e execução de servidores em uma máquina virtual. Quando ativado, o Oracle VM VirtualBox se conecta a uma de suas placas de rede instaladas (real) e troca pacotes de rede diretamente com o Router da rede doméstica (físico). ATENÇÃO: neste modo a sua máquina real tem acesso a máquina virtual pela rede.

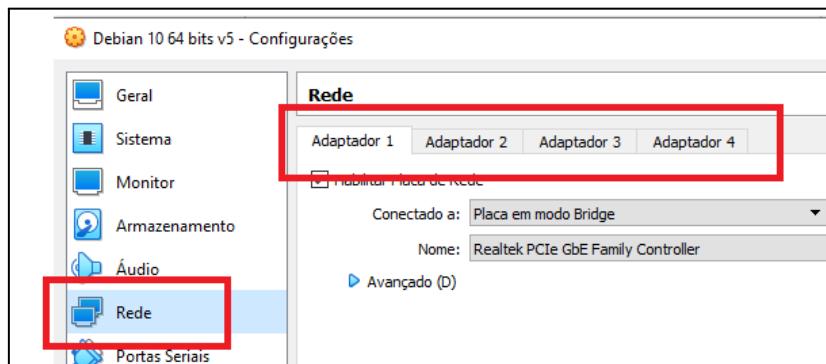
Internal networking: Isso pode ser usado para criar um tipo diferente de rede, uma rede interna dentro do VirtualBox isolada da rede doméstica real. Muito útil quando se deseja testar uma configuração ou executar aplicativos perigosos;

Host-only networking: Isso pode ser usado para criar uma rede contendo o host e um conjunto de máquinas virtuais, sem a necessidade da interface de rede física do host. Em vez disso, uma interface de rede virtual, semelhante a uma interface de loopback, é criada no host, fornecendo conectividade entre máquinas virtuais e o host. Sem nenhum tipo de configuração extra as máquinas convidadas não terão acesso a rede real;

Para configurar uma placa de rede (adaptador) de uma máquina virtual, primeiro deve-se desligar a máquina virtual, depois selecionar a máquina virtual e clicar em configurações.



Depois na aba Network, selecione os adaptadores, o correto é configurar somente a quantidade exata de adaptadores que é requisitado por prática.



Embora o VirtualBox nos dê muitas opções, neste curso serão usados apenas as configurações:

- **Network Address Translation (NAT);**
- **Bridged networking;**
- **Internal networking;**

É baseado na configuração do adaptador que configuramos a abrangência do acesso, por exemplo há configuração no qual o acesso da VM não pode passar do host, já existem necessidades que a VM tem que acessar toda a LAN real. Para simplificar a explicação, projetamos uma tabela de acesso.

Modo	VM→Host	VM←Host	VM1↔VM2	VM→LAN	VM←LAN
Network Address Translation	sim	não	não	sim	não
Bridged networking	sim	sim	sim	sim	sim
Internal networking	não	não	sim	não	não

Onde:

VM→Host: A máquina virtual é capaz de acessar o Host pela rede;

VM←Host: A máquina virtual é acessada pelo Host;

VM1↔VM2: Uma máquina virtual pode acessar outra pela rede;

VM→LAN: A Máquina virtual pode acessar a LAN real;

VM←LAN: Qualquer máquina na LAN real pode acessar a Máquina Virtual.

O aluno tem que entender que podemos ter no máximo quatro adaptadores, então pode-se mesclar as configurações acima e naturalmente, criar serviços. Um exemplo é um Gateway, no qual o aluno aprenderá a fazer, nesta configuração a máquina Gateway terá dois adaptadores, um em modo **Bridged networking** e a outra em modo **Internal Networking**.

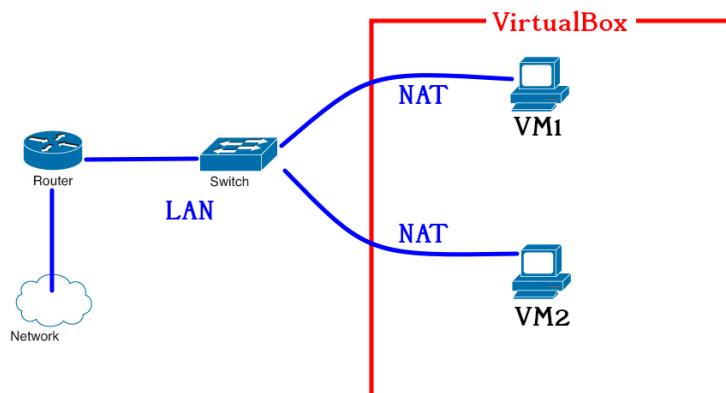
1.8.1 Adaptador em NAT

Network Address Translation (NAT) é a maneira mais simples de acessar uma rede LAN a partir de uma máquina virtual. Normalmente, não requer nenhuma configuração na rede host e no sistema convidado, ou seja, o VirtualBox faz tudo. Por esse motivo, é o modo de rede padrão no Oracle VM VirtualBox.

Uma máquina virtual com NAT habilitado funciona como um computador real que se conecta à Internet por meio de um roteador virtual. O roteador, neste caso, é o mecanismo de rede Oracle VM VirtualBox, que mapeia o tráfego de e para a máquina virtual de forma transparente. Uma Máquina virtual consegue acessar o computador real Host e a LAN do aluno, mas o inverso não é possível.

Modo	VM→Host	VM←Host	VM1↔VM2	VM→LAN	VM←LAN
Network Address Translation	sim	não	não	sim	não

No Oracle VM VirtualBox este roteador é colocado entre cada máquina virtual e o host. Essa separação maximiza a segurança, pois, por padrão, as máquinas virtuais não podem se comunicar entre si. A desvantagem do modo NAT é que, assim como uma rede privada atrás de um roteador, a máquina virtual é invisível e inacessível da Internet externa.



Esta configuração é ideal quando não é preciso acessar nenhum serviço na Virtual Machine como um serviço na rede local, ou seja, quando ela fica isolada dentro do Host mas com acesso aos serviços da rede LAN.

A máquina virtual deve estar em modo de configuração de rede por DHCP, todos os GNU/Linux modernos vem com esta configuração por padrão, então o Oracle VirtualBox por DHCP libera dados para a auto configuração da Virtual Machine. O endereço IP atribuído à máquina virtual geralmente está em uma rede completamente diferente da do host (LAN real). A configuração do adaptador em NAT obtém um endereço na rede 10.0.2.0/24 e se tiver um segundo adaptador de rede em modo NAT este segundo obtém o endereço na rede 10.0.3.0/24, e assim por diante.

1.8.2 Modo Bridge

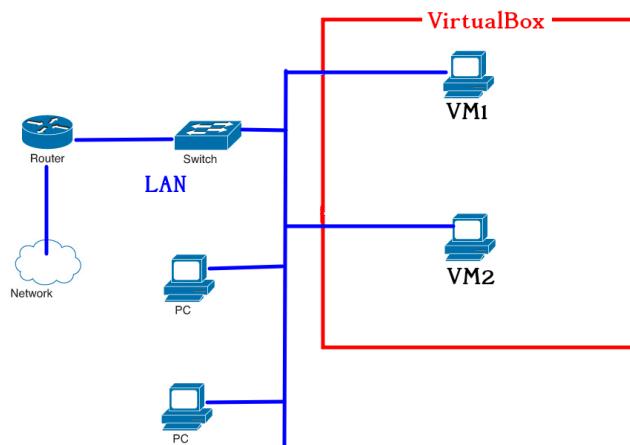
Com o Adaptador em Bridge o VirtualBox usa um driver de dispositivo real em seu host físico para se conectar na rede local real, para isso utiliza um driver denominado driver de filtro de rede.

Modo	VM→Host	VM←Host	VM1↔VM2	VM→LAN	VM←LAN
Bridged networking	sim	sim	sim	sim	sim

Isso permite que o VirtualBox intercepte dados da rede física ou envie dados para ela, criando efetivamente uma nova interface de rede no software.

Quando uma máquina virtual está usando essa nova interface virtual, parece ao sistema host como se a máquina virtual estivesse fisicamente conectada à interface por meio de um cabo de rede real. Então a Virtual Machine consegue acessar pela rede qualquer computador e vice versa.

O host pode enviar dados para a máquina virtual por meio dessa interface e receber dados dele, isso significa que você pode configurar o roteamento ou a ponte entre a máquina virtual e o resto da sua rede.



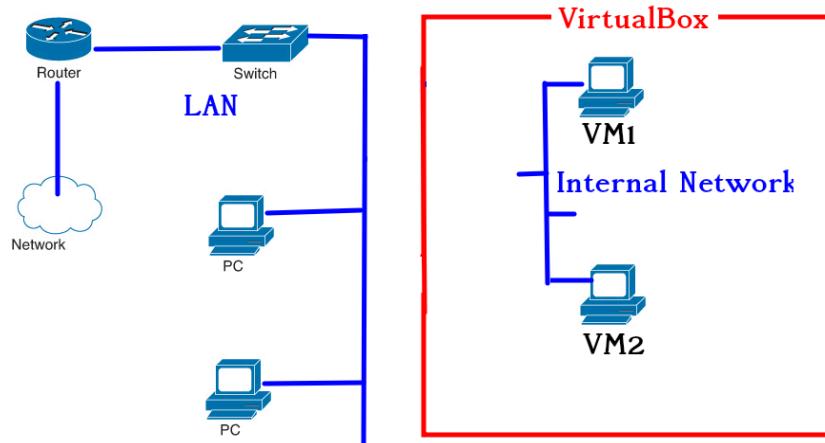
1.8.3 Rede Interna

As redes internas são criadas automaticamente conforme necessário, não há configuração central. Cada rede interna é identificada simplesmente por seu nome (ID).

Quando houver mais de um Adaptador virtual ativo com a mesma ID de rede interna, o driver de suporte do VirtualBox conectará automaticamente as placas e atuará como um switch de rede.

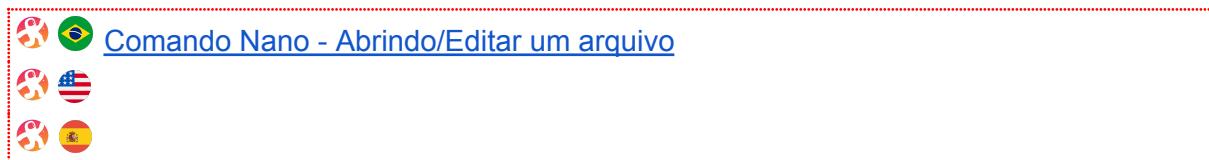
Modo	VM→Host	VM←Host	VM1↔VM2	VM→LAN	VM←LAN
Internal networking	não	não	sim	não	não

Utiliza-se esta rede para fazer uma configuração de LAN privada virtual entre as máquinas virtuais.



1.9 Editor de texto nano

Muitos amantes GNU/Linux pensarão em desqualificar este texto por conta do uso do comando nano para editar um texto, mas o leitor expert tem que entender que nano é muito mais amigável para um usuário leigo no assunto GNU/Linux, e este será o motivo da escolha do nano como editor principal. Mas saiba que vi e vim serão discutidos mais à frente.



O comando nano é simples e isso é o que nos importa neste momento, basta digitar o comando e o caminho do arquivo, conforme exemplo abaixo.

```
Debian 12 64 bits [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
userlinux@debian:~$ userlinux@debian:~$ userlinux@debian:~$ nano /tmp/arquivoquenaoexiste
1                               2
```

Onde:

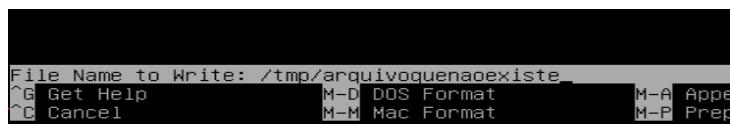
1. Comando nano. Nano é o editor de texto mais like;
2. Path do arquivo que quer alterar/criar;



Onde:

1. Versão do editor Nano;
2. Path do arquivo que está sendo editado;
3. Área de texto do arquivo;
4. Menu de opções;

Veja que nas opções você deve pressionar a tecla **Ctrl** da esquerda + a opção, no caso de salvar o arquivo **Ctrl+o**, o nano questiona sobre o nome do arquivo, pressione **ENTER**.



Para fechar utilize a opção **Ctrl+x**, leia as instruções do nano ao fechar o arquivo. Caso o arquivo não exista então ele cria, caso o arquivo exista ele abre para edição.

1.10 Recomendações e divisão dos capítulos

Este material engloba toda a prática de Linux bem como sua implementação em uma rede de computadores, consumindo e disponibilizando serviços, segmento este conteúdo em 2 momentos que devem ser seguidos conforme a prática se revelou ao longo dos anos, mas vou adicionar pré-requisitos externos (2 e 4), são estes:

1. **Livro Sistemas Operacionais Modernos** do autor **Andrew Stuart Tanenbaum**;
2. **Sistemas Operacionais prático com GNU/Linux**: Conteúdo sobre o GNU/Linux em si, este naturalmente isolado, para isso deve-se aplicar do capítulo 1 ao capítulo 13 deste livro;
3. **Redes de Computadores** do autor **Andrew Stuart Tanenbaum**;
4. **Redes de Computadores prático com GNU/Linux**: Conteúdo sobre como o GNU/Linux se comportar em uma rede de computadores e como estes interagem consumindo ou prestando serviços na rede, aplica-se do capítulo 14 em diante deste livro;

O curso tem a visão de trazer informações conceituais e alinhar tais conceitos com práticas, e foi também pensado para formação de alunos para todas as provas de certificação Linux LPI da Linux Professional Institute que pode ser acessado pelo link <https://www.lpi.org/pt-br/>.

O material exigirá 40 aulas de 4 horas para se completar todas as atividades e ver todos os conceitos.

2 Sistema Operacional Linux e Distribuições

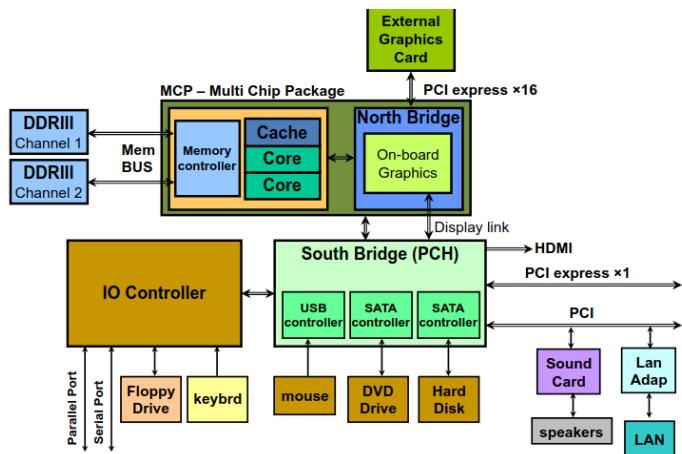
Antes de discutir o conceito de Linux, GNU e Distribuição, temos que desmistificar o que é um Sistema Operacional, o aprendiz sem estes conceitos não vai entender o funcionamento de certos recursos no Linux e muito menos saber distinguir Linux de Microsoft Windows.

Seria traumático trazer todo um curso de Arquitetura e Organização de Computadores, um curso de Hardware e um de Teorias de Sistemas Operacionais. Se o aprendiz já teve tais matérias na Faculdade/Universidade este está apto a pular o capítulo 2.

2.1 Arquitetura de um computador

Um Computador é composto de placas de circuito impresso que interconectam vários componentes e fornece conectores para dispositivos periféricos. A placa principal neste tipo de sistema, a placa-mãe, fornece as conexões que permitem a comunicação dos componentes do sistema. A placa mãe também é formada por vários dispositivos de I/O embutidos, mas há possibilidade de expansão.

Esses canais de comunicação são normalmente chamados de barramentos de computador. A Figura abaixo ilustra como os diferentes componentes se conectam para execução das operações de processamento de dados em um computador de uso geral.



Os dois componentes mais importantes que estão conectados à placa-mãe são o processador, que executa os programas, e a memória principal, que armazena temporariamente os programas executados e seus dados associados. O processador é comumente referido como unidade central de processamento (CPU). A CPU acessa dados que estão na memória principal e busca instruções também na memória principal, em seguida, executa essas instruções para processar os dados. Repare que é a mesma ideia descrita por **John von Neumann**.

Ler da memória principal costuma ser dramaticamente mais lento do que ler da própria memória interna da CPU. Como resultado, os sistemas modernos aproveitam várias camadas de memória rápida, chamadas cachês, para ajudar a compensar essa disparidade. Cada nível de cache (L1, L2 e assim por diante) é relativamente mais lento e

maior do que seu predecessor. Na maioria dos sistemas, esses caches são integrados ao processador e a cada um de seus núcleos. Se os dados não forem encontrados em um determinado cache, eles devem ser buscados no cache de próximo nível ou na memória principal.

A CPU depende de sua unidade de gerenciamento de memória (MMU) para ajudar a encontrar onde os dados estão armazenados. A MMU é a unidade de hardware que traduz o endereço que o processador solicita para seu endereço correspondente na memória principal. Conforme descreveremos mais adiante neste capítulo, as estruturas de dados para gerenciar a tradução de endereços também são armazenadas na memória principal.

Como uma determinada tradução pode exigir várias operações de leitura de memória, o processador usa um cache especial, conhecido como buffer lookaside de tradução (TLB), para a tabela de tradução MMU. Antes de cada acesso à memória, o TLB é consultado antes de solicitar à MMU para realizar uma operação de tradução de endereço cara.

2.1.1 North and Southbridge

A CPU depende do controlador de memória para gerenciar a comunicação com a memória principal. O controlador de memória é responsável por mediar solicitações potencialmente simultâneas de memória do sistema dos processadores e dispositivos de **Entrada e Saída (I/O)**. O controlador de memória pode ser implementado em um chip separado ou integrado no próprio processador.

Em PCs mais antigos, a CPU conectava-se à ponte norte (hub controlador de memória) usando o barramento frontal e a ponte norte conectada à memória principal por meio do barramento de memória. Dispositivos como por exemplo, placas de rede e controladores de disco foram conectados por meio de outro chip, chamado ponte sul ou hub controlador de I/O, que tinha uma única conexão compartilhada com a ponte norte para acesso à memória e à CPU.

Para melhorar o desempenho e reduzir os custos dos sistemas mais novos, a maioria dos recursos associados ao hub do controlador de memória agora estão integrados ao processador. A funcionalidade restante do chipset, previamente implementada na ponte sul, está concentrada em um chip conhecido como hub controlador da plataforma.

2.1.2 Acesso direto à memória

Para melhorar o desempenho geral, a maioria dos sistemas modernos fornece aos dispositivos de I/O a capacidade de transferir diretamente os dados armazenados na memória do sistema sem intervenção do processador. Esse recurso é chamado de acesso direto à memória (DMA). Antes de o DMA ser introduzido, a CPU era totalmente consumida durante as transferências de I/O e frequentemente agia como intermediária. Em arquiteturas modernas, a CPU pode iniciar uma transferência de dados e permitir que um controlador DMA gerencie a transferência de dados, ou um dispositivo de I/O pode iniciar uma transferência independente da CPU.

Além de seu impacto óbvio no desempenho do sistema, o DMA também tem ramificações importantes para a análise forense de memória. Ele fornece um mecanismo para acessar diretamente o conteúdo da memória física de um dispositivo periférico sem envolver o software não confiável em execução na máquina. Por exemplo, o barramento PCI oferece suporte a dispositivos que atuam como mestres de barramento, o que significa que eles podem solicitar o controle do barramento para iniciar transações. Como resultado, um dispositivo PCI com funcionalidade **Bus Master** e suporte DMA pode acessar a memória do sistema sem envolver a CPU.

Outro exemplo é a interface IEEE 1394, comumente conhecida como Firewire. O chip controlador de host IEEE 1394 fornece um barramento de expansão serial ponto a ponto destinado a conectar dispositivos periféricos de alta velocidade a um PC. Embora a interface IEEE 1394 seja normalmente encontrada nativamente apenas em sistemas de ponta, você pode adicionar a interface a ambos desktops e laptops usando placas de expansão.

2.1.3 Memória de Acesso Aleatório (RAM)

A memória principal de um PC é implementada com memória de acesso aleatório (RAM), que armazena o código e os dados que o processador acessa e armazena ativamente. Em contraste com o armazenamento de acesso sequencial normalmente associado a discos, o acesso aleatório se refere à característica de ter um tempo de acesso constante, independentemente de onde os dados estão armazenados na mídia. A memória principal na maioria dos PCs é RAM dinâmica (DRAM). É dinâmico porque aproveita a diferença entre um estado carregado e descarregado de um capacitor para armazenar um pouco de dados. Para que o capacitor mantenha esse estado, ele deve ser atualizado periodicamente, o que é uma tarefa que o controlador de memória normalmente executa.

A RAM é considerada memória volátil porque requer energia para que os dados permaneçam acessíveis. Assim, exceto no caso de ataques de inicialização a frio⁹, depois que um PC é desligado, a memória volátil é perdida. Esta é a principal razão pela qual a tática de resposta a incidentes: **puxar a tomada** não é recomendada se você planeja preservar as evidências sobre o estado atual do sistema.

2.1.4 Arquiteturas de CPU

Como mencionado anteriormente, a CPU é um dos componentes mais importantes de um sistema de computador. Para extrair com eficácia a estrutura da memória física e compreender como o código malicioso pode comprometer a segurança do sistema, você deve ter um conhecimento sólido do modelo de programação que a CPU fornece para acessar a memória.

Para que a CPU execute instruções e acesse os dados armazenados na memória principal, ela deve especificar um endereço único para esses dados. Os processadores discutidos neste livro aproveitam o endereçamento de bytes e a memória é acessada como uma sequência de bytes. O espaço de endereço se refere a um intervalo de endereços válidos usados para identificar os dados armazenados em uma alocação finita de memória. Em

⁹ Obtenha mais informações em <https://citp.princeton.edu/research/memory>

particular, este livro se concentra em sistemas que definem um byte como uma quantidade de 8 bits. Este esquema de endereçamento geralmente começa com o byte 0 e termina no deslocamento do byte final de memória na alocação.

O único espaço de endereço contínuo exposto a um programa em execução é conhecido como espaço de endereço linear. Com base nos modelos de memória discutidos no livro e seu uso de paginação, usamos os termos endereços lineares e endereços virtuais alternadamente.

A arquitetura IA-32 geralmente se refere à família de arquiteturas x86 que oferecem suporte à computação de 32 bits. Em particular, ele especifica o conjunto de instruções e o ambiente de programação para os processadores de 32 bits da Intel. O IA-32 é uma pequena máquina endian que usa endereçamento de bytes. O software executado em um processador IA-32 pode ter um espaço de endereço linear e um espaço de endereço físico de até 4 GB. Como você verá posteriormente, você pode expandir o tamanho da memória física para 64 GB usando o recurso IA-32 Physical Address Extension (PAE). Este é o estado principal do processador e também o modo em que a maioria dos sistemas operacionais modernos são executados.

A arquitetura IA-32 define uma pequena quantidade de memória extremamente rápida, chamada de registradores, que a CPU usa para armazenamento temporário durante o processamento. Cada núcleo do processador contém oito registros de uso geral de 32 bits para realizar operações lógicas e aritméticas, bem como vários outros registros que controlam o comportamento do processador. Esta seção destaca alguns dos registros de controle relevantes para a análise de memória.

O registro EIP, também conhecido como contador de programa, contém o endereço linear da próxima instrução a ser executada. A arquitetura IA-32 também possui cinco registros de controle que especificam a configuração do processador e as características da tarefa de execução. CR0 contém sinalizadores que controlam o modo de operação do processador, incluindo um sinalizador que ativa a página CR1 é reservado e não deve ser acessado. CR2 contém o endereço linear que causou uma falha de página. CR3 contém o endereço físico da estrutura inicial usada para tradução de endereço. Ele é atualizado durante as mudanças de contexto quando uma nova tarefa é agendada. CR4 é usado para habilitar extensões arquitetônicas, incluindo PAE.

Os processadores IA-32 implementam dois mecanismos de gerenciamento de memória: segmentação e paginação. A segmentação divide o espaço de endereço linear de 32 bits em vários segmentos de comprimento variável. Todas as referências de memória IA-32 são endereçadas usando um seletor de segmento de 16 bits, que identifica um descriptor de segmento específico e um deslocamento de 32 bits no segmento especificado. Um descriptor de segmento é uma estrutura de dados residente na memória que define a localização, tamanho, tipo e permissões para um determinado segmento. Cada núcleo do processador contém dois registros especiais, GDTR e LDTR, que apontam para tabelas de descriptores de segmento, chamadas de Tabela de Descritores Globais (GDT) e Tabela de Descritores Locais, respectivamente. Os registros de segmentação CS (para código), SS (para pilha) e DS, ES, FS e GS (cada um para dados) devem sempre conter seletores de segmento válidos.

Embora a segmentação seja obrigatória, os sistemas operacionais discutidos neste livro ocultam o endereçamento segmentado, definindo um conjunto de segmentos sobrepostos com endereço de base zero, criando assim a aparência de um único espaço de endereço linear "plano" contínuo. No entanto, as proteções de segmentação ainda são aplicadas para cada segmento e descritores de segmento separados devem ser usados para referências de código e dados.

A paginação fornece a capacidade de virtualizar o espaço de endereço linear. Ele cria um ambiente de execução no qual um grande espaço de endereço linear é simulado com uma quantidade modesta de memória física e armazenamento em disco. Cada espaço de endereço linear de 32 bits é dividido em seções de comprimento fixo, chamadas de páginas, que podem ser mapeadas na memória física em uma ordem arbitrária. Quando um programa tenta acessar um endereço linear, esse mapeamento usa diretórios de páginas residentes na memória e tabelas de páginas para converter o endereço linear em um endereço físico.

A arquitetura IA-32 também suporta páginas de 4 MB, cuja tradução requer apenas um diretório de página. Ao usar diferentes estruturas de paginação para diferentes processos, um sistema operacional pode fornecer a cada processo a aparência de um ambiente de programação única por meio de um espaço de endereço linear virtualizado.

2.2 Sistema Operacional

Segundo Tanembaum um Sistema Operacional é:

- Um gerenciador de recursos;
- Uma abstração da máquina (hardware);

Um gerenciador de recursos pois este organiza e segmenta os recurso para usuários e processos, evitando assim falhas de segurança e falhas internas como Deadlocks. No Kernel Linux um crash por má gestão de recursos é algo raro, trabalho com mutex e semáforos serão discutidos nos próximos capítulos.

É uma abstração do Hardware e não importa a marca ou modelo de seu disco rígido, para o seu código **Write** é **Write**, e o mesmo código pode ser utilizado para um SSD ou um Pendrive. Se não fosse isto, a cada nova evolução da indústria de hardware impactaria todo o código dos programadores. No livro do Tanenbaum de Sistemas Operacionais Modernos você encontra tais detalhes no capítulo de I/O.

2.2.1 Gerenciamento de processo

O Kernel Linux conta com um poderoso esquema de escalonamento de processos, chamados de *schedule*. No capítulo de Processos será descrito este algoritmo em formato de texto, mas saiba neste momento que há dois escopos:

- Processos de Sistema;
- Processos de Usuários;

E é baseado neste escopo que se define parte da prioridade do processo e esta ação impacta no processamento das requisições ou aplicações, toda esta explicação está em alto nível e abstrato pois será detalhado neste livro.

2.2.2 Gerenciamento de memória

Todos os elementos de um computador moderno evoluíram nos últimos anos, a média está em torno de 16 vezes a cada década¹⁰. A memória evoluiu em tamanho e em velocidade, isso garantiu aos processos maior agilidade afinal um processo depende de dois recursos em suma:

- Da velocidade de acesso a memória;
- Da velocidade dos dispositivos de I/O;

O tempo de vida de um processo impacta no uso da memória, então o comportamento do processo impacta e naturalmente a forma que foi escrito impacta diretamente no desempenho final.

A memória física em um sistema de computador é um recurso limitado e mesmo para sistemas que suportam hotplug de memória há um limite rígido na quantidade de memória que pode ser instalada. A memória física não é necessariamente contígua; pode ser acessível como um conjunto de intervalos de endereços distintos. Além disso, diferentes arquiteturas de CPU e até mesmo diferentes implementações da mesma arquitetura têm diferentes visões de como esses intervalos de endereços são definidos.

Tudo isso torna bastante complexo lidar diretamente com a memória física e para evitar essa complexidade foi desenvolvido um conceito de memória virtual.

A memória virtual abstrai os detalhes da memória física do software aplicativo, permite manter apenas as informações necessárias na memória física (paginação por demanda) e fornece um mecanismo para proteção e compartilhamento controlado de dados entre processos.

Com a memória virtual, todo e qualquer acesso à memória usa um endereço virtual. Quando a CPU decodifica uma instrução que lê (ou grava) da (ou para) memória do sistema, ela traduz o endereço virtual codificado nesta instrução em um endereço físico que o controlador de memória possa entender.

A gestão da memória é realizado por um mecanismos chamada Memory Management Unit (MMU)¹¹, este recurso do Kernel Linux é um sistema complexo que evoluiu ao longo dos anos e incluiu cada vez mais funcionalidades para suportar uma variedade de sistemas, desde microcontroladores sem MMU até supercomputadores. O gerenciamento de memória para sistemas sem MMU é chamado NOMMU definitivamente merece um tópico especial no capítulo de Processos. No entanto, embora alguns dos conceitos sejam os mesmos, aqui assumimos que uma MMU está disponível e uma CPU pode traduzir um endereço virtual em um endereço físico conforme descrito por Tanenbaum.

¹⁰ As redes de computadores também evoluíram nesta taxa;

¹¹ Temos um capítulo só para discutir MMU e como o Linux trata endereçamento de memórias;

2.2.4 Sistema de I/O e Sistema de Arquivos

Para um Linux, os dispositivos de I/O se dividem em dois grupos:

- Dispositivos que realizam leitura e escrita com unidades de 512 bytes;
- Dispositivos que realizam leitura e escrita com unidade de 1 byte;

O Sistema é uma máquina estendida, ou seja, uma abstração do hardware real. Com o Sistema Operacional podemos interagir de forma fácil com linguagens de alto nível como C++, operando assim esses dispositivos. Neste livro vou demonstrar o uso de linguagem C, C++ e Python para operar o sistema. Temos que entender que nem todo dispositivo de I/O possui a mesma velocidade, por exemplo os HD são infinitamente mais lentos que as memórias NVME, e isso impacta na execução do processo. Quando um programa é bem programado e o hardware é bom, os processos tendem a ficar menos tempo na memória principal e ter naturalmente menos elementos para decisão do schedule.

Um dispositivo de I/O tem como objetivo a escrita e leitura de seus registradores, tais registradores são endereços de memórias e há uma Hierarquia de memória conforme definida em William Stallings. E é comum um dispositivo de I/O ter mais de um registrador e estes podem ser acessados em endereços consecutivos, seja na área da memória referente ao kernel ou mapeado em registradores independentes no próprio hardware. Cada um destes dispositivos de I/O conectados ao barramento possui um conjunto de endereços, chamamos estes endereços de portas de I/O. As portas de I/O podem ser mapeadas para endereços de memória física, permitindo que o processador se comunique com o dispositivo por meio de instruções que trabalham diretamente com a memória.

As portas de I/O de cada dispositivo são estruturadas em um conjunto de registradores especializados para fornecer uma interface de programação uniforme, lembre-se que uma das definições de Sistema Operacional é que é uma abstração do hardware, e estamos usando a abstração neste momento. Assim, a maioria dos dispositivos terá os seguintes tipos de registradores:

Control registers: que recebem comandos do dispositivo;

Status registers: que contêm informações sobre o status interno do dispositivo;

Input registers: quais os dados são obtidos do dispositivo;

Output registers: nos quais os dados são gravados para transmiti-los ao dispositivo.

Um CPU não pode operar diretamente o I/O devido a brutal diferença de velocidade, então os dispositivos operam em seu tempo e naturalmente transmitem em tempos diferentes os resultados de suas ações no mundo real. Tal diferença de velocidade se dá ao seu dispositivo físico, e por isso deve-se sempre procurar os dispositivos que exigem menos tempo para operar no mundo real. Como os tempos de resposta às atividades solicitadas se tornam eventos de tempo indefinidos, temos um problema que é o tempo de resposta, o que nos leva ao problema de tratativa da resposta. O Sistema Operacional pode trabalhar com interrupções, para isso foi introduzido o **Interrupt ReQuest (IRQ)**, que é uma notificação de hardware pela qual o processador é informado da ocorrência de um determinado evento externo de I/O, tal como o término da ação.

Encontramos nos drivers dos dispositivos artefatos manipuladores de interrupções, lembre-se que afinal todos os dispositivos vão interferir no IRQs, isso limita muito e para isso os drivers devem solicitar a interrupção antes de usar e liberar quando não mais for necessário. Tudo isso é para evitar um Deadlock, e mesmo assim, acontecerá um dia.

No Linux existe uma camada de tratamento de IRQ, esta camada genérica oferece uma abstração completa do tratamento de interrupções para os drivers de dispositivos, é capaz de lidar com qualquer tipo de hardware, drivers, etc. Isso garante que o projetista que desenvolve drivers não precisa saber sobre os detalhes das interrupções de hardware e portanto pode utilizar-se seu artefato em diferentes plataformas.

No Linux existe um super-handler capaz de manipular qualquer lógica de interrupção, chama-se `__do_IRQ()`, sempre que uma interrupção for lançada irá chamar este handler de forma assíncrona. No capítulo de Gerenciamento de Processos, práticas de região crítica serão aplicadas.

2.3 Um Kernel e muitas Distribuições

O que é engraçado quando se observa um novo usuário Linux é que ele inicia questionando: Qual é o melhor Linux? E o mais engraçado é ver usuários Linux velhos defendendo a sua distribuição. A verdade é que a resposta é: **aquele Linux que você melhor se adapta**. Este material é focado em Debian por um motivo: Em sala de aula eu como professor tenho que ter tudo uniforme, até a versão do Debian tem que ser igual, pois é uma sala de aula.

Mas por que tantas distribuições? É mais uma questão de nunca estarmos contentes com o que temos, e naturalmente dá aquela vontade de melhorar o que temos, mudar o que temos, e depois, divulgar o que fazemos. Quando fica bom as alterações, é natural que arrebanhar muitas pessoas, e é assim que nascem as distribuições Linux.

Outra coisa que um usuário Microsoft Windows tem que saber, Linux é um Sistema Operacional feito por programadores para programadores, altamente customizável e com uma comunidade engajada em evoluir. Você verá aqui neste livro recursos que jamais veria no mundo Microsoft e tamanha maturidade, mas a única coisa que não consigo responder é: Porque o Linux é tão feio? Para isso eu não tenho resposta.

Distribuições Linux

No mundo Windows somos obrigados a utilizar apenas um Sistema Operacional, veja, o Microsoft Windows XP foi o principal sistema da Microsoft por quase 11 anos. Quando uma pessoa inicia sua vida no mundo Linux se depara com várias distribuições e ficam confusos, mas a pergunta que mais vejo é: Qual é o melhor Linux.

Se você já chegou aqui já sabe, não existe o melhor Linux, o que temos é o Linux mais adequado para mim ou para o uso que vou fazer, inclusive o autor deste livro utiliza várias distribuições Linux no mesmo dia no mesmo ambiente.

Uma distribuição Linux, frequentemente abreviada como distro, é um sistema operacional que inclui o Kernel Linux para sua funcionalidade. Embora o nome não implique distribuição

de produto em si, uma distro geralmente é obtida por meio de um site criado especificamente para esse propósito. As distros foram projetadas para uma ampla variedade de sistemas, desde computadores pessoais a servidores e de dispositivos embarcados a supercomputadores.

Uma distro normalmente inclui muitos componentes além do Kernel Linux. Normalmente, ela inclui um gerenciador de pacotes, um sistema de inicialização, ferramentas e bibliotecas GNU, documentações, utilitários de configuração de rede e o programa acesso TTY.

Para fornecer uma experiência de desktop, um servidor gráfico, o mais comum sendo o X.org Server ou, mais recentemente, um compositor Wayland. Além do servidor gráfico também é instalado uma camada gráfica para Desktop, o mais comum é o GNOME, KDE e XFCE.

Uma distro pode ser descrita como uma variedade particular de softwares aplicativos e utilitários, empacotados com o kernel Linux de tal forma que suas capacidades atendam às necessidades dos usuários. O software é geralmente adaptado à distribuição e então combinado em pacotes de software pelos mantenedores da distribuição. Os pacotes de software estão disponíveis online em repositórios, que são locais de armazenamento geralmente distribuídos ao redor do mundo.

FALAR DAS DISTROS

FALTADO FALAR DO DEBIAN

LAYOUT

TTY

3 Virtualização

A virtualização usa software para criar uma camada de abstração sobre o hardware do computador que permite que os elementos de hardware de um único computador - processadores, memória, armazenamento e muito mais - sejam divididos em vários computadores virtuais, comumente chamados de Máquina Virtual (VM).

Cada VM executa seu próprio sistema operacional (SO) e se comporta como um computador independente, embora esteja sendo executado em apenas uma parte do hardware do computador subjacente real. Todo o conteúdo externo ao texto pode ser obtido no link abaixo.



3.1 Introdução a Virtualização

A ideia de abstração é comum na área da Computação, veja que segundo Tanenbaum o Sistema Operacional (SO) é uma abstração do hardware e naturalmente uma simplificação da complexidade do Hardware. Em outro capítulo, Tanenbaum afirma que estas abstrações permitem que se faça um isolamento de processos em "Caixas de Areia" utilizando abstrações em abstrações.

A Virtualização de Sistemas Operacionais permite:

- Que se crie ambientes controlados para testes e experimentações;
- Isolar aplicações com algum risco negativo para o ambiente;
- Se execute aplicações legadas para Sistemas Operacionais descontinuados;
- Redução de parque de computadores reais com uso de Virtualização;
- Upgrade de sistemas;
- Aprendizado;

Hoje, a virtualização é uma prática padrão na arquitetura de TI corporativa. É também a tecnologia que impulsiona a economia da computação em nuvem. A virtualização permite que os provedores de nuvem atendam aos usuários com seu hardware físico de computador existente; ele permite que os usuários da nuvem comprem apenas os recursos de computação de que precisam, quando precisam, e escalem esses recursos de maneira econômica conforme suas cargas de trabalho aumentam.

A virtualização traz vários benefícios para operadoras de data center e provedores de serviços, podemos citar:

Eficiência de recursos: antes da virtualização, cada servidor de aplicativo exigia sua própria CPU física dedicada - a equipe de TI comprava e configurava um servidor separado para cada aplicativo que desejava executar. Invariavelmente, cada servidor físico seria subutilizado, e por outro lado, a virtualização de servidor permite executar vários aplicativos em um único computador físico sem sacrificar a confiabilidade, isso permite a utilização máxima da capacidade de computação do hardware físico;

Gerenciamento mais fácil: substituir computadores físicos por VMs definidas por software torna mais fácil usar e gerenciar políticas escritas no software. Isso permite que você crie fluxos de trabalho automatizados de gerenciamento de serviços de TI. Por exemplo,

ferramentas automatizadas de implantação e configuração permitem que os administradores definam coleções de máquinas virtuais e aplicativos como serviços, em modelos de software. Isso significa que eles podem instalar esses serviços repetidamente e de forma consistente, sem incômodos e demorados, e configuração manual sujeita a erros.

Tempo de inatividade mínimo: travamentos do sistema operacional e de aplicativos podem causar tempo de inatividade e prejudicar a produtividade do usuário. Os administradores podem executar várias máquinas virtuais redundantes lado a lado e fazer failover entre elas quando surgem problemas.

Provisionamento mais rápido: comprar, instalar e configurar hardware para cada aplicativo consome muito tempo. Desde que o hardware já esteja instalado, o provisionamento de máquinas virtuais para executar todos os seus aplicativos é significativamente mais rápido. Você pode até automatizá-lo usando um software de gerenciamento e incorporá-lo aos fluxos de trabalho existentes.

As máquinas virtuais (VMs) são ambientes virtuais que simulam uma computação física na forma de software. Eles normalmente incluem vários arquivos contendo a configuração da VM, o armazenamento para o disco rígido virtual e alguns instantâneos da VM que preservam seu estado em um determinado momento.

3.3 Hypervisors

Um hipervisor é a camada de software que coordena as VMs, esta camada serve como uma interface entre a VM e o hardware físico subjacente, garantindo que cada uma tenha acesso aos recursos físicos de que precisa para executar, e também garante que as VMs não interfiram uma com as outras interferindo no espaço de memória ou nos ciclos de computação uma das outras.

Existem dois tipos de Hypervisors:

- Os **hypervisors tipo 1** ou “bare-metal” interagem com os recursos físicos subjacentes, substituindo o sistema operacional tradicional por completo. Eles geralmente aparecem em cenários de servidor virtual.
- Os **hypervisors tipo 2** são executados como um aplicativo em um sistema operacional existente. Mais comumente usados em dispositivos terminais para executar sistemas operacionais alternativos, eles carregam uma sobrecarga de desempenho porque devem usar o sistema operacional host para acessar e coordenar os recursos de hardware subjacentes.

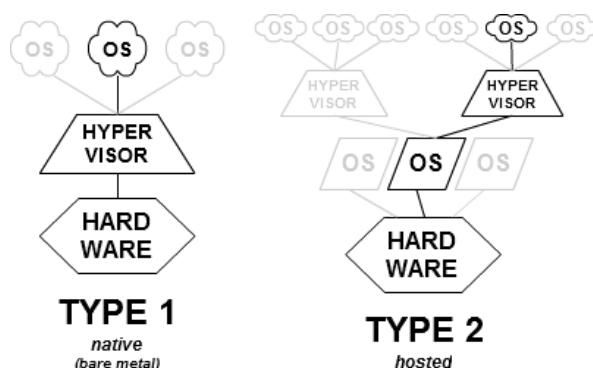


Figura 3.

3.4 Tipos de virtualização

Nesta seção, abordaremos os seguintes tipos de virtualização:

- Virtualização de desktop
- Virtualização de rede
- Virtualização de armazenamento
- Virtualização de dados
- Virtualização de aplicativos
- Virtualização de data center
- Virtualização de CPU
- Virtualização de GPU
- Virtualização Linux
- Virtualização de nuvem

3.4.1 Virtualização de desktop

A virtualização de desktop permite que você execute vários sistemas operacionais de desktop, cada um em sua própria VM no mesmo computador.

Existem dois tipos de virtualização de desktop, a **infraestrutura de desktop virtual (VDI)** executa vários desktops em VMs em um servidor central e os transmite para usuários que fazem login em dispositivos thin client. Desta forma, o VDI permite que uma organização forneça a seus usuários acesso a uma variedade de sistemas operacionais a partir de qualquer dispositivo, sem instalar sistemas operacionais em nenhum dispositivo.

A **virtualização de desktop local** executa um hipervisor em um computador local, permitindo ao usuário executar um ou mais sistemas operacionais adicionais naquele computador e alternar de um sistema operacional para outro conforme necessário, sem alterar nada no sistema operacional principal.

3.4.2 Virtualização de rede

A virtualização de rede usa software para criar uma “visão” da rede que um administrador pode usar para gerenciar a rede a partir de um único console. Ele abstrai elementos e funções de hardware (por exemplo, conexões, switches, roteadores, etc.) e os abstrai em software executado em um hipervisor, o administrador da rede pode modificar e controlar esses elementos sem tocar nos componentes físicos subjacentes, o que simplifica drasticamente o gerenciamento da rede.

Os tipos de virtualização de rede incluem rede definida por software (SDN), que virtualiza hardware que controla o roteamento de tráfego de rede (chamado de “plano de controle”) e virtualização de função de rede (NFV), que virtualiza um ou mais dispositivos de hardware que fornecem uma rede específica (por exemplo, um firewall, balanceador de carga ou analisador de tráfego), tornando esses dispositivos mais fáceis de configurar, provisionar e gerenciar.

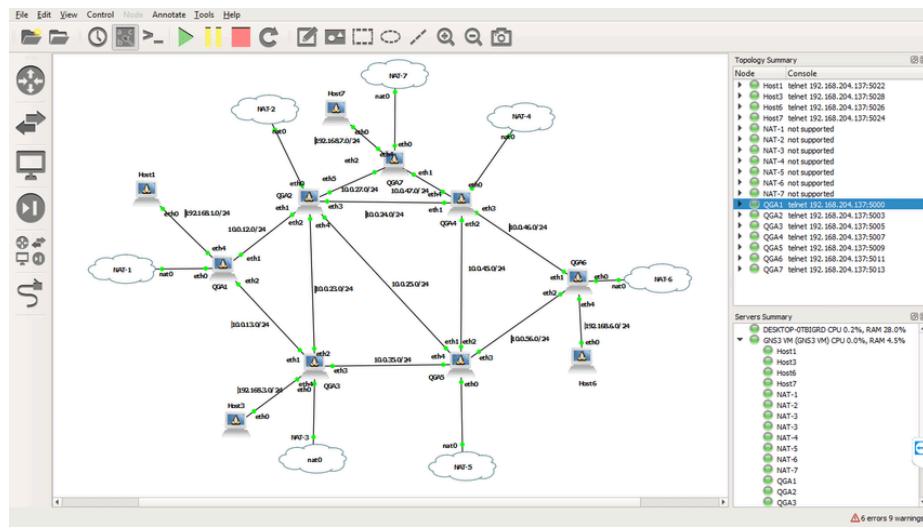


Figura 3.

Na figura acima o GNS3¹² está sendo utilizado para virtualizar um ambiente de redes para testes e homologação. Neste ambiente é possível importar VMs Linux, Routers, Switchs, etc..

3.4.3 Virtualização de armazenamento

A virtualização de armazenamento permite que todos os dispositivos de armazenamento na rede, sejam eles instalados em servidores individuais ou unidades de armazenamento autônomas, sejam acessados e gerenciados como um único dispositivo de armazenamento.

Especificamente, a virtualização de armazenamento agrupa todos os blocos de armazenamento em um único pool compartilhado, a partir do qual eles podem ser atribuídos a qualquer VM na rede, conforme necessário. A virtualização de armazenamento facilita o provisionamento de armazenamento para VMs e aproveita ao máximo todo o armazenamento disponível na rede.

3.4.4 Virtualização de dados

As empresas modernas armazenam dados de vários aplicativos, usando vários formatos de arquivo, em vários locais, desde a nuvem até hardware e sistemas de software no local. A virtualização de dados permite que qualquer aplicativo acesse todos esses dados - independentemente da fonte, formato ou localização.

As ferramentas de virtualização de dados criam uma camada de software entre os aplicativos que acessam os dados e os sistemas que os armazenam. A camada traduz a solicitação de dados de um aplicativo ou consulta conforme necessário e retorna resultados que podem abranger vários sistemas. A virtualização de dados pode ajudar a quebrar os repositórios de dados quando outros tipos de integração não são viáveis, desejáveis ou acessíveis.

¹² GNS3 é uma aplicação gratuita que pode ser obtida no site <https://www.gns3.com/>

3.4.5 Virtualização de aplicativos

A virtualização de aplicativos executa o software do aplicativo sem instalá-lo diretamente no sistema operacional do usuário. Isso difere da virtualização de desktop completa (mencionada acima) porque apenas o aplicativo é executado em um ambiente virtual - o sistema operacional no dispositivo do usuário final é executado normalmente. Existem três tipos de virtualização de aplicativos:

Virtualização de aplicativo local: O aplicativo inteiro é executado no dispositivo endpoint, mas é executado em um ambiente de tempo de execução em vez de no hardware nativo.

Streaming do aplicativo: o aplicativo reside em um servidor que envia pequenos componentes do software para serem executados no dispositivo do usuário final quando necessário.

Virtualização de aplicativo baseada em servidor O aplicativo é executado inteiramente em um servidor que envia apenas sua interface de usuário ao dispositivo cliente.

Virtualização de data center abstrai a maior parte do hardware de um data center em software, permitindo que um administrador divida um único data center físico em vários data centers virtuais para clientes diferentes.

Cada cliente pode acessar sua própria infraestrutura como serviço (IaaS), que seria executada no mesmo hardware físico subjacente. Os data centers virtuais oferecem uma entrada fácil na computação baseada em nuvem, permitindo que uma empresa configure rapidamente um ambiente de data center completo sem comprar hardware de infraestrutura.

3.4.6 Virtualização de CPU

A virtualização da CPU (unidade central de processamento) é a tecnologia fundamental que torna possíveis os hipervisores, as máquinas virtuais e os sistemas operacionais. Ele permite que uma única CPU seja dividida em várias CPUs virtuais para uso por várias VMs.

No início, a virtualização da CPU era totalmente definida por software, mas muitos dos processadores de hoje incluem conjuntos de instruções estendidas que suportam a virtualização da CPU, o que melhora o desempenho da VM.

3.4.7 Virtualização de GPU

Uma GPU (unidade de processamento gráfico) é um processador multi-core especial que melhora o desempenho geral da computação, assumindo o processamento gráfico ou matemático pesado. A virtualização de GPU permite que várias VMs usem todo ou parte do poder de processamento de uma única GPU para vídeo mais rápido, inteligência artificial (IA) e outros aplicativos gráficos ou matemáticos.

As GPUs de passagem disponibilizam toda a GPU para um único sistema operacional convidado, elas dividem os núcleos físicos da GPU entre várias GPUs virtuais (vGPUs) para uso por VMs baseadas em servidor.

3.4.8 Virtualização Linux

O Linux inclui seu próprio hipervisor, chamado de máquina virtual baseada em kernel (KVM), que suporta extensões de processador de virtualização da Intel e AMD para que você possa criar VMs baseadas em x86 a partir de um sistema operacional Linux host.

Como um sistema operacional de código aberto, o Linux é altamente personalizável. Você pode criar VMs executando versões do Linux personalizadas para cargas de trabalho específicas ou versões com segurança reforçada para aplicativos mais confidenciais.

3.4.9 Virtualização de nuvem

Conforme observado acima, o modelo de computação em nuvem depende da virtualização. Ao virtualizar servidores, armazenamento e outros recursos físicos de data center, os provedores de computação em nuvem podem oferecer uma variedade de serviços aos clientes, incluindo o seguinte:

Infraestrutura como serviço (IaaS): recursos virtualizados de servidor, armazenamento e rede que você pode configurar com base em seus requisitos.

Plataforma como serviço (PaaS): ferramentas de desenvolvimento virtualizadas, bancos de dados e outros serviços baseados em nuvem que você pode usar para construir seus próprios aplicativos e soluções baseados em nuvem.

Software como serviço (SaaS): aplicativos de software que você usa na nuvem. SaaS é o serviço baseado em nuvem mais abstruído do hardware.

3.6 Práticas da aula

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

3.6.1 Prática 98a917ce 01: Criando o ambiente da aula

Nesta prática o aluno deve utilizar instalar os softwares necessários e instalar o Debian:

1. Faça download:
 - a. VirtualBox: <https://www.virtualbox.org/wiki/Downloads>
 - b. Cd de instalação do Debian:
<https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-12.1.0-amd64-netinst.iso>
2. Instale o Debian e configure da seguinte forma: [Instalando o Debian 12 GNU/Linux](#)
 - a. English;
 - b. usuário: userlinux
 - c. senha: 123456
3. Instale o sudo, utilize este link: [Instalando e configurando o SUDO](#)
4. Instale o programa aied, utilize este link: [Validação de práticas por meio de Robô e suporte](#)

Execute o robô de validação aied conforme comando abaixo.

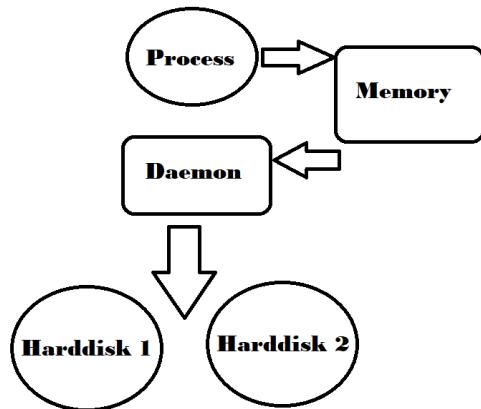
```
sudo aied validar 98a917ce checkpoint01
```

4 Sistema de arquivos no Debian GNU/Linux

Um sistema de arquivos GNU/Linux é uma coleção estruturada de arquivos em uma unidade de armazenamento secundário¹³ (ou várias), já uma partição é um segmento da memória e contém alguns dados específicos e em nossa máquina, pode haver várias partições da memória secundária. O sistema de arquivos em um computador de uso geral precisa armazenar dados sistematicamente e a forma que é armazenado é importante, pois para que possamos acessar facilmente os arquivos com o menor tempo possível. As principais memórias secundárias são:

- Hard Disk;
- Pendrive;
- Floppy Disk;
- CD ou DVD;
- SSD;

O computador salva os dados no armazenamento principal¹⁴, e este pode perder os dados se for desligado, o armazenamento de dados é preferido em sistemas secundários por este motivo. Porém a arquitetura dos computadores força que as instruções e os dados estejam na mesma memória, ou seja, na memória principal. No passado os processos residentes na memória principal realizavam operações de I/O diretamente na memória secundária, porém isso levava a corrupção de arquivos. Hoje usamos um modelo diferente, os processos escrevem o arquivo diretamente na memória principal, e então um Daemon persiste os dados da memória principal na memória secundária, conforme esquema abaixo.



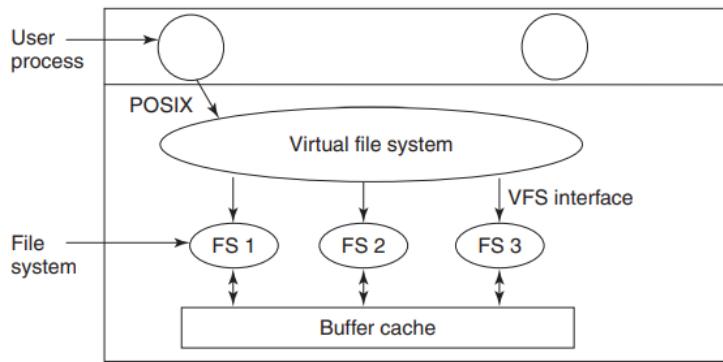
4.1 Virtual File System

O sistema de arquivos GNU/Linux é geralmente uma camada embutida de um sistema operacional Linux usada para lidar com o gerenciamento de dados do armazenamento. Isso ajuda a organizar o arquivo no armazenamento em disco. Ele gerencia o nome do arquivo, o tamanho do arquivo, a data de criação e muito mais informações sobre um arquivo. Um

¹³ Recomendo a leitura do capítulo de Hierarquia de Memória no livro **W. Stallings, Arquitetura e Organização de Computadores, Pearson, 2010**;

¹⁴ No início os processos foram projetados para serem executados por inteiro na posição 0 de uma memória, e naturalmente isto exige que todos os dados estejam nesta, o aluno deve ler o capítulo 1 do livro: Sistemas Operacionais Modernos do autor Tanenbaum.

processo que executa operações de I/O em um sistema secundário utiliza o Virtual File System para executar tais operações.



Um sistema de arquivos é projetado de forma que possa gerenciar e fornecer espaço para dados de armazenamento não voláteis. Todos os sistemas de arquivos exigem um namespace que é uma metodologia de nomenclatura e organização, já um namespace define o processo de nomenclatura, o comprimento do nome do arquivo ou um subconjunto de caracteres que podem ser usados para o nome do arquivo. Ele também define a estrutura lógica dos arquivos em um segmento de memória, como o uso de diretórios para organizar os arquivos específicos. Depois que um namespace é descrito, uma descrição de metadados deve ser definida para esse arquivo específico.

O sistema de arquivos do GNU/Linux tem uma estrutura de arquivos hierárquica, pois contém um diretório raiz e seus subdiretórios e todos os outros diretórios podem ser acessados a partir do diretório raiz a partir de comandos de navegação, um detalhe importante é que uma partição geralmente possui apenas um sistema de arquivos. No GNU/Linux a montagem do sistema de arquivo está codificada na biblioteca **fs.h** e com esta biblioteca pode-se montar ou desmontar qualquer hardware devidamente acoplado e operacional.

```

usuario@debian:~$ 
usuario@debian:~$ find /usr -name fs.h
/usr/include/linux/fs.h
usuario@debian:~$ 

```

O que fundamenta a inexistência de boas aulas de Sistemas Operacionais com Microsoft Windows é que a plataforma GNU/Linux é aberta e o aluno pode inspecionar todo o código, alterar e recompilar. Na imagem abaixo pode se observar o código do arquivo **fs.h**, saiba que também é possível verificar o versionamento deste arquivo no endereço oficial de **Linux Torvalds**¹⁵.

¹⁵ Acessível pela url: <https://github.com/torvalds/linux/blob/master/include/linux/fs.h>

```
/* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
#ifndef _LINUX_FS_H
#define _LINUX_FS_H

/*
 * This file has definitions for some important file table structures
 * and constants and structures used by various generic file system
 * ioctl's. Please do not make any changes in this file before
 * sending patches for review to linux-fsdevel@vger.kernel.org and
 * linux-api@vger.kernel.org.
 */

#include <linux/limits.h>
#include <linux/ioctl.h>
#include <linux/types.h>
#include <linux/fscrypt.h>

/* Use of MS_* flags within the kernel is restricted to core mount(2) code. */
#include <linux/mount.h>

/*
 * It's silly to have NR_OPEN bigger than NR_FILE, but you can change
 * the file limit at runtime and only root can increase the per-process
 * nr_file rlimit, so it's safe to set up a ridiculously high absolute
 * upper limit on files-per-process.
 *
 * Some programs (notably those using select()) may have to be
 * recompiled to take full advantage of the new limits..
 */

/* Fixed constants first: */
#undef NR_OPEN
#define INR_OPEN_CUR 1024           /* Initial setting for nfile rlimits */
#define INR_OPEN_MAX 4096           /* Hard limit for nfile rlimits */

#define BLOCK_SIZE_BITS 10
--More-- (9%)
```

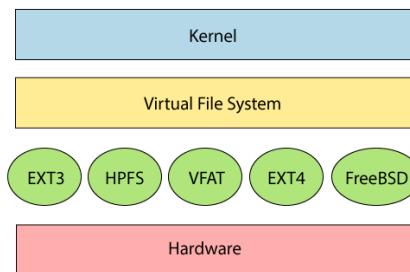
Caso queira importar o arquivo acima em algum código de algum programa que queira desenvolver que envolva montagem de sistemas de arquivos, é muito simples, basta informar no **include** o arquivo **linux/fs.h** e informar quais funções que pretende utilizar conforme listagem abaixo. Daí para frente é sua imaginação e lembre-se, o código fonte é uma obra de arte, da mesma forma que uma escultura ou um quadro.

1. #include <linux/fs.h>
- 2.
3. extern int register_filesystem(struct file_system_type *);
4. extern int unregister_filesystem(struct file_system_type *);

Quando uma solicitação é feita para montar um sistema de arquivos em um diretório em seu namespace, o Virtual File System chamará o método **mount()** apropriado para o sistema de arquivos específico.

A estrutura de dados precisa oferecer suporte a uma estrutura de diretório hierárquica, esta estrutura é usada para descrever o espaço em disco e localização de arquivos. Também contém outros detalhes sobre os arquivos, como tamanho do arquivo, data e hora de criação, atualização e última modificação. Além disso, ele armazena informações avançadas sobre a seção do disco, como partições e volumes. Os dados avançados e as estruturas que eles representam contêm as informações sobre o sistema de arquivos

armazenado na unidade. O sistema de arquivos Linux contém uma arquitetura de implementação de software do sistema de arquivos em duas partes.



Na figura acima o Virtual File System fornece um único conjunto de comandos para o kernel e os desenvolvedores acessam o sistema de arquivos, conforme já visto. Este sistema de arquivos virtual requer o driver de sistema específico para fornecer uma interface com o sistema de arquivos. O sistema de arquivos requer uma API para acessar as chamadas de função para interagir com os componentes do sistema de arquivos, como arquivos e diretórios. API facilita tarefas como criação, exclusão e cópia de arquivos. Ele facilita um algoritmo que define a organização dos arquivos em um sistema de arquivos.

```

1. struct super_operations {
2.     struct inode *(*alloc_inode)(struct super_block *sb);
3.     void (*destroy_inode)(struct inode *);
4.     void (*dirty_inode)(struct inode *, int flags);
5.     int (*write_inode)(struct inode *, int);
6.     void (*drop_inode)(struct inode *);
7.     void (*delete_inode)(struct inode *);
8.     void (*put_super)(struct super_block *);
9.     int (*sync_fs)(struct super_block *sb, int wait);
10.    int (*freeze_fs)(struct super_block *);
11.    int (*unfreeze_fs)(struct super_block *);
12.    int (*statfs)(struct dentry *, struct kstatfs *);
13.    int (*remount_fs)(struct super_block *, int *, char *);
14.    void (*clear_inode)(struct inode *);
15.    void (*umount_begin)(struct super_block *);
16.    int (*show_options)(struct seq_file *, struct dentry *);
17.    ssize_t (*quota_read)(struct super_block *, int, char *, size_t, loff_t);
18.    ssize_t (*quota_write)(struct super_block *, int, const char *, size_t,
   loff_t);
19.    int (*nr_cached_objects)(struct super_block *);
20.    void (*free_cached_objects)(struct super_block *, int);
21. };

```

Onde:

alloc_inode: este método é chamado por alloc_inode() para alocar memória para struct inode e inicializá-lo;

destroy_inode: este método é chamado por destroy_inode() para liberar recursos alocados para struct inode;

dirty_inode: este método é chamado pelo VFS para marcar um inode sujo e naturalmente poderá ser descartado.

write_inode: este método é chamado quando o VFS precisa gravar um inode no disco;

drop_inode: chamado quando o último acesso ao inode é descartado, elimina o inode;

delete_inode: chamado quando o VFS deseja excluir um inode;

put_super: chamado quando o VFS deseja liberar o superbloco (ou seja, desmontar);

sync_fs: chamado quando o VFS está gravando todos os dados sujos associados a um superbloco;

freeze_fs: chamado quando o VFS está bloqueando um sistema de arquivos e forçando-o a um estado **consistent**;

unfreeze_fs: chamado quando o VFS está desbloqueando um sistema de arquivos e tornando-o gravável novamente.

statfs: chamado quando o VFS precisa obter estatísticas do sistema de arquivos.

remount_fs: chamado quando o sistema de arquivos é remontado;

clear_inode: chamado então o VFS limpa o inode. Opcional

umount_begin: chamado quando o VFS está desmontando um sistema de arquivos.

show_options: chamado pelo VFS para mostrar opções de montagem para /proc/<pid>/ mounts;

quota_read: chamado pelo VFS para ler o arquivo de cota do sistema de arquivos.

quota_write: chamado pelo VFS para gravar no arquivo de quota do sistema de arquivos.

nr_cached_objects: chamado pela função de redução do cache sb para o sistema de arquivos para retornar o número de objetos em cache liberáveis que ele contém;

free_cache_objects: chamado pela função de redução do cache sb para que o sistema de arquivos varra o número de objetos indicados para tentar liberá-los.

4.2 Diretórios padrões de um GNU/Linux Debian

No GNU/Linux, o sistema de arquivos cria uma estrutura em árvore e todos os arquivos são organizados como uma árvore e seus ramos, se mais de um sistema de armazenamento secundário for adicionado ambos serão mapeados dentro da mesma árvore. O diretório superior denominado diretório raiz (/)¹⁶. Uma boa recomendação é a leitura do capítulo de Sistema de Arquivos do livro Sistemas Operacionais Modernos do autor Andrew Stuart Tanenbaum.

- | | |
|---|--|
|  | Sistema de Arquivos do Linux - Parte 1 |
|  | Linux File System - Part 1 |
|  | Sistema de archivos Linux - Parte 1 |

Observações importantes sobre o Sistema de Arquivos:

Especificando caminhos: o GNU/Linux não usa a barra invertida (\) para separar os componentes; ele usa barra (/) como alternativa. Por exemplo, como no Windows, os dados podem ser armazenados em **C:\Meus Documentos\Trabalho**, ao passo

¹⁶ No passado, durante a instalação o especialista poderia modificar o símbolo de raiz, mas com o passar do tempo o símbolo / se solidificou e se tornou padrão.

que, no GNU/Linux, eles seriam armazenados em **/home/userlinux/Documents/Trabalho**.

Partição, diretórios e unidades: o GNU/Linux não usa letras de unidade para organizar a unidade como o Microsoft Windows faz, em um GNU/Linux não podemos dizer se estamos endereçando uma partição, um dispositivo de rede ou um diretório "comum" e um Drive pois todos os dispositivos estão montados dentro de uma mesma árvore.

Sensibilidade a maiúsculas e minúsculas: o sistema de arquivos GNU/Linux diferencia maiúsculas de minúsculas, ele distingue entre nomes de arquivo em letras maiúsculas e minúsculas.

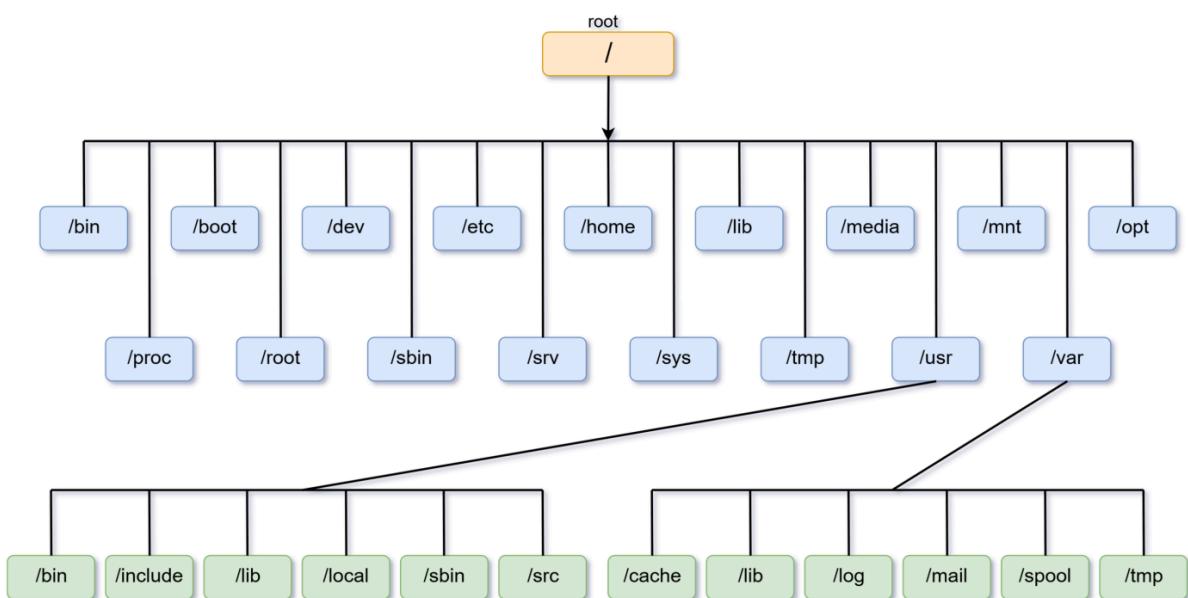
Extensões de arquivo: No GNU/Linux, um arquivo pode ter a extensão '.txt', mas não é necessário que um arquivo tenha uma extensão de arquivo, ao trabalhar com o Shell, ele cria alguns problemas para os iniciantes diferenciarem entre arquivos e diretórios.

Arquivos ocultos: o GNU/Linux distingue entre arquivos padrão e arquivos ocultos, principalmente os arquivos de configuração estão ocultos no sistema operacional GNU/Linux, normalmente, não precisamos acessar ou ler os arquivos ocultos e os arquivos ocultos no GNU/Linux são representados por um ponto (.) Antes do nome do arquivo (por exemplo, .ignore).

Por anos usuários ativos dos projetos que envolvem Linux discutiram a organização dos arquivos, no ponto de vista do autor deste texto a segurança e estabilidade de um servidor GNU/Linux depende de duas características do mundo Linux, que são:

- Simplicidade em tudo, reduzindo decisões complexas e possíveis vulnerabilidades;
- Sistema de arquivos extremamente organizado;

Em suma, a organização é fundamental e abaixo estou listando os principais diretórios que um expert em GNU/Linux deve conhecer e naturalmente, manter assim a organização.





Onde:

Diretório / (chamado de raiz): Raiz da hierarquia primária e é o primeiro diretório. Ele contém todos os outros diretórios, ou seja, os subdiretórios e apenas o usuário root tem permissão para escrever aqui;

Diretório /bin (binários do usuário): Ele contém os executáveis binários relacionados aos comandos Linux comuns usados por todos os usuários no modo de usuário único estão localizados neste diretório. Alguns arquivos presentes neste diretório são: ls, cp, grep, ping, cat, etc;

Diretório /boot (arquivos de inicialização boot): Ele contém arquivos do carregador de boot, kernel, initrd, grub e outros arquivos e diretórios estão localizados neste diretório;

Diretório /dev (arquivos de dispositivo): Ele contém os arquivos essenciais relacionados aos dispositivos conectados ao sistema, isso inclui dispositivos de terminal, USB, dispositivos de rede e quaisquer outros dispositivos de I/O que estejam conectados ao sistema;

Diretório /etc (arquivos de configuração): Este diretório tem um significado, ou seja, etc significa 'edit to config' este diretório contém os arquivos de configuração exigidos pelos programas instalados, os arquivos são configurações específicas do host e de todo o sistema, necessárias para o funcionamento adequado do sistema. Este diretório também contém scripts de shell para inicialização e desligamento do sistema que são usados para iniciar ou parar programas individuais. Alguns programas grandes e complexos estão neste diretório;

Diretório /home (diretórios pessoais): Este diretório contém os diretórios pessoais do usuário, contendo arquivos salvos e configurações pessoais. Cada usuário terá um diretório separado com seu nome de usuário nesse diretório, exceto o usuário root, porque toda vez que um novo usuário é criado, um diretório é criado no nome do usuário dentro do diretório inicial;

Diretório /lib (bibliotecas do sistema): Este diretório contém bibliotecas que são essenciais para os binários em /bin e /sbin, os nomes dos arquivos de biblioteca são ld* ou lib*.so.*, etc;

Diretório /media (dispositivos de mídia removíveis): Diretório temporário de montagem para mídia removível, como CD-ROM, por exemplo: /media/cdrom para CD-ROM; /media/floppy para unidades de disquete; /media/cdrecorder para gravador de CD;

Diretório /mnt (diretório de montagem): Diretório de montagem temporário onde o administrador do sistema pode montar sistemas de arquivos;

Diretório /opt (pacotes de software de aplicativo opcionais): Este diretório contém aplicativos complementares de fornecedores individuais, tal como Tomcat, Monero, etc;

Diretório /proc (informações do processo): Este é um sistema de arquivos virtual que fornece informações sobre o processo e o kernel. Esses arquivos neste diretório são gerados, preenchidos e excluídos automaticamente pelo sistema;

Diretório /root (diretório raiz): Este é o diretório inicial do usuário root;

Diretório /sbin (binários do sistema): Este diretório contém binários essenciais do sistema. Os comandos do GNU/Linux que estão localizados neste diretório são usados pelo administrador do sistema, para manutenção do sistema e propósito de configuração, por exemplo: fsck, reboot, fdisk, ip, init, etc;

Diretório /srv (dados de serviço): Este diretório contém dados específicos do site servidos pelo sistema, como dados e scripts para servidores da web, dados oferecidos por servidores FTP e repositórios para sistemas de controle de versão, ou seja, dados relacionados a serviços específicos de servidor, por exemplo: /srv/cvs contém dados relacionados ao CVS, etc;

Diretório /sys (sistema): Este diretório contém informações sobre dispositivos, drivers e alguns recursos do kernel.

Diretório /tmp (arquivos temporários): Este diretório contém arquivos temporários criados pelo sistema e os usuários que serão inicializados quando o sistema for reinicializado;

Diretório /usr (programas do usuário): Este diretório contém dados de usuário somente leitura, como binários, bibliotecas, documentação e código-fonte para programas de segundo nível, como utilitários de usuário e aplicativos;

Diretório /usr/bin: contém arquivos binários para programas do usuário. Se você não conseguir encontrar um binário de usuário em /bin, devemos procurar em /usr/bin;

Diretório /usr/include: contém arquivos de inclusão padrão;

Diretório /usr/lib: contém bibliotecas para os binários em /usr/bin e /usr/sbin;

Diretório /usr/local: hierarquia terciária para dados locais. contém programas de usuários que você instala a partir de código fonte;

Diretório /usr/sbin: contém arquivos binários para administradores de sistema. Se você não conseguir encontrar um binário do sistema em /sbin, você deve procurar em /usr/sbin. Ele também contém binários de sistema não essenciais, por exemplo: daemons para serviços de rede;

Diretório /usr/share: contém dados independentes de arquitetura (compartilhados);

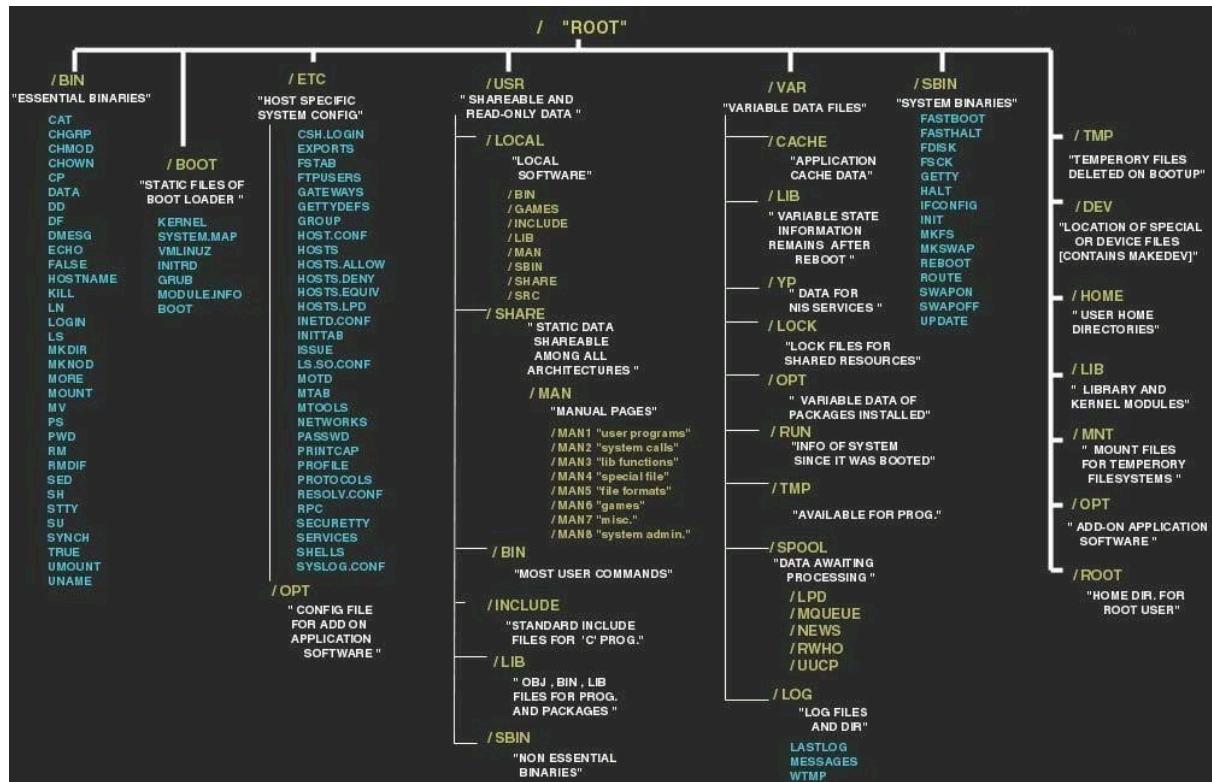
Diretório /usr/src: contém o código fonte como fontes do kernel do GNU/Linux, arquivos de cabeçalho e documentação;

Diretório /usr/X11: contém arquivos relacionados ao X Window System;

Diretório /var (arquivos variáveis): Este diretório contém arquivos cujo conteúdo deve ser alterado continuamente durante a operação normal do sistema - como logs, arquivos de spool e arquivo de e-mail temporário;

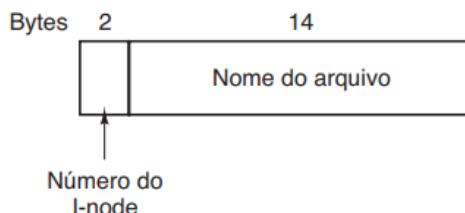
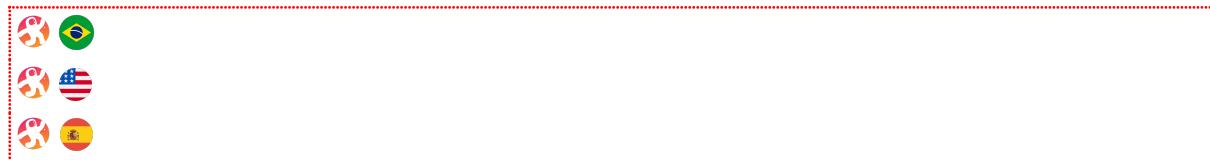
Diretório /var/log: contém arquivos de log do sistema.

É importante que antes de uma prova de certificação se repasse todos estes diretórios, recomendo rever a imagem abaixo que possui uma síntese desta organização.

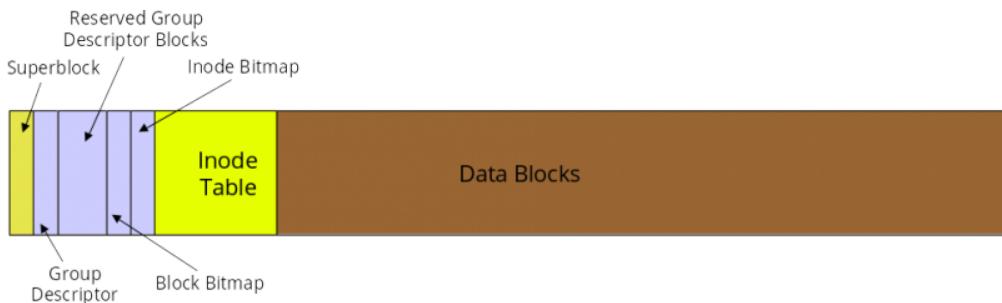


4.3 Conceito de i-node

Um i-node é uma estrutura de dados que descreve objetos do sistema de arquivos, que pode ser uma de várias coisas, incluindo um arquivo, diretório, um link, etc.. De acordo com Tanenbaum no Unix estes **i-node** eram formados por um registro contendo um número e o nome de um arquivo, conforme figura abaixo.



No sistema de arquivos existe uma tabela contendo estas entradas i-node (ver figura abaixo) e é nesta tabela que vamos encontrar os i-nodes. Saiba que o número de i-nodes tem um limite, geralmente é uma divisão média do espaço da partição pelo tamanho médio dos blocos, geralmente o tamanho médio dos blocos por padrão é 4096 bytes.



Em um sistema 64 bits pode-se aumentar o número de **i-node** reduzindo o tamanho dos blocos, sendo que o mínimo é 1024 bytes. Se o número máximo de **i-node** for alcançado nenhum arquivo mais poderá ser criado e as alterações começam a falhar, mesmo o sistema sendo iniciado tudo ficará instável. Em servidores que possuem aplicações verbosas que lançam log por arquivos individuais, isso pode ocorrer com frequência, o autor deste conteúdo já se deparou com tais problemas duas vezes em mais de uma década de experiência. No Linux, no i-node encontramos alguns atributos (ver figura abaixo), os principais são:

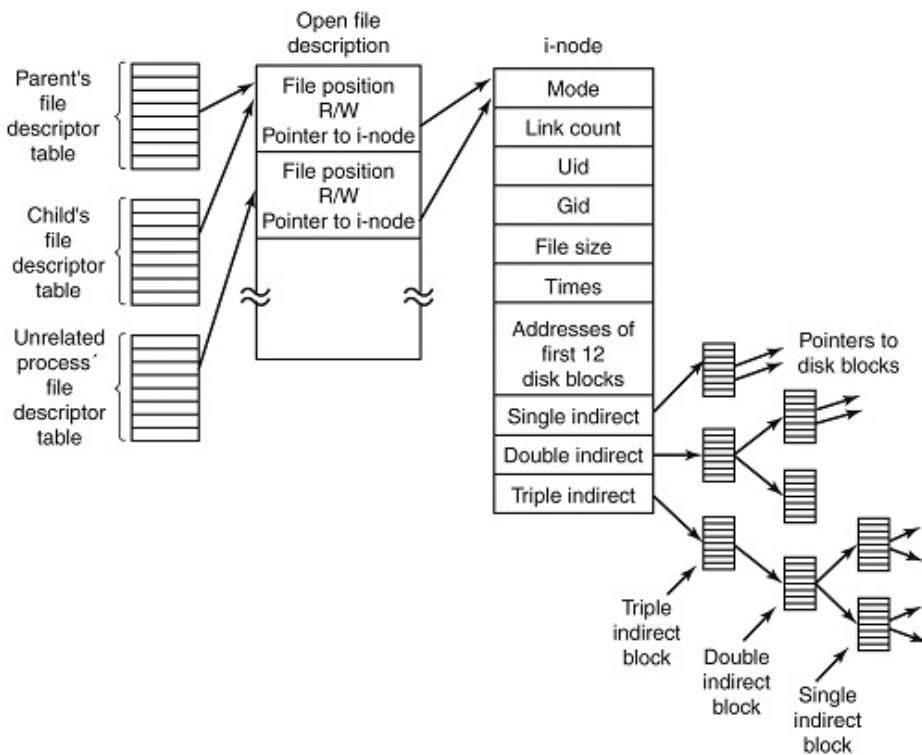
- nome do arquivo;
- o tamanho do arquivo;
- três horários:
 - criação;
 - último acesso;
 - última modificação.
- proprietário;
- grupo;
- informação de proteção;
- contagem do número de entradas de diretórios que apontam para o i-node.

Linux tem distinção de arquivos pequenos e arquivos grandes, em arquivos pequenos com menos de **40KB** os dados são armazenados diretamente no **i-node**, pois o **i-node** possui 10 endereços de 4KB (**supondo que a formatação foi realizada com 4KB por bloco e formatação ext3 ou ext4**). A grande maioria dos arquivos de configuração possuem menos de **40KB** e muitos arquivos de usuário também, o que agiliza a busca no sistema secundário de armazenamento e o devido carregamento para a memória.

Já para arquivos maiores, o **i-node** referencia tabelas contendo endereços que apontam para os dados em si, e temos

- **singly indirect**: 1 bloco completo de endereços de 1024 pointers, armazenando arquivos de até 4MB;
- **double indirect**: 1 bloco completo de single indirect, ou seja, 1024 single indirects, armazenando arquivos de até 4GB;
- **triple indirect**: 1 bloco completo de double indirect, ou seja, 1024 double indirects, armazenando arquivos de até 4TB.

Na figura abaixo temos um esquema de um **i-node**, com atributos e endereços de blocos.

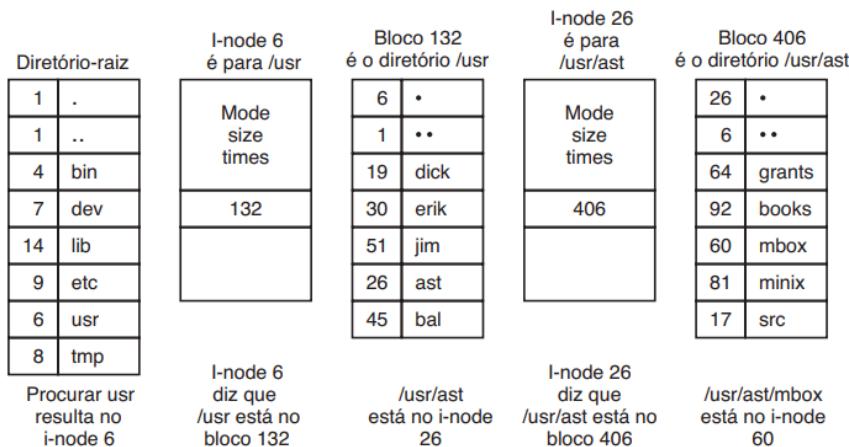


Quando um arquivo é aberto, o sistema deve tomar o nome do arquivo fornecido e localizar seus blocos de disco. Vamos considerar como o nome do caminho `/usr/ast/mbox` é procurado. Usaremos o UNIX¹⁷ como exemplo, mas o algoritmo é basicamente o mesmo para todos os sistemas de diretórios hierárquicos:

1. O sistema carrega o **i-node** referente ao diretório raiz /, este **i-node** aponta para o bloco 1 (neste exemplo);
2. Com o **i-node** carregado o bloco 1 será carregado para a memória, repare que é um diretório, então pode-se listar o diretório;
3. Digamos que queremos carregar o subdiretório `usr` na raiz /, então carregamos o **i-node** 6 que possui um endereço apontando para o bloco 132;
4. Carregamos o bloco 132 para a memória, então pode-se listar este bloco como um diretório;
5. Queremos carregar o subdiretório `ast` no diretório `/usr`, para isso carregamos o **i-node** 26 que possui um endereço para um bloco 406;
6. Carregamos para a memória o conteúdo do bloco 406, que se trata de um diretório;

Este algoritmo está exemplificado na figura abaixo, fica claro ao ver o número do i-node.

¹⁷ Utilizando como base o livro de Sistemas Operacionais Modernos do autor Tanenbaum;



No GNU/Linux assim como no Unix, o diretório `./` representa a entrada na tabela **i-node** do diretório corrente, e `../` representa a entrada na tabela i-node que leva ao diretório pai deste i-node. Algo muito interessante é que no caso do diretório raiz tanto o i-node do diretório `./` quanto o **i-node** do diretório pai apontam para o mesmo bloco de dados.

Na prática o i-node referente a raiz do sistema nem sempre será 1, em uma distro instalada em 2022 o i-node referente a raiz está com o número 2, isso vai depender naturalmente do processo de instalação que está automatizado por um wizard já estudado no capítulo 1.

Número do i-node

```

total 32
2 drwxr-xr-x 18 root root 4096 Aug 31 19:27 .
2 drwxr-xr-x 18 root root 4096 Aug 31 19:27 ..
12 lrwxrwxrwx 1 root root 7 Aug 31 19:23 bin -> usr/bin
1046529 drwxr-xr-x 3 root root 4096 Aug 31 19:31 boot
1 drwxr-xr-x 17 root root 3300 Sep 27 19:21 dev
1177345 drwxr-xr-x 66 root root 4096 Sep 27 19:21 etc
1308161 drwxr-xr-x 3 root root 4096 Aug 31 19:31 home
20 lrwxrwxrwx 1 root root 30 Aug 31 19:27 initrd.img -> boot/initrd.i
19 lrwxrwxrwx 1 root root 30 Aug 31 19:24 initrd.img.old -> boot/init
14 lrwxrwxrwx 1 root root 7 Aug 31 19:23 lib -> usr/lib
15 lrwxrwxrwx 1 root root 9 Aug 31 19:23 lib32 -> usr/lib32
16 lrwxrwxrwx 1 root root 9 Aug 31 19:23 lib64 -> usr/lib64
17 lrwxrwxrwx 1 root root 10 Aug 31 19:23 libx32 -> usr/libx32
11 drwxr-xr-x 2 root root 16384 Aug 31 19:23 lost+found
1700609 drwxr-xr-x 3 root root 4096 Aug 31 19:23 media
1308162 drwxr-xr-x 2 root root 4096 Aug 31 19:23 mnt
5007297 l

```

4.4 Tipos de arquivos

O UNIX e incluindo o GNU/Linux possuem arquivos regulares, diretórios, arquivos especiais de carácter e arquivos especiais de bloco.

- Arquivos regulares ('-'):
 - Arquivos de dados;
 - Arquivos executáveis;
- Diretórios ('d');
- Arquivos especiais:
 - Dispositivos de Blocos ('b');
 - Dispositivos de Caracteres ('c');
 - Links simbólicos ('l');
- Comunicação inter processos IPC:

- Socket ('s');
- Pipes de comunicação ('p');

Um diretório é um arquivo que contém informações sobre a estrutura do sistema de arquivos conforme vimos no tópico de i-node, arquivos regulares possuem dados para usuários ou é um executável. Segundo o capítulo de Arquivos do livro de Sistemas Operacionais Modernos, o que distingue na visão do sistema operacional se o arquivo regular é um arquivo de dados ou um executável é a existência de um bit mágico ativo em uma determinada posição quando é um executável, conforme figura abaixo.



Todo arquivo criado tem um dono e um grupo dono, como exemplo vamos supor que exista um usuário chamado aluno pertencente ao grupo dos alunos, então quando o aluno cria um novo arquivo regular este arquivo é criado com o seguinte critério de acesso:

- O dono do arquivo (aluno) pode: ler e escrever no arquivo;
- O grupo dono do arquivo (alunos) pode: ler o arquivo;
- Outros usuários podem: ler o arquivo;

Essa regra é definida pelo Umask¹⁸ do sistema operacional. Já no caso deste mesmo usuário criar um diretório, então o novo diretório terá o seguinte esquema de permissão de acesso:

- O dono do diretório (aluno) pode: ler, escrever e executar;
- O grupo dono do diretório (alunos) pode: ler e executar;
- Outros podem: ler e executar;

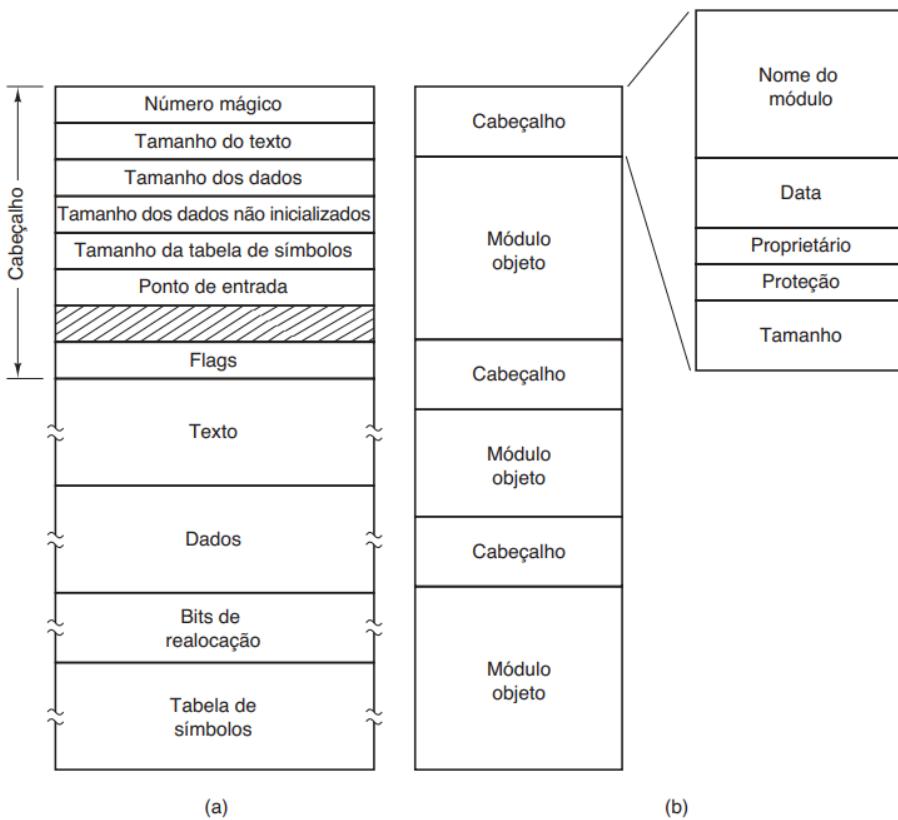
Pode soar estranho o termo “executar um diretório”, mas fará sentido quando estudarmos as permissões, por enquanto entenda que “executar” seria “listar”.

4.4.1 Arquivos regulares

Quando um arquivo regular é criado ele ou é criado em UTF-8 ou em ASCII, a predefinição deste deste requisito é realizado quando se define o charset do Sistema Operacional durante a instalação, conforme **Figura 1.11** (um passo durante a instalação do GNU/Linux Debian).

¹⁸ Mais detalhes no link <https://wiki.debian.org/Debate/umask>

(a) Um arquivo executável. (b) Um repositório (archive).



Para um usuário Microsoft Windows pode soar estranho a ideia que grande parte “dos arquivos que executam” na verdade sejam scripts, ou seja, não sejam arquivos compilados como um programa. Afinal o mundo Microsoft Windows o foco é “caixa fechada” enquanto no mundo Linux o foco é “caixa aberta”. Neste cenário dinâmico ao longo dos anos surgiram inúmeros entusiastas que criaram de forma rápida e simples poderosos scripts que suplantaram a necessidade de um programa.

Mas saiba, mesmo sendo um script com permissão de execução, este por ser interpretado por um programa ainda é um arquivo regular e não um arquivo executável, executável é o /bin/bash que o interpreta. Na figura abaixo o script que possui permissão de execução é identificado como um texto UTF-8.

```
well@wpo:/tmp$ file meuscript.sh
meuscript.sh: Bourne-Again shell script, UTF-8 Unicode text executable
well@wpo:/tmp$
```

Quando um arquivo é binário executável e possui a permissão de execução este é então interpretado por um mecanismo binário do próprio sistema operacional, conforme visto na figura abaixo.

```
usuario@debian:/tmp$ file copiar
copiar: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=2e16ba4b2f43f5ee0d483791d4ad706f7ba9063e, for GNU/Linux 3.2.0, not stripped
usuario@debian:/tmp$
```

Para exemplificar, no exemplo a seguir será criado um arquivo de dados chamado copiar.cpp que deverá estar no diretório /tmp/, neste arquivo o aluno irá codificar um código em C++ e depois executará o processo de compilação, gerando assim um arquivo regular executável chamado copiar que deverá estar no mesmo diretório /tmp/.

Este novo programa criado executará processos de cópia de arquivos, embora o GNU/Linux já tenha este comando o objetivo é discutir:

- Como codificar um programa em C++;
- Como compilar um programa gerando um arquivo regular executável;
- Boas práticas de manipulação de arquivos no Sistema Operacional.

O primeiro ponto interessante é a forma errada que clássicos livros de programação ensinam, tomo como exemplo **Java como Programar** do autor **Deitel&Deitel**, basicamente o autor ensina a abrir um arquivo e jogar em um array de bytes, isso é incondizente com questões de segurança em sistemas operacionais, pois se o arquivo for grande o suficiente para elevar um exceptio do tipo OutofMemory a aplicação e talvez até o sistema vão entrar em colapso.

A ideia é abrir um arquivo e de 4 em 4 KB¹⁹ ir copiando de um arquivo para o outro. Quando abrimos um arquivo em um GNU/Linux na verdade não o carregamos para a memória, mas criamos um endereço numérico para uma referência de arquivo em memória que teoricamente não reside na memória. Por este motivo, a função **open()** retorna um número inteiro para nos referir ao arquivo que queremos operar.

Para começar, é importante que tenha domínio sobre o editor nano, conteúdo já discutido no tópico [Editor de texto nano](#). Então no terminal execute os dois comandos abaixo na sequência.

1. cd /tmp
2. nano copiar.cpp

Primeiro passo foi navegar para o diretório /tmp pois é neste diretório que podemos fazer testes, afinal tudo que está neste diretório será automaticamente excluído na próxima inicialização. E em seguida passamos para o comando nano o nome do arquivo **copiar.cpp**, que não existe ainda, mas a partir do momento que salvar, ele existirá. Edite o seguinte código neste arquivo, conforme listagem abaixo:

1. #include <fcntl.h>
2. #include <stdlib.h>
3. #include <unistd.h>
- 4.

¹⁹ O Debian GNU/Linux utiliza 4KB como unidade de bloco de formatação;

```

5. int main(int argc, char *argv[]);
6. #define BUF_SIZE 4096
7. #define OUTPUT_MODE 0640
8.
9. int main(int argc, char *argv[])
10. {
11.     int in_fd, out_fd, rd_count, wt_count;
12.     char buffer[BUF_SIZE];
13.     if (argc != 3) exit(1);
14.     in_fd = open(argv[1], O_RDONLY);
15.     if (in_fd < 0) exit(2);
16.     out_fd = creat(argv[2], OUTPUT_MODE);
17.     if (out_fd < 0) exit(3);
18.
19.     while (true) {
20.         rd_count = read(in_fd, buffer, BUF_SIZE);
21.         if (rd_count <= 0) break;
22.         wt_count = write(out_fd, buffer, rd_count);
23.         if (wt_count <= 0) exit(4);
24.     }
25.
26.     close(in_fd);
27.     close(out_fd);
28.
29.     if (rd_count == 0)
30.         exit(0);
31.     else
32.         exit(5);
33. }

```

Conforme já explicado os comandos **open()** e **create()** deverão retornar dois números inteiros para referenciar os arquivos na memória (**in_fd** e **out_fd**), estes números são então utilizados no **read()**, **write()** e **close()**. Dentro do laço de repetição **while()** consumimos apenas 4096 bytes para realizar a cópia do conteúdo, não importa o quão grande seja o arquivo. Se achar que quer consumir mais memória mas reduzir o número de repetições do **while()** pode multiplicar **BUF_SIZE** por algum número, talvez 10 ou 100, fica a seu critério.

Este novo programa que estamos criando apenas deverá realizar a cópia de arquivos, então no mesmo diretório crie um novo arquivo chamado **arquivo1.txt** (com o nano) e escreva qualquer coisa, pois servirá apenas para ser copiado.

Para este exemplo é utilizado os seguintes headers que já existem em um GNU/Linux:

- **fcntl.h**²⁰: O header **<fcntl.h>** define constantes para argumentos cmd e algumas rotinas, no código será útil para definir: **O_RDONLY**, **open()** e **creat()**;
- **stdlib.h**²¹: O header **<stdlib.h>** definido como referência ISO C, possui recursos tais como constantes e funções que manterão o padrão **POSIX**. É útil no código para referência ao **exit()**;
- **unistd.h**²²: O header **<unistd.h>** define constantes e funções diversas, será útil neste código para uso das funções: **close()**, **read()** e **write()**.

²⁰ Referência completa: <https://man7.org/linux/man-pages/man0/fcntl.h.0p.html>

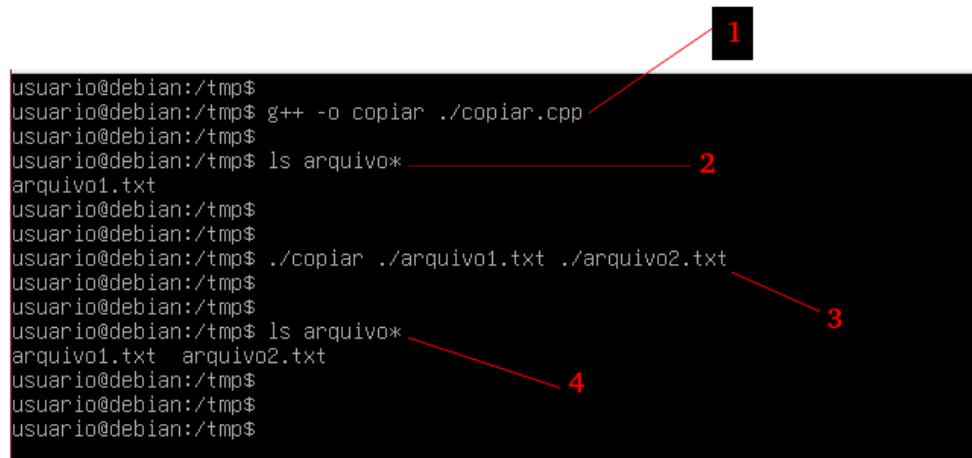
²¹ Referência completa: <https://man7.org/linux/man-pages/man0/stdlib.h.0p.html>

²² Referência completa: <https://man7.org/linux/man-pages/man0/unistd.h.0p.html>

Pronto, agora o código e o arquivo que será alvo de nosso programa já existem, precisamos agora de ter um compilador C++, para isso utilize o comando apt para instalar o g++ conforme comando de terminal abaixo.

```
1. sudo apt install g++ -y
```

Com o compilador g++ instalado, agora o processo de compilação será simples, basta executar a sequência de comandos, conforme figura abaixo.



The terminal session shows the following steps:

1. Compilation of the C++ code: `g++ -o copiar ./copiar.cpp`
2. Listing files: `ls arquivo*` showing `arquivo1.txt`
3. Execution of the compiled program: `./copiar ./arquivo1.txt ./arquivo2.txt`
4. Listing files again: `ls arquivo*` showing both `arquivo1.txt` and `arquivo2.txt`

```

usuario@debian:/tmp$ g++ -o copiar ./copiar.cpp
usuario@debian:/tmp$ ls arquivo*
arquivo1.txt
usuario@debian:/tmp$ ./copiar ./arquivo1.txt ./arquivo2.txt
usuario@debian:/tmp$ ls arquivo*
arquivo1.txt arquivo2.txt
usuario@debian:/tmp$
```

Onde:

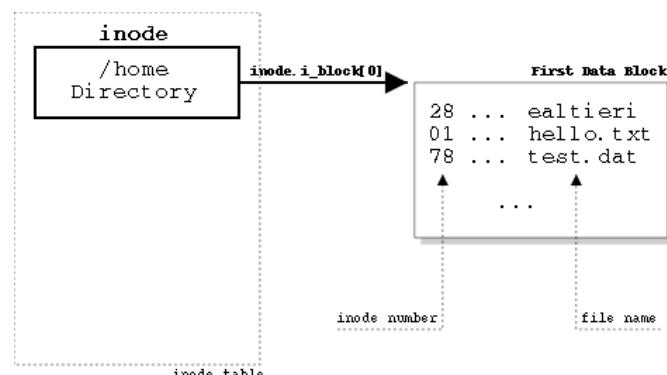
1. Compilamos o código escrito em C++;
2. Verificando se existe um arquivo para copiar;
3. Utilizando o programa recém compilado e escrito por você para copiar o arquivo1.txt;
4. Verificando a existência do segundo arquivo, cópia do primeiro.

Ao executar o comando `./copiar` e informar o arquivo que será copiado o executável então deverá realizar a cópia gerando o **arquivo2.txt** que não existia até então.

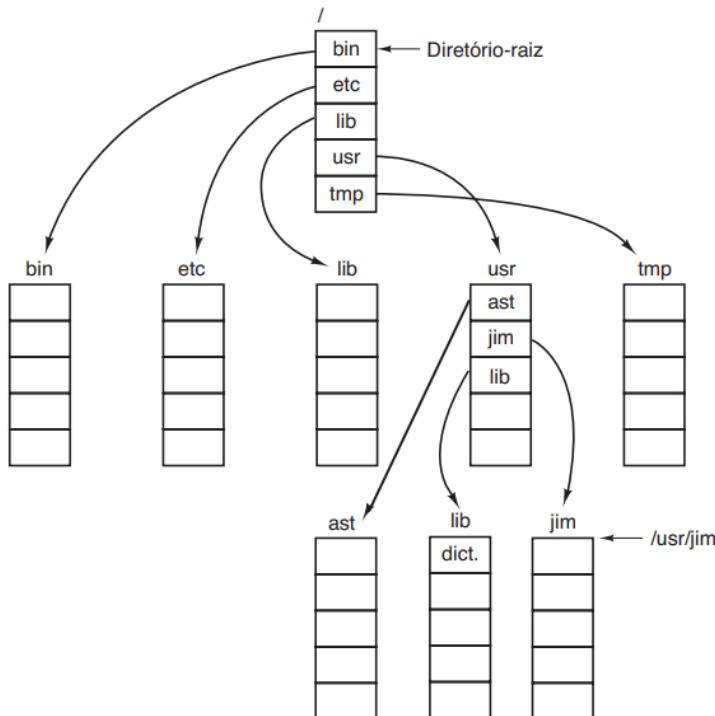
4.4.2 Diretórios

Um diretório é um arquivo utilizado para indexar outros arquivos, e naturalmente estes arquivos podem ser arquivos regulares ou outros diretórios.

Faz-se necessária uma hierarquia pois a capacidade de armazenamento destes arquivos é muito grande e naturalmente deverá ser organizado.



Com essa abordagem, o usuário pode ter tantos diretórios quantos²³ forem necessários para agrupar seus arquivos de maneira natural. Além disso, se múltiplos usuários compartilham um servidor de arquivos comum, como é o caso em muitas redes de empresas, cada usuário pode ter um diretório-raiz privado para sua própria hierarquia.



Neste exemplo o programa deverá realizar uma listagem de diretórios e arquivos que estão localizados na raiz do GNU/Linux, neste caso `/`. Para este exemplo será utilizadas referências à algumas bibliotecas, conforme lista abaixo:

- **iostream**²⁴: A biblioteca `<iostream>` conhecida como **Standard I/O Streams Library** tem definições de objetos e funções destinadas a manipular dispositivos de I/O, tal como monitores e teclados. Neste código está sendo utilizado o `cout`;
- **filesystem**²⁵: A biblioteca `<filesystem>` fornece recursos para executar operações em sistemas de arquivos e seus componentes, como caminhos, arquivos regulares e diretórios, foi publicada como a especificação técnica ISO/IEC TS 18822:2015 e finalmente fundida com ISO C++ a partir de C++17. Utilizada para referenciar `filesystem` e `fs`.

Basicamente é navegar para o diretório `/tmp` e criar um arquivo chamado `listar.cpp` e neste arquivo digitar o código da listagem abaixo.

```

1. #include <iostream>
2. #include <filesystem>
3.
4. namespace fs = std::filesystem;

```

²³ Existe um limite de i-node total de um sistema de arquivos, trata-se de uma relação entre tamanho da **partição/tamanho** do bloco;

²⁴ Referência completa: <https://en.cppreference.com/w/cpp/header/iostream>

²⁵ Referência completa: <https://en.cppreference.com/w/cpp/filesystem>

```

5.
6. int main()
7. {
8.     std::string path = "/";
9.     for (const auto & entry : fs::directory_iterator(path))
10.         std::cout << entry.path() << std::endl;
11. }
```

Com o **g++** compile e em seguida execute. O Output deste programa é uma lista de arquivos e diretórios que estão na raiz do seu GNU/Linux. Conforme a teoria um diretório pode listar arquivos e outros diretórios, então no próximo exemplo **listarrecursivo.cpp** o programa terá a capacidade de listar diretórios e subdiretórios de forma recursiva e imprimir todos no console. Para não exibir uma lista quase interminável, será restrito ao diretório **/home**.

```

1. #include <iostream>
2. #include <filesystem>
3.
4. namespace fs = std::filesystem;
5.
6. void listar(std::string path)
7. {
8.     for (const auto & entry : fs::directory_iterator(path))
9.     {
10.         std::cout << entry.path() << std::endl;
11.         if( is_directory( entry.path() ) )
12.         {
13.             listar( entry.path() );
14.         }
15.     }
16. }
17.
18. int main()
19. {
20.     listar("/home");
21. }
```

O destaque neste código é dado a função **is_directory()** definida na biblioteca **filesystem**, como o código é recursivo espera-se que pare naturalmente, a não ser que tenha um link apontando para um diretório anterior, mas o assunto link será abordado.

4.4.3 Arquivos especiais de bloco e de carácter

Sistemas especiais de Bloco e Carácter são arquivos que referenciam dispositivos de I/O no GNU/Linux, estes arquivos são referenciados em **/dev** conforme figura abaixo onde estão sendo listados todos os dispositivos especiais de bloco.

```
usuario@debian:/tmp$ ls -l /dev | grep -i ^b
brw-rw---- 1 root disk 7, 0 Oct 1 12:46 loop0
brw-rw---- 1 root disk 7, 1 Oct 1 12:46 loop1
brw-rw---- 1 root disk 7, 2 Oct 1 12:46 loop2
brw-rw---- 1 root disk 7, 3 Oct 1 12:46 loop3
brw-rw---- 1 root disk 7, 4 Oct 1 12:46 loop4
brw-rw---- 1 root disk 7, 5 Oct 1 12:46 loop5
brw-rw---- 1 root disk 7, 6 Oct 1 12:46 loop6
brw-rw---- 1 root disk 7, 7 Oct 1 12:46 loop7
brw-rw---- 1 root disk 8, 0 Oct 1 12:46 sda
brw-rw---- 1 root disk 8, 1 Oct 1 12:46 sda1
brw-rw---- 1 root disk 8, 2 Oct 1 12:46 sda2
brw-rw---- 1 root disk 8, 5 Oct 1 12:46 sda5
brw-rw----+ 1 root cdrom 11, 0 Oct 1 12:47 sr0
usuario@debian:/tmp$
```

Dispositivos de bloco conforme teoria descrita no Tanenbaum são dispositivos de I/O que operam com blocos de Bytes (caracteres) em suas operações de **write()** e **read()**, já os dispositivos especiais de carácter operam com apenas um byte (carácter).

No próximo exemplo, o programa vai listar os arquivos de **/dev** e imprimir no terminal com **cout** os arquivos que são de especiais de carácter e de bloco. Crie um arquivo **listarespecial.cpp** e codifique neste arquivo o código descrito na listagem abaixo.

```
1. #include <iostream>
2. #include <filesystem>
3.
4. namespace fs = std::filesystem;
5.
6.
7. int main()
8. {
9.     std::string path = "/dev";
10.    for (const auto & entry : fs::directory_iterator(path))
11.    {
12.        if (fs::is_block_file( entry.path() ))
13.        {
14.            std::cout << entry.path() << ": arquivo especial de bloco" <<
15.            std::endl;
16.            continue;
17.        if (fs::is_character_file( entry.path() ))
18.        {
19.            std::cout << entry.path() << ": arquivo especial de carácter" <<
20.            std::endl;
21.            continue;
22.        }
23.    }
24. }
```

O destaque deste código é o uso de **is_block_file()** e **is_character_file()** descritos na biblioteca **filesystem**.

4.5 Informações sobre arquivos (file e stat)

Como no GNU/Linux, extensão não é um requisito obrigatório para um arquivo e é muito comum se localizar arquivos sem extensão.



Utiliza-se o comando **file** para se compreender do que se trata o arquivo. Exemplo:

1. **file /etc/passwd**

A saída deste comando informa que o arquivo é um arquivo de texto, conforme figura abaixo.

```
usuario@debian:/tmp$ file /etc/passwd
/etc/passwd: ASCII text
usuario@debian:/tmp$ _
```

Figura 4.

O **stat** é um comando que fornece informações sobre o arquivo e o sistema de arquivos, este comando **Stat** fornece informações como:

- tamanho do arquivo;
- permissões de acesso;
- ID do usuário e ID do grupo;
- hora de criação;
- hora de acesso ao arquivo.

O comando **Stat** possui outro recurso, pelo qual também pode fornecer informações do sistema de arquivos. Esta é a melhor ferramenta para utilizar quando queremos as informações de algum arquivo.

```
usuario@debian:/tmp$ stat /etc/passwd
  File: /etc/passwd
  Size: 1142          Blocks: 8          IO Block: 4096   regular file
Device: 8,1      Inode: 262265      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/  root)  Gid: (    0/  root)
Access: 2023-10-01 12:21:40.247561318 -0300
Modify: 2023-10-01 12:21:40.243560650 -0300
Change: 2023-10-01 12:21:40.243560650 -0300
 Birth: 2023-10-01 12:21:40.243560650 -0300
usuario@debian:/tmp$
```

Figura 4.

No próximo exemplo a seguir, será listado algumas propriedades do diretório **/tmp**, basicamente será reproduzido o próprio comando **stat** do sistema operacional, para isso será necessário incluir o header **sys/stat.h** para ter acesso a função **stat()**. Crie um arquivo chamado **exemplostat.cpp** e codifique a listagem abaixo.

```

1. #include<iostream>
2. #include<sys/stat.h>
3.
4. using namespace std;
5.
6. int main()
7. {
8.     std::string path = "/tmp";
9.     struct stat var;
10.    const char* chr = path.c_str();
11.    int ret= stat( chr , &var );
12.
13.    if( ret < 0 )
14.    {
15.        cout << "A chamada do sistema 'stat' foi encerrada com um código de erro: "
16.        << ret << endl;
17.    }
18.    else
19.    {
20.        cout << "ID do device: " << var.st_dev << endl;
21.        cout << "Número Inode: " << var.st_ino << endl;
22.        cout << "Mode: " << var.st_mode << endl;
23.        cout << "UID: " << var.st_uid << endl;
24.        cout << "GID: " << var.st_gid << endl;
25.        cout << "Size: " << var.st_size << endl;
26.
27.    struct stat
28.    {
29.        dev_t      st_dev;      /* ID do dispositivo que contém o arquivo */
30.        ino_t      st_ino;      /* número inode */
31.        mode_t     st_mode;     /* proteção */
32.        nlink_t    st_nlink;    /* número de links físicos */
33.        uid_t      st_uid;      /* ID de usuário do proprietário */
34.        gid_t      st_gid;      /* ID do grupo do proprietário */
35.        off_t      st_size;     /* tamanho total, em bytes */
36.        blksize_t  st_blksize;  /* block size for filesystem I/O */
37.        blkcnt_t   st_blocks;   /* número de blocos de 512 Bytes alocados */
38.    };
39. }

```

Compile este código com g++ com o nome **exemplostat**, rode o comando.

```

usuario@debian:/tmp$ 
usuario@debian:/tmp$ ./exemplostat
Device id: 2049
Inode number: 129803
Mode: 17407
UID: 0
GID: 0
Size: 4096
usuario@debian:/tmp$ 

```

GNU/Linux e C/C++ tem muita aderência, em tudo que fazemos com comandos GNU/Linux podemos refazer com C/C++.

4.6 Navegando no sistema de Arquivos (cd, pwd, ls)

Quando o sistema de arquivos é montado na memória principal é mapeado todos os dispositivos secundários e é possível movimentar um cursor nesta árvore. Mas é comum se ter dúvidas de qual posição está o cursor. O comando **pwd** é utilizado para imprimir no terminal a posição do cursor no sistema de arquivos, conforme figura abaixo.

```
usuario@debian:~$ pwd
/home/usuario
usuario@debian:~$ _
```

Ao executar **pwd** o comando retornou que o cursor está em **/home/usuario**, embora pareça um comando sem utilidade, na verdade ele é muito importante, em operações que envolvem manipulação do sistema de arquivos e que tal manipulação será executada em um nível crucial para execução do sistema, é importante olhar antes e confirmar onde está, para evitar problemas.

No próximo exemplo o programa escrito em C++ deverá exibir o diretório que está no cursor no sistema de arquivos, um resultado final igual ao comando **pwd**, para isso será utilizado a função **getcwd()**. Crie o arquivo **mypwd.cpp** e codifique segundo a listagem abaixo.

```
1. #include <iostream>
2. #include <unistd.h>
3.
4. int main() {
5.     char tmp[256];
6.     getcwd(tmp, 256);
7.
8.     std::cout << "Directory: " << tmp << std::endl;
9.     return 0;
10. }
```

Compile e execute de qualquer ponto do sistema de arquivos, verá que retorna o diretório em que está o cursor no sistema de arquivos e não o diretório que está o programa compilado.

```
1. g++ -o mypwd mypwd.cpp
2. ./mypwd
```

Outro comando muito interessante que revela arquivos e diretórios que estão dentro do diretório corrente do cursor é o **comando ls**. Com um simples "ls" é possível listar estes elementos em forma de grade, conforme figura abaixo.

```
usuario@debian:/ $ ls
bin  etc  initrd.img.old  lib32  mnt  root  srv  usr      vmlinuz.old
boot  home  lib  lost+found  opt  run  sys  var
dev  initrd.img  lib64  media  proc  sbin  tmp  vmlinuz
usuario@debian:/ $
```

Alguns GNU/Linux exibem cores para distinguir arquivos, neste caso estou usando um GNU/Linux Debian e podemos ver azul para diretórios e azul claro para links. Mas é comum se precisar de mais informações, como dono do arquivo, permissão de acesso entre outras informações, ao exibir em lista tais informações são exibidas, veja o parâmetro **-l** na figura abaixo.

Tipo de arquivo	Referencias na tabela i-node		Modificado em	Nome
	Permissao	Dono		
usuario@debian:~/	ls -l			
	total 60			
lrwxrwxrwx	1 root root	7 Nov 20 12:36	bin	→ usr/bin
drwxr-xr-x	3 root root	4096 Nov 20 12:39	boot	
drwxr-xr-x	16 root root	3160 Feb 8 19:18	dev	
drwxr-xr-x	70 root root	4096 Feb 8 19:18	etc	
drwxr-xr-x	3 root root	4096 Nov 20 12:39	home	
lrwxrwxrwx	1 root root	29 Nov 20 12:37	initrd.img	→ boot/initrd.img-4.19.0-12-
lrwxrwxrwx	1 root root	29 Nov 20 12:36	initrd.img.old	→ boot/initrd.img-4.19.0-
lrwxrwxrwx	1 root root	7 Nov 20 12:36	lib	→ usr/lib
lrwxrwxrwx	1 root root	9 Nov 20 12:36	lib64	→ usr/lib64

O tamanho do arquivo está em bytes na figura acima, caso precise ver o tamanho em uma versão mais “humana” adicione o parâmetro **-h** ao comando conforme figura abaixo.

```
usuario@debian:~/
```

```
ls -lh
```

```
total 60K
```

```
lrwxrwxrwx 1 root root 7 Nov 20 12:36 bin → usr/bin
```

```
drwxr-xr-x 3 root root 4.0K Nov 20 12:39 boot
```

```
drwxr-xr-x 16 root root 3.1K Feb 8 19:18 dev
```

```
drwxr-xr-x 70 root root 4.0K Feb 8 19:18 etc
```

```
drwxr-xr-x 3 root root 4.0K Nov 20 12:39 home
```

```
lrwxrwxrwx 1 root root 29 Nov 20 12:37 initrd.img → boot/init
```

```
lrwxrwxrwx 1 root root 29 Nov 20 12:36 initrd.img.old → boot/init
```

Veja que agora a listagem exibe a quantidade na maior unidade de medida possível, que neste caso está em Kbytes. Para listar o número i-node no sistema de arquivos utilize o parâmetro **-i** conforme figura abaixo.

Numero do i-node no sistema de arquivos

```
usuario@debian:~/
```

```
ls -li
```

```
total 60
```

```
12 lrwxrwxrwx 1 root root 7 Nov 20 12:36 bin → usr/bin
```

```
655361 drwxr-xr-x 3 root root 4096 Nov 20 12:39 boot
```

```
3 drwxr-xr-x 16 root root 3160 Feb 8 19:18 dev
```

```
917505 drwxr-xr-x 70 root root 4096 Feb 8 19:18 etc
```

```
655362 drwxr-xr-x 3 root root 4096 Nov 20 12:39 home
```

```
23 lrwxrwxrwx 1 root root 29 Nov 20 12:37 initrd.img →
```

```
22 lrwxrwxrwx 1 root root 29 Nov 20 12:36 initrd.img.old
```

```
14 lrwxrwxrwx 1 root root 7 Nov 20 12:36 lib → usr/lib
```

```
15 lrwxrwxrwx 1 root root 9 Nov 20 12:36 lib64 → usr/l
```

```
16 lrwxrwxrwx 1 root root 10 Nov 20 12:36 lib32 → usr/
```

Para ordenação, utiliza-se os parâmetros:

- **S**: ordena arquivos por tamanho;
- **X**: Ordena os arquivos por extensão;
- **t**: ordena por tempo de modificação do arquivo;
- **r**: ordem inversa/contrária.

Para listar arquivos e diretórios ocultos basta utilizar o parâmetro **-a** conforme exemplo abaixo.

```
well@wpo:~/projects/c$ ls -la
total 48
drwxrwxr-x 10 well well 4096 fev 2 22:19 .
drwxrwxr-x 14 well well 4096 fev 3 21:27 ..
drwxrwxr-x 3 well well 4096 fev 2 22:53 aied
-rwxrwxr-x 1 well well 672 jan 25 23:44 build.sh
drwxr-xr-x 2 root root 4096 fev 2 18:50 dist
drwxrwxr-x 2 well well 4096 jan 18 20:08 engine
drwxrwxr-x 2 well well 4096 jan 18 20:08 functions
drwxrwxr-x 8 well well 4096 fev 3 10:08 .git
-rw-rw-r-- 1 well well 624 jan 18 20:08 .gitignore
drwxrwxr-x 4 well well 4096 jan 18 20:08 infect
drwxrwxr-x 5 well well 4096 jan 18 20:08 rctk
drwxr-xr-x 5 root root 4096 fev 2 22:49 tmp
well@wpo:~/projects/c$
```

Ocultos

Para evitar ter que ficar navegando com o cursor no sistema de arquivos apenas para ver se arquivos existem, pode-se informar um diretório no final do comando como parâmetro para que o comando **ls** listar aquele diretório.

```
usuario@debian:~/ls -l /var/log
total 1236
-rw-r--r-- 1 root root 0 Feb 8 11:38 alternatives.log
-rw-r--r-- 1 root root 17600 Nov 20 12:41 alternatives.log.1
drwxr-xr-x 2 root root 4096 Feb 8 11:40 apt
-rw-r----- 1 root adm 4848 Feb 8 19:18 auth.log
-rw-r----- 1 root adm 8581 Feb 8 11:38 auth.log.1
-rw-rw---- 1 root utmp 0 Feb 8 11:38 btmp
-rw-rw---- 1 root utmp 384 Nov 24 23:37 btmp.1
-rw-r----- 1 root adm 16084 Feb 8 19:33 daemon.log
-rw-r----- 1 root adm 51125 Feb 8 11:38 daemon.log.1
-rw-r----- 1 root adm 6385 Feb 8 19:18 debug
-rw-r----- 1 root adm 31925 Feb 8 11:38 debug.1
-rw-r--r-- 1 root root 1242 Feb 8 11:40 dpkg.log
-rw-r--r-- 1 root root 201997 Nov 20 12:41 dpkg.log.1
-rw-r--r-- 1 root root 24024 Nov 20 12:39 faillog
drwxr-xr-x 3 root root 4096 Nov 20 12:39 installer
-rw-r----- 1 root adm 43459 Feb 8 19:18 kern.log
-rw-r----- 1 root adm 215961 Feb 8 11:38 kern.log.1
-rw-rw-r-- 1 root utmp 292292 Feb 8 19:18 lastlog
```

4.7 Copiando e movendo arquivos (mv e cp)

Uma operação comum em sistemas operacionais é a operação de cópia e movimentação de arquivos, isso pois estamos constantemente reorganizando os arquivos no Sistema de Arquivos a fim de facilitar a busca para o ser humano.

Para copiar um arquivo de um local para outro é simples, basta informar o comando **cp** bem como os seguintes parâmetros na sequência: caminho do arquivo e diretório no qual se pretende guardar uma cópia do arquivo.

```
usuario@debian:~$ ls
meuarquivo
usuario@debian:~$ cp meuarquivo /tmp
usuario@debian:~$ ls /tmp
meuarquivo  systemd-private-4600c7ac299c4b5
usuario@debian:~$
```

Apenas mostrar que o arquivo existe

Fazer a copia do arquivo

Mostrar que foi copiado

Copiar com outro nome? Sim, uma operação muito útil pois constantemente estamos editando arquivos de configuração e caso haja falhas temos um arquivo para voltar aquela versão anterior de configuração.

```

Mostro que o arquivo existe
Faco a copia
Mostro os dois arquivos

usuario@debian:~$ ls /etc/network/interfaces
/etc/network/interfaces
usuario@debian:~$ sudo cp /etc/network/interfaces /etc/network/interfaces.old
[sudo] password for usuario:
usuario@debian:~$ ls /etc/network
if-down.d  if-post-down.d  if-pre-up.d  interfaces  interfaces.d  interfaces.old
usuario@debian:~$
```

Veja que além do caminho do arquivo foi passado o diretório de destino bem como o nome do arquivo lá no diretório de destino, neste caso, ambos estão no mesmo diretório.

Para copiar um diretório bem como seus arquivos internos de forma recursiva basta utilizar o parâmetro **-r** conforme figura abaixo, o exemplo abaixo não é uma operação que se realize, foi realizada apenas para demonstração.

Copiando o diretório /etc/network de forma recursiva para /tmp

```

Copia do diretório

usuario@debian:~$ sudo cp -r /etc/network /tmp/
usuario@debian:~$ ls /tmp
meuarquivo  systemd-private-4600c7ac299c4b5aa94d1eba89d
network
usuario@debian:~$
```

Caso queira copiar o conteúdo de um diretório de forma recursiva então deve-se informar o diretório, o parâmetro **-r** e adicionar **/***, conforme exemplo abaixo.

O uso do /* para informar que quer copiar o conteúdo

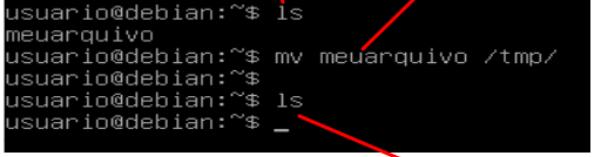
```

Arquivos em /tmp

usuario@debian:~$ ls /tmp
if-down.d  interfaces.d
if-post-down.d  interfaces.old
if-pre-up.d  meuarquivo
if-up.d  network
interfaces  systemd-private-4600c7ac299c4b5aa94d1eba89d
usuario@debian:~$
```

A movimentação de arquivo elimina o arquivo original após sucesso na cópia para outra posição no sistema de arquivos. O comando é simples, conforme exemplo abaixo.

Mostrar a existencia de um arquivo



Realizar a movimentacao

Mostrar que o arquivo nao esta mais na posicao

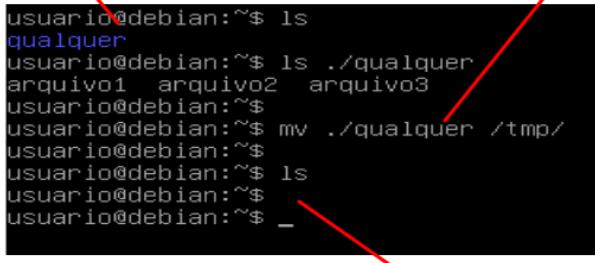
```

usuario@debian:~$ ls
meuarquivo
usuario@debian:~$ mv meuarquivo /tmp/
usuario@debian:~$ ls
usuario@debian:~$ -

```

Para movimentar um diretório com arquivos basta informar o diretório e o destino, conforme exemplo abaixo.

Mostrar um diretorio com arquivos



Fazer a movimentacao do diretorio com o comando mv

Nao existe mais o diretorio

```

usuario@debian:~$ ls
qualquer
usuario@debian:~$ ls ./qualquer
arquivo1 arquivo2 arquivo3
usuario@debian:~$ 
usuario@debian:~$ mv ./qualquer /tmp/
usuario@debian:~$ 
usuario@debian:~$ ls
usuario@debian:~$ 
usuario@debian:~$ -

```

Veja que não foi preciso se informar o parâmetro **-r** como foi feito no comando cp. Nesse próximo exemplo vamos mover um arquivo por um programa feito em C++, mas antes de codificar o código, crie um arquivo em **/home/userlinux/** chamado **teste.txt**, escreva qualquer texto. Agora em **/tmp** crie um novo arquivo chamado **mover.cpp** e codifique a listagem abaixo. Compile com **g++** e execute.

```

1. #include <iostream>
2. #include <cstdio>
3.
4. int main(int argc, char *argv[])
5. {
6.     if (argc != 3) exit(1);
7.
8.     if (std::rename(argv[1], argv[2]) != 0)
9.         perror("Erro ao renomear arquivo");
10.    else
11.        std::cout << "Arquivo renomeado" << std::endl;
12.    return 0;
13. }

```

Este código executa um rename() no arquivo, então desta forma será movido de **/home/userlinux/** para **/tmp/**, para isso execute o comando:

```
1. g++ -o /tmp/mover /tmp/mover.cpp
2. /tmp/mover /home/userlinux/teste.txt /tmp/teste.txt
```

4.8 Lendo arquivos (cat, tac, more, sed, tail, split)

Existem duas operações que a grande maioria dos usuários GNU/Linux confundem, são estas opções:

- ler um arquivo;
- editar um arquivo.

Um erro clássico é abrir um arquivo para apenas ler e utilizar uma ferramenta que abre o arquivo em modo edição, isso leva ao risco²⁶ de falha no programa e naturalmente a corrupção do arquivo. Caso queira ler um arquivo utilize o comando cat, conforme exemplo abaixo.

```
usuario@debian:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/uucp
```

Outro uso para o comando cat é exibir o final de string de cada linha do arquivo, isso pois é comum a corrupção de arquivos quando se copia e cola texto provenientes de Internet, utilize para isso o parâmetro **-e**.

Parametro **-e**

Existe no final de cada string o caractere \$

```
usuario@debian:~$ cat /etc/passwd -e
root:x:0:0:root:/root:/bin/bash$
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin$
bin:x:2:2:bin:/bin:/usr/sbin/nologin$
sys:x:3:3:sys:/dev:/usr/sbin/nologin$
sync:x:4:65534:sync:/bin:/sync$
games:x:5:60:games:/usr/games:/usr/sbin/nologin$
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin$
```

Caso apareça M\$ no final da string então o arquivo possui marcação de arquivos MS-DOS e naturalmente não será lido corretamente pelo GNU/Linux

Veja na imagem abaixo um erro ao realizar o download de um script Linux escrito em um aplicativo em um Microsoft Window.

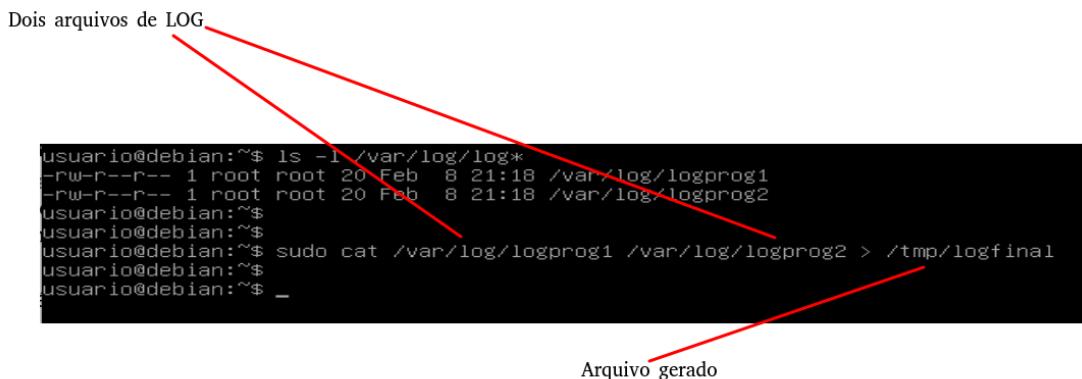
²⁶ Risco reduzido em sistemas de arquivos que utilizam formatação que usam journaling;

```
usuario@debian:~$ cat /tmp/layout.sh -e
#!/bin/bash^M$^M
setterm -term linux -back black -fore white --blink on -clear^M$^M
setterm --foreground black --background white --blink on^M$^M
clear^M$^M
^M$^M
usuario@debian:~$ _
```

Outro uso interessante do comando **cat** é concatenação de arquivos, imagine um cenário onde inúmeros arquivos de LOG são gerados por N sistemas, é interessante que ao enviar o estado da máquina para um administrador se junte estes arquivos de LOG. Na figura abaixo temos 2 logs no diretório de LOGs, do programa 1 e do programa 2.

```
usuario@debian:~$ ls -l /var/log/log*
-rw-r--r-- 1 root root 20 Feb 8 21:18 /var/log/logprog1
-rw-r--r-- 1 root root 20 Feb 8 21:18 /var/log/logprog2
usuario@debian:~$
```

Para concatenar os arquivos com **cat** é simples, basta informar os 2 arquivos de log e a saída que será um terceiro arquivo, conforme figura abaixo.



Em alguns casos é necessário se ler arquivo com a sequência de linhas invertidas, ou seja, ler e exibir de forma invertida (as linhas), utiliza-se o comando **tac** da mesma forma que se foi utilizado o comando **cat**, veja que **tac** é **cat** invertido.

Neste exemplo, o programa deverá abrir um arquivo e listar o texto no output, mas é preciso que se crie um arquivo chamado **teste.txt** no diretório **/tmp/**, escreva qualquer frase neste arquivo. Crie um arquivo chamado **exibir.cpp** e codifique o código conforme listagem abaixo, compile e execute.

```
1. #include <fstream>
2. #include <string>
3. #include <iostream>
4.
5. int main(int argc, char *argv[])
6. {
7.     if (argc != 2)
8.         exit(1);
9.     std::ifstream file( argv[1] );
10.    std::string str;
11.    while (std::getline(file, str))
```

```

12.         std::cout << str << std::endl;
13.
14. }
```

Verá que o resultado da execução será semelhante ao cat, agora você pode customizar este código de acordo com sua necessidade, lembre-se que o GNU/Linux é uma poderosa ferramenta que qualquer um pode expandir suas funcionalidades. Vamos exibir quais foram os últimos usuários adicionados no sistema utilizando **tac**, veja como na figura abaixo.

```

usuario@debian:~$ tac /etc/passwd
systemd-coredump:x:999:999:systemd Core Dumper:,
usuario:x:1000:1000:usuario,,,:/home/usuario:/b
messagebus:x:104:110::/nonexistent:/usr/sbin/no
systemd-resolve:x:103:104:systemd Resolver,,,:/r
systemd-network:x:102:103:systemd Network Manage
systemd-timesync:x:101:102:systemd Time Synchroni
_lapt:x:100:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/s
gnats:x:41:41:Gnats Bug-Reporting System (admin)
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr
backup:x:34:34:backup:/var/backups:/usr/sbin/no
www-data:x:33:33:www-data:/var/www:/usr/sbin/no
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nolog
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
games:x:5:60:games:/usr/games:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
root:x:0:0:root:/root:/bin/bash
usuario@debian:~$ _
```

Para realizar a visualização de um arquivo grande é necessário se utilizar o comando **more**, basta informar o comando **more** e como parâmetro o arquivo que se precisa visualizar conforme figura abaixo.

Ao digitar o comando **more** e o caminho do arquivo

O comando exibe algumas linhas e para exibir mais linhas basta pressionar **ENTER**

```

# Uncomment the next line to enable IPv4 forwarding
#net.ipv4.ip_forward=1

# Uncomment the next line to enable IPv6 forwarding
# Enabling this option disables IPv6 autoconfiguration
# based on Router Advertisements
#net.ipv6.conf.all.forwarding=1

#####
--More-- (49%)
```

O comando **more** é interativo e por isso precisa ir pressionando **ENTER** para visualizar mais linhas. Agora imagine que o arquivo seja grande e esteja em constante alteração por processos, um arquivo é clássico neste sentido, é o arquivo de log gerado do GNU/Linux: **/var/log/syslog**. Para acompanhar as mudanças nestes arquivos utiliza-se o comando **tail**, conforme exemplo abaixo.

Comando tail com os parametros -f e caminho do arquivo

```
usuario@debian:~$ sudo tail -f /var/log/syslog
[sudo] password for usuario:
Feb  8 20:31:14 debian kernel: [ 4385.784067] e1000: enp0s3 N
Feb  8 20:31:15 debian kernel: [ 4386.391926] usb 1-1: new fu
-pci
Feb  8 20:31:15 debian kernel: [ 4386.693833] usb 1-1: New USE
t=0021, bcdDevice= 1.00
Feb  8 20:31:15 debian kernel: [ 4386.693834] usb 1-1: New USE
rialNumber=0
Feb  8 20:31:15 debian kernel: [ 4386.693835] usb 1-1: Product
Feb  8 20:31:15 debian kernel: [ 4386.693836] usb 1-1: Manufac
Feb  8 20:31:15 debian kernel: [ 4386.705894] input: VirtualBo
00:00:06.0/usb1/1-1/1-1:1.0/0003:00EE:0021.0002/input/input9
Feb  8 20:31:15 debian kernel: [ 4386.706278] hid-generic 000
D v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-1/i
Feb  8 20:31:18 debian kernel: [ 4389.816297] e1000: enp0s3 N
ow Control: RX
Feb  8 21:17:01 debian CRON[552]: (root) CMD ( cd / && run-p
|
```

Cursor fica parado esperando a modificacao do arquivo, utilize Ctrl-c para sair.

Ultimas linhas do arquivo

Outra operação importante sobre grandes arquivos é a segmentação destes, podemos obter vários arquivos menores a partir de um grande arquivo com o uso do comando **split**.

Um grande arquivo de 8 linhas :)

Gerando arquivos menores de 4 linhas

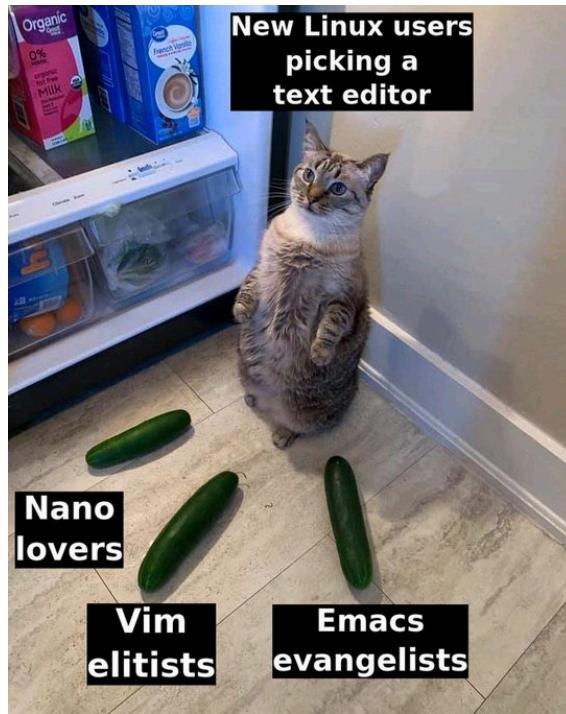
```
usuario@debian:~$ ls
umgrandearquivo
usuario@debian:~$ split -l 4 ./umgrandearquivo parte
usuario@debian:~$ ls
partea  parteab  umgrandearquivo
usuario@debian:~$
```

Arquivos gerados com o nome parte + uma letra

4.9 Editores de texto (nano, vim e vi)

Abrimos um arquivo para edição quando se tem 100% de certeza que o arquivo precisa ser editado, inclusive se utiliza os comandos anteriores para se obter esta certeza absoluta, dois programas são clássicos no GNU/Linux, são estes:

- vi, vim;
- nano.



O comando nano é simples, basta digitar o comando e o caminho do arquivo, conforme exemplo abaixo.

Comando nano e o nome do arquivo

```
usuario@debian:~$ nano /tmp/arquivoquenaoexiste
```

Veja aqui o nome do arquivo

Vai digitando o texto do arquivo

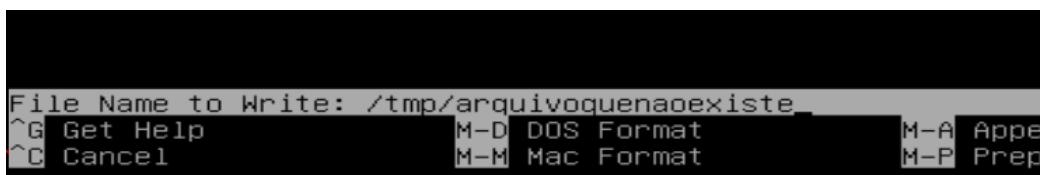
Opções, como salvar, fechar, etc..

GNU nano 3.2 /tmp/arquivoquenaoexiste

coloca aqui o texto, pode ir digitando...
mais linhas
e outras linhas

Get Help Exit Write Out Read File Where Is Replace Cut Text Uncut Text Justify To Spell Cur Pos Go To Line Undo Redo

Veja que nas opções você deve pressionar a tecla Ctrl da esquerda + a opção, no caso de salvar o arquivo Ctrl+o, o nano questiona sobre o nome do arquivo, pressione ENTER.



Para fechar utilize a opção **Ctrl+x**, leia as instruções do nano ao fechar o arquivo. Caso o arquivo não exista então ele cria, caso o arquivo exista ele abre para edição.

Você pode usar o editor **vi** para editar um arquivo existente ou criar um novo arquivo do zero. No exemplo abaixo estou abrindo um arquivo já existente:

1. **vi /etc/network/interfaces**

Você notará um til (~) em cada linha após o cursor, este til representa uma linha não utilizada. Se uma linha não começar com um til e parecer em branco, há um espaço, tabulação, nova linha ou algum outro caractere não visível presente.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp
~
```

Figura 4.

Ao trabalhar com o editor **vi**, geralmente encontramos os dois modos a seguir:

- **Modo de comando:** Este modo permite realizar tarefas administrativas como salvar os arquivos, executar os comandos, mover o cursor, cortar (arrancar) e colar as linhas ou palavras, bem como localizar e substituir. Neste modo, tudo o que você digita é interpretado como um comando sobre o arquivo;
- **Modo de inserção:** Este modo permite inserir texto no arquivo, tudo o que é digitado neste modo é interpretado como entrada e colocado no arquivo.

O comando **vi** sempre inicia no modo de comando, e para inserir texto, você deve estar no modo de inserção, basta digitar **i**, já para sair do modo de inserção, pressione a tecla **ESC**, que o levará de volta ao modo de comando. Se você não tiver certeza de qual modo está, pressione a tecla **ESC** duas vezes, isso o levará ao modo de comando. Você abre um arquivo usando o editor **vi**. Comece digitando alguns caracteres e depois vá para o modo de comando para entender a diferença.

O comando para sair do vi digite :**q** e caso tenha alterado o arquivo ele vai exibir uma mensagem, mas caso queira ignorar a mensagem e fechar sem salvar use :**q!** seguido de **ENTER**. Se o seu arquivo tiver sido modificado de alguma forma, o editor irá avisá-lo sobre isso e não permitirá que você saia, então se quer salvar use o comando :**w** seguido de **ENTER**. Você pode combinar o comando acima com o comando quit ou usar :**wq** e **ENTER**.

Para mover-se dentro de um arquivo sem afetar seu texto, você deve estar no modo de comando, as opções são:

- k** Move o cursor uma linha para cima
- j** Move o cursor uma linha para baixo
- h** Move o cursor para a esquerda uma posição de caractere
- l** Move o cursor para a posição de um caractere à direita

Para editar o arquivo, você precisa estar no modo de inserção. Existem muitas maneiras de entrar no modo de inserção a partir do modo de comando -

- i** Insere texto antes da localização atual do cursor
- I** Insere texto no início da linha atual
- a** Insere texto após a localização atual do cursor
- A** Insere texto no final da linha atual
- o** Cria uma nova linha para entrada de texto abaixo da localização do cursor
- O** Cria uma nova linha para entrada de texto acima da localização do cursor

Aqui está uma lista de comandos importantes, que podem ser usados para excluir caracteres e linhas em um arquivo aberto

- x** Exclui o caractere sob a localização do cursor
- X** Exclui o caractere antes da localização do cursor
- dw** Exclui da localização atual do cursor para a próxima palavra
- d^** Exclui da posição atual do cursor até o início da linha
- d\$** Exclui da posição atual do cursor até o final da linha
- D** Exclui da posição do cursor até o final da linha atual
- dd** Exclui a linha onde o cursor está

Como mencionado acima, a maioria dos comandos no **vi** podem ser precedida pelo número de vezes que você deseja que a ação ocorra. Por exemplo, **2x** exclui dois caracteres sobre a localização do cursor e **2dd** exclui duas linhas em que o cursor está.

Você também tem a capacidade de alterar caracteres, palavras ou linhas no **vi** sem excluí-los. Aqui estão os comandos relevantes:

- CC** Remove o conteúdo da linha, deixando você no modo de inserção.
- cw** Muda a palavra em que o cursor está do cursor para o final da palavra em minúscula.
- R** Substitui o caractere sob o cursor. vi retorna ao modo de comando após a substituição ser inserida.
- R** Substitui vários caracteres começando pelo caractere atualmente sob o cursor. Você deve usar Esc para interromper a substituição.
- s** Substitui o caractere atual pelo caractere digitado. Depois disso, você fica no modo de inserção.

S Exclui a linha em que o cursor está e a substitui pelo novo texto. Depois que o novo texto for inserido, o vi permanecerá no modo de inserção.

Você pode copiar linhas ou palavras de um lugar e colá-las em outro lugar usando os seguintes comandos:

- yy** Copia a linha atual.
- yw** Copia a palavra atual do caractere em que o cursor w minúsculo está, até o final da palavra.
- p** Coloca o texto copiado após o cursor.
- P** Coloca o texto arrancado antes do cursor.

O editor vi possui dois tipos de pesquisas:

- string
- character (regex)

Para uma pesquisa de string, utilize os comandos / e ?. Ao iniciar esses comandos você deve digitar a string específica a ser procurada.

Para pesquisa por regex devem ser precedidos por uma barra invertida \ para serem incluídos como parte da expressão regular, segue opções:

- ^** Pesquisa no início da linha (Use no início de uma expressão de pesquisa).
- .** Corresponde a um único caractere.
- *** Corresponde a zero ou mais do caractere anterior.
- \$** Fim da linha (Use no final da expressão de pesquisa).
- [** Inicia um conjunto de expressões correspondentes ou não correspondentes.
- <** Isso é colocado em uma expressão com escape de barra invertida para encontrar o final ou o início de uma palavra.
- >** Isso ajuda a ver a descrição do caractere '<' acima.

O **vi** tem a capacidade de executar comandos de dentro do editor, para executar um comando, você só precisa ir para o modo de comando e digitar :! seguido do comando. Por exemplo, se quiser verificar se existe um arquivo antes de tentar salvá-lo com esse nome, você pode digitar :! ls e você verá a saída de ls na tela.

Se você quiser especificar/declarar qualquer nome específico para o arquivo, poderá fazê-lo especificando-o após :w

4.10 Obtendo informações sobre o arquivo (file, stat, cksum, cmp)

Como no GNU/Linux inúmeros arquivos não possuem extensão é comum se precisar do arquivo file para se saber do que se trata o arquivo, vou utilizar o comando file em um diretório na e veja a figura abaixo.

```
usuario@debian:~$ file /var/log
/var/log: directory
usuario@debian:~$ _
```

Agora se executar o mesmo comando em um arquivo de script será exibido a informação que é um script executável.

```
usuario@debian:~$ file /etc/init.d/cron
/etc/init.d/cron: POSIX shell script, ASCII text executable
```

E se executar contra um binário deverá informar detalhes de compilação, conforme figura abaixo.

```
usuario@debian:~$ file /etc/aied/aied_32
/etc/aied/aied_32: ELF 32-bit LSB pie executable, Intel 80386, version 1 (SYSV), dynamically linked,
 interpreter /lib/ld-linux.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=b860ee0c7f15a21ba4ec9e8e13b11a0f
1a82b5c9, not stripped
```

O comando `file` consulta o header do arquivo que traz informações sobre seu tipo.

Enquanto no mundo Microsoft Windows é levado em consideração a extensão do arquivo no mundo GNU/Linux é o contexto do arquivo que revela o que é o arquivo.

Obtendo um código hash do arquivo antes da edição

Editando um arquivo com o nano, só foi trocado uma letra do arquivo /

```
usuario@debian:~$ cksum umgrandearquivo  
1315219495 42 umgrandearquivo
```

```
usuario@debian:~$ nano umgrandearquivo
```

Tamanho do arquivo

```
usuario@debian:~$ cksum umgrandearquivo  
1211290790 42 umgrandearquivo
```

Novo hash do arquivo, mas mesmo tamanho

Uma prática comum conforme já dito é antes da edição de um arquivo de configuração realizar uma cópia do mesmo, e caso a edição da tenha levado o sistema a algum estado indesejado ou o administrador volta o arquivo original ou analisa as linhas editadas, o comando `cmp` permite que se faça uma comparação entre 2 arquivos, e naturalmente se compararmos um arquivo antigo antes da edição e o arquivo após a edição obtemos uma lista de linhas que foram alteradas, conforme exemplo na figura abaixo.

```

usuario@debian:~$ ls -l
total 8
-rw-r--r-- 1 usuario usuario 42 Feb  8 22:47 umgrandearquivo
-rw-r--r-- 1 usuario usuario 42 Feb  8 22:47 umgrandearquivo.old
usuario@debian:~$ 
usuario@debian:~$ 
usuario@debian:~$ cmp umgrandearquivo umgrandearquivo.old
umgrandearquivo umgrandearquivo.old differ: byte 11, line 3
usuario@debian:~$ 
usuario@debian:~$ 

```

Dois arquivos, um e uma copia antiga realizada antes de uma possivel edicao

Foi realizada a edicao

Imprimindo no terminal informacoes de linhas editadas

4.11 Criando e removendo diretórios e arquivos (touch, rm, mkdir, rmdir)

O comando **touch** é um comando padrão usado no sistema operacional GNU/Linux para criar, alterar e modificar o timestamp de arquivos e também criar arquivos vazios (**Empty**). Basicamente, existem dois comandos diferentes para criar um arquivo no sistema Linux, que é o seguinte:

Comandos cat ou echo: É usado para criar o arquivo com conteúdo;

Comando touch: é usado para criar um arquivo sem nenhum conteúdo. Este comando pode ser usado quando o usuário não tem dados para armazenar no momento da criação do arquivo.

```

usuario@debian:~$ ls
usuario@debian:~$ 
usuario@debian:~$ touch umarquivo
usuario@debian:~$ 
usuario@debian:~$ ls -l
total 0
-rw-r--r-- 1 usuario usuario 0 Feb  9 01:04 umarquivo
usuario@debian:~$ 

```

O comando touch com o parâmetro **a** é usado apenas para alterar o tempo de acesso.

Chegou a hora de remover arquivos e diretórios, muita calma nessa hora, o comando **rm** tem essa função e remove cada arquivo especificado na linha de comando, mas saiba que por padrão, ele não remove diretórios sem o uso de parâmetros. Já o **rmdir** remove diretórios, mas o diretório deve estar vazio.

Quando **rm** é executado com o parâmetro **r**, ele exclui recursivamente quaisquer diretórios correspondentes, seus subdiretórios e todos os arquivos que eles contêm. O processo de remoção desvincula um nome de arquivo em um sistema de arquivos de seus dados associados na tabela **i-node** e naturalmente marca esse espaço no dispositivo de armazenamento como utilizável por gravações futuras. Em outras palavras, quando você remove um arquivo, os dados no arquivo não são alterados, mas não estão mais associados nas tabelas **i-node**.

Os dados em si não são destruídos, mas após serem desvinculados do **rm**, eles se tornam inacessíveis por meios normais, e com o passar do tempo o Sistema Operacional utiliza

partes destes espaços, então se remover um arquivo e desejar recuperar por programas, evite ficar salvando e criando novos arquivos, faça a recuperação imediatamente. Para remover um arquivo é simples, utilize o comando **rm** acrescido do caminho do arquivo, veja na figura abaixo.

```
usuario@debian:~$ ls -l
total 0
-rw-r--r-- 1 usuario usuario 0 Feb  9 01:04 umarquivo
usuario@debian:~$ rm umarquivo
usuario@debian:~$ ls -l
total 0
usuario@debian:~$
```

Na figura abaixo está sendo removido um diretório com 2 arquivos.

Comando ls exibe apenas 1 diretório

```
usuario@debian:~$ ls -l
total 4
drwxr-xr-x 2 usuario usuario 4096 Feb  9 01:15 diretorio
usuario@debian:~$ ls -l ./diretorio/
total 8
-rw-r--r-- 1 usuario usuario 10 Feb  9 01:15 arquivo1
-rw-r--r-- 1 usuario usuario 10 Feb  9 01:15 arquivo2
usuario@debian:~$ rm -r ./diretorio
usuario@debian:~$ ls -l
total 0
usuario@debian:~$
```

Nao existe mais o diretório

Removendo o diretório de forma recursiva

Neste unico diretório temos 2 arquivos

No exemplo acima foi removido todo o conteúdo do diretório desejado bem como os arquivos, mas caso queira remover o conteúdo do diretório sem remover o diretório (para preservar o esquema de permissão de acesso) utilize */** conforme exemplo.

```
usuario@debian:~$ rm -r ./diretorio/*
```

O comando **rmdir** remove cada diretório especificado na linha de comando, se eles estiverem vazios, ou seja, cada diretório removido não deve conter arquivos ou diretórios internos. Se algum diretório especificado não estiver vazio, **rmdir** não o removerá e continuará a tentar remover quaisquer outros diretórios que você especificou. Os diretórios são processados na ordem em que você os especifica na linha de comando, da esquerda para a direita.

O comando **mkdir** permite ao usuário criar diretórios, este comando pode criar vários diretórios de uma vez, bem como definir as permissões para os diretórios. É importante

observar que o usuário que executa este comando deve ter permissões suficientes para criar um diretório no diretório pai, ou ele pode receber um erro de 'permissão negada'.

```
usuario@debian:~$ mkdir ./diretorio
usuario@debian:~$
```

O parâmetro **p** informa ao comando que ele deve criar diretórios pai conforme necessário, conforme exemplo abaixo.

```
usuario@debian:~$ 
usuario@debian:~$ mkdir -p ./diretorio/subdir1/subdir2
usuario@debian:~$
```

4.12 Permissão de acesso e dono (chmod, chown, chgrp, umask)

O modelo de permissão de acesso de arquivos é muito simples, e esta simplicidade associada a uma boa organização de diretórios e arquivos garantem ao GNU/Linux o status de sistema seguro.

Tipo de arquivo	Referencias na tabela i-node		Modificado em	Nome
	Permissao	Dono		

```
usuario@debian:~/Desktop$ ls -l
total 60
lrwxrwxrwx  1 root root   7 Nov 20 12:36 bin -> usr/bin
drwxr-xr-x  3 root root 4096 Nov 20 12:39 boot
drwxr-xr-x 16 root root 3160 Feb  8 19:18 dev
drwxr-xr-x 70 root root 4096 Feb  8 19:18 etc
drwxr-xr-x  3 root root 4096 Nov 20 12:39 home
lrwxrwxrwx  1 root root   29 Nov 20 12:37 initrd.img -> boot/initrd.img-4.19.0-12-
lrwxrwxrwx  1 root root   29 Nov 20 12:36 initrd.img.old -> boot/initrd.img-4.19.0-
lrwxrwxrwx  1 root root    7 Nov 20 12:36 lib -> usr/lib
lrwxrwxrwx  1 root root    9 Nov 20 12:36 lib64 -> usr/lib64
```

Arquivos e diretórios tem 3 tipos de permissão (PADRÃO):

Read (r): permissão de leitura;

Write (w): permissão de escrita;

Execute (x): permissão de executar se for um arquivo e permissão de listagem caso seja um diretório;

Cada arquivo e diretório define seu controle de acesso por 3 classes:

User (u): é o proprietário do arquivo.

Group (g): é o grupo dono do arquivo, e dentro do grupo pode conter vários usuários.

Others (o): são todos os outros usuários que não se enquadram nos dois casos acima.



Outra forma de trabalhar com permissões é usando o modo octal que nada mais é do que a representação de sequências de letras (rwx) com números, ele é baseado no seguinte esquema:

0	Nenhuma permissão de acesso	-
1	Permissão de execução	x
2	Permissão de gravação	w
3	Permissão de gravação e execução	wx
4	Permissão de leitura	r
5	Permissão de leitura e execução	rx
6	- Permissão de leitura e gravação	rw
7	- Permissão de leitura, gravação e execução	rwx

Em sistemas operacionais GNU/Linux, o comando chmod é usado para alterar o modo de acesso de um arquivo.

O grupo que possui este arquivo pode somente ler
e os outros tambem

```
usuario@debian:~$ ls -l
total 8
-rw-r--r-- 1 usuario usuario 4 Feb  9 01:46 arquivo
-rw-r--r-- 1 usuario usuario 25 Feb  9 04:02 script.sh
usuario@debian:~$
```

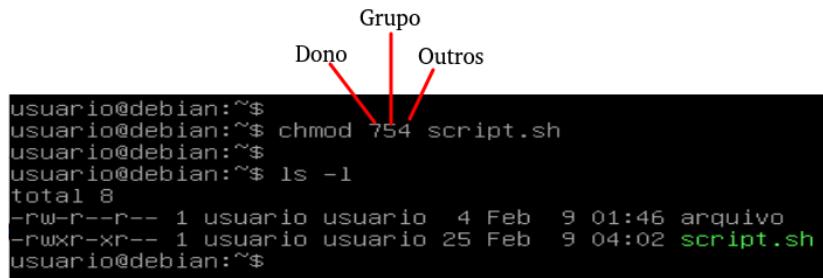
O dono pode ler e escrever

Para dar permissão de execução para este script, utiliza-se o comando chmod conforme exemplo da figura abaixo.

```
Adiciona
          |
  Usuario e Grupo  Execucao
          |
usuario@debian:~$ 
usuario@debian:~$ chmod ug+x script.sh
usuario@debian:~$ 
usuario@debian:~$ 
usuario@debian:~$ ls -l
total 8
-rw-r--r-- 1 usuario usuario 4 Feb  9 01:46 arquivo
-rwxr-xr-- 1 usuario usuario 25 Feb  9 04:02 script.sh
usuario@debian:~$ _
```

Agora o dono do arquivo bem como o grupo a qual o arquivo pertence
podem executar o script

O exemplo acima pode ser realizado utilizando a forma numérica conforme o exemplo
abaixo, ambos produzem o mesmo resultado.



```

usuario@debian:~$ 
usuario@debian:~$ chmod 754 script.sh
usuario@debian:~$ 
usuario@debian:~$ ls -l
total 8
-rw-r--r-- 1 usuario usuario 4 Feb  9 01:46 arquivo
-rwxr-xr-- 1 usuario usuario 25 Feb  9 04:02 script.sh
usuario@debian:~$ 

```

Diferentes usuários no sistema operacional têm propriedade e permissão para garantir que os arquivos sejam seguros e colocam restrições sobre quem pode modificar o conteúdo dos arquivos. No próximo exemplo, o programa vai escrever no output os caracteres relacionados à permissão de acesso, para o dono do arquivo, o grupo dono do arquivo e os outros, para isso importa-se a biblioteca `filesystem` para uso do `perms`. Crie um novo arquivo `exibirperm.cpp` em `/tmp`, digite a listagem abaixo e compile com `g++`.

```

1.
2. #include <filesystem>
3. #include <fstream>
4. #include <iostream>
5.
6. void demo_perms(std::filesystem::perms p)
7. {
8.     using std::filesystem::perms;
9.     auto show = [=](char op, perms perm)
10.    {
11.        std::cout << (perm::none == (perm & p) ? '-' : op);
12.    };
13.
14.    show('r', perm::owner_read);
15.    show('w', perm::owner_write);
16.    show('x', perm::owner_exec);
17.    show('r', perm::group_read);
18.    show('w', perm::group_write);
19.    show('x', perm::group_exec);
20.    show('r', perm::others_read);
21.    show('w', perm::others_write);
22.    show('x', perm::others_exec);
23.    std::cout << '\n';
24. }
25.
26. int main(int argc, char *argv[])
27. {
28.     if (argc != 2) exit(1);
29.     std::cout << "Permissão do arquivo: ";
30.     demo_perms(std::filesystem::status(argv[1]).permissions());
31. }

```

Exemplo de uso:

```

1. g++ -o exibirperm exibirperm.cpp
2. ./exibirperm /etc/passwd

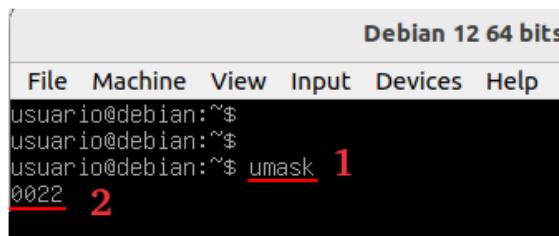
```

Por padrão, quando você cria um arquivo como um usuário normal, ele recebe as permissões de **rw-r--r--**. Você pode usar o comando **umask** (**stands for user mask**) para determinar as permissões padrão para arquivos recém-criados.

O umask é o valor subtraído das permissões 666 (rw-rw-rw-) ao criar novos arquivos ou de 777 (rwxrwxrwx) ao criar novos diretórios. Por exemplo, se o umask padrão for **022**, novos arquivos serão criados com as permissões **644** (rw-r--r--) e novos diretórios com as permissões **755** (rwxr-xr-x). Para exibir o valor atual de umask, execute o comando umask sem nenhuma opção:

```
1. umask
```

Em um GNU/Debian 12 o resultado do umask que será retornado é **0022**, quatro dígitos, pois nesta versão já se aplica por default as permissões especiais que serão explicadas.



```
Debian 12 64 bits
File Machine View Input Devices Help
usuario@debian:~$ umask 1
0022 2
```

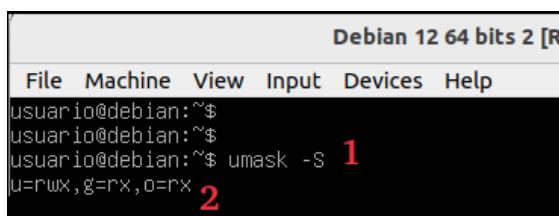
Onde:

1. Comando umask;
2. Umask padrão de um Debian 12;

Caso queira uma descrição mais humana, utilize o parâmetro **-S**.

```
1. umask -S
```

No output dá para ver o cálculo para arquivo.



```
Debian 12 64 bits 2 [R]
File Machine View Input Devices Help
usuario@debian:~$ umask -S 1
u=rwx,g=rx,o=rx 2
```

Onde:

1. Comando umask com parâmetro **-S**;
2. Versão humana para 755.

Para exemplificar, ao criar um diretório novo e listar com ls, é possível ver o cálculo **0777 - 0022**, resultando em **0755**, conforme figura abaixo.

```

Debian 12 64 bits 2 [Running] - Oracle V
File Machine View Input Devices Help
usuario@debian:/tmp$ mkdir diretorio 1
usuario@debian:/tmp$ ls -l 2
total 12
drwxr-xr-x 2 usuario usuario 4096 Oct 14 20:38 diretorio 3

```

Onde:

1. Comando mkdir criando um novo diretório;
2. Exibindo com ls a permissão de acesso;
3. A permissão **0755** resultante.

Já para arquivos, ao subtrair **0022** de **0666** terá como resultado **0644**, conforme imagem abaixo.

```

Debian 12 64 bits 2 [Running] - Oracle V
File Machine View Input Devices Help
usuario@debian:/tmp$ touch arquivo.txt 1
usuario@debian:/tmp$ ls -l 2
total 8
-rw-r--r-- 1 usuario usuario 0 Oct 14 20:41 arquivo.txt 3

```

Onde:

1. Comando touch criando um arquivo vazio;
2. Comando ls exibindo a permissão de acesso do arquivo;
3. A permissão **0644** do novo arquivo.

Este valor por padrão de umask está definido no arquivo **/etc/login.defs** do Debian GNU/Linux, conforme imagem abaixo.

```

Debian 12 64 bits 2 [Running] - Oracle V
File Machine View Input Devices Help
# used as group permissions, e. g. 022 will become 002.
#
# Prefix these values with "0" to get octal, "0x" to get hex.
#
ERASECHAR      0177
KILLCHAR       025
UMASK          022
#
# HOME_MODE is used by useradd(8) and newusers(8) to set the
# home directories.

```

Cada usuário possui algumas propriedades associadas a ele, como um ID do usuário e um diretório inicial. Podemos adicionar usuários a um grupo para facilitar o processo de gerenciamento de usuários. Um grupo pode ter zero ou mais usuários.

O dono do arquivo se chama: usuario

O comando chown altera o dono para root

```
usuario@debian:~$ ls -l
total 8
-rw-r--r-- 1 usuario usuario 4 Feb  9 01:46 arquivo
-rwxr-xr-- 1 usuario usuario 25 Feb  9 04:02 script.sh
usuario@debian:~$ 
usuario@debian:~$ sudo chown root ./script.sh
[sudo] password for usuario:
usuario@debian:~$ 
usuario@debian:~$ ls -l
total 8
-rw-r--r-- 1 usuario usuario 4 Feb  9 01:46 arquivo
-rwxr-xr-- 1 root      usuario 25 Feb  9 04:02 script.sh
usuario@debian:~$
```

Novo usuario dono do arquivo agora e root

O comando chgrp no GNU/Linux é usado para alterar a propriedade do grupo de um arquivo ou diretório. Todos os arquivos pertencem a um proprietário e a um grupo e você pode definir o proprietário usando o comando **chown** e o grupo pelo comando **chgrp**.

O grupo a qual este arquivo pertence e o grupo usuario

Mudar o grupo de usuario para o grupo users

```
usuario@debian:~$ ls -l
total 8
-rw-r--r-- 1 usuario usuario 4 Feb  9 01:46 arquivo
-rwxr-xr-- 1 root      usuario 25 Feb  9 04:02 script.sh
usuario@debian:~$ 
usuario@debian:~$ sudo chgrp users ./script.sh
usuario@debian:~$ 
usuario@debian:~$ ls -l
total 8
-rw-r--r-- 1 usuario usuario 4 Feb  9 01:46 arquivo
-rwxr-xr-- 1 root      users    25 Feb  9 04:02 script.sh
usuario@debian:~$ 
usuario@debian:~$ _
```

Novo grupo

4.13 SetUID, SetGID e Sticky Bits

Conforme explicado, o GNU/Linux usa uma combinação de bits para armazenar as permissões de um arquivo, podemos alterar as permissões usando o comando chmod, que essencialmente altera os caracteres 'r', 'w' e 'x' associados ao arquivo.

Além disso, a propriedade dos arquivos também depende do uid (ID do usuário) e do gid (ID do grupo) do criador, conforme já demonstrado no tópico anterior.

4.13.1 Bit SetUID

O sistema possui um bit presente para arquivos que têm permissões de execução e o bit SetUID indica simplesmente que, ao executar o executável, ele definirá suas permissões para as do usuário dono (proprietário), em vez de defini-lo para o usuário que executou.

Então neste exemplo um usuário chamado **usuario** pode executar um executável do root como se fosse root sem o auxílio do comando sudo ou de estar na sessão do root.

1. `chmod u+s path_do_arquivo`

Utiliza-se para isso os parâmetros u+s no comando chmod conforme exemplo abaixo. Neste exemplo está sendo adicionado a permissão **para fins de demonstração**, não sendo indicado para ser realizado em servidores em produção.

systemctl pode ser executado por qualquer usuário

```
usuario@debian:~$ ls -l /bin/systemctl
-rwxr-xr-x 1 root root 895792 Apr 27 2020 /bin/systemctl
usuario@debian:~$ systemctl restart cron.service
Failed to restart cron.service: Access denied
See system logs and 'systemctl status cron.service' for details.
usuario@debian:~$ 
usuario@debian:~$ 
usuario@debian:~$ sudo chmod u+s /bin/systemctl
usuario@debian:~$ 
usuario@debian:~$ ls -l /bin/systemctl
-rwsr-xr-x 1 root root 895792 Apr 27 2020 /bin/systemctl
usuario@debian:~$ 
usuario@debian:~$ systemctl restart cron
usuario@debian:~$ 
usuario@debian:~$ _
```

Vamos ativar o bit SetUID

Mas quando o systemctl altera um arquivo do root ocorre um erro para quem não é o root

Agora, mesmo que um usuário qualquer utilize o systemctl para alterar algum arquivo do root, sucesso

4.12.2 Bit SetGID

O bit SetGID modifica tanto os arquivos quanto os diretórios, quando usado em um arquivo, ele executa com os privilégios do usuário que o possui ao invés de executar com aqueles do grupo do usuário que o executou.

Quando o bit é definido para um diretório, o conjunto de arquivos nesse diretório terá o mesmo grupo que o grupo do diretório pai, e não o do usuário que criou esses arquivos. Isso é usado para compartilhamento de arquivos, pois agora eles podem ser modificados por todos os usuários que fazem parte do grupo do diretório pai. Utiliza-se para isso os parâmetros g+s no comando chmod, a operação é semelhante ao exemplo do Bit SetUID.

1. `chmod g+s path_do_arquivo`

4.13.3 Bit Sticky

O sticky bit foi inicialmente introduzido para 'colar' um segmento de texto de um programa executável no espaço de troca, mesmo depois de o programa ter concluído a execução, para acelerar as execuções subsequentes do mesmo programa, mas no entanto, hoje em dia, significa algo totalmente diferente.

Quando um diretório tem o sticky bit definido, seus arquivos podem ser excluídos ou renomeados apenas pelo proprietário do arquivo, pelo proprietário do diretório e pelo usuário root. Um exemplo deste uso é o diretório /tmp onde qualquer processo sendo executado por qualquer usuário pode ser usado para armazenar arquivos.

```
1. chmod o+t path_do_diretorio
```

Utiliza-se para isso os parâmetros +t no comando chmod.

```
usuario@debian:~$ ls -l /
total 60
lrwxrwxrwx 1 root root 7 Nov 20 12:36 bin -> usr/bin
drwxr-xr-x 3 root root 4096 Nov 20 12:39 boot
drwxr-xr-x 16 root root 3160 Feb 9 01:47 dev
drwxr-xr-x 70 root root 4096 Feb 9 01:48 etc
drwxr-xr-x 3 root root 4096 Nov 20 12:39 home
lrwxrwxrwx 1 root root 29 Nov 20 12:37 initrd.img -> boot/i
lrwxrwxrwx 1 root root 29 Nov 20 12:36 initrd.img.old -> bo
lrwxrwxrwx 1 root root 7 Nov 20 12:36 lib -> usr/lib
lrwxrwxrwx 1 root root 9 Nov 20 12:36 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 Nov 20 12:36 lib32 -> usr/lib32
drwx----- 2 root root 16384 Nov 20 12:36 lost+found
drwxr-xr-x 3 root root 4096 Nov 20 12:36 media
drwxr-xr-x 2 root root 4096 Nov 20 12:36 mnt
drwxr-xr-x 2 root root 4096 Nov 20 12:36 opt
dr-xr-xr-x 72 root root 0 Feb 9 03:42 proc
drwx----- 4 root root 4096 Feb 8 11:45 root
drwxr-xr-x 15 root root 460 Feb 9 01:48 run
lrwxrwxrwx 1 root root 8 Nov 20 12:36 sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Nov 20 12:36 srv
dr-xr-xr-x 12 root root 0 Feb 9 05:16 sys
drwxrwxrwt 8 root root 4096 Feb 9 01:47 tmp
drwxr-xr-x 12 root root 4096 Nov 20 12:36 usr
drwxr-xr-x 11 root root 4096 Nov 20 12:36 var
lrwxrwxrwx 1 root root 26 Nov 20 12:37 vmlinuz -> boot/vmlinuz
lrwxrwxrwx 1 root root 26 Nov 20 12:36 vmlinuz.old -> boot/vmlinuz.old
usuario@debian:~$
```

Bit ativo em /tmp

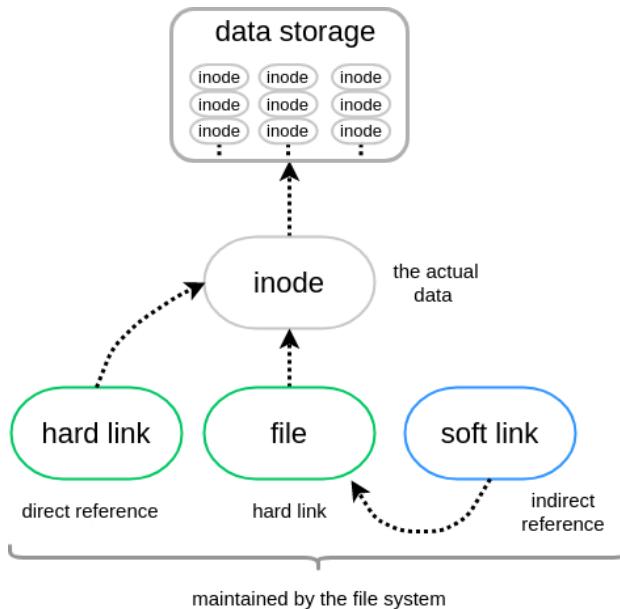
4.14 Link simbólico e Hard Link (ln e ln -s)

O comando **ln** é usado para criar links, antes de entrar na aplicação do comando **ln** em detalhes vamos entender o que é link para uma compreensão clara do comando **ln**.

Um link é um ponteiro para um arquivo no sistema secundário de armazenamento, como ponteiros em qualquer linguagem de programação, os links são ponteiros que apontam para um arquivo ou diretório onde estiver. Criar links é uma espécie de atalho para acessar um arquivo.

Existem dois tipos de links:

- **Soft Link** ou também conhecido com Symbolic links;
- **Hard Links**.



4.14.1 Hard Link

É o apontamento direto do para o arquivo no sistema de arquivos para um arquivo no sistema secundário de armazenamento, é permitido que um mesmo arquivo sistema secundário de armazenamento aparece inúmeras vezes no sistema de arquivos (na RAM) porém em pontos diferentes na árvore, todos os Hard Links de um arquivo possuem o mesmo número **i-node**.

Por exemplo, se tivermos um arquivo **a.txt** e se criarmos um Hard Link para o arquivo em outra posição do sistema de arquivos, em seguida, excluímos o arquivo **a.txt**, ainda podemos acessar o arquivo usando o Hard Link. Em um cenário de dois diretórios e 1 arquivo conforme exemplo abaixo será criado um Hard Link deste arquivo no segundo diretório.

```

usuario@debian:~$ pwd
/home/usuario
usuario@debian:~$ tree
.
└── diretorio1
    └── arquivo
└── diretorio2

2 directories, 1 file
usuario@debian:~$ 

```

i-node do arquivo no Sistema de Arquivos

Listando o diretório1 que contém o arquivo

Fazendo o Hard Link com o comando ln e criando no diretório2 com outro nome

```
usuario@debian:~$ ls -li ./diretorio1
total 4
667667 -rw-r--r-- 1 usuario usuario 25 Feb  9 11:53 arquivo
usuario@debian:~$ 
usuario@debian:~$ ln ./diretorio1/arquivo ./diretorio2/hardArquivo
usuario@debian:~$ 
usuario@debian:~$ ls -li ./diretorio2
total 4
667667 -rw-r--r-- 2 usuario usuario 25 Feb  9 11:53 hardArquivo
usuario@debian:~$
```

Repare que o Hard Link possui o mesmo i-node

Agora será mudado a permissão de acesso ao arquivo **hardArquivo**, será permitido que o grupo altere todas as referências ao **i-node** alterado receberão o impacto.

Exibindo a permissão para o arquivo hardArquivo

Alterando a permissão de acesso do arquivo hardArquivo

```
usuario@debian:~$ 
usuario@debian:~$ ls -l ./diretorio2
total 4
-rw-r--r-- 2 usuario usuario 25 Feb  9 11:53 hardArquivo
usuario@debian:~$ 
usuario@debian:~$ chmod g+w ./diretorio2/hardArquivo
usuario@debian:~$ 
usuario@debian:~$ ls -l ./diretorio2
total 4
-rw-rw-r-- 2 usuario usuario 25 Feb  9 11:53 hardArquivo
usuario@debian:~$ 
usuario@debian:~$ ls -l ./diretorio1
total 4
-rw-rw-r-- 2 usuario usuario 25 Feb  9 11:53 arquivo
usuario@debian:~$ 
usuario@debian:~$ -
```

Veja que a permissão mudou tanto para o hardArquivo (alvo do chmod) e também para o arquivo, isso ocorre pois ambos apontam para o mesmo i-node

4.14.2 Symbolic links

Um link simbólico é semelhante ao recurso de atalho de arquivo usado nos sistemas operacionais em geral. Cada arquivo link contém um valor **i-node** separado que aponta para o arquivo original. Da mesma forma que os Hard Links, todas as alterações nos dados de um dos arquivos são refletidas no outro mas a alteração é feita no arquivo e não no link.

Por exemplo, se tivermos um arquivo **a.txt** e criarmos um Symbolic Link do arquivo e, em seguida, excluir o arquivo, não podemos acessar o arquivo por meio do link. Em um cenário

de dois diretórios e 1 arquivo conforme exemplo abaixo será criado um Link Simbólico deste arquivo no segundo diretório.

```
usuario@debian:~$ pwd
/home/usuario
usuario@debian:~$ tree
.
└── diretorio1
    └── arquivo
└── diretorio2

2 directories, 1 file
usuario@debian:~$
```

Figura 4.

Para isso basta adicionar o parâmetro **s** ao comando **ln**, conforme figura abaixo.

Exibindo a arvore para entender a organizacao

Criando um link simbolico do arquivo do diretorio2

```
usuario@debian:~$ tree
.
└── diretorio1
    └── arquivo
└── diretorio2

2 directories, 1 file
usuario@debian:~$ ln -s ./diretorio1/arquivo ./diretorio2/linksimbolico
usuario@debian:~$ ls -li ./diretorio1
total 4
667667 -rw-rw-r-- 1 usuario usuario 25 Feb  9 11:53 arquivo
usuario@debian:~$ usuario@debian:~$ ls -li ./diretorio2
total 0
667666 lrwxrwxrwx 1 usuario usuario 20 Feb  9 12:07 linksimbolico -> ..
usuario@debian:~$ usuario@debian:~$
```

O i-node é diferente

Veja que a permissao de acesso aos arquivos é diferente

Uma observação importante é que o Link Simbólico tem permissão total, então pode ser acessado e executado, mas quando o arquivo original é carregado o Sistema Operacional valida a permissão de acesso ao arquivo, e naturalmente se o usuário não tiver permissão para acessar o arquivo será negado.

4.15 Práticas da aula de Sistema de Arquivos

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

4.15.1 Prática 8e9c361c 01: Criando diretórios e arquivos no GNU/Linux

Nesta prática o aluno deve ter a expertise de criar diretórios e arquivos, mas antes o aluno deve navegar até o diretório do usuário e seguir o roteiro:

1. Tenha certeza que o diretório não possui arquivos do usuário;

```
usuario@debian:~$ ls
usuario@debian:~$ _
```

Figura 4.

2. Criar dentro do diretório do usuário os seguintes diretórios (usando o comando `mkdir`): diretório1, diretório2, diretório3 conforme figura:

```
usuario@debian:~$ pwd
/home/usuario
usuario@debian:~$ tree
.
└── diretório1
    ├── diretório2
    └── diretório3

3 directories, 0 files
usuario@debian:~$ _
```

Figura 4.

3. Dentro do diretório1 crie um arquivo chamado arquivo1 com o conteúdo “111”;
4. Dentro do diretório1 crie um arquivo chamado arquivo2 com o conteúdo “222”;
5. Dentro do diretório1 crie um arquivo chamado arquivo3 com o conteúdo “333”;
6. Dentro do diretório2 crie um arquivo chamado arquivo4 com o conteúdo “444”;
7. Dentro do diretório2 crie um arquivo chamado arquivo5 com o conteúdo “555”;
8. Dentro do diretório2 crie um arquivo chamado arquivo6 com o conteúdo “666”;

Execute o robô de validação aied conforme comando abaixo.

```
sudo aied validate 8e9c361c checkpoint01
```

Veja a prática validada na figura abaixo.

```
usuario@debian:~$ sudo aied validar 8e9c361c checkpoint01
Ação que será executada: validar

Prática: Prática da aula de sistemas de arquivos - Criando arquivos
Será enviado o OUTPUT do comando: ls -l /home/usuario/
Será enviado o OUTPUT do comando: ls -l /home/usuario/diretório1/
Será enviado o OUTPUT do comando: ls -l /home/usuario/diretório2/

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: ls -l /home/usuario/
1.1   Validar por text      diretório1
1.2   Validar por text      diretório2
1.3   Validar por text      diretório3

2 - Comando: ls -l /home/usuario/diretório1/
2.1   Validar por text      arquivo1
2.2   Validar por text      arquivo2
2.3   Validar por text      arquivo3

3 - Comando: ls -l /home/usuario/diretório2/
3.1   Validar por text      arquivo4
3.2   Validar por text      arquivo5
```

4.15.2 Prática 8e9c361c 02: Copiando arquivo, removendo arquivo e movendo arquivo

Nesta prática o aluno deve utilizar os comandos `cp`, `rm` e `mv` para manipular os arquivos no Sistema de Arquivos, partindo da prática anterior o aluno deve:

1. Copiar o arquivo arquivo1 que está em diretório1 para o diretório3;

2. Remover o arquivo arquivo4 que está em diretório2;
3. Mover o arquivo arquivo5 que está em diretório2 para o diretório3;

Execute o robô de validação aied conforme comando abaixo.

```
sudo aied validate 8e9c361c checkpoint02
```

Veja a prática validada na figura abaixo.

```
usuario@debian:~$ 
usuario@debian:~$ sudo aied validar 8e9c361c checkpoint02
Ação que será executada: validar

Prática: Prática da aula de sistemas de arquivos - Copiando, movendo e removendo
Será enviado o OUTPUT do comando: ls -l /home/usuario/diretorio1/
Será enviado o OUTPUT do comando: ls -l /home/usuario/diretorio2/
Será enviado o OUTPUT do comando: ls -l /home/usuario/diretorio3/

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: ls -l /home/usuario/diretorio1/
1.1   Validar por text      arquivo1
1.2   Validar por text      arquivo2
1.3   Validar por text      arquivo3

2 - Comando: ls -l /home/usuario/diretorio2/
2.1   Validar por text      arquivo4
2.2   Validar por text      arquivo5

3 - Comando: ls -l /home/usuario/diretorio3/
3.1   Validar por text      arquivo1
3.2   Validar por text      arquivo5

usuario@debian:~$
```

4.15.3 Prática 8e9c361c 03: Criar links

Nesta prática o aluno deve utilizar o comando ln para criar um link simbólico e um hard link, partindo da prática anterior o aluno deve:

1. Criar um link simbólico com o nome arquivo3 no diretório3 apontando para o arquivo arquivo3 no diretório1;
2. Criar um Hard Link com o nome arquivo6 no diretório3 que aponta para o arquivo arquivo6 do diretório2;

O resultado final do diretório 3 deve ser semelhante:

```
usuario@debian:~$ ls -li ./diretorio3
total 12
792398 -rw-r--r-- 1 usuario usuario 4 Feb  9 14:49 arquivo1
792399 lrwxrwxrwx 1 usuario usuario 21 Feb  9 14:54 arquivo3 -> ./diretorio1/arquivo3
667670 -rw-r--r-- 1 usuario usuario 4 Feb  9 12:34 arquivo5
667671 -rw-r--r-- 2 usuario usuario 4 Feb  9 12:47 arquivo6
usuario@debian:~$
```

Execute o robô de validação aied conforme comando abaixo.

```
sudo aied validate 8e9c361c checkpoint03
```

Veja a prática validada na figura abaixo.

```
usuario@debian:~$ sudo aied validar 8e9c361c checkpoint03
Ação que será executada: validar

Prática: Prática da aula de sistemas de arquivos - Links
Será enviado o OUTPUT do comando: ls -l /home/usuario/diretorio3/

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: ls -l /home/usuario/diretorio3/
1.1   Validar por text      arquivo1
1.2   Validar por text      arquivo3
1.3   Validar por text      arquivo5
1.4   Validar por text      arquivo6
1.5   Validar por text      ./diretorio1/arquivo3

usuario@debian:~$
```

4.15.4 Prática 8e9c361c 04: Diretório e arquivos com permissão especial

Nesta prática o aluno deve utilizar o aprendizado sobre permissões especiais para customizar as permissões de arquivos do usuário (entenda-se como ~):

1. Criar um diretório chamado tmp no diretório do usuário;
2. Criar um diretório chamado bin no diretório do usuário;
3. Para o diretório tmp criado, ative a permissão especial Stick Bit;
4. Copie o arquivo binário copiar (feito neste capítulo) para dentro do diretório bin;
5. Permita que somente o dono do arquivo possa executar, mas ninguém;
6. Dê a permissão especial SetUID para o programa /etc/aied/aied_64;

Execute o robô de validação aied conforme comando abaixo.

```
sudo aied validate 8e9c361c checkpoint04
```

Veja a prática validada na figura abaixo.

```
usuario@debian:~$ sudo aied validar 8e9c361c checkpoint04
Ação que será executada: validar

Prática: Prática da aula de sistemas de arquivos - CHMOD
Será enviado o OUTPUT do comando: ls -l /home/usuario/
Será enviado o OUTPUT do comando: ls -l /home/usuario/bin/
Será enviado o OUTPUT do comando: ls -l /etc/aied/

Deseja continuar? (s para SIM) s
s1 - Comando: ls -l /home/usuario/
1.1   Validar por regex      /drwxr-xr-x(.*)bin/
1.2   Validar por regex      /drwxrwxrwt(.*)tmp/
2 - Comando: ls -l /home/usuario/bin/
2.1   Validar por regex      /-rwxr-xr-x(.*)copiar/
3 - Comando: ls -l /etc/aied/
3.1   Validar por regex      /-rwsr-xr-x(.*)aied_64/

Total de acertos: 4 do total de 4 validações equivalente a 100%.
Identificação: 65cie5ide3
AIED v(8)
```

Adicionar questões de segurança

5 Usuários GNU/Linux

Sistemas multiusuário são sistemas que permitem o acesso de vários usuários a um computador e inclusive simultaneamente. Um exemplo é um sistema deste tipo é o Unix, em que vários usuários remotos têm acesso ao prompt do shell ao mesmo tempo. Outro exemplo usa várias sessões X Window espalhadas por vários terminais alimentados por uma única máquina. Os sistemas de gerenciamento são implicitamente projetados para serem usados por vários usuários, normalmente um administrador de sistema ou mais e uma comunidade de usuários finais.



5.2 Multiusuário e Multitarefas

UNIX é um Sistema Operacional multiusuário, onde cada usuário usa um terminal diferente, chamados de **TTY**. Um terminal consiste em um teclado e um monitor e está conectado ao computador principal conhecido como Computador de Host. Os recursos, como Hard Disk, memória, CPU (Processador), impressora etc, são acessíveis a todos os usuários e uma única máquina host suporta vários terminais. Temos as seguintes observações sobre a concorrência de processos e usuários:

- O compartilhamento de tempo refere-se à alocação de recursos de computador de maneira dependente do tempo;
- Cada usuário trabalha em um terminal e cada terminal fornece acesso direto à CPU (processador) no Computador Host;
- Cada programa de usuário recebe um curto período de tempo da CPU, um por um;
- A velocidade da CPU é tão rápida que um usuário tem a ilusão de que ele sozinho usando o computador;
- Uma razão para usar o compartilhamento de tempo é que não é economicamente viável permitir que um único usuário envolva um computador grande de forma interativa;
- Mesmo que os vários usuários estejam usando o mesmo sistema de computador ao mesmo tempo, um único sistema de CPU pode executar apenas um conjunto de instruções por vez;
- Com um sistema de compartilhamento de tempo, apenas um programa pode estar no controle da CPU a qualquer momento.

Já o fato de ter a capacidade de multiprocessamento (por pseudo paralelismo), diz ao fato de que o processador não precisa processar todo o processo para com o término de um processo iniciar outro, e assim sucessivamente. Já computadores modernos com mais de um núcleo podem até processar um processo até seu fim sem o uso de pseudo-paralelismo.

Segundo Tanenbaum os processadores inicialmente partiram de um pseudo-paralelismo para o processamento distribuído, isso por ter apenas um núcleo e posteriormente partindo

para um multiprocessamento em paralelo como a entrada de hardwares que permitiram isso.

GNU/Linux é um sistema operacional multitarefa, ou seja, permite o multiprocessamento em paralelo e onde um processo do kernel chamado Escalonador mantém registro de todos os programas em execução e aloca o tempo do processador de acordo, efetivamente executando vários programas simultaneamente e por isso tem a possibilidade de operação

Cada processo é alocado em sua própria memória e é proibido pelo kernel de acessar a memória fora da área alocada para o usuário que está rodando o processo. Processos diferentes compartilham dados por meio de Pipes comuns na memória, em vez de escrever diretamente no espaço de memória uns dos outros, consequentemente, se um programa travar, nenhum outro processo será afetado. E ser multitarefa é importante para ambientes multi-usuários, afinal múltiplos usuários querem a execução quase que concorrente de seus processos.

5.3 Usuários do GNU/Linux

Um usuário ou conta de um sistema é identificado exclusivamente²⁷ por um número denominado UID, e no GNU/Linux existem dois tipos de usuários:

- usuário **root** que é conhecido como super usuário;
- usuários normais.

Um usuário root ou super usuário pode acessar todos os arquivos, enquanto o usuário normal tem acesso limitado aos arquivos, lembre-se que durante o processo de instalação todos os arquivos foram criados para o usuário 0 (zero). Um superusuário pode adicionar, excluir e modificar uma conta de usuário.

Venho lutando contra o uso exacerbado da conta root, o que pode levar a todo um risco para a organização, no passado toda distribuição GNU/Linux iniciava somente com um usuário, o tal root, e depois por boa prática o usuário cria um novo usuário normal para uso cotidiano. Várias distros então nos últimos anos começaram a fazer o correto, já no processo de instalação do sistema operacional, passaram a criar este segundo usuário, na esperança da consciência.

Um aluno que usa a conta root nas práticas de Linux merece ser reprovado, um administrador de TI que usa a conta root rotineiramente merece ser demitido e naturalmente um dono de uma empresa que usa a conta root rotineiramente merece perder tudo que construiu ao longo de sua vida, com infundáveis processos judiciais, **isso já é uma observação Hacker.**

Um UID é um número atribuído pelo GNU/Linux a cada usuário no sistema e este número é usado para identificar o usuário no sistema e determinar quais recursos do sistema o usuário pode acessar, estas identificações estão armazenadas no arquivo **/etc/passwd** conforme figura abaixo.

²⁷ Na prática é possível ter mais de um usuário com o mesmo UID, [veja essa vulnerabilidade neste link](#).

```

usuario@debian:~$ cat /etc/passwd
root::0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/n
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/l
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nolo
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time Synchronization
systemd-network:x:102:103:systemd Network Management,,
systemd-resolve:x:103:104:systemd Resolver,,,:/run/syst
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
usuario:x:1000:1000:usuario,,,:/home/usuario:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/s
usuario@debian:~$
```

O terceiro campo representa o UID, observe como o usuário root tem o UID com o valor 0. A maioria das distribuições de GNU/Linux reserva os primeiros 100 UIDs para uso do sistema, já os novos usuários recebem UIDs a partir de 500 ou 1000. Estes valores são definidos no arquivo **/etc/login.defs** e utilizado no script adduser.

```

Debian 12 64 bits 2 [Running] - Oracle VM Vi
File Machine View Input Devices Help
#
# Min/max values for automatic uid selection in useradd
#
UID_MIN          1000
UID_MAX          60000
# System accounts
#SYS_UID_MIN     100
#SYS_UID_MAX     999
# Extra per user uids
SUB_UID_MIN      100000
SUB_UID_MAX      600100000
SUB_UID_COUNT    65536
```

5.4 O arquivo /etc/passwd

O arquivo **/etc/passwd** armazena informações essenciais, que são necessárias durante o login, em outras palavras, ele armazena informações das contas dos usuários.

O arquivo **/etc/passwd** é um texto simples que pode ser lido como o comando **cat**. Ele contém uma lista das contas do sistema, fornecendo para cada conta algumas informações úteis, como ID do usuário, ID do grupo, diretório inicial, shell e muito mais. O arquivo **/etc/passwd** deve ter permissão de leitura geral, já que muitos utilitários de comando o utilizam para mapear IDs de usuário para nomes de usuário mas no entanto, o acesso de gravação ao **/etc/passwd** deve limitar apenas para a conta de superusuário.



O arquivo **/etc/passwd** contém uma entrada por linha para cada usuário do sistema e todos os campos são separados por dois pontos. Geralmente, a entrada do arquivo **/etc/passwd** tem o seguinte layout:

```
GNU nano 7.2
root:x:0:0:root:/root:/bin/bash
  1 2 3 4 5   6   7
```

1. **Nome de usuário:** É usado quando o usuário faz o login. Deve ter entre 1 e 32 caracteres.
2. **Senha:** Um caractere x indica que existe uma entrada no arquivo **/etc/shadow** que pode conter uma senha e detalhes de acesso;
3. **ID de usuário (UID):** Cada usuário deve receber uma ID de usuário (UID). UID 0 (zero) é reservado para root e UIDs 1-99 são reservados para outras contas predefinidas. Outros UID 100-999 (no Debian) são reservados pelo sistema para contas/grupos administrativos e do sistema, depois para cada novo usuário um número superior a 999 e inferior ao INT_MAX do sistema²⁸.
4. **ID do grupo (GID):** O ID do grupo default (armazenado no arquivo **/etc/group**)
5. **Informações de ID do usuário:** o campo de comentário. Ele permite que você adicione informações extras sobre os usuários, como nome completo do usuário, número de telefone, etc.
6. **Diretório inicial:** O caminho absoluto para o diretório em que o usuário estará quando fizer login.
7. **Shell :** O caminho absoluto de um comando ou shell (**/bin/bash**). Normalmente, é um shell. Observe que não precisa ser um shell. Por exemplo, o sysadmin pode usar o shell nologin, que atua como um shell substituto para as contas de usuário. Se shell estiver definido como **/sbin/nologin** o usuário tentar efetuar login no sistema Linux diretamente, o shell **/sbin/nologin** fecha a conexão.

No item 3 da lista acima é definido o UID da camada de usuários comuns, este valor está definido fisicamente no arquivo **/etc/adduser.conf** em versões mais antigas do Debian GNU/Linux ou **/etc/login.defs** nas versões modernas, utilize o comando **more** visto no capítulo de Sistemas de Arquivos e localize as linhas abaixo.

²⁸ Existe uma vulnerabilidade no estouro do valor do INT, veja em:
<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-19788>

```
# package, may assume that UIDs 1
FIRST_SYSTEM_UID=100
LAST_SYSTEM_UID=999

FIRST_SYSTEM_GID=100
LAST_SYSTEM_GID=999
#
# FIRST_[GU]ID to LAST_[GU]ID inc
illy
## allocated user accounts/groups.
FIRST_UID=1000
LAST_UID=59999
```

No item 1 da listagem anterior é dito que o usuário deve ter um login único formado só de letras minúsculas com no mínimo 1 caractere e no máximo 32, é possível definir estes parâmetros no arquivo **/etc/adduser.conf** em uma REGEX conforme imagem abaixo (que está comentada).

```
# Non-system user- and groupnames are checked against this regul
# expression.
# Default: NAME_REGEX="^ [a-z] [-a-zA-Z_]*\$?\$"
#NAME_REGEX="^ [a-z] [-a-zA-Z_]*\$?\$"
```

Todos os recursos (arquivos, processo, etc..) estão associados aos usuários e a política de acesso garante ao GNU/Linux o status de sistema seguro e estável, mas é lógico que um usuário sem o devido treinamento pode transformar este ambiente organizado em um ambiente caótico, similar ao encontrado no Microsoft Windows.

No GNU/Linux usuário é usuário, mas podemos configurar os parâmetros do usuário em **/etc/passwd** e **/etc/group** para os seguintes contextos:

- **Usuário comum**: onde ficamos a maior parte do tempo e possuímos privilégios apenas ao diretório do usuário;
- **Usuário administração**: um superusuário ou um usuário comum associado a algum grupo que possui algum privilégio sobre alguns recursos, seja arquivos, processos e serviços;
- **Usuário de sistema**: contas de usuários que não podem realizar o login;

Geralmente é preciso através de scripts obter alguma informação deste arquivo, no script bash abaixo é demonstrado como fazer um parser entre estas colunas para posterior análise por outro algoritmo. Scripts serão apresentados ao longo do material.

```
1.#!/bin/bash
2. # script que lista os usuários e separa por campo
3. #
4. while IFS=: read -r f1 f2 f3 f4 f5 f6 f7
5. do
6.     echo "Usuário $f1 usa o shell $f7 e armazena arquivos em $f6 "
6. done < /etc/passwd
```

5.5 O arquivo /etc/shadow

Sua senha criptografada não é armazenada no arquivo **/etc/passwd**²⁹. Ele é armazenado no arquivo **/etc/shadow**, nos bons velhos tempos, não havia grande problema com essa permissão geral de leitura pois todos podiam ler as senhas criptografadas e o hardware era muito lento para quebrar uma senha bem escolhida e, além disso, o pressuposto básico costumava ser o de uma comunidade de usuários amigável.

Quase, todos os sistemas operacionais Linux/UNIX modernos usam algum tipo de suíte de senhas shadow, onde **/etc/passwd** tem asteriscos (*) em vez de senhas criptografadas, e as senhas criptografadas estão em **/etc/shadow** que é legível pelo superusuário.

```
usuario@debian:~$ tac /etc/passwd
systemd-coredump:x:999:999:systemd Core Dumper::/usr/sbin/nologin
usuario:x:1000:1000:usuario,,,:/home/usuario:/bin/bash
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
_lapt:x:100:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
```

O nome do usuário e a ligação entre as 2 listas

```
usuario@debian:~$ sudo tac /etc/shadow
[sudo] password for usuario:
systemd-coredump:!:18586::::
usuario:$6$uafkssmu5nHa1fMn$M2jzpzNK6zpaNJA7s/2ug8oX1hr
2EGdpCHytGJ3w.:18586:0:99999:7:::
messagebus:*:18586:0:99999:7:::
systemd-resolve:*:18586:0:99999:7:::
systemd-network:*:18586:0:99999:7:::
```

A estrutura do arquivo **/etc/shadow** é a seguinte:

GNU nano 7.2	/etc/shadow *						
1	2	3	4	5	6	7	8
/							

Onde,

1. **Nome de usuário**: é o seu nome de login;
2. **Senha**: é sua senha criptografada. A senha deve ter no mínimo 8-12 caracteres, incluindo caracteres especiais, dígitos, letras minúsculas do alfabeto e mais, segue formatos:
 - a. \$1\$ é MD5;
 - b. \$2a\$ é Blowfish;
 - c. \$2y\$ é Blowfish;
 - d. \$5\$ é SHA-256;
 - e. \$6\$ é SHA-512;
 - f. \$y\$ é yescrypt.
3. **Última alteração de senha (última alteração)**: Dias desde 1º de janeiro de 1970 em que a senha foi alterada pela última vez;

²⁹ No passado, em distribuições GNU/Linux o password ficava em /etc/passwd mas por este arquivo ter a permissão r para outros usuários, os especialistas removeram o password estes arquivo.

4. **Mínimo:** O número mínimo de dias necessários entre as alterações de senha, ou seja, o número de dias restantes antes que o usuário tenha permissão para alterar sua senha;
5. **Máximo:** O número máximo de dias em que a senha é válida (depois que o usuário é forçado a alterar sua senha);
6. **Aviso:** O número de dias antes que a senha expire para que o usuário seja avisado de que sua senha deve ser alterada;
7. **Inativo:** o número de dias após a expiração da senha em que a conta é desativada
8. **Expiração:** dias desde 1º de janeiro de 1970 que a conta foi desativada, ou seja, uma data absoluta que especifica quando o login não pode mais ser usado.

Atualmente (em 2025) a criptografia mais avançada é a Yescrypt, **Yescrypt** é um esquema de hash de senha escalável projetado pela Solar Designer, que é baseado no Colin Percival's scrypt³⁰. Recomendado para novos hashes de Sistemas Operacionais, consiste nas seguintes partes:

Prefix: "\$y\$"

Hashed passphrase format: \\${y\\$[./A-Za-z0-9]+\\${[./A-Za-z0-9]{,86}\\$[./A-Za-z0-9]{43}}

Maximum passphrase length: unlimited

Hash size: 256 bits

Salt size: up to 512 (128+ recommended) bits

CPU time cost parameter: 1 to 11 (logarithmic)

5.6 Arquivo /etc/group

Para visualizar todos os grupos presentes no sistema e sua relação com usuários³¹, basta abrir o arquivo /etc/group. Cada linha neste arquivo representa informações para um grupo.

```
1. cat /etc/group
```

Outra opção é usar o comando getent que exibe entradas de bancos de dados configurados em arquivo **/etc/nsswitch.conf** incluindo o banco do grupo que podemos usar para consultar uma lista de todos os grupos. Para obter uma lista de todos os grupos, digite o seguinte comando:

```
1. getent group
```

A saída é a mesma da exibição do conteúdo do arquivo /etc/group. Se você estiver usando LDAP para autenticação de usuário, o getent exibirá todos os grupos do arquivo /etc/group e do banco de dados LDAP.

Você também pode usar awk (linha 1 da listagem abaixo) ou cut (linha 2 da listagem abaixo) para imprimir o campo que desejar:

³⁰ Mais detalhes em <https://en.wikipedia.org/wiki/Scrypt>

³¹ Somente para grupos complementares, visto que grupo default de um usuário fica no arquivo /etc/passwd

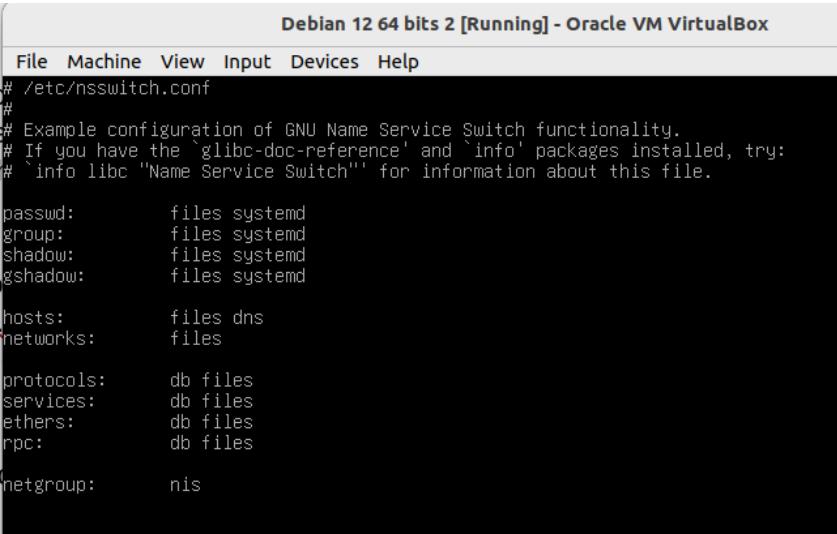
```
1. getent group | awk -F: '{ print $1}'
2. getent group | cut -d: -f1
```

O comando **awk** é um executor de scripts em linha, então é muito importante para o administrador linux conhecer seu poder, rapidamente podemos injetar um script em uma saída de comando e obter o que precisamos em segundos, sem a necessidade de fazer um arquivo script para tratar a saída. Já o **cut** é um comando para cortar uma string em pedaços, dando um token.

5.7 Arquivo /etc/nsswitch.conf

O arquivo **/etc/nsswitch.conf** é um arquivo de configuração do GNU/Linux que especifica como o sistema deve alternar entre diferentes name service providers. O arquivo pode ser usado para configurar quais serviços devem ser usados para pesquisa de nome de host, pesquisas de senha e assim por diante.

O arquivo **/etc/nsswitch.conf** é lido pelo Interruptor de Serviço de Nome (NSS) biblioteca quando o sistema é iniciado. A biblioteca NSS usa as informações em **/etc/nsswitch.conf** para determinar quais provedores de serviços de nome devem ser usados para cada tipo de pesquisa. Este arquivo é uma parte crítica do sistema operacional GNU/Linux, e quaisquer alterações no arquivo podem potencialmente causar sérios problemas.



```
Debian 12 64 bits 2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the 'glibc-doc-reference' and 'info' packages installed, try:
# `info libc "Name Service Switch"' for information about this file.

passwd:      files systemd
group:       files systemd
shadow:      files systemd
gshadow:     files systemd

hosts:       files dns
networks:    files

protocols:   db files
services:    db files
ethers:      db files
rpc:         db files

netgroup:    nis
```

5.8 Adicionando usuários no GNU/Linux com adduser e useradd

O comando **adduser** pode ser utilizado para adicionar usuários e até grupos ao sistema operacional, sua sintaxe é simples conforme listagem abaixo.

```
1. sudo adduser [opções] LOGIN_DO_USUARIO_PRETENDIDO
2. sudo adduser --group [opções] NOME_DO_GRUPO_PRETENDIDO
3. sudo adduser [opções] LOGIN_DO_USUARIO_PRETENDIDO grupo
```

Na linha 1 da listagem acima o comando **adduser** está sendo utilizado para criar um usuário no sistema e por não ser informado qual o grupo **default** do novo usuário será criado um novo grupo com o mesmo nome e associado. Já na linha 2 o comando adduser está sendo utilizado para criar um novo grupo, para isso utiliza-se o parâmetro **--group** e sim, o adduser é utilizado para criar grupos. Na linha 3 um novo usuário está sendo criado e já está sendo associado a um grupo default caso já tenha sido criado anteriormente.

São possíveis opções:

--conf <caminho do arquivo>: esta opção faz com que o adduser utilize outro arquivo de roteiro de criação diferente do default que é /etc/adduser.conf;
--group: informa que será criado um grupo e não um usuário;
--system: informa que será criado um usuário de sistema local;
--home: diretório default do usuário;

Vamos ver alguns exemplos:

```
1. sudo adduser --group alunos
2. sudo adduser aluno
3. sudo adduser --home /hd2/alunos/aluno aluno
```

Na figura abaixo o comando da linha 2 (da listagem acima) está sendo executado no terminal, logo após o comando inicia-se uma interação entre terminal e administrador para configurar a nova conta do usuário.

```
usuario@aied:/tmp$ sudo adduser aluno
[sudo] password for usuario:
Adding user `aluno' ...
Adding new group `aluno' (1001) ...
Adding new user `aluno' (1001) with group `aluno' ...
Creating home directory `/home/aluno' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for aluno
Enter the new value, or press ENTER for the default
    Full Name []: Aluno
    Room Number []: 101
    Work Phone []: 5555-5555
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
usuario@aied:/tmp$ _
```

O comando adduser é um script Perl que pode ser visto pelo aluno, basta usar **more** e **cat**.

```
usuario@aied:~$ file /sbin/adduser
/sbin/adduser: Perl script text executable
usuario@aied:~$
```

Assim como o adduser o useradd cria usuários no sistema, mas diferente do anterior este é um programa em C compilado³² e pertencente ao projeto **shadow-utils**. Existem distribuições que possuem somente um destes, mas o Debian tende a ser amigo de todos e por isso encontramos ambos no Sistema Operacional.

³² O código em C do useradd pode ser visto na url:

<https://github.com/Distrotech/shadow-utils/blob/distrotech-shadow-utils/src/useradd.c>

```
usuario@debian:~$ file /usr/sbin/useradd
/usr/sbin/useradd: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=8b4a2c4bab80053bb1dc9576dc41e86e0793bb9b, for GNU/Linux 3.2.0, stripped
usuario@debian:~$ _
```

A sintaxe é simples, veja a listagem abaixo de opções:

1. useradd [opções] LOGIN
2. useradd -D
3. useradd -D [opções]

Quando executado sem o parâmetro **-D**, o comando useradd cria uma nova conta de usuário usando os valores especificados na linha de comando mais os valores padrão do sistema. Dependendo das opções de linha de comando, o comando useradd atualizará os arquivos do sistema e também poderá criar o diretório home do novo usuário.

As opções aplicáveis ao comando useradd são:

-b, --base-dir BASE_DIR: O diretório home padrão para o sistema se **-d HOME_DIR** não está especificado. Se esta opção não for especificada, useradd usará o diretório base especificado pela variável HOME em **/etc/default/useradd** ou **/home** por padrão em um Debian GNU/Linux;

-c, --comment COMENTÁRIO: Qualquer sequência de texto. Geralmente, é uma breve descrição do login e atualmente é usado como campo para o nome completo do usuário;

-d, --home HOME_DIR: O novo usuário será criado usando HOME_DIR como o valor para o diretório de login do usuário. O padrão é anexar o username ao BASE_DIR e use isso como o nome do diretório de login;

-e, --expiredate EXPIRE_DATE: A data em que a conta de usuário será desativada. A data é especificada no formato AAAA-MM-DD, e se não especificado, useradd usará a data de validade padrão especificada pela variável EXPIRE em **/etc/default/useradd**;

```
#
# Other former uses of this variable such as setting the umask when
# user==primary group are not used in PAM environments, such as Debian
#
USERGROUPS_ENAB yes
```

-g, --gid GRUPO: O nome do grupo ou o número do grupo padrão do novo usuário. Se não especificado, o comando **useradd** irá consumir o arquivo **/etc/login.defs** para obter a variável **USERGRUPS_ENAB**. Se esta variável estiver definida como **yes**, um grupo será criado para o usuário, com o mesmo nome que seu nome de logon. Se o variável é definida como **false**, o comando useradd irá definir o grupo primário do novo usuário para o valor especificado pela variável **GROUP** em **/etc/default/useradd**, ou 100 por padrão;

-G, --grupos GRUPO1[,GRUPO2,...[GRUPON]]: Uma lista de grupos suplementares dos quais o usuário também é membro. Cada grupo é separado do próximo por uma vírgula, sem espaço intermediário. Os grupos estão sujeitos às mesmas restrições que o grupo dado com a opção **-g**;

-r, --sistema: Criar uma conta do sistema. Os usuários do sistema serão criados sem informações em **/etc/shadow**, e seus identificadores numéricos são escolhidos de acordo

como intervalo definido na variável **SYS_UID_MIN** e **SYS_UID_MAX** no arquivo **/etc/login.defs**;

-s, --shell PATH_DO_SHELL: A variável shell que informa qual programa será executado quando o novo usuário logar. O padrão é deixar esse campo em branco, o que faz com que o sistema selecione o shell de login padrão especificado pela variável **SHELL** em **/etc/default/useradd**;

```
# Default values for useradd(8)
#
# The SHELL variable specifies the default login shell on your
# system.
# Similar to DSHELL in adduser. However, we use "sh" here because
# useradd is a low level utility and should be as general
# as possible
# SHELL=/bin/sh 1
#
# The default group for users
# 100=users on Debian systems
# Same as USERS_GID in adduser
# This argument is used when the -n flag is specified.
# The default behavior (when -n and -g are not specified) is to
# primary user group with the same name as the user being added
# system.
# GROUP=100
#
# The default home directory. Same as DHOME for adduser
# HOME=/home 2
#
# The number of days after a password expires until the account
# is permanently disabled
# INACTIVE=-1 3
#
# The default expire date
# EXPIRE= 4
#
# The SKEL variable specifies the directory containing "skeletal"
# files; in other words, files such as a sample .profile that w
```

Onde:

1. Define o shell padrão se nenhum shell for especificado;
2. Diretório home padrão para os usuários;
3. Se a nova conta inicia desativado;
4. Data padrão que a nova conta será expirada.

Para a saber a lista de grupos de um usuário utilize o comando **group** seguido do **username**, veja a figura abaixo.

```
usuario@debian:~$ groups usuario 1
usuario : usuario cdrom floppy audio dip video plugdev users
2 3
```

Onde:

1. Comando **group** para listar os grupos do usuário **userlinux**;
2. Default group do usuário **userlinux**;
3. Grupos complementares do **userlinux**.

5.9 Criando grupos com adduser e addgroup

Grupos podem ser adicionados com o comando adduser ou addgroup, com adduser basta adicionar o comando **--group** conforme figura abaixo.

```
usuario@aled:~$ sudo adduser --group alunos
[sudo] password for usuario:
Adding group `alunos' (GID 1001) ...
Done.
usuario@aled:~$
```

Quando o comando é executado este consulta o arquivo **/etc/adduser.conf** para obter a informação de qual o GID inicial, conforme a imagem acima.9

```
usuario@aled:~$ 
usuario@aled:~$ file /sbin/addgroup
/sbin/addgroup: symbolic link to adduser
usuario@aled:~$ 
usuario@aled:~$
```

O comando addgroup em um Debian é um link simbólico para adduser, conforme figura acima, então quando é invocado ele chama o adduser e o resultado acaba sendo o mesmo, conforme figura abaixo.

```
usuario@aled:~$ sudo addgroup professores
[sudo] password for usuario:
Adding group `professores' (GID 1002) ...
Done.
usuario@aled:~$ 
usuario@aled:~$
```

5.10 Deletando grupos e usuários com deluser e delgroup

Grupos são excluídos e há vários momentos em que essa ação é necessária, a principal é o uso errôneo dos comandos **adduser** e **useradd** sem o default **group**. Grupos podem ser excluídos tanto com deluser como com delgroup, a regra é que um grupo que é default group de algum usuário existente não pode ser escolhido.

No exemplo abaixo vou demonstrar ao tentar remover o grupo userlinux, mas como existe um usuário chamado userlinux que possui o grupo userlinux como grupo default, a exclusão não é possível.

```
1 usuario@debian:~$ groups usuario
2 usuario : usuario cdrom floppy audio dip video plugdev users
usuario@debian:~$ sudo delgroup usuario
3 Removing group `usuario' ...
4 groupdel: cannot remove the primary group of user 'usuario'
delgroup: '/sbin/groupdel usuario' returned error code 8. Exiting.
usuario@debian:~$
```

Onde:

1. Exibindo grupos de um usuário
2. O usuário userlinux possui como default group o grupo também chamado userlinux;
3. Tentando remover o grupo userlinux;
4. Falha ao remover, veja que a mensagem é clara sobre esta restrição.

Mas se o grupo que quer excluir não for usado como grupo default de nenhum usuário e é usado apenas como grupo complementar de usuários, será excluído normalmente.

```
usuario@debian:~$ groups usuario 1
usuario : usuario cdrom floppy audio dip video plugdev users
usuario@debian:~$ 2
usuario@debian:~$ sudo delgroup floppy 3
Removing group `floppy' ...
Done. 4
```

Onde:

1. Comando group para ver todos os grupos do usuário userlinux;
2. Localizando um grupo desnecessário para este GNU/Linux;
3. Removendo um grupo, mesmo que tenha usuários utilizando como grupo complementar;
4. Sucesso, removido.

Essa divisão entre grupos de complemento e grupo default é bem evidenciado na posição onde esta informação é salva, veja que o **default group** está definido no arquivo **/etc/passwd** enquanto os grupos complementares estão definidos no arquivo **/etc/group**. Se a opção **--only-if-empty** é utilizada, o grupo não será removido se estiver sendo utilizado como grupo complementar de algum usuário.

Pode-se remover grupos também com o comando **deluser**, mas para isso deve-se utilizar o argumento **--group GRUPO**. Por padrão, **deluser** removerá o usuário sem remover o diretório home deste usuário, nem caixas de e-mail (em **/var/mail**) e muito menos qualquer spool como o de impressão (em **/var/spool**), isso pois um comando errado pode se tornar um problema maior. Se quer remover tudo, então utilize a opção **--remove-all-files** como opção, mas removerá tudo, caso queira ser menos agressivo na remoção e remover apenas o diretório home do usuário utilize a opção **--remove-home**.

```
usuario@debian:~$ sudo deluser --remove-all-files mariana 1
Looking for files to backup/remove ...
Removing files ... 2
Removing crontab ...
Removing user `mariana' ... 3
Done.
```

Onde:

1. Removendo um usuário com o comando **deluser**, informando que deve ser removido todos os arquivos;
2. A remoção dos arquivos pessoais do usuário em questão;
3. Removendo o usuário dos arquivos **/etc/passwd**, **/etc/shadow** e **/etc/group**.

5.11 Editando um usuário com usermod

O comando **usermod** é um comando que permite modificar as informações de login de um usuário. A sintaxe do comando assume a seguinte forma:

1. `sudo usermod [options] USER`

Possíveis opções:

- d DIRETÓRIO [-m]**: cria um novo diretório home para o usuário. A opção **-m** faz com que os arquivos do diretório atual do usuário sejam movidos para o novo diretório;
- e yyyy-mm-dd**: altera a data de expiração da conta do usuário;
- g GRUPO**: altera o GID do default group do usuário para o valor especificado;
- G GRUPO1[,GRUPO2, ...]**: define o GID dos outros grupos aos quais o usuário pertence;
- i NOME**: altera o nome de identificação do usuário (o usuário não pode estar logado);
- s SHELL**: altera o shell do usuário;
- u UID**: altera o número de UID do usuário.

Apenas **root** ou usuários cadastrados no sudoers podem executar o comando **usermod** e modificar uma conta de usuário e, em caso de sucesso, o comando não exibe nenhuma saída.

5.11.1 Adicionar um usuário a um grupo complementar

O caso de uso mais típico de usermode é adicionar um usuário a um grupo. Para adicionar um usuário existente a um grupo secundário, use as **-a -G** opções após o nome do grupo e o nome de usuário conforme exemplo:

```
1. sudo usermod -a -G alunos aluno
```

Se você deseja adicionar o usuário a vários grupos de uma vez, especifique os grupos após a **-G** opção separados por ,(vírgulas) sem espaços em branco. Por exemplo, para adicionar o usuário aluno aos grupos alunos e users, você executaria o seguinte comando:

```
1. sudo usermod -a -G alunos,users aluno
```

Sempre use a opção **-a** ao adicionar um usuário a um novo grupo, se você omitir a opção **-a** o usuário será removido dos grupos não listados após a opção **-G**. O resultado desta operação é a alteração do arquivo **/etc/group**.

5.11.2 Alterar o default group do usuário

Para alterar o grupo default de um usuário, execute o comando **usermod** com o parâmetro **-g** seguida do nome do grupo e do nome de usuário conforme exemplo:

```
1. sudo usermod -g GROUP USER
```

No exemplo o usuário aluno está sendo alterado para ter como default group alunos.

```
1. sudo usermod -g alunos aluno
```

Cada usuário pode pertencer a somente um grupo default. O resultado desta operação é a alteração do arquivo **/etc/passwd**.

5.1.3 Alteração das informações do usuário

Para alterar as informações de comentário sobre o usuário aluno, execute o comando com a opção **-c**, e em seguida pelo novo comentário e nome de usuário:

```
1. sudo usermod -c "O aluno que faz a aula" aluno
```

Essas informações são armazenadas na coluna 5 do arquivo **/etc/passwd** conforme tópico anterior.

5.11.4 Alterando um diretório inicial do usuário

Na maioria dos sistemas GNU/Linux, os diretórios dos arquivos do usuário são nomeados de acordo com o nome do usuário e criados em um diretório específico, no caso do Debian o padrão é em **/home/** quando a variável **DEFAULT_HOME** em **/etc/login.defs** é definida como **yes**. Se por algum motivo for necessário alterar o diretório inicial do usuário, execute o comando **usermod** com a opção **-d** seguida pelo caminho absoluto do novo diretório inicial e o nome do usuário:

```
1. sudo usermod -d /hd2/alunos/aluno aluno
```

Por padrão, o comando não move o conteúdo do diretório inicial do usuário para o novo. Para mover o conteúdo, use a opção **-m**. Se o novo diretório ainda não existir, ele será criado:

```
2. sudo usermod -d /hd2/alunos/aluno -m aluno
```

Um exemplo clássico que é muito conhecido é a troca do diretório do usuário **www-data** forçando que o diretório inicial seja **/var/www** por motivos de segurança.

```
1. sudo usermod -d /var/www www-data
```

5.11.5 Mudando o Shell Padrão do Usuário

O shell padrão é o shell que é executado após o login no sistema. Por padrão, na maioria dos sistemas GNU/Linux, o shell padrão é definido como **/bin/bash**, mas é comum usar também **/bin/sh** ou **/bin/dash**. Para alterar o shell padrão do usuário, execute o comando com a opção **-s** seguida do caminho absoluto do shell e o nome do usuário:

```
1. sudo usermod -s /usr/sbin/nologin aluno
```

5.11.6 Alterando um UID de usuário

UID é um número atribuído a cada usuário. É usado pelo sistema operacional para se referir a um usuário e já foi definido neste texto com um número teoricamente único. Para alterar o UID do usuário, execute o comando com a opção **-u** seguida do novo UID e o nome do usuário:

```
1. sudo usermod -u 1300 aluno
```

O UID dos arquivos pertencentes ao usuário estão localizados no diretório inicial do usuário, e o arquivo da caixa de correio do usuário será alterado automaticamente. A propriedade de todos os outros arquivos deve ser alterada manualmente.

5.11.7 Alterar um nome de usuário

Embora não seja muito frequente, às vezes você pode querer alterar o nome de um usuário existente. A opção **-l** é usada para alterar o nome de usuário:

```
1. sudo usermod -l aluno aprendiz
```

No exemplo acima o aluno está sendo renomeado para aprendiz. Ao alterar o nome de usuário, você também pode alterar o diretório pessoal do usuário para refletir o novo nome de usuário.

5.11.8 Definir uma data de expiração do usuário

A data de expiração é a data em que a conta do usuário será desabilitada. Para definir a data de expiração do usuário, use a opção **-e**. A data de validade deve ser definida usando o formato YYYY-MM-DD. Por exemplo, para desabilitar o usuário aluno2, você executaria o seguinte comando:

```
1. sudo usermod -e "2022-02-21" aluno2
```

Para desativar a expiração de uma conta, defina uma data de expiração vazia:

```
1. sudo usermod -e "" aluno2
```

Use o comando **chage** com o parâmetro **-l** para ver a data de validade do usuário:

```
1. sudo chage -l aluno2
```

5.11.9 Bloqueio e desbloqueio de uma conta de usuário

A opção **-L** permite que você bloqueeie uma conta de usuário. O comando deverá inserir um ponto de exclamação ! na frente da senha criptografada no arquivo **/etc/shadow**, o usuário não poderá fazer login no sistema usando autenticação de senha. Se você deseja bloquear a conta e desabilitar todos os métodos de login, você também precisa definir a data de expiração para 1.

```
1. sudo usermod -L aluno2
2. sudo usermod -L -e 1 aluno2
```

Para desbloquear um usuário, execute o comando com o parâmetro **-U** conforme exemplo:

```
1. sudo usermod -U aluno2
```

5.12 Alterando a senha de um usuário

No GNU/Linux, você pode alterar a senha de uma conta de usuário com o comando `passwd`, se é sua própria senha então poderá sem necessidade de `sudo`, mas se está alterando a senha de outro usuário será necessário o uso do `sudo`. As senhas são criptografadas automaticamente, e estas informações são armazenadas no arquivo `/etc/shadow` conforme visto em tópicos anteriores.



Atualmente na data de escrita deste material (ano 2023), segundo a NIST 800-63³³ uma senha recomenda é uma senha que não foi exposta em algum Data Leak, embora existam sites que validam isto, não é recomendado que realmente utilize o sistema online, mas você pode extrair LISTAS de senhas vazadas e naturalmente fazer a validação de senhas antes de seu uso.

5.12.1 Consultando dados sobre o password de um usuário

Antes de alterar qualquer aspecto da autenticação de um usuário, entenda que é possível consultar o status de uma conta, com o comando `passwd` e usando o parâmetro `-S`, conforme exemplo abaixo.

```
usuario@debian:~$ passwd -S
usuario P 2023-09-28 0 99999 7 -1
 1 2 3 4 5 6 7
```

Onde:

1. O nome de login;
 2. Se a conta de usuário:
- L** tem uma senha bloqueada;
- NP** não tem senha;

³³ Norma disponível em <https://pages.nist.gov/800-63-3/>

P tem uma senha útil.

3. Data da última alteração de senha;
4. Tempo de vida mínimo da senha antes que ela possa ser alterada;
5. Tempo máximo de vida útil da senha antes de ser alterado;
6. Número de dias antes da expiração da vida útil da senha quando o usuário começará a receber avisos;
7. Número de dias após a expiração da vida útil da senha quando o usuário está desativado.

5.12.2 Altere sua senha de usuário

Para alterar a senha da sua própria conta de usuário, execute o comando `passwd` sem nenhum argumento:

```
1. passwd
```

Você será solicitado a inserir sua senha atual. Se a senha estiver correta, o comando solicitará que você insira e confirme a nova senha. As senhas não são mostradas na tela quando você as insere, para um usuário iniciante o desespero pode bater por não estar vendo a senha ser digitada mesmo que ele digite no teclado.

5.12.3 Alterar a senha de outro usuário

Como mencionamos, apenas o usuário `root` e os usuários com acesso ao `sudo` podem alterar a senha de outra conta de usuário. O exemplo a seguir pressupõe que você esteja conectado como um usuário com privilégios `sudo`. Para alterar a senha de outra conta de usuário, execute o comando `passwd` seguido do nome de usuário.

```
1. sudo passwd aluno2
```

Você será solicitado a inserir e confirmar a nova senha:

```
Enter new UNIX password:  
Retype new UNIX password:
```

5.12.4 Forçar o usuário a alterar a senha no próximo login

Por padrão, as senhas são definidas para nunca expirar. Para forçar um usuário a alterar sua senha na próxima vez que fizer login, use o comando `passwd` com `--expire` seguida do nome de usuário do usuário:

```
1. sudo passwd --expire aluno2
```

O comando acima irá expirar imediatamente a senha do usuário e na próxima vez que o usuário tentar fazer o login com a senha antiga, será exibida uma mensagem forçando-o a alterar a senha.

5.13.5 Removendo a senha de um usuário

Se você quiser fazer uma conta de usuário sem senha (para um serviço por exemplo ou até a conta root), você pode usar o parâmetro **-d** (**--delete**) no comando **passwd**. Esta é uma maneira rápida de desativar uma senha para uma conta.

```
1. sudo passwd --delete www-data
```

Embora o usuário **www-data** não tenha senha, mas digamos que possua, então podemos tirar assim, como exemplo anterior. Usuários de serviços não devem ter:

- Um password;
- Um shell definido.

5.13.6 Bloqueando e Desbloqueando o password de usuários

Você pode impedir que os usuários façam login bloqueando a senha deles, para isso use o comando **passwd** com o parâmetro **-l** (**--lock**) e em seguida pelo nome de usuário.

```
1. sudo passwd --lock aluno
```

Mas há um problema, se este GNU/Linux possui o **openssh-server** instalado e configurado e previamente o usuário **aluno** (recentemente bloqueado com **--lock**) tiver uma chave assimétrica para acesso sem senha, este conseguirá acessar o servidor, então por motivo de segurança, desabilitamos a opção shell.

```
1. sudo usermod -s /bin/false aluno
```

Para desbloquear é fácil, basta executar o comando **passwd** com o parâmetro **--unlock** e como neste caso foi alterado o shell default do usuário, também retornar ao shell anterior.

```
1. sudo passwd --unlock aluno
2. sudo usermod -s /bin/bash aluno
```

5.13 Obtendo informações de usuários com finger

Saber quem está usando seu sistema GNU/Linux, o comando **finger** não é um comando instalado por padrão no Debian GNU/Linux foco deste livro, também está ausente nas distribuições Ubuntu, Fedora e Manjaro. A instalação é simples, utilizando o **apt**, veja exemplo:

```
1. sudo apt install finger -y
```

O comando **finger** fornece algumas informações sobre os usuários atualmente logados em seu sistema operacional GNU/Linux. O exemplo abaixo exibe uma lista de usuários com sessão aberta no GNU/Linux.

```
usuario@aied:~$ finger -l
Login: usuario                               Name: usuario
Directory: /home/usuario                      Shell: /bin/bash
On since Mon Mar  8 10:23 (-03) on tty1   6 seconds idle
  (messages off)
No mail.
No Plan.
usuario@aied:~$
```

Figura 4.

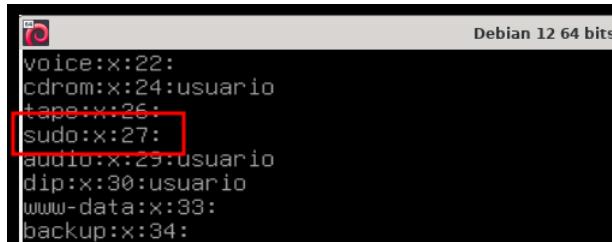
Caso queira obter dados de um único usuário, basta informar o usuário no comando finger.

```
usuario@aied:~$ finger root
Login: root                               Name: root
Directory: /root                           Shell: /bin/bash
Last login Thu Nov 19 23:20 (-03) on tty1
No mail.
No Plan.
usuario@aied:~$
```

Figura 4.

5.14 Comando sudo, arquivo sudoers e grupo sudo

Não é recomendável o uso da conta root, este fato já é bem recorrente no mundo Linux. O que se faz é criar contas de administradores associando estes administradores a grupos de administradores, um grupo muito conhecido é o grupo sudo, conforme figura abaixo.



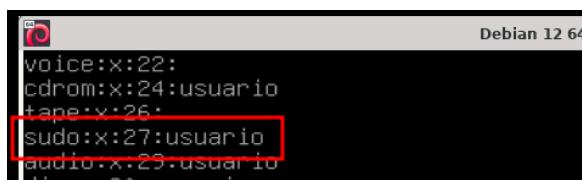
```
Debian 12 64 bits
cat /etc/group
voice:x:22:
cdrom:x:24:usuario
tape:x:26:
sudo:x:27:
audio:x:29:usuario
dip:x:30:usuario
www-data:x:33:
backup:x:34:
```

Figura 4.

Este grupo é criado por padrão quando se instala o sudo, conforme capítulo 1 deste livro, inclusive neste livro foi ensinado a adicionar o usuário userlinux no sudoers por meio da edição do arquivo **/etc/sudoers**. Agora que o aluno possui um conhecimento maior de GNU/Linux, vamos mudar este esquema, para isso adicione o usuário chamado userlinux ao grupo sudo conforme comandos abaixo.

1. sudo usermod -a -G sudo userlinux
2. sudo reboot

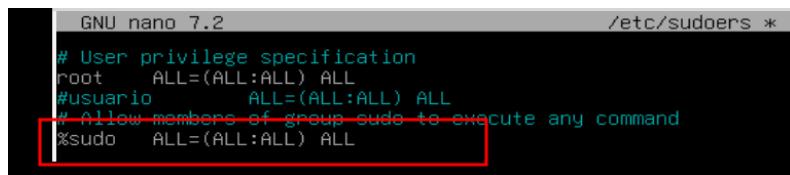
Ao listar as linhas do arquivo **/etc/group** é possível observar o usuário no grupo sudo.



```
Debian 12 64 bits
cat /etc/group
voice:x:22:
cdrom:x:24:usuario
tape:x:26:
sudo:x:27:usuario
audio:x:29:usuario
dip:x:30:usuario
```

Figura 4.

Na figura abaixo foi comentado a linha **userlinux ALL=(ALL:ALL) ALL**, mas o correto é remover, foi deixado comentado para mostrar o que deve ser removido. Foi adicionada a linha **%sudo ALL=(ALL:ALL) ALL** conforme imagem abaixo.



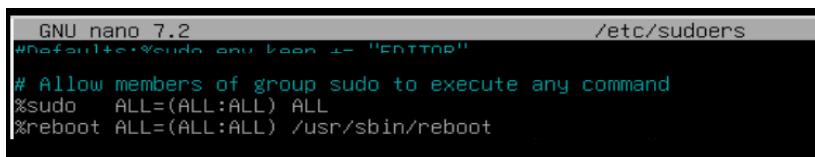
```
GNU nano 7.2 /etc/sudoers *
# User privilege specification
root    ALL=(ALL:ALL) ALL
#usuario    ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL
```

Figura 4.

Onde:

- O primeiro ALL** refere-se a regra aplicada a todos os hosts;
- O segundo ALL** informa que o usuário sudo pode executar comandos como qualquer usuário;
- O terceiro ALL** informa que o usuário sudo pode executar comandos como qualquer grupo;
- O quarto ALL** informa que todos os comandos podem ser executados.

A prática é a criação de vários grupos de administração com restrições de comando ou de permissões aqui no **/etc/sudoers**, e então associar os administradores de acordo com o escopo (comandos) de cada administrador. No exemplo abaixo é possível ver um grupo de usuários que poderão executar o sudo e invocar especificamente o comando **/usr/sbin/reboot**.



```
GNU nano 7.2 /etc/sudoers
# User privilege specification
root    ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL
%reboot ALL=(ALL:ALL) /usr/sbin/reboot
```

Lógico que o grupo deve existir e então é só adicionar usuários ao grupo reboot. Caso precise de mais comandos, basta adicionar ao término da linha divididos por vírgula sem espaços.

5.15 Práticas do capítulo usuários GNU/Linux

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

5.15.1 Prática fcb0b800 01: Criando usuários e grupos

Nesta prática o aluno deve ter a expertise de criar usuários e grupos, mas antes o aluno deve ter o comando sudo devidamente instalado e configurado conforme link abaixo.

O aluno deve:

1. Criar um grupo chamado **alunos**;
2. Criar um usuário chamado **joao** e durante a criação defina como grupo default o grupo **alunos**, utilize o comando useradd para isso;

3. Crie um usuário chamado **leticia** sem informar grupo default;
4. Crie um usuário chamado **wanderson** sem informar grupo default;
5. Crie um novo diretório chamado **/hd2/usuarios/** na raiz, vai precisar ter permissão de acesso;
6. Crie um usuário chamado **bia** sem informar grupo default mas informando que o default directory deste usuário é **/hd2/usuarios/bia**;
7. Informar um password para os usuários: **wanderson, bia e leticia**

Execute o robô de validação aied conforme comando abaixo.

1. sudo aied validar fcb0b800 checkpoint01

O resultado esperado é próximo ao da figura abaixo, isso pois o texto é dinâmico e eventuais testes podem ser adicionados.

```
usuario@aied:~$ sudo aied validar fcb0b800 checkpoint01
AASÁo que serÃi executada: validar

PrÃtica: PrÃtica da aula de usuarios linux - Criando usuarios e grupos
SerÃi enviado o OUTPUT do comando: cat /etc/passwd
SerÃi enviado o OUTPUT do comando: ls -l /hd2/usuarios/
SerÃi enviado o OUTPUT do comando: cat /etc/shadow

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: cat /etc/passwd
1.1   Validar por text    joao:x:
1.2   Validar por text    leticia:x:
1.3   Validar por text    wanderson:x:
1.4   Validar por text    bia:x:
1.5   Validar por text    /hd2/usuarios/bia

2 - Comando: ls -l /hd2/usuarios/
2.1   Validar por text    bia bia

3 - Comando: cat /etc/shadow
3.1   Validar por text    joao:!: 
3.2   Validar por text    leticia:$6$ 
3.3   Validar por text    bia:$6$ 
3.4   Validar por text    

Total de acertos: 10 do total de 10 validaÃ§Ãµes equivalente a 100%.
IdentificaÃ§Ã£o: 868f3e877d
AIED v(5)
```

5.15.2 Prática fcb0b800 02: Editando usuários

Nesta prática o aluno deve ter a expertise de alterar usuários, a lista de atividades da prática:

1. Com usermod faça com que o grupo default de leticia, wanderson e bia seja alunos;
2. Defina como diretório padrão para wanderson o diretório **/hd2/usuarios/wanderson**;
3. Defina como diretório padrão para leticia o diretório **/hd2/usuarios/leticia**;
4. Defina como diretório padrão para joao o diretório **/hd2/usuarios/joao**;
5. Trave a conta do usuário leticia com usermod e parâmetro L;
6. Defina que o usuário leticia deve ter como default shell **/bin/false**;

Execute o robô de validação aied conforme comando abaixo.

```
1. sudo aied validar fcb0b800 checkpoint02
```

O resultado esperado é próximo ao da figura abaixo, isso pois o texto é dinâmico e eventuais testes podem ser adicionados.

```
usuario@aied:~$ sudo aied validar fcb0b800 checkpoint02
Ação que será executada: validar

Prática: Prática da aula de usuários linux - Editando usuários
Seriamente enviado o OUTPUT do comando: groups bia
Seriamente enviado o OUTPUT do comando: groups leticia
Seriamente enviado o OUTPUT do comando: groups wanderson
Seriamente enviado o OUTPUT do comando: cat /etc/passwd
Seriamente enviado o OUTPUT do comando: cat /etc/shadow

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: groups bia
1.1   Validar por text      bia : alunos
2 - Comando: groups leticia
2.1   Validar por text      leticia : alunos
3 - Comando: groups wanderson
3.1   Validar por text      wanderson : alunos
4 - Comando: cat /etc/passwd
4.1   Validar por text      /hd2/alunos/joao
4.2   Validar por text      /hd2/alunos/wanderson
4.3   Validar por text      /hd2/alunos/leticia
5 - Comando: cat /etc/shadow
5.1   Validar por text

Total de acertos: 7 do total de 7 validações equivalentes a 100%.
Identificação: 868f3e877d
AIED v(5)

usuario@aied:~$
```

5.15.3 Prática fcb0b800 03: Alterar os arquivos passwd e shadow

Nesta prática o aluno deve ter a expertise de alterar usuários tanto no arquivo shadow quanto no arquivo passwd:

1. Para o usuário root, altere a senha para * no arquivo shadow;
2. Altere no passwd o shell padrão do usuário www-data como /bin/false

Execute o robô de validação aied conforme comando abaixo.

```
1. sudo aied validar fcb0b800 checkpoint03
```

O resultado esperado é próximo ao da figura abaixo, isso pois o texto é dinâmico e eventuais testes podem ser adicionados.

```

usuario@debian:~$ sudo aied validar fcb0b800 checkpoint03
[sudo] password for usuario:
Ação que será executada: validar

Prática: Alterar arquivo passwd e shadow
Será enviado o OUTPUT do comando: cat /etc/passwd
Será enviado o OUTPUT do comando: cat /etc/shadow

Deseja continuar? (s para SIM) s
s1 - Comando: cat /etc/passwd
1.1   Validar por regex      /www -data(.*)? /bin /false/
2 - Comando: cat /etc/shadow
2.1   Validar por text      root:*

Total de acertos: 2 do total de 2 validações equivalente a 100%.
Identificação: 65c1e51de3
AIED v(8)

```

5.15.4 Prática fcb0b800 04: Usuário tomcat para o serviço tomcat

Nesta prática o aluno deve ter a expertise para criar um usuário de serviço:

1. Crie um grupo chamado tomcat;
2. Crie um usuário chamado tomcat, mas com as seguintes características:
 - a. Não possuir senha
 - b. Diretório default /opt/tomcat
 - c. Shell padrão /bin/false
 - d. Grupo default tomcat
3. Crie o diretório /opt/tomcat e altere o diretório tenha como grupo o grupo tomcat.

Execute o robô de validação aied conforme comando abaixo.

```
2. sudo aied validar fcb0b800 checkpoint04
```

O resultado esperado é próximo ao da figura abaixo, isso pois o texto é dinâmico e eventuais testes podem ser adicionados.

```

usuario@debian:~$ sudo aied validar fcb0b800 checkpoint04
Ação que será executada: validar

Prática: Usuário tomcat para o serviço tomcat
Será enviado o OUTPUT do comando: cat /etc/passwd
Será enviado o OUTPUT do comando: cat /etc/shadow
Será enviado o OUTPUT do comando: groups tomcat
Será enviado o OUTPUT do comando: ls -l /opt/ | grep tomcat

Deseja continuar? (s para SIM) s
s1 - Comando: cat /etc/passwd
1.1   Validar por regex      /tomcat(.*)? /opt /tomcat: /bin /false/
2 - Comando: cat /etc/shadow
2.1   Validar por text      tomcat:!
3 - Comando: groups tomcat
3.1   Validar por text      tomcat : tomcat
4 - Comando: ls -l /opt/ | grep tomcat
4.1   Validar por text      root tomcat

Total de acertos: 4 do total de 4 validações equivalente a 100%.
Identificação: b9e0eec03a
AIED v(8)

```

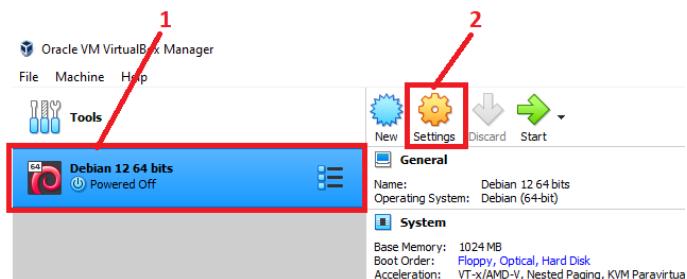
adicionar questões de segurança

6 Formatando e montando um dispositivo de bloco

Um disco rígido é genericamente denominado "dispositivo de bloco" porque os discos rígidos leem e gravam dados em blocos de tamanho fixo, segundo Tanenbaum geralmente 512 bytes por operação `read()` ou `write()`, e isso diferencia estes dispositivos dos dispositivos de caractere, como uma impressora serial, teclado, microfone ou câmera que não opera por blocos.

6.1 Adicionando um novo disco no VirtualBox

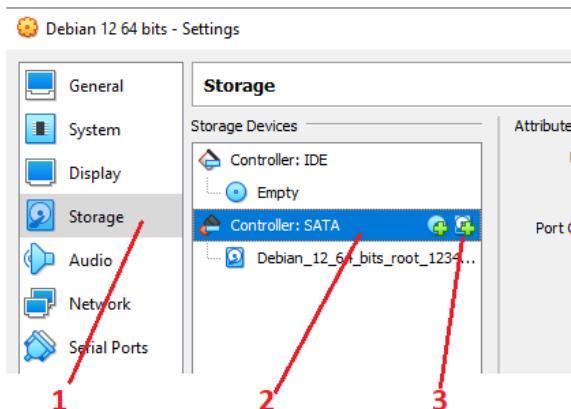
Quando temos um computador físico compramos um disco e adicionamos na SATA, já um dispositivo SSD adicionamos também na SATA e uma M2 no slot m2. No caso do VirtualBox temos que adicionar um novo disco virtual, conforme exemplo abaixo, para isso, com a Virtual Machine desligada entre em suas configurações conforme figura abaixo.



Onde:

1. Selecione a máquina virtual que será utilizada na prática;
2. Acesse a opção de Settings.

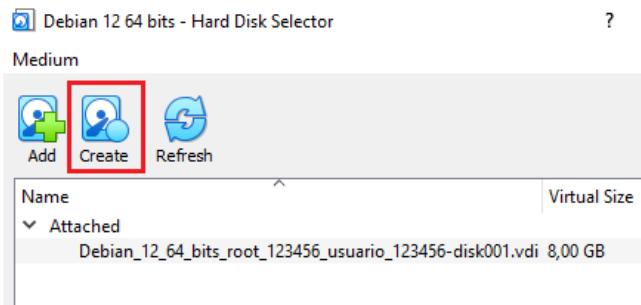
Na aba Storage clique em Adicionar um novo Hardisk conforme figura abaixo, mas fique atento, o Harddisk possui muitos pratos (versões antigas do VirtualBox) ou uma caixinha em versões mais recentes.



Onde:

1. Opção Storage da máquina virtual;
2. Barramento SATA;
3. Opção de adicionar um Hard Disk.

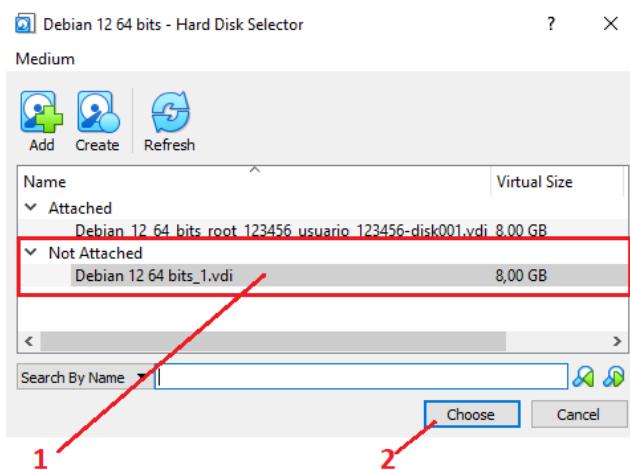
Com certeza não é nenhum dos discos que já existe, será um novo, então clique em Create.



Siga o Wizard de criação (ver figura abaixo), padrão, next > next > finish.



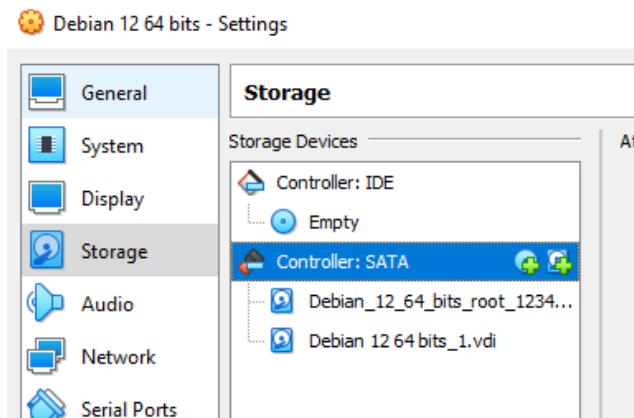
Ao término do Wizard, o seu novo disco estará disponível na lista de discos não conectados (ver figura abaixo), selecione este disco não conectado e clique em “Choose”.



Onde:

1. Selecione o novo disco que foi criado e que não está em uso;
2. Clique no botão Choose para utilizar este disco.

Veja que agora encontra-se 2 discos, feche a configuração e inicie a Virtual Machine.



A ordem acima seria a mesma ordem de conexão SATA de uma placa real, ou seja a **Debian_12_64_bits_root_123456.vdi** está conectada na SATA 0 (zero) e a **Debian 12 64 bits.vdi** está conectada na SATA 1. No livro do Tanenbaum no capítulo de Dispositivos de I/O ele aborda um tema chamado “nomeação uniforme”, aqui vamos mostrar essa teoria explicando na prática.

Um Sistema Operacional não consegui distinguir seu Hard Disk de seu SSD, pois ambos são dispositivos de bloco, então como ele irá nomear os dispositivos? Simples pela posição em que foi ligado na placa, ele conta tudo, o barramento, o conector, etc.. Então o primeiro disco que está na **SATA 0** será chamado de **sda** e o segundo disco que está na **SATA 1** será chamado de **sdb**. Pendrive USB são considerados dispositivos de bloco e são adicionados depois, então neste caso se tiver um pendrive seria **sdc**.

6.2 Dispositivo especial de bloco

Quando ligamos o computador, nosso primeiro trabalho é localizar o disco adicionado, uma maneira rápida de listar os dispositivos de blocos é utilizando o comando **lsblk**, conforme exemplo abaixo.

```
usuario@aied:~$ lsblk
NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda    8:0    0  18G  0 disk
|---sda1  8:1    0 17.5G  0 part /
|---sda2  8:2    0   1K  0 part
`---sda5  8:5    0  509M  0 part [SWAP]
sr0   11:0    1 1024M  0 rom
usuario@aied:~$
```

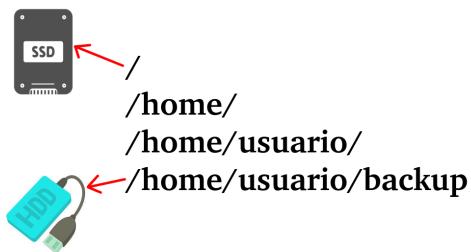
Este comando imprime uma lista de todos os dispositivos de bloco, bem como informações sobre eles, são estas informações:

- **NAME**: nomes de dispositivos;
- **MAJ: MIN**: números de dispositivos principais ou secundários;
- **RM**: Se o dispositivo é removível (1 se sim, 0 se não);
- **SIZE**: O tamanho do dispositivo;
- **RO**: Se o dispositivo é somente leitura;
- **TYPE**: O tipo de dispositivo;
- **MOUNTPOINT**: Ponto de montagem do dispositivo.

Não usamos diretamente um disco, após adicionar um disco criamos partições e formatamos, só depois de formatar é que montamos a partição no sistema de arquivos. Estas partições são indicadas por números, no exemplo da figura abaixo temos 3 partições, são **sda1**, **sda2** e **sda5**. Usaremos a partição **/dev/sda1** como exemplo.

```
usuario@debian:~$ lsblk
NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda    8:0    0   8G  0 disk
└─sda1  8:1    0    7G  0 part /
  └─sda2  8:2    0    1K  0 part
    └─sda5  8:5    0  975M 0 part [SWAP]
sr0   11:0    1 1024M 0 rom
```

Uma partição em um sistema de bloco é uma espécie de limite no dispositivo que informa a cada sistema de arquivos que espaço ele pode ocupar. Diferente do Microsoft Windows, o GNU/Linux monta as diferentes partições dos diferentes dispositivos de bloco em pontos na árvore de diretórios, ou seja, não é letras com C:\, D:\ etc..



Na figura acima um Hard Disk removível foi montado no diretório **/home/usuario/backup**, é uma montagem específica para um usuário e tudo que este usuário salvar neste diretório, na verdade está sendo persistido no disco removível. Lembre-se que no Linux a árvore de arquivo fica parcialmente carregada na memória do computador durante o seu uso.

Todo profissional de TI que pretende chegar a algum lugar de destaque deve aprender a automatizar atividades, ele não encontrará uma solução exata para sua necessidade pois os desenvolvedores não conhecem todas as necessidades específicas de todas as pessoas no mundo, resumindo, você terá que programar. No próximo exemplo de código vamos escrever no output qual partição que está sendo utilizada na montagem da raiz do GNU/Linux via programação C++, será utilizada a biblioteca **optional** para acesso ao método **get_device_of_mount_point()** que retorna o nome da partição, a biblioteca **fstream** para ler o arquivo **/proc/mounts** com **ifstream** e **iostream** para uso do **cout**.

Comece criando um novo arquivo com o nano com o nome **devices.cpp** no diretório **/tmp**, e será compilado com o nome **devices** e será executado. Para codificar o arquivo **devices.cpp** utilize a listagem abaixo.

```

1. #include <iostream>
2. #include <fstream>
3. #include <optional>
4.
5. std::optional<std::string> get_device_of_mount_point(std::string_view path) {
6.     std::ifstream mounts{"/proc/mounts"};
  
```

```

7.     std::string mountPoint;
8.     std::string device;
9.
10.    while (mounts >> device >> mountPoint) {
11.        if (mountPoint == path)
12.            return device;
13.    }
14.    return std::nullopt;
15. }
16.
17. int main()
18. {
19.     if (const auto device = get_device_of_mount_point("/"))
20.         std::cout << *device << "\n";
21.     else
22.         std::cout << "Not found\n";
23. }

```

Veja na figura abaixo o processo de edição (1), compilação (2) e execução (3) do programa.

```

usuario@debian:/tmp$ nano devices.cpp
usuario@debian:/tmp$ g++ -o devices devices.cpp
usuario@debian:/tmp$ ./devices

```

6.4 Tabela de Partição

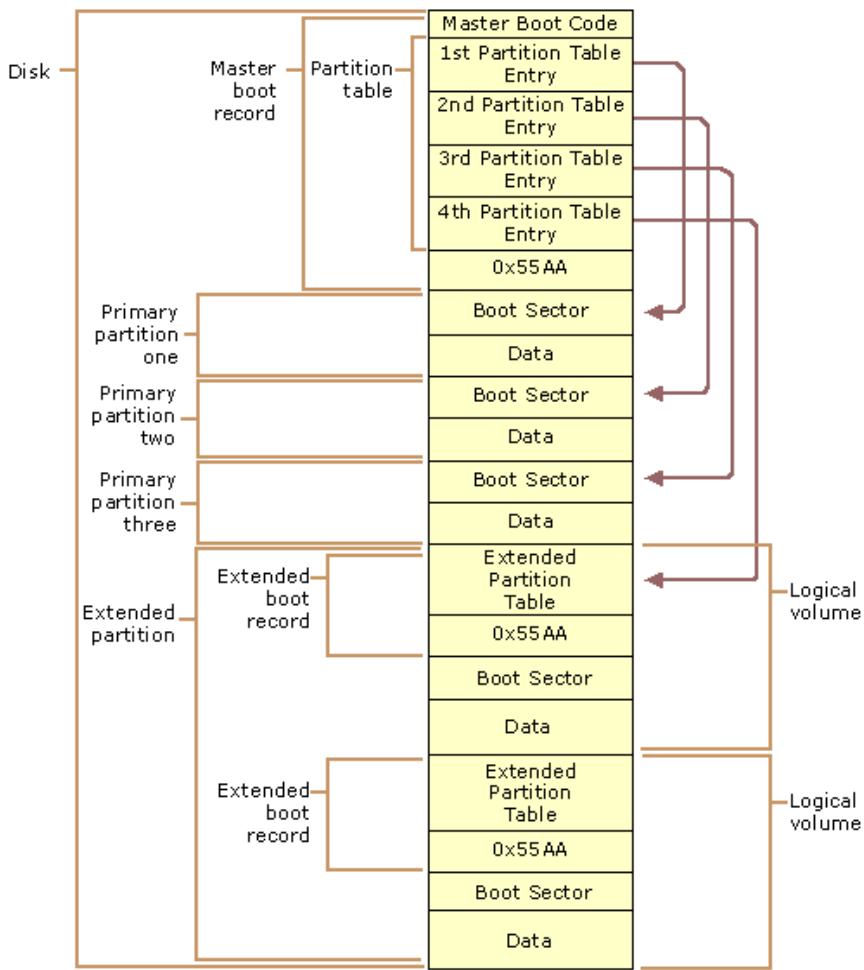
É natural que a formação física de fábrica não atenda a todas as necessidades de formato de todos os Sistemas Operacionais, e por isso realizamos uma formatação lógica sobre a formatação física. E fazemos a gestão de tais espaços por meio de partições.

O primeiro esquema de partições foi definido como MBR e anos depois (para não falar décadas) por questões de segurança e organização um novo modelo de organização de partições foi criado, chama-se GPT.

6.4.1 Master Boot Record

Uma tabela de partição é uma estrutura de dados de 64 bytes que fornece informações básicas para o sistema operacional de um computador sobre a divisão da unidade de bloco, exemplo HDD e SSD, em partições primárias. Uma partição é uma divisão de um dispositivo de bloco em seções logicamente independentes. Tomando como exemplo um HDD único em um computador, este pode ser dividido em partições para melhor gerenciamento, onde, as partições primárias são as primeiras quatro partições em um HDD.

A tabela de partição faz parte do MBR, que é um pequeno programa executado quando um computador é inicializado para localizar o sistema operacional e carregá-lo na memória. O MBR também contém outras informações necessárias para o BIOS. A estrutura do MBR pode ser vista na figura abaixo, repare a limitação de 4 partições primárias.



O MBR e, portanto, a tabela de partição, são armazenados no setor de inicialização, que é o primeiro setor físico de um dispositivo de bloco. Um setor é um segmento de uma trilha em um disco magnético (ou seja, um disquete ou uma bandeja em um disco rígido).

No caso de um disco rígido, uma trilha é qualquer um dos círculos concêntricos na mídia magnética em um disco ou prato sobre o qual uma cabeça magnética passa enquanto a cabeça está parada, mas o disco está girando. Um prato é um disco fino de alumínio ou vidro de alta precisão que é revestido em ambos os lados com um material magnético de alta sensibilidade e que é usado por um disco rígido para armazenar dados.

O MBR lê a tabela de partição para determinar qual partição é a partição ativa. A partição ativa é a partição que contém o sistema operacional que um computador tenta carregar para a memória por padrão quando é inicializado ou reinicializado.

A tabela de partição começa na posição hexadecimal **0x1BE** no setor de inicialização, e contém quatro entradas, cada uma com 16 bytes de comprimento, uma para cada partição.

Uma das primeiras decisões que tomamos ao instalar uma distribuição GNU/Linux é:

- o particionamento de seu disco;
- o sistema de arquivos a ser usado,

- implementar criptografia para segurança que varia com a mudança na arquitetura;
- e plataforma.

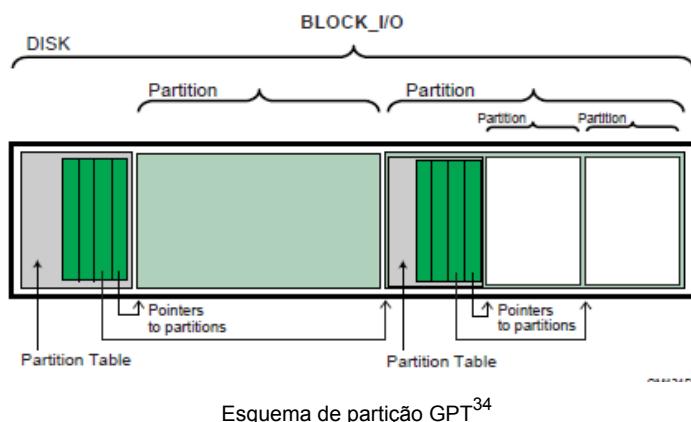
O MBR é um esquema de partição largamente utilizado nas décadas de 80 e 90, mas impõe limitações para os hardwares modernos, por exemplo, o tamanho para endereçamento de bloco e informações possuem limitações de 32 bits. Para Hard Disk com setores de 512 bytes, este esquema armazena em sua tabela no máximo 2 TiB (2 elevado a 2 vezes 512).

6.4.2 EFI system partition, UEFI e GPT

EFI System Partition (Extensible Firmware Interface) é um formato de partição para armazenamento de dados em sistema secundário de armazenamento e é utilizado em computadores que possuem UEFI (Unified Extensible Firmware Interface). Quando o computador realiza o BOOT o firmware UEFI carrega os arquivos armazenados na ESP para inicializar o Sistema Operacional.

O ESP contém o bootloader ou a imagem do kernel para iniciar o Sistema Operacional. Esse processo não é tão trivial, afinal deve-se carregar drivers dos devices, arquivos de configuração. O GUID Partition Table (GPT) é um padrão de layout para tabelas de partções da parte física do computador, faz parte da padronização UEFI no entanto, também é usado para alguns BIOS, devido às limitações das tabelas de partição do registro mestre de inicialização (MBR).

Todos os sistemas operacionais modernos de computadores pessoais oferecem suporte a GPT. Alguns, incluindo macOS e Microsoft Windows na arquitetura x86, suportam inicialização a partir de partções GPT apenas em sistemas com firmware EFI, mas o FreeBSD e a maioria das distribuições Linux podem inicializar a partir de partções GPT em sistemas com BIOS ou interface de firmware EFI.



6.5 Tipos de formatação

Uma distribuição padrão do GNU/Linux oferece a opção de particionar o disco e formatar as partções com os formatos de arquivo listados abaixo, cada um com um significado especial associado a ele:

³⁴ Acessível https://uefi.org/specs/UEFI/2.10/13_Proocols_Media_Access.html

- ext2
- ext3
- ext4
- jfs
- ReiserFS
- XFS
- Btrfs

O comando **mkfs** (Make File System) é utilizado para formatar partições criando sistemas de arquivos, é nestas partições que alocamos diretórios e arquivos. Podemos utilizar este comando para criar partições lógicas em Hard Disk, SSD, M2, Pendrive, etc.. ou seja, todo e qualquer dispositivo especial de bloco.

1. `mkfs [-V] [-t fstype] [fs-options] PATH_DA_PARTIÇÃO [blocks]`

Principais parâmetros são:

- V Usado para exibir status, chama-se modo verbose;
- c Valida a integridade do sistema de arquivos;
- t TIPO Tipo de formatação, exemplo: ext4, vfat, etc..

No passado comandos como **mkfs.ext2** e **mkfs.ext3** chamavam o mkfs parametrizando automaticamente o tipo como **ext2** e **ext3** respectivamente. Na listagem abaixo pode-se observar que existem além do **mkfs** os comandos **mkfs.bfs**, **mkfs.cramfs**, **mkfs.minix** e **mke2fs**.

```
usuario@debian:~$ ls -l /usr/sbin/mkfs*
-rwxr-xr-x 1 root root 14648 Mar 23 2023 /usr/sbin/mkfs
-rwxr-xr-x 1 root root 35136 Mar 23 2023 /usr/sbin/mkfs.bfs
-rwxr-xr-x 1 root root 43256 Mar 23 2023 /usr/sbin/mkfs.cramfs
lrwxrwxrwx 1 root root      6 Mar  5 2023 /usr/sbin/mkfs.ext2 -> mke2fs
lrwxrwxrwx 1 root root      6 Mar  5 2023 /usr/sbin/mkfs.ext3 -> mke2fs
lrwxrwxrwx 1 root root      6 Mar  5 2023 /usr/sbin/mkfs.ext4 -> mke2fs
-rwxr-xr-x 1 root root 112968 Mar 23 2023 /usr/sbin/mkfs.minix
```

Figura 4.

O comando **mke2fs** é usado para criar sistemas de arquivos **ext2**, **ext3** ou **ext4** em um sistema de armazenamento secundário e de operação de bloco, o device é o arquivo especial de bloco correspondente ao dispositivo (por exemplo, **/dev/sda**). O comando **mkfs** faz parte do pacote **util-linux**³⁵ e permite criar sistemas de arquivos de muitas variedades diferentes, já o comando **mke2fs** faz parte do pacote **e2fsprogs** e permite criar sistemas de arquivos **ext2**, **ext3** e **ext4**.

Pomos chamar as opções padrões, simplesmente chamando **mkfs.ext2**, **mkfs.ext3** e **mkfs.ext4**, por exemplo, se chamado como **mkfs.ext3**, um journaling será criado como se a opção **-j** tivesse sido especificada. Os padrões dos parâmetros para o sistema de arquivos recém-criado, se não forem substituídos pelas opções listadas abaixo, são controlados pelo arquivo de configuração **/etc/mke2fs.conf**.

1. `mke2fs [-c | -l filename] [-b block-size] [-f fragment-size] [-g blocks-per-group] [-G number-of-groups] [-i bytes-per-inode] [-l inode-size] [-j] [-J journal-options] [-K] [-N number-of-inodes] [-n] [-m reserved-blocks-percentage] [-o creator-os] [-O feature[,...]]`

³⁵ Detalhes do pacote podem ser observados no link: <https://www.kernel.org/pub/linux/utils/util-linux/>

```
[ -q ] [ -r fs-revision-level ] [ -E extended-options ] [ -v ] [ -F ] [ -L volume-label ] [ -M
last-mounted-directory ] [ -S ] [ -t fs-type ] [ -T usage-type ] [ -U UUID ] [ -V ] device [ 
blocks-count ]
```

-b Os valores de tamanho de bloco válidos são 1024, 2048 e 4096 bytes por bloco. Se omitido, o tamanho do bloco é heuristicamente determinado por o tamanho do sistema de arquivos e o uso esperado do sistema de arquivos (um computador de usuário corresponde a 4KB);

-c Verifique o dispositivo para blocos defeituosos antes de criar o sistema de arquivos;

-i Especifique a relação bytes/inode. mke2fs cria um inode para cada bytes por inodo bytes de espaço no disco. Quanto maior o bytes por inodo proporção, menos inodes serão criados;

-j Crie o sistema de arquivos com um diário ext3. Se o J a opção não é especificada, os parâmetros padrão do diário serão usados para criar um diário de tamanho adequado;

t tipo fs Especifique o tipo de sistema de arquivos (ou seja, ext2, ext3, ext4, etc.) que deve ser criado;

6.5.1 Padrão ext2

O ext foi inicialmente projetado pelo desenvolvedor de software francês Rémy Card³⁶ como um substituto para o sistema de arquivos estendido (ext). Tendo sido projetado de acordo com os mesmos princípios do Berkeley Fast File System da BSD, e o ext foi o primeiro sistema de arquivos de nível comercial para Linux substituindo o MinixFS de Tanenbaum utilizado até a versão 0.63 do Linux (julho de 1992). Logo que foi lançado o ext passou a ter como concorrente o ext2 no Linux 0.99.7 (março de 1993).

A implementação canônica do ext2 é o driver do sistema de arquivos ext2fs no Kernel do Linux. Outras implementações existem no GNU, MINIX 3, alguns kernels BSD, no MiNT e como drivers Microsoft Windows e macOS de terceiros.

O sistema de arquivos ext2 era padrão em várias distribuições GNU/Linux, incluindo Debian e Red Hat Linux, até ser suplantado por ext3, que é quase completamente compatível com ext2 e é um sistema de arquivos com journaling. O padrão ext2 ainda é o sistema de arquivos de escolha para mídia de armazenamento baseada em flash porque a falta de um journaling aumenta o desempenho e minimiza o número de gravações, lembre-se que os dispositivos flash podem suportar um número limitado de ciclos de gravação. Desde 2009, o kernel Linux suporta journaling de e o padrão ext4 fornece benefícios não encontrados no ext2, como arquivos maiores e tamanhos de volume.

O espaço em ext2 é dividido em blocos, e esses blocos, são agrupados em grupos de blocos, análogos aos grupos de cilindros no Sistema de Arquivos, e normalmente, existem milhares de blocos em um grande sistema de arquivos. Os dados de qualquer arquivo normalmente estão contidos em um único grupo de blocos, quando possível. Isso é feito para minimizar o número de buscas de disco ao ler grandes quantidades de dados contínuos.

³⁶ O material original do autor pode ser obtido neste link
<https://web.mit.edu/tytso/www/linux/ext2intro.html>

Tamanho bloco	1 KiB	2 KiB	4 KiB	8 KiB
Max. tamanho arquivo	16 GiB	256 GiB	2 TiB	2 TiB
Max. tamanho do sistema de arquivos	4 TiB	8 TiB	16 TiB	32 TiB

6.5.2 Padrão ext3

O padrão Ext3 é um padrão para o sistema de arquivos com **journaling** que é comumente usado pelo Kernel do Linux e costumava ser o sistema de arquivos padrão para muitas distribuições populares do GNU/Linux.

Desenvolvido Stephen Tweedie que revelou pela primeira vez que estava trabalhando na extensão do ext2 com Journaling em um artigo de 1998, e mais tarde em uma postagem na lista de discussão do kernel de fevereiro de 1999. O sistema de arquivos foi mesclado com o Kernel Linux principal em novembro de 2001.

Sua principal vantagem sobre o ext2 é o registro no journaling, que melhora a confiabilidade e elimina a necessidade de verificar o sistema de arquivos após um desligamento incorreto. O desempenho do ext3 é menos atraente do que os sistemas de arquivos concorrentes do Linux, como ext4, JFS, ReiserFS e XFS, mas o ext3 tem uma vantagem significativa por permitir atualizações do dispositivo que já possui ext2 sem ter que fazer backup e restaurar dados. Os benchmarks sugerem que ext3 também usa menos energia da CPU do que ReiserFS e XFS. O Ext3 também é considerado mais seguro do que os outros sistemas de arquivos Linux (até então), devido à sua relativa simplicidade e base de teste mais ampla.

O padrão ext3 adiciona os seguintes recursos ao padrão ext2:

- Journaling
- Crescimento do sistema de arquivos em tempo de execução;
- Indexação HTree para diretórios maiores

Sem esses recursos, qualquer sistema de arquivo ext3 também é um sistema de arquivo ext2 válido, ou seja, **eram intercambiados sem a necessidade de reformatação**. Esta situação permitiu que utilitários de manutenção de sistema de arquivos bem testados e maduros para manutenção e reparo de sistemas de arquivos ext2 também fossem usados com ext3 sem grandes mudanças. Os sistemas de arquivos ext2 e ext3 compartilham o mesmo conjunto padrão de comandos e utilitários.

Tamanho bloco	1 KiB	2 KiB	4 KiB	8 KiB
Max. tamanho arquivo	16 GiB	256 GiB	2 TiB	2 TiB
Max. tamanho do sistema de arquivos	4 TiB	8 TiB	16 TiB	32 TiB

6.5.3 Padrão ext4

O padrão Ext4 é um sistema de arquivos com journaling para Linux, desenvolvido como o sucessor do ext3. O ext4 era inicialmente uma série de extensões compatíveis com versões anteriores para ext3, muitas delas originalmente desenvolvidas por **Cluster File Systems**, destinadas a estender os limites de armazenamento e adicionar outras melhorias de desempenho. No entanto, outros desenvolvedores do Kernel Linux se opuseram a aceitar extensões para ext3 por razões de estabilidade, e propuseram bifurcar o código-fonte do ext3, renomeá-lo como ext4 e realizar todo o desenvolvimento lá, sem afetar os usuários ext3 existentes. Essa proposta foi aceita e, em 28 de junho de 2006 pelo mantenedor do ext3, anunciou o novo plano de desenvolvimento do ext4.

Uma versão preliminar de desenvolvimento do ext4 foi incluída na versão 2.6.19 do Kernel Linux. Em 11 de outubro de 2008, os patches que marcam o ext4 como código estável foram mesclados nos repositórios de código-fonte do Linux 2.6.28, denotando o fim da fase de desenvolvimento e recomendando a adoção do ext4. Kernel 2.6.28, contendo o sistema de arquivos ext4, foi finalmente lançado em 25 de dezembro de 2008. Em 15 de janeiro de 2010, o Google anunciou que iria atualizar sua infraestrutura de armazenamento de ext2 para ext4. Em 14 de dezembro de 2010, o Google também anunciou que usaria ext4, em vez de YAFFS, no Android 2.3.

O sistema de arquivos ext4 pode suportar volumes com tamanhos de até 1 EiB e arquivos únicos com tamanhos de até 16 TiB com o tamanho de bloco padrão de 4 KiB. Os limites máximos de tamanho de arquivo, diretório e sistema de arquivos aumentam pelo menos proporcionalmente com o tamanho do bloco do sistema de arquivos até o tamanho máximo de bloco de 64 KiB disponível em CPUs ARM.

As extensões substituem o esquema de mapeamento de bloco tradicional usado pelo ext2 e ext3. Uma extensão é uma gama de blocos físicos contíguos, melhorando o desempenho de arquivos grandes e reduzindo a fragmentação. Uma única extensão no ext4 pode mapear até 128 MiB de espaço contíguo com um tamanho de bloco de 4 KiB.

O ext4 não limita o número de subdiretórios em um único diretório, exceto pelo limite de tamanho inerente do próprio diretório, para permitir diretórios maiores e desempenho contínuo, ext4 no Linux 2.6.23 e posterior ativa os índices HTree³⁷ (uma versão especializada de uma B-tree) por padrão, que permite que diretórios de até aproximadamente 10 a 12 milhões de entradas sejam armazenados no índice HTree de 2 níveis.

6.7 Praticando leitura de blocos (falta)

³⁷ Um bom artigo sobre HTree <https://dl.acm.org/doi/pdf/10.1145/141484.130307>

```

1. #include <stdio.h>
2. #include <linux/fs.h>
3. #include <sys/types.h>
4. #include <sys/stat.h>
5. #include <fcntl.h>
6.
7. #define SECTOR_NO 10 /*read 10th sector*/
8.
9. int main()
10. {
11.     int sector_size;
12.     char *buf;
13.     int n = SECTOR_NO;
14.
15.     int fd = open("/dev/sdal", O_RDONLY|O_NONBLOCK);
16.     ioctl(fd, BLKSSZGET, &sector_size);
17.     printf("%d\n", sector_size);
18.     lseek(fd, n*sector_size, SEEK_SET);
19.
20.     buf = malloc(sector_size);
21.     read(fd, buf, sector_size);
22.
23.     return 0;
24. }

```

6.8 Acesso direto aos arquivos e Journaling

(será escrito na próxima versão)

6.9 Listando dispositivos especiais de bloco com lsblk

Comando **lsblk** é usado para exibir detalhes sobre dispositivos de bloco e esses dispositivos de bloco são basicamente aqueles arquivos especiais que representam dispositivos conectados ao computador e que operam em blocos. Ele consulta o sistema de arquivos virtual **sys** e **udev** (Linux dynamic device management)³⁸ para obter informações do Sistema Operacional. E basicamente exibe a saída em uma estrutura semelhante a uma árvore. Este comando vem pré-instalado com o pacote **util-linux**.

Muitas distribuições do GNU/Linux não têm o comando **lsblk** pré-instalado, para instalá-lo, use o seguinte comando:

```
1. sudo apt install util-linux -y
```

Para exibir os dispositivos especiais de bloco é simples, execute o comando **lsblk**:

```
1. sudo lsblk
```

Ele exibe a lista de dispositivos de bloco em seu sistema.

³⁸ Recomendo a leitura da documentação oficial: <https://wiki.debian.org/udev>

```
usuario@aied:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0 18G  0 disk
└─sda1   8:1    0 17.5G 0 part /
└─sda2   8:2    0 1K   0 part
└─sda5   8:5    0 509M 0 part [SWAP]
sdb      8:16   0   8G  0 disk
sr0     11:0    1 1024M 0 rom
usuario@aied:~$
```

Figura 4.

Para exibir dispositivos de bloco vazios também:

```
1. sudo lsblk -a
```

Isso exibirá todos os dispositivos de bloco junto com os vazios. Para imprimir informações de tamanho em bytes.

```
1. sudo lsblk -b
```

Ele exibe todos os dispositivos de bloco junto com seus tamanhos em bytes. Para imprimir o modelo de zona para dispositivos.

```
1. sudo lsblk -z
```

Para pular as entradas do escravo.

```
1. sudo lsblk -d
```

Para evitar-escravos isso irá ignorar todas as entradas do escravo. Para usar caracteres ASCII para formatação de árvore.

```
1. sudo lsblk -i
```

Isso exibirá a formatação da árvore com valores de caracteres ASCII. Para imprimir informações sobre o proprietário do dispositivo, grupo e modo de dispositivos de bloqueio.

```
1. sudo lsblk -m
```

Isso mostrará todos os detalhes do proprietário do dispositivo, grupo e modo de dispositivos de bloqueio. Para imprimir colunas selecionadas de dispositivos de bloco.

```
1. sudo lsblk -o SIZE, NAME, MOUNTPOINT
```

Isso imprimirá apenas as colunas especificadas. Para ocultar os títulos das colunas.

```
1. sudo lsblk -dn
```

6.10 Obtendo informações sobre dispositivos de Bloco com blkid

O comando blkid é a interface de linha de comando para trabalhar com a biblioteca libblkid do Sistema Operacional. Ele pode determinar o tipo de conteúdo (por exemplo, sistema de arquivos, swap) que um dispositivo de bloco contém e também atributos tais como tokens, NAME, pares de valores dos metadados. Um destes valores do metadado que vamos utilizar na prática é o UUID.

O blkid tem duas formas principais de operação: ou procurando por um dispositivo com um par de valor, ou exibindo pares para um ou mais dispositivos. Veja a saída do comando.

```
usuario@aied:~$ sudo blkid
[sudo] password for usuario:
/dev/sda1: UUID="7415f2b1-d281-4f80-a1be-8f7525afba57" TYPE="ext4" PARTUUID="a8c50257-01"
/dev/sda5: UUID="0fee3085-04a4-4d70-9464-39838a0a4f31" TYPE="swap" PARTUUID="a8c50257-05"
usuario@aied:~$
```

NO próximo exemplo vamos usar o head blkid.h para obter o nome o ponto de montagem de uma partição. Usamos este head para ter acesso as primitivas blkid_probe_get_partitions() e blkid_partlist_numof_partitions(). Crie um novo arquivo com o nano chamado getuuid.c e codifique a listagem abaixo.

```
1. #include <stdio.h>
2. #include <string.h>
3. #include <err.h>
4. #include <blkid/blkid.h>
5.
6. int main (int argc, char *argv[]) {
7.     blkid_probe pr = blkid_new_probe_from_filename(argv[1]);
8.     if (!pr) {
9.         err(1, "Failed to open %s", argv[1]);
10.    }
11.
12.    // Get number of partitions
13.    blkid_partlist ls;
14.    int nparts, i;
15.
16.    ls = blkid_probe_get_partitions(pr);
17.    nparts = blkid_partlist_numof_partitions(ls);
18.    printf("Number of partitions:%d\n", nparts);
19.
20.    if (nparts <= 0) {
21.        printf("Please enter correct device name! e.g. \"/dev/sdc\"\n");
22.        return;
23.    }
24.
25.    // Get UUID, label and type
26.    const char *uuid;
27.    const char *label;
28.    const char *type;
29.
30.    for (i = 0; i < nparts; i++) {
```

```

31.     char dev_name[20];
32.     sprintf(dev_name, "%s%d", argv[1], (i+1));
33.     pr = blkid_new_probe_from_filename(dev_name);
34.     blkid_do_probe(pr);
35.     blkid_probe_lookup_value(pr, "UUID", &uuid, NULL);
36.     blkid_probe_lookup_value(pr, "LABEL", &label, NULL);
37.     blkid_probe_lookup_value(pr, "TYPE", &type, NULL);
38.     printf("Name=%s, UUID=%s, LABEL=%s, TYPE=%s\n", dev_name, uuid, label, type);
39. }
40. blkid_free_probe(pr);
41. return 0;
42. }
```

Mas para compilar precisa-se instalar no Debian a biblioteca de apoio para programação libblkid-dev, para isso use o comando apt conforme listagem abaixo.

```
1. sudo apt install libblkid-dev -y
```

Para compilar com GCC deve-se adicionar a shared library -lblkid conforme comando abaixo, depois poderá executar naturalmente seu novo programa.

```
1. gcc -o getuuid getuuid.c -lblkid
```

6.11 Arquivos partitions, mounts e mtab

O arquivo **/proc/partitions** contém informações sobre cada partição atualmente anexada ao sistema. Como **/proc/mounts**, não é um arquivo real, mas parte do sistema de arquivos virtuais. O formato de **/proc/partitions** contém colunas e cada coluna é a seguinte:

major Representa a classe do dispositivo para que possa ser mapeada para um driver apropriado;

minor Separa partições em dispositivos físicos. Isso corresponde ao número no final do nome da partição;

#blocks Quantos blocos físicos a partição possui (ou ocupa);

name O nome da partição.

O exemplo do arquivo **/proc/partitions**:

```
usuario@debian:~$ cat /proc/partitions
major minor  #blocks  name

      8        0    8388608  sda
      8        1    7387136  sda1
      8        2        1  sda2
      8        5   998400  sda5
     11        0   1048575  sr0
```

O arquivo **/proc/mounts** lista o status de todos os sistemas de arquivos atualmente montados em um formato semelhante ao fstab, o ponto de montagem, o tipo de sistema de arquivos, etc. Na verdade, não é um arquivo real, mas parte do sistema de arquivos virtual que representa o status de objetos montados conforme relatado pelo Kernel do Linux.

```
usuario@debian:~$ cat /proc/mounts
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
udev /dev devtmpfs rw,nosuid,relatime,size=472816k,nr_inodes=118204,mode=755,inode64
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,nosuid,nodev,noexec,relatime,size=98380k,mode=755,inode64 0 0
/dev/sda1 / ext4 rw,relatime,errors=remount-ro 0 0
securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev,inode64 0 0
tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k,inode64 0 0
cgroup2 /sys/fs/cgroup cgroup2 rw,nosuid,nodev,noexec,relatime nsdelegate memory,reci
```

O arquivo **/etc/mtab** é muito semelhante ao arquivo **/proc/mounts**, pois informa o status dos sistemas de arquivos montados atualmente. No entanto, **/proc/mounts** é tipicamente mais preciso e inclui informações mais atualizadas sobre os sistemas de arquivos.

```
usuario@debian:~$ cat /etc/mtab
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
udev /dev devtmpfs rw,nosuid,relatime,size=472816k,nr_inodes=118204,mode=755,inode64
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,pt
tmpfs /run tmpfs rw,nosuid,nodev,noexec,relatime,size=98380k,mode=
/dev/sda1 / ext4 rw,relatime,errors=remount-ro 0 0
securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,
tmpfs /dev/shm tmpfs rw,nosuid,nodev,inode64 0 0
tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k,i
```

6.12 Criando partições em discos com fdisk

A primeira coisa que você precisa fazer após instalar fisicamente um novo SSD ou disco rígido é particioná-lo. Uma unidade precisa ter pelo menos uma partição para que você possa formatá-la e armazenar arquivos nela. No GNU/Linux, existem várias ferramentas que você pode usar para criar partições, **fdisk** é a mais usada.

O comando **fdisk** é um utilitário de linha de comando baseado em menu que permite criar e manipular tabelas de partição em um disco rígido. Esteja ciente de que **fdisk** é uma ferramenta perigosa e deve ser usada com extremo cuidado. Apenas root ou usuários com privilégios ou uso do comando **sudo** podem manipular as tabelas de partição.

6.12.1 Listar Partições no GNU/Linux

Para listar a tabela de partição de um dispositivo, invoque o comando **fdisk** com a opção **-l**, seguido do nome do dispositivo. Por exemplo, para listar a tabela de partição e partições de **/dev/sda** que você executaria:

```
1. sudo fdisk -l /dev/sda
```

Veja a saída:

```
usuario@aied:~$ sudo fdisk -l /dev/sda
Disk /dev/sda: 18 GiB, 19327352832 bytes, 37748736 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xa8c50257

Device      Boot   Start     End   Sectors  Size Id Type
/dev/sda1        *      2048 36702207 36700160 17.5G 83 Linux
/dev/sda2          36704254 37746687 1042434 509M  5 Extended
/dev/sda5          36704256 37746687 1042432 509M 82 Linux swap / Solaris
usuario@aied:~$ _
```

Quando nenhum dispositivo é fornecido como argumento, o fdisk irá imprimir as tabelas de partição de todos os dispositivos listados no arquivo **/proc/partitions**.

```
usuario@aied:~$ sudo fdisk -l
Disk /dev/sda: 18 GiB, 19327352832 bytes, 37748736 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xa8c50257

Device      Boot   Start     End   Sectors  Size Id Type
/dev/sda1        *      2048 36702207 36700160 17.5G 83 Linux
/dev/sda2          36704254 37746687 1042434 509M  5 Extended
/dev/sda5          36704256 37746687 1042432 509M 82 Linux swap

Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

6.12.2 Criando Tabela de Partição

Para iniciar o particionamento da unidade, execute fdisk com o nome do dispositivo. Neste exemplo, trabalharemos em **/dev/sdb**, lembrando que este é o disco recém adicionado.

1. `sudo fdisk /dev/sdb`

O prompt de comando mudará e a caixa de diálogo fdisk onde você pode digitar os comandos será aberta:

```
usuario@aied:~$ sudo fdisk /dev/sdb
Welcome to fdisk (util-linux 2.33.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x635dba20.
Command (m for help): _
```

Caso não saiba quais são as opções ou não lembra (normal), digite **m** e pressione ENTER.

```

d  delete a partition
F  list free unpartitioned space
I  list known partition types
n  add a new partition
p  print the partition table
t  change a partition type
v  verify the partition table
i  print information about a partition

Misc
m  print this menu
u  change display/entry units
x  extra functionality (experts only)

Script
I  load disk layout from sfdisk script file
O  dump disk layout to sfdisk script file

Save & Exit
w  write table to disk and exit
q  quit without saving changes

Create a new label
g  create a new empty GPT partition table
G  create a new empty SGI (IRIX) partition table
o  create a new empty DOS partition table
s  create a new empty Sun partition table

Command (m for help): ...

```

Se você estiver particionando uma nova unidade, antes de começar a criar partições, é necessário criar uma tabela de partição. **ATENÇÃO:** Pule esta etapa se o dispositivo já tiver uma tabela de partição e você quiser mantê-la.

O comando **fdisk** suporta vários esquemas de particionamento, MBR e GPT são os dois padrões de esquema de partição mais populares, que armazenam as informações de particionamento em uma unidade de uma maneira diferente. GPT é um padrão mais recente que permite e tem muitas vantagens em relação ao MBR. Os principais pontos a serem considerados ao escolher qual padrão de particionamento usar:

- Use MBR para inicializar o disco no modo BIOS legado.
- Use GPT para inicializar o disco no modo UEFI.
- O padrão MBR suporta a criação de uma partição de disco de até 2 TiB. Se você tiver um disco de 2 TiB ou maior, use GPT.
- MBR tem um limite de 4 partições primárias. Se você precisar de mais partições, uma das partições primárias pode ser definida como uma partição estendida e conter partições lógicas adicionais. Com o GPT, você pode ter até 128 partições. GPT não oferece suporte a partições estendidas ou lógicas.

Para criar uma nova partição utiliza-se o comando **n** (conforme figura abaixo), o **fdisk** irá questionar se deseja uma partição primária ou estendida. Como foi solicitado (figura abaixo) uma partição primária com a opção: **p**, o **fdisk** então questiona sobre qual partição primária, pela teoria temos possibilidade de 1 a 4. O sistema é interativo, irá lhe questionar, repare dentro do parênteses que há uma escolha padrão caso não informe nenhum valor.

```

Uma nova particao
Estou usando uma primaria

Command (m for help): n
Partition type
  p  primary (0 primary, 0 extended, 4 free)
  e  extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-16777215, default 2048):
Last sector, +/sectors or +/-size{K,M,G,T,P} (2048-16777215, defau

Created a new partition 1 of type 'Linux' and of size 8 GiB.
Command (m for help): 

Depois utilizando as opcoes padroes

```

Termine a operação com o comando de escrita **w**, o comando **fdisk** irá escrever as tabelas de partições.

Escrevendo as alterações

```

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

usuario@aied:~$
```

A tabela de partição é escrita

6.12 Formatando uma partição para uso com mkfs

O comando **mkfs** é usado para criar um sistema de arquivos ext2, ext3 ou ext4, geralmente em uma partição de disco. dispositivo é o arquivo especial correspondente ao dispositivo. Se parâmetros forem omitidos, o **mkfs** calcula automaticamente o tamanho do sistema de arquivos e faz a formatação.

Opções:

- **-b tamanho do bloco** Especifique o tamanho dos blocos em bytes. Os valores de tamanho de bloco válidos são 1024, 2048 e 4096 bytes por bloco. Se omitido, o tamanho do bloco é determinado heuristicamente pelo tamanho do sistema de arquivos e o uso esperado do sistema de arquivos;
- **-c** verifica se há blocos danificados no dispositivo antes de criar o sistema de arquivos. Se esta opção for especificada duas vezes, um teste de leitura e gravação mais lento será usado em vez de um teste rápido de somente leitura.
- **-q** Execução silenciosa;
- **-t fs-type** Especifique o tipo de sistema de arquivos (ou seja, ext2, ext3, ext4, etc.) que deve ser criado;
- **-U UUID** Crie o sistema de arquivos com o UUID especificado.
- **-v** execução detalhada (verboso).

Mas antes tenha certeza de qual arquivo especial de bloco usará na formatação, lembre-se que é um processo que deverá alterar as tabelas de referência i-node.

Usando o comando ls para exibir particao criada

```
usuario@aied:~$ ls -l /dev/sd*
brw-rw---- 1 root disk 8,  0 Mar 12 11:58 /dev/sda
brw-rw---- 1 root disk 8,  1 Mar 12 11:58 /dev/sda1
brw-rw---- 1 root disk 8,  2 Mar 12 11:58 /dev/sda2
brw-rw---- 1 root disk 8,  5 Mar 12 11:58 /dev/sda5
brw-rw---- 1 root disk 8, 16 Mar 12 12:06 /dev/sdb
brw-rw---- 1 root disk 8, 17 Mar 12 12:06 /dev/sdb1
usuario@aied:~$
```

No comando abaixo o **mkfs** está sendo definido com **.ext4** e apontando para **/dev/sdb1** que foi recentemente criado no fdisk.

Use o comando mkfs para formatar o padrao ext4 e aponte para /dev/sdb1

```
usuario@aied:~$ sudo mkfs.ext4 /dev/sdb1
mke2fs 1.44.5 (15-Dec-2018)
Creating filesystem with 2096896 4K blocks and 524288 inodes
Filesystem UUID: d0d6f1bc-cf22-463d-bd4a-63368be8ac08
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

usuario@aied:~$
```

Informacoes sobre i-node e super bloco

6.13 Montando e desmontando um dispositivo no Sistema de Arquivos com mount e umount

O disco existe, foi criado a partição e foi formatado, mas ele não é acessível. Para acessar é preciso criar no Sistema de Arquivos um ponto de montagem, digamos que o objetivo deste novo disco seja **backups**, então crie na raiz **/backup** com **mkdir**. Como no futuro ele ficará fixo ele não será montado em **/mnt** e sim diretamente na raiz.

```
usuario@aied:~$ sudo mkdir /backup
usuario@aied:~$
```

Para realizar a montagem, você precisa apenas saber a extensão do dispositivo e o nome do arquivo especial de bloco recentemente formatado, o comando é simples conforme figura abaixo.

```
usuario@aied:~$ 
usuario@aied:~$ sudo mount -t ext4 /dev/sdb1 /backup
usuario@aied:~$
```

Confirme utilizando o comando **df**, conforme figura abaixo.

O comando df exibe informações de dispositivos

```
usuario@aied:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            480M   0    480M  0% /dev
tmpfs           99M  3.0M   96M  3% /run
/dev/sda1        18G  2.1G   15G 13% /
tmpfs           494M   0    494M  0% /dev/shm
tmpfs           5.0M   0    5.0M  0% /run/lock
tmpfs           494M   0    494M  0% /sys/fs/cgroup
tmpfs           99M   0    99M  0% /run/user/1000
/dev/sdb1        7.9G  36M  7.4G 1% /backup
usuario@aied:~$
```

Montagem feita em /backup

Caso não queira mais o disco ou queira removê-lo, faça a desmontagem utilizando o comando `umount`, basta informar o ponto de montagem que será removida do Sistema de Arquivos, conforme figura abaixo.

```
usuario@aied:~$ sudo umount /backup
usuario@aied:~$ 
usuario@aied:~$ 
usuario@aied:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            480M   0    480M  0% /dev
tmpfs           99M  3.0M   96M  3% /run
/dev/sda1        18G  2.1G   15G 13% /
tmpfs           494M   0    494M  0% /dev/shm
tmpfs           5.0M   0    5.0M  0% /run/lock
tmpfs           494M   0    494M  0% /sys/fs/cgroup
tmpfs           99M   0    99M  0% /run/user/1000
```

Se estiver executando o comando `mount` de um cdrom (exemplo `sr0`), não use tipo, afinal o CD que irá ser carregado possui um padrão internacional e independe de Sistema Operacional, conforme explicado no livro do autor Tanenbaum (Sistemas Operacionais Modernos).

6.14 O arquivo /etc/fstab

O arquivo **/etc/fstab** é um dos arquivos mais importantes em um sistema baseado em Linux, uma vez que armazena informações estáticas sobre sistemas de arquivos, seus pontos de montagem e opções de montagem. A primeira coisa que devemos saber sobre este arquivo é que ele deve ser lido apenas por programas e nunca escrito, exceto pelo administrador do sistema em raras necessidades.

Cada linha no arquivo descreve um sistema de arquivos e contém campos usados para fornecer informações sobre seu ponto de montagem, as opções que devem ser usadas ao montá-lo, etc. e cada campo pode ser separado por outro por espaços ou tabulações.

O arquivo é /etc/fstab e é protegido

```

GNU nano 3.2          /etc/fstab          Modified
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options>      <dump> <pass>
# / was on /dev/sda1 during installation
UUID=7415f2b1-d281-4f80-a1be-8f7525afba57 /          ext4      errors=remoun$#
# swap was on /dev/sda5 during installation
UUID=0fee3085-04a4-4d70-9464-39838a0a4f31 swap      sw      $#
/dev/sr0      /media/cdrom0  udf,iso9660 user,noauto  0      0
#
UUID=d0d6f1bc-cf22-463d-bd4a-63368be8ac08 /backup ext4 defaults 0 0

```

Tem que adicionar o UUID, ponto de montagem e padrão de formatação

O primeiro campo em cada entrada fstab contém informações sobre o dispositivo de bloco local ou remoto que deve ser montado. A maneira mais comum de fazer referência a um dispositivo de bloco é usando seu nó dentro do diretório **/dev**, por exemplo, para fazer referência à primeira partição do dispositivo **sda** usamos **/dev/sda1**. Na imagem acima podemos ver esse tipo de montagem para o **/dev/sr0**, trata-se do CD-ROM. O problema desta abordagem é que estes nomes são uniformes, conforme já discutido, então a troca da posição da conexão na placa mãe pode alterar este nome. Lembrando que o CD-ROM está nessa imagem somente para exibição, recomendo que use o **mount** e o **umount**.

Formas alternativas de referenciar um dispositivo de bloco são usando seu **LABEL** ou **UUID**(Universal Unique Identifier). O **UUID** é o método absolutamente preferido, uma vez que garante a referência univocamente a um sistema de arquivos, como seu nome indica. Para obter informações sobre os sistemas de arquivos, podemos executar o **lsblk**.

Use o **blkid** para obter o **UUID** das particoes

```

usuario@aied:~$ sudo blkid
/dev/sda1: UUID="7415f2b1-d281-4f80-a1be-8f7525afba57" TYPE="ext4" PARTUUID="a8c50257-01"
/dev/sda5: UUID="0fee3085-04a4-4d70-9464-39838a0a4f31" TYPE="swap" PARTUUID="a8c50257-05"
/dev/sdb1: UUID='d0d6f1bc-cf22-463d-bd4a-63368be8ac08' TYPE="ext4" PARTUUID="635dba20-01"

```

Identificação da partição, tem que anotar certo

Em cada entrada do fstab, **o segundo campo** especifica o ponto de montagem no sistema de arquivos: qual diretório no sistema deve ser usado para acessar seu conteúdo, para esta prática será utilizado o **/backup** recentemente criado.

O terceiro campo de uma entrada fstab especifica o tipo de sistema de arquivos em uso no dispositivo de bloco bruto ou partição. O sistema de arquivos deve estar entre os suportados pelo sistema operacional como, por exemplo: ext3, ext4, xfs etc.

O quarto campo de cada entrada no arquivo fstab é usado para fornecer uma lista de opções a serem usadas ao montar o sistema de arquivos. Para usar o conjunto padrão de opções de montagem, especificamos default como um valor. As opções padrão são:

- **rw**: ler escrever;
- **suid**: respeitar os bits setuid e setgid ;
- **dev**: interpretar caracteres e bloquear dispositivos no sistema de arquivos;
- **exec**: permite a execução de binários e scripts;
- **auto**: monte o sistema de arquivos quando a opção -a do comando mount for usada;
- **nouser**: tornar o sistema de arquivos não montável por um usuário padrão;
- **async**: executa operações de I/O no sistema de arquivos de forma assíncrona.

O quinto campo em cada entrada pode ser 0 ou 1. O valor é usado pelo programa de backup de despejo (se instalado) para saber qual sistema de arquivos deve ser despejado.

O sexto campo é usado para estabelecer a ordem pela qual outro utilitário, fsck deve verificar os sistemas de arquivos na inicialização. O valor de 1 deve sempre ser usado para o sistema de arquivos raiz; para todos os outros que podemos usar 2 (se queremos a verificação). Se este valor não for fornecido, o padrão é 0 e o sistema de arquivos não será verificado.

Na figura abaixo em destaque temos uma nova entrada relacionada ao disco recém adicionado **/dev/sdb** que foi criada a partição **/dev/sdb1** e formatada com ext4. Será montada sempre que o GNU/Linux for iniciado no diretório **/backup** no sistema de arquivos.

O arquivo é /etc/fstab e é protegido

```

GNU nano 3.2          /etc/fstab          Modified
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options>      <dump> <pass>
# / was on /dev/sda1 during installation
UUID=7415f2b1-d281-4f80-a1be-8f7525afba57 /          ext4      errors=remoun$ 
# swap was on /dev/sda5 during installation
UUID=0fee3085-04a4-4d70-9464-39838a0a4f31 swap      sw      $ 
/dev/sr0      /media/cdrom0 udf,iso9660 user,noauto      0      0
UUID=d0d6f1bc-cf22-463d-bd4a-63368be8ac08 /backup ext4 defaults 0 0

```

Tem que adicionar o UUID, ponto de montagem e padrão de formatação

Execute um reboot, e ao reiniciar o GNU/Linux veja que agora automaticamente está montado no diretório **/backup**.

```
Usuario@aied:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            480M    0  480M  0% /dev
tmpfs           99M  1.7M  98M  2% /run
/dev/sda1        18G  2.1G  15G  13% /
tmpfs           494M    0  494M  0% /dev/shm
tmpfs           5.0M    0  5.0M  0% /run/lock
tmpfs           494M    0  494M  0% /sys/fs/cgroup
/dev/sdb1        7.9G  36M  7.4G  1% /backup
tmpfs           99M    0  99M  0% /run/user/1000
usuario@aied:~$ _
```

Em caso de falha é possível analisar o ocorrido, basta executar um CAT no arquivo **/var/log/syslog** e aplicar um filtro com grep, conforme exemplo abaixo.

```
well@wpo:~$ cat /var/log/syslog | grep sdb1
Nov 22 11:11:44 wpo kernel: [  3.848931]  sdb: sdb1
Nov 22 11:13:48 wpo kernel: [  3.831318]  sdb: sdb1
Nov 22 11:13:48 wpo kernel: [  13.289210] EXT4-fs (sdb1): Unrecognized mount option "default" or missing value
Nov 22 11:17:04 wpo kernel: [  13.370826] EXT4-fs (sdb1): Unrecognized mount option "default" or missing value
Nov 22 11:17:04 wpo kernel: [  13.370826] EXT4-fs (sdb1): Unrecognized mount option "default" or missing value
Nov 22 11:17:04 wpo kernel: [  13.370826] EXT4-fs (sdb1): mounted filesystem with ordered data mode. Opts: (null).
Nov 22 11:18:17 wpo kernel: [  86.836910] EXT4-fs (sdb1): mounted filesystem with ordered data mode. Opts: (null).
Nov 22 11:31:29 wpo kernel: [  14.936071] EXT4-fs (sdb1): Unrecognized mount option "default" or missing value
Nov 22 11:31:29 wpo kernel: [  14.936071] EXT4-fs (sdb1): Unrecognized mount option "default" or missing value
well@wpo:~$
```

6.15 Prática do capítulo de Formatação

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

6.15.1 Prática 8b65b431 01: Adicionando um novo disco

Nesta prática o aluno deve ter a expertise para adicionar um novo disco, formatar e montar no **/etc/fstab**, mas antes o aluno deve ter o comando sudo devidamente instalado.

O aluno deve:

1. Adicione um novo disco na Virtual Machine pelo VirtualBox;
2. Crie uma partição primária no novo disco;
3. Formate a nova partição com padrão ext4;
4. Altera o arquivo **/etc/fstab** para durante a inicialização do GNU/Linux seja feita a montagem em **/backup**, para editar o arquivo **/etc/fstab** use o sudo e o nano;
5. Reinicie a máquina.

Execute o robô de validação aied conforme comando abaixo.

```
1. sudo aied validar 8b65b431 checkpoint01
```

O resultado esperado é próximo ao da figura abaixo, isso pois o texto é dinâmico e eventuais testes podem ser adicionados.

```

usuario@aied:~$ sudo aied validar 8b65b431 checkpoint01
Ação que será executada: validar

Prática: Prática da aula de formatação
Será enviado o OUTPUT do comando: cat /etc/fstab
Será enviado o OUTPUT do comando: df -h

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: cat /etc/fstab
1.1   Validar por text          /backup

2 - Comando: df -h
2.1   Validar por text          /dev/sdb1

Total de acertos: 2 do total de 2 validações equivalentes a 100%.
Identificação: 868f3e877d
AIED v(5)

```

6.15.2 Prática 8b65b431 02: Adicionando um cdrom

Nesta prática o aluno deve ter a expertise para usar o comando de montagem e montar um CD no CD-ROM já existente, essa montagem será feita no diretório **/home/userlinux/cdrom**.

O aluno deve:

1. Faça download da iso nesta url:
(<https://drive.google.com/file/d/1xk-6ZDSbB8d6JZzSqy7NjUyM6p3xBLHU/view?usp=sharing>)
2. Configure o VirtualBox para abrir no CDROM esta imagem iso;
3. Crie o diretório **/home/userlinux/cdrom**
4. Monte o CD no CD-ROM **/dev/sr0** em **/home/userlinux/cdrom/**, para isso use o mount.

Execute o robô de validação aied conforme comando abaixo.

```
1. sudo aied validar 8b65b431 checkpoint02
```

O resultado esperado é próximo ao da figura abaixo, isso pois o texto é dinâmico e eventuais testes podem ser adicionados.

```

usuario@debian:~/cdrom$ sudo aied validar 8b65b431 checkpoint02
Ação que será executada: validar

Prática: Prática da aula de formatação - Cdrom
Será enviado o OUTPUT do comando: df -h
Será enviado o OUTPUT do comando: cat /home/usuario/cdrom/arquivo.txt

Deseja continuar? (s para SIM) s
s1 - Comando: df -h
1.1   Validar por regex          / /dev /sr0(.*)? /home /usuario /cdrom/
2 - Comando: cat /home/usuario/cdrom/arquivo.txt
2.1   Validar por text          aiedonline

Total de acertos: 2 do total de 2 validações equivalentes a 100%.
Identificação: 65c1e51de3
AIED v(8)

```

7 Gerenciamento de Processos no GNU/Linux

Arquivos executáveis são arquivos regulares que se situam em sistemas secundários de armazenamento, tal como Hard Disk, SSD e mídias removíveis. Os arquivos regulares se subdividem em 2 grupos, são estes:

- Arquivos repositório de dados;
- Arquivos executáveis;

O que diferencia um arquivo executável de um arquivo de dados de um executável é que este possui um bit-mágico ativo e o Sistema Operacional comprehende que este arquivo deve ser tratado de forma diferente pelo seu mecanismo de carregamento.

Um arquivo executável, também chamado de executável ou binário, é a forma pronta para execução, e este é uma sequência de instruções comprehensíveis pela CPU de um computador, e estas instruções é o que indica quais operações do computador devem realizar-se num conjunto de dados.

Um arquivo de dados é uma coleção nomeada de dados relacionados que aparece para o usuário como um único bloco de dados contínuo e que é retido no armazenamento. Armazenamento refere-se a dispositivos de computador ou mídia que podem reter dados por períodos relativamente longos de tempo. Isso contrasta com a memória, que retém seu conteúdo apenas brevemente e que fisicamente consiste em RAM.

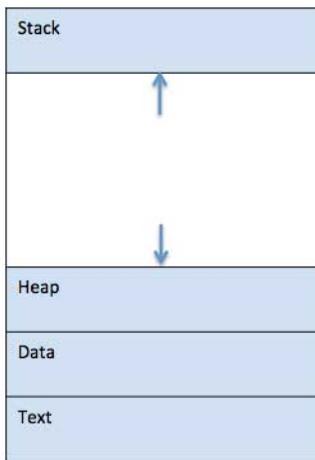
Os arquivos executáveis consistem em instruções que foram traduzidas de seu código-fonte original (um código humano) em código de máquina, também chamado de linguagem de máquina ou código-objeto por meio do uso de um programa especializado chamado **compilador** para que a CPU possa usá-los diretamente e sem tradução adicional. O código de máquina consiste inteiramente em zeros e uns, que representam os estados desligados e ligado aos circuitos lógicos da CPU e das células de memória.

Este material não demonstra o processo interno de compilação, somente os passos na visão de um desenvolvedor, recomenda-se o livro **Compilers: Principles, Techniques, and Tools**³⁹.

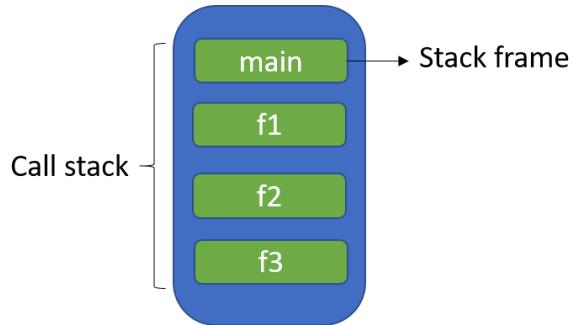
Os arquivos de código-objeto e quaisquer outros arquivos necessários são então vinculados usando um vinculador para criar o executável. Os vinculadores são geralmente incluídos nos compiladores e a vinculação é realizada automaticamente. Os arquivos executáveis são normalmente armazenados em um dos vários diretórios, são estes, **/bin**, **/sbin**, **/usr/bin**, **/usr/sbin** e **/usr/local/bin**. Embora não seja necessário que eles estejam nesses diretórios para serem operáveis⁴⁰, geralmente é mais conveniente por estarem no **PATH** do sistema. Quando um programa é iniciado, seu arquivo executável é copiado para a memória pelo sistema operacional para que seu código de máquina fique imediatamente disponível para a CPU. Mas não é uma cópia simples, segue-se uma estrutura de montagem na memória, conforme figura abaixo.

³⁹ Livro acessível pela url: https://en.wikipedia.org/wiki/Compilers:_Principles,_Techniques,_and_Tools

⁴⁰ Um comando que pode ser executado de qualquer ponto do sistema de arquivos



O seu código em linguagem de máquina fica alocado na base, e lá encontramos as instruções do programa, logo em seguida encontramos as variáveis globais. Variáveis globais é uma má prática, mas existe e tenho que explicar. Tratam-se de variáveis que podem ser acessadas por qualquer ponto de seu sistema. Logo em seguida temos uma área expansível de baixo para cima que é a área da pilha de funções, este cresce de baixo para cima. Cada nova função é empilhada nesta estrutura de dados. E por último temos a memória heap para variáveis do programa, a cada nova variável é alocada neste ponto da memória, lembre-se que as variáveis podem crescer.



Em sistemas operacionais em que o tipo de arquivo é indicado pelo acréscimo de uma extensão após seu nome, os executáveis são indicados por extensões como **.exe**, **.com** ou **.bin**. Essas extensões geralmente não são necessárias em sistemas operacionais do tipo Unix como o Linux.

Embora os programas de aplicativos geralmente venham à mente quando o termo executável é usado, este termo também se refere a **scripts**, utilitários e até mesmo Sistemas Operacionais. Um script é um pequeno programa escrito em uma linguagem de script, ou seja, um tipo de linguagem de programação simplificada. Os scripts são armazenados em arquivos de texto simples que devem ser interpretados por um executor ou shell (um programa compilado), em vez de serem compilados antecipadamente, o Linux faz uso extensivo de scripts para controlar a operação do sistema.

Um arquivo muito importante no Linux é o `vmlinuz`, trata-se de uma imagem com vários executáveis que formam o Kernel do Linux, um Kernel é um conjunto de programas que constitui o núcleo central de um Sistema Operacional de computador, e além de ser um

executável, o vmlinuz também é inicializável. Isso significa que ele é capaz de carregar o Sistema Operacional na memória para que o computador se torne utilizável e outros executáveis possam ser executados.

7.1 Ciclo de Vida do Processo

Quando um processo é executado, ele passa por diferentes estados. Esses estágios podem diferir em diferentes sistemas operacionais, e os nomes desses estados também não são padronizados.

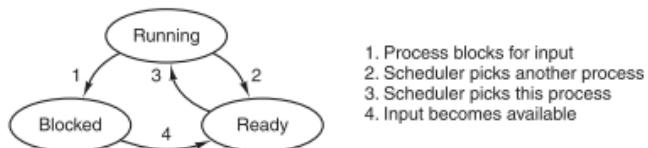


Figure 2-2. A process can be in running, blocked, or ready state. Transitions between these states are as shown.

No livro de Sistemas Operacionais do autor Tanenbaum ele aborda três estados somente, falando em um contexto genérico ou de implementação do Minix OS (ver figura acima). No Kernel do Linux os processos podem ter 5 estados:

Running or Runnable: No estado de execução, o processo ocupa um núcleo de CPU para executar seu código e lógica, ele é identificado no comando PS pela letra **R**;

Uninterruptible Sleep: Espera que os recursos estejam disponíveis antes que ele se mova para um estado executável e não reaja a nenhum sinal, ele é identificado no comando PS pela letra **D**;

Interruptable Sleep: Estado de sono interruptível (s) que reagirá aos sinais e à disponibilidade de recursos, ele é identificado no comando PS pela letra **S**;

Stopped: De um estado em execução ou executável, poderíamos colocar um processo no estado parado (T) usando o sinal SIGSTOP ou SIGTSTP, ele é identificado no comando PS pela letra **T**;

Zombie: Quando um processo tiver concluído a sua execução ou for encerrado, o mecanismo enviará o sinal SIGCHLD para o processo pai e entre no estado zumbi, ele é identificado no comando PS pela letra **Z**;

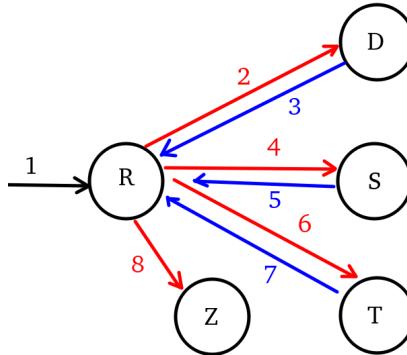
Quando executamos o comando PS podemos ver uma coluna com estas informações, segue conforme imagem abaixo. O comando PS será explicado ainda neste capítulo.

ueLL@usb:~\$ ps aux grep python									
root	852	0.0	0.0	28060	19736	?	Ss	10:56	0:00 python3 /opt/kfishmonger/p
root	967	0.0	0.0	26804	19084	?	Ss	10:56	0:00 python3 /opt/kfishmonger/p
root	968	0.0	0.0	243080	18088	?	Ssl	10:56	0:00 python3 /opt/kfishmonger/p
root	969	0.0	0.0	95464	14380	?	Ssl	10:56	0:00 python3 /opt/kfishmonger/p
root	971	0.0	0.0	21848	14624	?	Ss	10:56	0:00 python3 /opt/kfishmonger/p
root	1028	5.7	0.1	5139760	36320	?	Sl	10:56	2:26 python3 /opt/kfishmonger/p
root	1031	0.0	0.0	2580	904	?	S	10:56	0:00 /bin/sh -c python3 /opt/kf
root	1033	0.0	0.0	26520	18592	?	S	10:56	0:00 python3 /opt/kfishmonger/p
ueLL	1629	0.0	0.0	174252	20056	?	Sl	10:56	0:00 python3 /opt/kfishmonger/p
ueLL	158519	2.1	0.1	46412	38556	pts/0	S+	11:38	0:00 python3 /home/ueLL/desenv/

Já no comando top, temos temos outras letras para indicar o estado do processo, mas este comando será explicado ainda neste capítulo.

top - 12:16:06 up 1 day, 13:16, 1 user, load average: 0.04, 0.02, 0.00										
Tasks: 273 total, 1 running, 272 sleeping, 0 stopped, 0 zombie										
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st										
MiB Mem : 31933.2 total, 23624.8 free, 3495.7 used, 5290.0 buff/cache										
MiB Swap: 976.0 total, 606.8 free, 369.2 used, 28437.4 avail Mem										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
11780	well	20	0	11632	5320	3175	R	0.7	0.0	0:00.07 top
1	root	20	0	167628	9824	6935	S	0.0	0.0	0:02.71 systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.04 kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 slub_flushwq
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 netns
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/0:0-H-eve
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 mm_percpu_wq
11	root	20	0	0	0	0	I	0.0	0.0	0:00.00 rcu_tasks_kthrea
12	root	20	0	0	0	0	I	0.0	0.0	0:00.00 rcu_tasks_rude_k
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00 rcu_tasks_trace_
14	root	20	0	0	0	0	S	0.0	0.0	0:14.16 ksoftirqd/0

Podemos ver pelo mapa de estado de um processo que no Linux temos um ambiente mais complexo do que o relatado por Tanenbaum, as letras D, S, T, Z e R já foram explicadas e referem-se ao estado do processo, nos números temos os eventos que levam ao estado já explicado.



Onde:

1. Um processo é carregado para execução, e se tiver prioridade superior será posto imediatamente para execução, caso contrário irá esperar seu momento na fila de prioridade;
2. Aguardando recursos e sinais (Signal POSIX), pode ser um processo esperando por um evento de leitura de disco ou aguardando dados da placa de rede;
3. O processo é "acordado" com uma interrupção de sistema informando que seu recurso já está disponível no endereço de memória, a falha deste evento pode levar processos ao problema "Dormir e Acordar";
4. Aguardando o recurso mas pode receber interrupção;
5. Um sinal mandando o processo continuar execução e o recurso está disponível, a falha deste evento pode levar processos ao problema "Dormir e Acordar";
6. Ocorre quando um sinal POSIX chamado SIGSTOP é enviado para o processo;
7. Ocorre quando um sinal POSIX chamado SIGCONT é enviado para o processo;
8. Quando um dos sinais POSIX que solicitam a finalização do processo é enviado ou quando o próprio processo executa o comando exit();

Repare que neste ponto do livro, mesmo sem explicar o comando top (ver imagem acima) você já tem conhecimento para reconhecer vários campos, o conhecimento teórico é libertador e ele irá lhe fazer enxergar o que poucos conseguem enxergar.

Quando o Tanenbaum explica em seu livro que o processo voluntariamente sai de execução e termina o aluno não entende, mas vai ficar claro agora. Um processo recebe sinais do Sistema Operacional, e partindo do pressuposto que está tudo correto, ele irá sair de sua execução quando receber o SIGNAL POSIX adequado, tais sinais serão explicados.

Talvez o mais complicado de se ensinar para o aluno são os dois estados de "dormir". Vamos imaginar que um processo esteja precisando armazenar dados em um serviço na rede e que naturalmente irá precisar de IO responsável por isso, segundo o Tanenbaum qualquer dispositivo de IO é imensamente mais lento que o processador então o processo que está em execução terá que abdicar de seu quantum restante de processamento, para maximizar o uso da CPU. Então o processo entra no estado D (por isso coloquei D antes de S na imagem acima). O processo não pode ser interrompido neste estado pois ele está aguardando o hardware para entrar em sua região crítica, e a região crítica é um dos quatro fatores que precisam ser satisfeitos para um Deadlock. É neste momento que o dispositivo de IO está trabalhando para

Por outro lado, o estado de sono interruptível (S) é o estado de sono interruptível (s) que reagirá aos sinais e à disponibilidade de recursos. Um processo entra no (S) estado, ou Sono Interruptível, quando aguarda um evento ou condição que não esteja diretamente relacionado a uma operação de IO ou quando não possui todos os recursos de IO para finalização.

Quando um processo tiver concluído a sua execução ou for encerrado, ele irá enviar o sinal POSIX SIGCHLD sinalizando para o processo pai que está finalizando e entrará no estado zumbi. O processo zumbi, também conhecido como processo extinto, permanecerá nesse estado até que o processo pai o limpe da tabela de processos. Para limpar o processo filho encerrado da tabela de processo, o processo pai deve ler o valor de saída do processo filho usando as chamadas (primitivas do sistema): wait() ou waitpid().

Conforme já explicado o diretório /proc possui vários arquivos referentes ao estado do seu Linux, tanto processos quanto a recursos de IO e aos processos. Trata-se de um sistema de arquivos virtual. Podemos obter estes arquivos e ler diretamente informações de execução de um processo, alguns arquivos são ASCII já outros são binários ou stream de bits. Para exemplificar vamos usar o Process ID 1, que em um Debian 12 é o systemd, já em um Debian 9 é o init.

Podemos acessar seu status lendo o arquivo /proc/1/status e obter dados de processamento, endereços de memória, IO, sinais POSIX enviados, etc.. Conforme exemplo abaixo que estou exibindo o estado do processo.

```
well@big1:~$ cat /proc/1/status | grep State
State: S (sleeping)
well@big1:~$
```

7.2 Bloco de Controle de Processo (PCB)

Um processo chamado Schedule irá decidir sobre o escalonamento de outros processos no Linux, para isso ele utiliza uma série de informações devidamente organizadas em uma

estrutura de dados, uma espécie de fila com classificação de prioridade. Além disso, vários dados são úteis para tomar tal decisão, como por exemplo a espera por um dispositivo de IO. Então cada processo tem um bloco de controle de processo (PCB) no Kernel para manter as informações relacionadas ao processo, e este bloco de controle de processo é a estrutura `task_struct`. Abaixo podemos ver um trecho do arquivo `sched.h`⁴¹ (biblioteca `schedule`), no qual temos algumas variáveis.

```

813  struct task_struct {
814      #ifdef CONFIG_THREAD_INFO_IN_TASK
815          /*
816          * For reasons of header soup (see current_thread_info()), this
817          * must be the first element of task_struct.
818          */
819          struct thread_info           thread_info;
820      #endif
821          unsigned int                __state;
822
823          /* saved state for "spinlock sleepers" */
824          unsigned int                saved_state;
825
826          /*
827          * This begins the randomizable portion of task_struct. Only
828          * scheduling-critical items should be added above here.
829          */
830          randomized_struct_fields_start
831
832          void                         *stack;
833          refcount_t                   usage;
834          /* Per task flags (PF_*), defined further below: */
835          unsigned int                 flags;
836          unsigned int                 ptrace;

```

Cada processo coloca suas informações na estrutura de dados `task_struct`, que contém esses conteúdos:

Process ID (PID): O único indicador que descreve esse processo, usado para distinguir outros processos;

status : Status da tarefa, código de saída, sinal de saída, etc.

Priority: A prioridade relativa a outros processos.

Program counter: O endereço da próxima instrução a ser executada no programa.

Memory pointer: Inclui ponteiros para programar código e dados relacionados ao processo, bem como ponteiros para blocos de memória compartilhados com outros processos

Context data: Os dados no registro do processador quando o processo está em execução.

Informações de status de I/O: Incluindo o pedido de I/O exibido, o dispositivo de I/O atribuído ao processo e a lista de arquivos usados pelo processo.

Execution times and data: Pode incluir o tempo total do processador, o número total de relógios usados, limite de tempo, número da conta, etc.

⁴¹ Arquivo pode ser acessado pela URL:

<https://github.com/torvalds/linux/blob/master/include/linux/sched.h>

Todos os processos em execução no sistema são armazenados no Kernel na forma de uma lista encadeada chamada `task_struct`. As informações do processo podem ser visualizadas no diretório de arquivos do sistema `/proc`.

7.1 Compiladores

Um compilador é um programa de computador especializado em converter código-fonte escrito em uma linguagem de programação em outra linguagem, geralmente linguagem de máquina para que possa ser compreendido por processadores.

O código-fonte é a versão do software, como é originalmente escrito por um ser humano em texto simples. O código-fonte pode ser escrito em qualquer uma das inúmeras linguagens de programação, algumas das mais populares são C, C++, Perl, Python, etc.. A saída de um compilador é conhecida como código-objeto.

7.1.1 Tipos de compiladores

Os compiladores podem ser classificados de várias maneiras, incluindo seu idioma de entrada, seu código-alvo, a plataforma em que são executados e se são software proprietário ou software livre.

Embora a maioria dos compiladores seja projetada para converter código-fonte em código de máquina, também existem compiladores que traduzem o código-fonte escrito em uma linguagem de alto nível para o escrito em outra. Outros compiladores traduzem o código-fonte em uma linguagem intermediária que ainda precisa de processamento adicional durante a execução.

Por exemplo, alguns compiladores foram desenvolvidos para converter programas escritos em alguma linguagem de alto nível em um programa equivalente escrito em C. Isso é útil porque pode aumentar a portabilidade de programas escritos em linguagens menos comuns. O motivo é que, uma vez que um programa tenha sido convertido para C, é fácil compilar para quase todas as plataformas, pois os compiladores C estão disponíveis para quase todas as plataformas.

Um compilador que se destina a produzir código de máquina para ser executado na mesma plataforma em que o próprio compilador é executado às vezes é chamado de compilador de código nativo. Um compilador cruzado, que produz código de máquina destinado a ser executado em uma plataforma diferente daquela em que é executado, pode ser muito útil ao introduzir novas plataformas de hardware.

7.1.2 Compilando Versus Interpretando

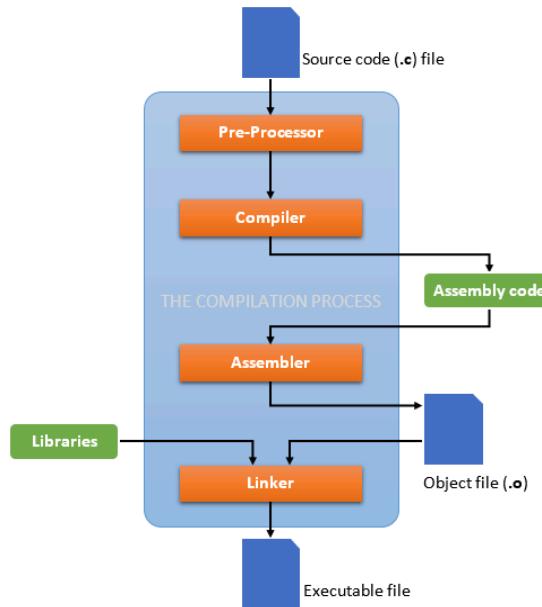
Nem todo o código-fonte é compilado, como algumas linguagens de programação, o código-fonte é frequentemente executado diretamente usando um interpretador em vez de primeiro compilar e depois executar o código de máquina resultante. Um interpretador é um programa que lê o código-fonte uma instrução de cada vez, traduz a instrução em código de máquina, executa a instrução do código de máquina e continua com a próxima instrução.

Geralmente é mais rápido executar código compilado do que executar um programa sob um interpretador. Isso ocorre principalmente porque o interpretador deve analisar cada instrução no código-fonte cada vez que o programa é executado e, em seguida, realizar a conversão desejada, enquanto isso não é necessário com o código compilado porque o código-fonte foi totalmente analisado durante a compilação. No entanto, pode levar menos tempo para interpretar o código-fonte do que o total necessário para compilar e executá-lo e, portanto, a interpretação é frequentemente usada ao desenvolver e testar o código-fonte para novos programas. São exemplos de Interpretadores:

- bash - (Linux, Mac OS X)
- csh - (UNIX)
- ksh - (UNIX)
- sh - (UNIX)
- zsh - (Linux)
- dash - (Debian)
- php (eu já vi)
- python

7.1.3 Criando um executável com GCC

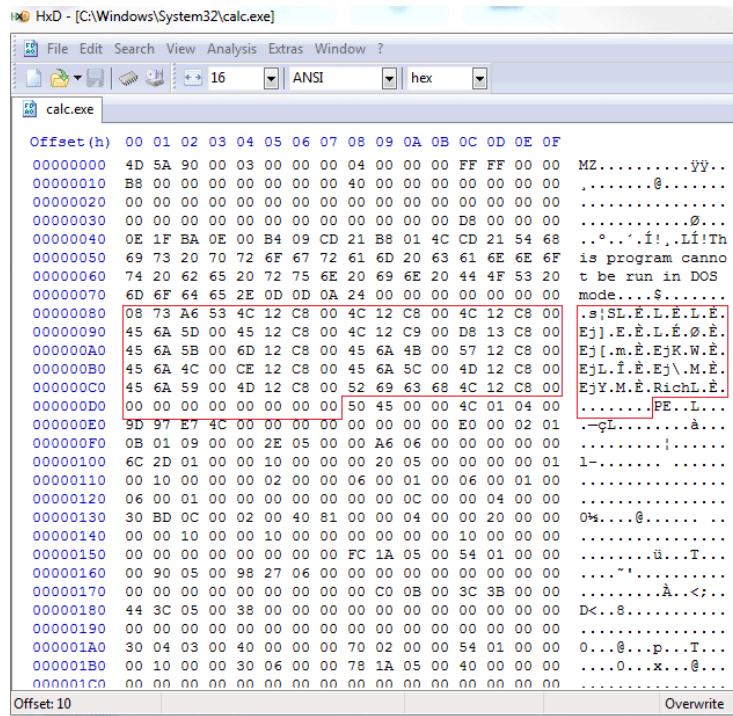
Neste exemplo será utilizado o compilador GNU/GCC, este é o compilador mais utilizado em ambientes Unix e naturalmente Linux. Este compilador compila C ANSI e também C++ e fornece ao desenvolvedor vários níveis de checagem de erros de código-fonte produzindo informações para depuração que otimizam o produto final. Na figura abaixo temos todo o processo de compilação descrito em um esquema.



A compilação envolve até quatro estágios:

- **Pré-processamento:** O pré-processamento é responsável por expandir as macros e incluir os arquivos de cabeçalho no arquivo fonte. O resultado é um arquivo que contém o código fonte expandido;
- **Compilação:** é responsável por traduzir o código fonte pré-processado em linguagem assembly (linguagem de máquina) para um processador específico;
- **Assembler:** nesta etapa o GCC converte o código de máquina de um processador específico em um arquivo objeto;
- **Linker:** é responsável por ligar os arquivos objeto para criar um arquivo executável.

O GCC é capaz de pré-processar e compilar vários arquivos em um ou mais arquivos em assembler. O arquivo de assembly gera um ou mais arquivos chamados de objeto e estes são ligados às bibliotecas (linking) para tornarem um arquivo executável. Na figura abaixo temos o header de um arquivo binário compilado.



No GNU/Linux os arquivos são definidos por este header.

7.1.3.1 O GCC⁴² – GNU Compiler Collection

O GNU Compiler Collection são compiladores completos da linguagem C ANSI com suporte para o C e C++, Objective C, Java, e Fortran. O GCC oferece também diferentes níveis de checagem de erros de código-fonte e informações para depuração e otimizações do programa objeto.

Ele também suporta a moderna plataforma de processadores Intel IA-64 e a versão 7.0 inclui novas APIs e bibliotecas C++. O manual também foi substancialmente reescrito e melhorado, e o pré-processamento é responsável por expandir as macros e incluir os arquivos de cabeçalho no arquivo fonte.

O pré-processamento do simples exemplo abaixo irá gerar um arquivo com mais de 800 linhas de código expandido. Então para exemplificar crie um arquivo chamado **teste.c** e codifique ele com o nano conforme listagem abaixo. No exemplo é utilizado o header **stdio.h** para uso do **printf()**.

```
1. #include <stdio.h>
2.
3. int main() {
```

⁴² Caso não tenha o GCC instalado utilize este tutorial:
<https://linuxize.com/post/how-to-install-gcc-compiler-on-ubuntu-18-04/>

```

4.     printf("Aied é 10, Aied é TOP, tá no Youtube");
5. };
6.

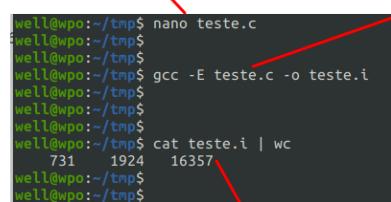
```

A opção “-E” do GCC diz ao compilador para fazer apenas o pré-processamento do código fonte.

```
1. gcc -E ./teste.c -o ./teste.i
```

Veja que o arquivo pre-processado teste.i gerou 841 linhas de código pré-processado:

Criando o arquivo com o código fonte



pré-processamento

código pré-processado

7.1.3.2 Compilação no GCC

O próximo estágio chamado de “compilação propriamente dita” é responsável por traduzir o código-fonte pré-processado em linguagem assembly (linguagem de máquina) para um processador específico.

Para que você veja o resultado do estágio de compilação, a opção “-S” (maiúsculo) do GCC gera o código de máquina para um processador específico. O código abaixo é o exemplo acima já pré-processado que foi compilado em assembly para o processador.

```
1. gcc -S ./teste.i
```

Ao gerar o código assembly teste utilizando o comando file, conforme exemplo abaixo.

```

well@wpo:~/tmp$ file teste.s
teste.s: assembler source, ASCII text
well@wpo:~/tmp$
```

É possível se ler o arquivo utilizando o comando cat.

```
well@wpo:~/tmp$ cat teste.s
    .file   "teste.c"
    .text
    .section      .rodata
    .align 8
.LC0:
    .string "Aied \303\251 10, Aied \303\251 TOP
.text
.globl  main
.type   main, @function
main:
.LFB0:
    .cfi_startproc
    endbr64
    pushq  %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset %rbp, -16
    movq  %rsp, %rbp
    .cfi_def_cfa_register 6
    leaq   .LC0(%rip), %rdi
    movl  $0, %eax
```

7.1.3.3 Assembler do Código no gcc

O estágio seguinte é chamado de assembler, nesta etapa o GCC converte o código de máquina de um processador específico em um arquivo objeto. Se neste código existirem chamadas externas de funções, o gcc deixa seus endereços indefinidos para serem preenchidos posteriormente pelo estágio de ligação.

O seguinte comando irá fazer a compilação do código assembly do exemplo anterior para o código de máquina:

```
1. as ./teste.s -o ./teste.o
```

O comando “**as**” é um compilador assembler disponível no pacote GNU GCC. O resultado será um arquivo “**teste.o**” que contém instruções de máquina do exemplo codificado com a referência indefinida para a função externa **printf**. A linha “**call printf**” do código assembly informa que esta função está definida em uma biblioteca e deverá ser chamada durante o processamento.

7.1.3.4 Linker do código com as bibliotecas

O último estágio chamado de ligação, ou **linker**, é responsável por ligar os arquivos objeto para criar um arquivo executável.

Ele faz isto preenchendo os endereços das funções indefinidas nos arquivos objeto com os endereços das bibliotecas externas do sistema operacional. Isto é necessário porque os arquivos executáveis precisam de muitas funções externas do sistema e de bibliotecas do C para serem executados.

As bibliotecas podem ser ligadas ao executável preenchendo os endereços das bibliotecas nas chamadas externas ou de forma estática quando as funções das bibliotecas são copiadas para o executável. No primeiro caso, o programa utilizará bibliotecas de forma compartilhada e ficará dependente delas para funcionar.

Este esquema economiza recursos, pois uma biblioteca utilizada por muitos programas precisa ser carregada somente uma vez na memória. O tamanho do executável será pequeno. No segundo caso, o programa é independente, uma vez que as funções que necessita estão no seu código.

Este esquema permite que quando houver uma mudança de versão de biblioteca o programa não será afetado. A desvantagem será o tamanho do executável e necessidades de mais recursos. Internamente a etapa de ligação é bem complexa, mas o GCC faz isto de forma transparente através do comando:

```
1. gcc ./teste.o -o ./teste
```

O resultado será um executável chamado teste, você pode invocar este executável digitando no terminal `./teste` conforme ilustração abaixo..

```
well@wpo:~/tmp$ ./teste
Aied é 10, Aied é TOP, tá no Youtubewell@wpo:~/tmp$
```

Para executar um executável compilado na própria máquina pelo GCC ou por G++⁴³ é fácil, pois o próprio compilador já adiciona a permissão do GNU/Linux a permissão X. Basta utilizar o `./` antes do nome do programa e pronto.

O `./` é necessário pois senão o GNU/Linux vai procurar em `/sbin`, `/bin`, `/usr/local/bin` e `/usr/local/sbin`, então o `./` é para indicar ao linux que o executável está aonde você está no sistema de arquivos (ver figura acima). Este compilado é extremamente dependente do Sistema operacional, será executado pelo `/lib64/ld-linux-x86-64.so.2` conforme figura abaixo, no vídeo abaixo é possível ver como alterar estes caminhos padrões do GNU/Linux editando arquivos do usuário.

```
well@wpo:~/tmp$ file teste
teste: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked
d, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=2e4a725a4c9704f880f5a5
6dc0af0512e89682c, for GNU/Linux 3.2.0, not stripped
well@wpo:~/tmp$
```

LLD é um vinculador do projeto LLVM que é um substituto imediato para vinculadores de sistema e é executado muito mais rápido do que eles. Ele também fornece recursos úteis para desenvolvedores de conjuntos de ferramentas. O vinculador suporta ELF (Unix), PE/COFF (Windows), Mach-O (macOS) e WebAssembly em ordem decrescente de integridade.

```
well@wpo:~/tmp$ ldd teste
linux-vdso.so.1 (0x00007ffe21d8f000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fc3bdf57000)
/lib64/ld-linux-x86-64.so.2 (0x00007fc3be170000)
well@wpo:~/tmp$
```

As bibliotecas `libc.so.6` e `ld-linux.so.2` são referenciadas dinamicamente pelo programa `teste` e é por isso que chamamos de compilação com referências dinâmicas. Caso queira

⁴³ Caso não tenha instalado o G++:

<https://linuxconfig.org/how-to-install-g-the-c-compiler-on-ubuntu-18-04-bionic-beaver-linux>

adicionar bibliotecas automaticamente para se ter menos dependência do Sistema Operacional, então utiliza-se a opção “-static” na compilação, conforme exemplo abaixo:

```
1. gcc -static ./teste.c -o ./teste1
```

Veja que ao adicionar as bibliotecas ao executável seu tamanho cresce substancialmente.

```
well@wpo:~/tmp$ ls -lh ./teste*
-rwxrwxr-x 1 well well 17K mar 18 18:12 ./teste
-rwxrwxr-x 1 well well 852K mar 18 18:12 ./teste1
-rw-rw-r-- 1 well well 89 mar 18 17:59 /teste.c
```

7.2 Locando arquivos headers, bibliotecas e comandos

A beleza da arte GNU/Linux é que pode-se estender as bibliotecas do GNU/Linux e criar novos componentes para o próprio Sistema Operacional, os componentes podem ser:

- Um arquivo header;
- Uma biblioteca compilada.
- Um comando que realiza uma ação específica;

Um header é um artefato que pode ser utilizado por desenvolvedores para estender, reutilizando código. Estes arquivos geralmente têm a extensão **.h** estão em texto plano.

```
usuario@debian:~$ ls /usr/include/
aio.h      error.h      libgen.h    netpacket
aliases.h  execinfo.h   libintl.h   netrom
alloca.h   fcntl.h     limits.h   netrose
argp.h     features.h   link.h     nfs
argz.h    features-time64.h linux     nl_types.h
ar.h       fenv.h      locale.h   nss.h
arpa      finclude     malloc.h   obstack.h
asm-generic fmtmsg.h   math.h     paths.h
```

Os programadores podem desenvolver rotinas para publicar nas distribuições GNU/Linux alocando os arquivos neste diretório. Mas o comum é exportar tais objetos já compilados, conforme figura abaixo.

```
usuario@debian:~$ ls /usr/include/c++/12/
algorithm  cmath          cwchar      iostream
any        codecvt        cwctype     istream
array      compare        cxxabi.h   iterator
atomic     complex        debug       latch
backward   complex.h     decimal     limits
barrier    concepts       deque      list
bit        condition_variable exception  locale
bits       coroutine      execution  map
bitset    csetjmp       expected   math.h
cassert   csignal       experimental memory
```

Agora, caso seja um comando, ou seja, um programa compilado e pronto para o uso é comum alocar tal programa em um dos diretórios presentes na variável PATH. Nos próximos capítulos a variável PATH será detalhada.

```
usuario@debian:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
usuario@debian:~$ _
```

Há a possibilidade de ter o artefato compilado sem a função `main()`, neste caso a biblioteca é utilizada em programas com uma importação, por exemplo a biblioteca `cURL`.

7.2.1 Criando um arquivo .h

Em uma empresa de desenvolvimento de aplicativos em C/C++ é comum a construção de uma grande base de código em arquivos `.h`, encontramos nestes arquivos funções e é natural que cada arquivo `.h` possui uma temática. O próprio Sistema Operacional está repleto destes arquivos e todos os comandos utilizados aqui são programas que utilizam estes `.h`.

Para exemplificar, se você programa em Java ou Python e quer obter dados dos dispositivos de bloco, em especial o UUID das partições, teria que criar um `Popen()` ou um `Process()` e executar o comando `blkid`, capturar o output e por regex obter o UUID fazendo um parser do texto. Em C/C++ basta importar a biblioteca `blkid/blkid.h` e invocar a função, não tem que fazer parser, no máximo conhecer uma `struct` de retorno.

Vamos exemplificar isso então em C++, para isso crie um novo arquivo chamado `myblkid.cpp`. Vamos utilizar a biblioteca `err.h` para tratar erros conforme linha 12 do código abaixo, e a biblioteca `blkid.h`⁴⁴ para as funções `blkid_new_probe_from_filename(const char*)`, `blkid_do_probe(*blkid_probe)` e `blkid_probe_lookup_value (blkid_probe, const char *, const char **, size_t *)`. Veja o código abaixo.

```
1. #include <err.h>
2. #include <blkid/blkid.h>
3. #include <string>
4.
5. int main (int argc, char *argv[]) {
6.     blkid_probe pr;
7.     const char *uuid;
8.     std::string partition = "/dev/sdal";
9.
10.    pr = blkid_new_probe_from_filename( partition.c_str() );
11.    if (!pr) {
12.        err(2, "Failed to open %s", partition);
13.    }
14.
15.    blkid_do_probe(pr);
16.    blkid_probe_lookup_value(pr, "UUID", &uuid, NULL);
17.
18.    printf("UUID=%s\n", uuid);
19.
20.    blkid_free_probe(pr);
21.
22.    return 0;
```

⁴⁴ Documentação sobre as funções disponíveis na URL:

<https://cdn.kernel.org/pub/linux/utils/util-linux/v2.29/libblkid-docs/libblkid-Low-level-probing.html>

23. }

Para compilar irá precisar adicionar a biblioteca `lblkid`, conforme comando de compilação abaixo.

```
1. g++ -o myblkid myblkid.cpp -lblkid
```

Ao executar verá o UUID da partição `/dev/sda1`, conforme figura abaixo. Lembre-se que este código teoricamente único é criado durante a formatação da partição então cada computador terá um código diferente.

```
uell@usb:/tmp$ g++ -o myblkid myblkid.cpp -lblkid
uell@usb:/tmp$ 
uell@usb:/tmp$ 
uell@usb:/tmp$ sudo ./myblkid
UUID=db5214a7-8476-48dd-a0b8-7cc132abec6b
```

Para criar um arquivo `head` e este ser utilizado pelos na compilação de aplicativos, primeiro deve-se mover o arquivo `.h` para `/usr/include/`, para exemplificar vamos criar um `head` chamado `fibonacci.h` e adicionar um código para um cálculo simples.

```
GNU nano 7.2                               /usr/include/fibonacci.h *
int fibonacci( int n ) {
    if (n <= 1 ) return n;
    return fibonacci( n - 1 ) + fibonacci( n - 2 );
}
```

Para usar é simples, basta adicionar em seu código um `include` para `<fibonacci.h>`. Repare que foi usado o símbolo `<` e `>` pois é um `head` que está no sistema e não localmente, se fosse localmente seria usado aspas duplas.

```
GNU nano 7.2                               calcfb.cpp *
#include <stdio.h>
#include <fibonacci.h>

int main( int argc, char* argv[] ) {
    printf( "F%d: %d \n", 4 ,fibonacci( 4 ) );
}
```

Para compilar é simples, afinal a biblioteca está no sistema e apta ao uso, então basta compilar com `g++`. O `g++` irá procurar todas as referências dos `includes`.

```
uell@usb:~/tmp$ 
uell@usb:~/tmp$ g++ -o calcfb calcfb.cpp
uell@usb:~/tmp$ 
uell@usb:~/tmp$ ./calcfb
F4: 3
```

7.3 A implementação de threads no Linux

Threads são abstrações de programação moderna, estes programas modernos são formados por várias Threads em um espaço de endereço de memória compartilhada entre elas (threads) e o objetivo é dar suavidade de execução de operações para os usuários. As Threads também podem compartilhar recursos, tal como arquivos, I/O, etc.. As threads permitem que o programa tenha programação simultânea e, em sistemas com vários processadores, o verdadeiro paralelismo.

O GNU/Linux tem uma implementação única de threads, para o kernel Linux, não existe o conceito de thread (**de processo**), ele implementa todos as threads como processos padrão concorrentes. O kernel do Linux não fornece nenhuma semântica de agendamento especial ou estruturas de dados para representar threads, em vez disso, uma thread é apenas um processo que compartilha certos recursos com outros processos. Cada thread possui um **task_struct** exclusivo e aparece para o kernel como um processo normal.

Essa abordagem para threads contrasta bastante com sistemas operacionais como Microsoft Windows ou Sun Solaris, que têm suporte de kernel explícito para threads. O nome **processo leve** resume a diferença de filosofias entre o Linux e outros sistemas. Para esses outros sistemas operacionais, os threads são uma abstração para fornecer uma unidade de execução mais leve e rápida do que o processo pesado.

Para o kernel Linux, threads são simplesmente uma maneira de compartilhar recursos entre processos. Por exemplo, suponha que você tenha um processo que consiste em quatro threads. Em sistemas com suporte de thread explícito, pode haver um descritor de processo que, por sua vez, aponta para os quatro threads diferentes. O descritor de processo descreve os recursos compartilhados, como um espaço de endereço ou arquivos abertos. Os tópicos então descrevem os recursos que eles possuem. Por outro lado, no Linux, existem simplesmente quatro processos e, portanto, quatro estruturas normais de **task_struct**. Os quatro processos são configurados para compartilhar certos recursos.

Threads são criados como tarefas normais, com a exceção de que a chamada do sistema **clone()** recebe sinalizadores correspondentes a recursos específicos a serem compartilhados.

7.3.1 Threads de kernel

Freqüentemente, é útil para o kernel realizar algumas operações em segundo plano. O kernel realiza isso por meio de threads de kernel.

A diferença significativa entre os threads do kernel e os processos normais é que os threads do kernel não têm um espaço de endereço. Eles operam apenas no espaço do kernel e não mudam de contexto para o espaço do usuário. Os threads do kernel são, entretanto, planejáveis e preemptivos como processos normais.

7.3.2 Implementando uma thread em C++

A classe `Thread` representa a construção de threads de execução em C++ e está disponível biblioteca `pthread`.

POSIX Threads (`pthread`), é um modelo de execução que existe independentemente de uma linguagem, bem como um modelo de execução paralela. Ele permite que um programa controle vários fluxos de trabalho diferentes que se sobrepõem no tempo. Cada fluxo de trabalho é conhecido como `thread`, e a criação e o controle sobre esses fluxos são obtidos fazendo chamadas para a API POSIX Threads. POSIX Threads é uma API definida pelo padrão POSIX.1c, extensão de Threads definida na IEEE Std 1003.1c-1995.

Implementação da API estão disponíveis em muitos sistemas operacionais em conformidade com POSIX, como o Unix, como FreeBSD, NetBSD, OpenBSD, Linux, macOS, Android, Solaris, tipicamente agrupados como uma biblioteca `libpthread` ou `pthread`.

Threads iniciam a execução imediatamente após a construção do objeto (no código abaixo o objeto é o `t1`) de thread (pendente de qualquer atraso no agendamento do sistema operacional), começando na função fornecida como um argumento do construtor (no exemplo abaixo a função é a `task1()`). O valor de retorno da função é ignorado e se terminar lançando uma exceção, `std::terminate` será chamado. A função pode comunicar seu valor de retorno ou uma exceção ao chamador via `std::promise` ou modificando variáveis compartilhadas.

Os objetos `std::thread` também podem estar no estado que não representa nenhum thread, um thread de execução pode não estar associado a nenhum objeto de thread. No próximo exemplo será demonstrado como criar uma thread simples, crie um arquivo chamado `thread.cpp` e codifique com o nano a listagem abaixo. A biblioteca `thread` está sendo utilizada para criar o objeto `t1` com o construtor e também o uso do `join()`.

```

1. #include <iostream>
2. #include <thread>
3.
4. using namespace std;
5.
6. // Uma função que sera executada como uma Thread
7. void task1(std::string msg)
8. {
9.     std::cout << "A thread está falando: " << msg;
10. }
11.
12. int main()
13. {
14.     //Invocando uma thread não bloqueante para ser executada em paralelo.
15.     thread t1(task1, "Olá");
16.
17.     //Aqui você tem seu programa pesado, que ocorre em paralelo com a thread
18.
19.     // O método JOIN aguarda a finalização da Thread para liberar recursos juntos
20.     t1.join();

```

21. }



A compilação é Simples, para isso deve-se informar em linha de comando que terá seré adicionado a referêcia ao pacote de Threads e vamos utilizar C++ Standard Library 11.

```
1. g++ ./thread.cpp -o ./thread -pthread -std=c++11
```

Na figura abaixo temos a sequênciá completa de edição, compilação e execução do programa.

```
usuario@debian:/tmp$ nano thread.cpp 1
usuario@debian:/tmp$ 
usuario@debian:/tmp$ g++ -o thread thread.cpp 2
usuario@debian:/tmp$ 
usuario@debian:/tmp$ 
usuario@debian:/tmp$ ./thread 3
```

Onde:

1. Criando o arquivo `thread.cpp` e editando o código;
2. Compilando o programa de `thread`;
3. Executando o binário que testa a `thread`.

7.4 Criando e gerenciando processos filho em C++

Uma outra forma de dividir esforço de computação em dois ou mais processos filhos, embora não seja a opção para todas as situações, uma boa chamada para processo filhos é o desenvolvimento de vírus e worms.

7.4.1 Função `fork()`

A chamada da função `fork()` no C/C++ irá gerar um novo processo filho que é um processo idêntico ao pai (baseado no mesmo executável binário), exceto que o filho vai possuir um novo ID de processo no sistema. O processo é copiado do pai (executável) para a memória e uma nova estrutura de processo é atribuída pelo kernel, e o valor de retorno da função é o que discrimina os dois processos.

No próximo exemplo, um processo pai vai criar um processo filho, e cada processo irá escrever uma frase no output, o objetivo é mostrar que ambos são executados. Crie um diretório chamado `dist` e um novo arquivo chamado `usefork.cpp` e edite o código abaixo. A biblioteca `unistd` está sendo utilizada para execução da função `fork()`.

```
1. #include <iostream>
2. #include <unistd.h>
3.
4. using namespace std;
5. int variavelGlobal = 2;
6.
7. int main() {
```

```

8.     string identidade;
9.     int variavelFuncao = 20;
10.
11.    // O ID retornado pelo fork() é zero quando o processo filho é criado
12.    pid_t pID = fork();
13.
14.    // Se é zero, então é um processo filho
15.    if (pID == 0) {
16.        identidade = "Processo filho: ";
17.        variavelGlobal++;
18.        variavelFuncao++;
19.    }
20.    // Se o pID retornado pelo fork for menor que zero, então houve falhas
21.    else if (pID < 0) {
22.        std::cerr << "Failed to fork" << std::endl;
23.        exit(1);
24.    }
25.    // Caso não seja nenhum dos dois, então é o processo pai
26.    else {
27.        identidade = "Processo pai:";
28.    }
29.
30.    // O código abaixo por não estar na sequência de IFs acima (desvio de bloco de
   código condicional) então é executado por ambos os processos, o pai e o filho.
31.    std::cout << identidade;
32.    std::cout << " Variavel Global: " << variavelGlobal ;
33.    std::cout << " Variável Funcao: " << variavelFuncao << std::endl;
34.    return 0;
35. }
```



Para compilar utilize o comando abaixo.

```

1. g++ -o ./usefork ./usefork.cpp
2. ./usefork
```

Na imagem abaixo pode-se ver o processo de edição, compilação e execução do programa.

```

usuario@debian:/tmp$ nano usefork.cpp 1
usuario@debian:/tmp$ g++ -o usefork usefork.cpp 2
usuario@debian:/tmp$ ./usefork 3
```

Onde:

4. Criando o arquivo usefork.cpp e editando o código;
5. Compilando o programa de thread;
6. Executando o binário mostra o uso de processo filho.

O resultado do output será:

```
usuario@debian:/tmp$ ./usefork
Processo pai: Variavel Global: 2 Variável Funcao: 20 1
usuario@debian:/tmp$ Processo filho: Variavel Global: 3 Variável Funcao: 21
2
```

A memória é compartilhada reduzindo a sobrecarga de geração de um novo processo com uma cópia exclusiva de toda a memória. Isso é normalmente usado ao usar **fork()** para **exec()**.

7.4.2 Aguardando finalização de procedimento filho com **wait()** e **waitpid()**

As condições de corrida podem ser criadas devido à imprevisibilidade de quando o agendador do kernel escalona os processos. Pode-se usar **wait()** para aguardar a execução de um procedimento realizado por outro processo.

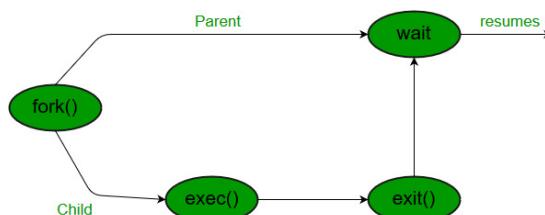
Alguma memória duplicada por um processo filho, como ponteiros de arquivo, causará uma saída mesclada de ambos os processos, para que um processo aguarde o uso de recursos por outro use a função **wait()** para que os processos não acessem o recurso ao mesmo tempo ou abram arquivo descritores exclusivos, alguns, como **stdout** ou **stderr**, serão compartilhados, a menos que sejam sincronizados com **wait()** ou algum outro mecanismo.

O fechamento do arquivo na saída é outra questão delicada, um processo de encerramento fechará os arquivos antes de sair da aplicação e os bloqueios de arquivo definidos pelo processo pai não são herdados pelo processo filho.

Uma chamada para **wait()** então bloqueia o processo pai até que um de seus processos filhos saia ou um sinal seja recebido. Depois que o processo filho termina, o pai continua sua execução após esperar a instrução de chamada do sistema.

O processo filho pode ser encerrado devido a um destes motivos:

- Ele chama **exit();**
- Ele retorna (um int) em **main();**
- Ele recebe um sinal cuja ação padrão é encerrar.



Se qualquer processo tiver mais de um processo filho, depois de chamar **wait()**, o processo pai deverá estar em estado de espera se nenhum filho for encerrado. Se apenas um processo filho for encerrado, o retorno de um **wait()** retorna o ID do processo filho encerrado. Se mais de um processo filho for encerrado, **wait()** retorna um ID arbitrariamente de algum desses processos filho.

Quando **wait()** retorna, eles também definem o status de saída por meio do ponteiro, se o status não for NULL. Se algum processo não tiver nenhum processo filho, **wait()** retorna imediatamente -1.

No exemplo abaixo, tanto o processo pai quanto o processo filho deverão imprimir seus PIDs, o **wait()** está sendo utilizado no código referente ao pai para aguardar a execução do processo filho. O header **sys/wait.h** está sendo referenciado para uso desta função. Com o nano no diretório tmp crie o arquivo **usewait.cpp** e codifique a listagem abaixo.

```

1. #include <iostream>
2. #include <unistd.h>
3. #include<sys/wait.h>
4.
5. using namespace std;
6.
7. int main() {
8.     // O ID retornado pelo fork() é zero quando o processo filho é criado
9.     pid_t pID = fork();
10.    pid_t cpid;
11.
12.    if ( pID == 0 ) {
13.        std::cout << "Saindo do processo filho. " << std::endl;
14.        return 0;
15.    } else {
16.        cpid = wait(NULL);
17.    }
18.
19.    std::cout << "PID do pai: " << getpid() << std::endl;
20.    std::cout << "PID do filho: " << cpid << std::endl;
21.    return 0;
22. }
```



Para compilar utilize o comando abaixo.

```

3. g++ -o ./usewait ./usewait.cpp
4. ./usewait
```

O resultado do output será:

```

usuario@debian:~/tmp$ ./usewait
Saindo do processo filho.
PID do pai: 1554
PID do filho: 1555
```

status sobre o processo filho relatado pela sua saída podem ser "capturados" por macros internas, e podemos assim saber se:

- O término foi normal/anormal
- A causa do término;
- Status de saída;

Para encontrar informações sobre o status, usamos:

WIFEXITED(status): O processo filho saiu normalmente;

WEXITSTATUS(status): O código de retorno enviado pelo processo filho;

WIFSIGNALED(status): O término da execução do processo filho se deu pelo recebimento de um sinal;

WTERMSIG(status): fornece o número do sinal;

WIFSTOPPED(status): O processo foi parado;

WSTOPSIG (status): Fornece o número do sinal de parada.

Para exemplificar crie com o nano um arquivo chamado **usewait_exit.cpp** e codifique o código abaixo. O processo pai vai esperar com a chamada bloqueante ao **wait()** que aguarda a finalização do filho.

Após isso o IF vai realizar o teste com **WIFEXITED()**⁴⁵ que retorna um número diferente de zero para uma saída com **exit()** do processo filho, ou seja, se tudo der certo no processo filho, ele irá rodar esse IF. Mas caso contrário, caso um sinal POSIX não tenha sido tratado, então deverá retornar um valor para **WIFSIGNALED()** e um **psignal()** deverá ser enviado para o processo indicando a saída.

Logo em seguida com **getpid()** e a variável **cpid**, tanto o PID do processo pai quanto o PID do processo filho são impressos na tela.

```

1. #include <iostream>
2. #include <stdlib.h>
3. #include<sys/wait.h>
4. #include <signal.h>
5.
6. int main() {
7.
8.     // O ID retornado pelo fork() é zero quando o processo filho é criado
9.     pid_t pID = fork();
10.    pid_t cpid;
11.    int stat;
12.
13.    if ( pID == 0 ) {
14.        std::cout << "Saindo do processo filho. " << std::endl;
15.        exit(1);
16.    } else {
17.        cpid = wait(NULL);
18.    }
19.
20.    if (WIFEXITED(stat)){
21.        std::cout << "WEXIT " << WEXITSTATUS(stat) ;
22.    } else if (WIFSIGNALED(stat)) {
23.        psignal(WTERMSIG(stat), "Sinal de saída: ");
24.    }
25.
26.    std::cout << "PID do pai: " << getpid() << std::endl;
27.    std::cout << "PID do filho: " << cpid << std::endl;

```

⁴⁵ Mais detalhes em

https://www.gnu.org/software/libc/manual/html_node/Process-Completion-Status.html

```
28.     return 0;
29. }
```



Compile e execute o comando conforme listagem abaixo.

```
1. g++ -o ./usewait_exit ./usewait_exit.cpp
2. ./usewait_exit
```

O resultado do output será:

```
usuario@debian:/tmp$ ./usewait_exit
Saindo do processo filho.
WEXIT OPID do pai: 1149
PID do filho: 1150
usuario@debian:/tmp$
```

Para um processo pai aguardar um grupo de processos filhos utiliza-se **waitpid()** passando como parâmetro o PID do processo filho. Mas neste exemplo serão abertos 5 processos filhos e o processo pai deverá aguardar todos, isso será feito utilizando um FOR. Para exemplificar crie um arquivo chamado **waitpid.cpp** com o nano e codifique o código abaixo.

```
1. #include <iostream>
2. #include<sys/wait.h>
3.
4. using namespace std;
5.
6. int main() {
7.     pid_t pid[5];
8.     int i, stat;
9.
10.    for (i = 0; i < 5; i++) {
11.        pid[i] = fork();
12.        if( pid[i] == 0) {
13.            sleep(1);
14.            exit(100 + i);
15.        }
16.    }
17.
18.    for (i = 0; i < 5; i++) {
19.        pid_t cpid = waitpid(pid[i], &stat, 0);
20.        if (WIFEXITED(stat)) {
21.            std::cout << "O filho " << cpid << " terminou com o status: " <<
22.            WEXITSTATUS(stat) << std::endl;
23.        }
24.    }
25.    return 0;
26. }
```



Compile e execute o programa, conforme listagem abaixo.

```
1. g++ -o ./waitpid ./waitpid.cpp
2. ./waitpid
```

O resultado do output será:

```
usuario@debian:/tmp$ ./waitpid
O filho 1207 terminou com o status: 100
O filho 1208 terminou com o status: 101
O filho 1209 terminou com o status: 102
O filho 1210 terminou com o status: 103
O filho 1211 terminou com o status: 104
usuario@debian:/tmp$ █
```

7.4.3 Executando comandos de Shell com system() popen() e exec()

A chamada **system()** executa um comando shell do sistema operacional conforme descrito por uma string contendo o comando literal. Esta função é implementada usando **fork()**, **exec()** e **waitpid()**.

A string de comando é executada chamando **/bin/sh -c COMANDO_NA_STRING**, e durante a execução do comando, **SIGCHLD** será bloqueado e **SIGINT** e **SIGQUIT** serão ignorados. A chamada "bloqueia" e espera que a tarefa seja executada antes de continuar. No exemplo **system.cpp** que será realizado, será executada a função **system()** e o seu output será injetado no output do programa.

```
1. #include <iostream>
2.
3. int main()
4. {
5.     system("ls -l");
6.     std::cout << "Executado" << std::endl;
7.     return 0;
8. }
```



Compile e execute o programa conforme listagem abaixo.

```
1. g++ -o ./dist/system ./system.cpp
2. ./dist/system
```

Veja a execução do comando.

```
usuario@debian:/tmp$ ./system
total 32
-rwxr-xr-x 1 usuario usuario 16592 Oct 25 17:32 system
-rw-r--r-- 1 usuario usuario    115 Oct 25 17:32 system.cpp
drwx----- 3 root    root    4096 Oct 25 16:14 systemd-private-42e47980dc95427c
9375b0e82f8ef0ed-systemd-logind.service-cy6GIW
drwx----- 3 root    root    4096 Oct 25 16:15 systemd-private-42e47980dc95427c
9375b0e82f8ef0ed-systemd-timesyncd.service-iihxNq
Executado
usuario@debian:/tmp$
```

A chamada **popen()** abre um processo criando e criando um pipe de comunicação (um arquivo tipo s dentro do sistema operacional), o **fork()** então executa o shell. A vantagem de usar **popen()** é que permitirá obter o output do comando executado.

Este exemplo abre um pipe que executa o comando shell "ls -l" e os resultados são lidos e impressos pelo processo pai.

- r Para ter acesso ao stdin
- w Para ter acesso ao stdout

Para ter acesso ao stderr adicione no comando 2>&1 conforme exemplo: ls -l 2>&1

```
1. #include <iostream>
2.
3. int main() {
4.     FILE *fpipe;
5.     char *command = (char *)"ls -l";
6.     char line[256];
7.
8.     if ( !(fpipe = (FILE*)popen(command,"r")) ) {
9.         perror("Falha ao abrir um pipe");
10.        exit(1);
11.    }
12.
13.    while ( fgets( line, sizeof line, fpipe) ) {
14.        std::cout << "Linha: " << line ;
15.    }
16.
17.    pclose(fpipe);
18.    return 0;
19. }
```



Para compilar utilize o comando abaixo (tem que existir um diretório chamado dist no mesmo diretório do arquivo .cpp).

```
1. g++ -o ./dist/pop ./pop.cpp
```

```
well@wpo:~/projects/git/ccpp/process/syspop$ ./dist/pop
Linha: total 12
Linha: drwxrwxr-x 2 well well 4096 mar 19 06:05 dist
Linha: -rw-rw-r-- 1 well well 342 mar 19 06:05 pop.cpp
Linha: -rw-rw-r-- 1 well well 337 mar 19 05:57 system.cpp
```

O comando `exec()` e `execve()` iniciarão um programa de dentro de um programa, eles também possuem várias funções front-end para `execve()`. A rotina `execlp()` executará a mesma finalidade, exceto que usará a variável de ambiente `PATH` para determinar qual executável processar.

7.5 Sinais POSIX

Os sinais POSIX⁴⁶ são usados para notificar um processo ou thread de um evento específico. Podem ocorrer quando um subsistema de hardware, como uma interface de I/O, gera uma interrupção para um processador quando a I/O é concluída ou quando um processo precisa emitir um sinal para outro processo. Esses eventos, por sua vez, fazem com que o processador entre em troca de contexto para tratar a interrupção, de modo que o processamento subsequente possa ser feito no sistema operacional com base na origem e na causa da interrupção. Vamos ver um exemplo:



Onde:

1. O processo 1 envia para o processo 2 um sinal SIGTERM, mas para isso deve-se ter permissão;
2. O processo 2 voluntariamente sai de execução;

Os sinais do sistema operacional, têm uma grande história de mudanças de design no código do sinal e várias implementações do UNIX. Isso se deveu em parte a algumas deficiências na implementação inicial de sinais, bem como ao trabalho de desenvolvimento paralelo feito em diferentes versões do UNIX, principalmente BSD UNIX e AT&T System V.

A implementação de sinais corretos e confiáveis já existe há muitos anos, onde um manipulador de sinal instalado permanece persistente e não é redefinido pelo kernel, os padrões POSIX forneceram um conjunto bastante definido de interfaces para usar sinais no código, e hoje a implementação de sinais do Linux é totalmente compatível com POSIX.

A ocorrência de um sinal pode ser síncrona ou assíncrona ao processo ou thread, dependendo da fonte do sinal e da razão ou causa subjacente. Os sinais síncronos ocorrem como resultado direto do fluxo de instruções em execução, onde um erro irrecuperável que requer o término imediato do processo. Esses sinais são direcionados ao encadeamento que causou o erro com seu fluxo de execução.

⁴⁶ Signal Handling baseado no projeto GNU
https://www.gnu.org/software/libc/manual/html_node/Signal-Handling.html

Cada sinal tem um nome de sinal único, uma abreviatura que começa com SIG (SIGINT para sinal de interrupção, por exemplo) e um número de sinal correspondente. Além disso, para todos os sinais possíveis, o sistema define uma disposição ou ação padrão a ser executada quando ocorre um sinal.

Linux oferece suporte aos sinais padrão listados abaixo. A segunda coluna da tabela indica qual padrão (se houver): "P1990" indica que o sinal está descrito no padrão POSIX.1-1990 original; "P2001" indica que o sinal foi adicionado em SUSv2 e POSIX.1-2001.

	Signal	Standard	Action	Comment
	SIGABRT	P1990	Core	Abort signal from abort
	SIGALRM	P1990	Term	Timer signal from alarm
	SIGBUS	P2001	Core	Bus error (bad memory access)
	SIGCHLD	P1990	Ign	Child stopped or terminated
	SIGCLD	-	Ign	A synonym for SIGCHLD
18	SIGCONT	P1990	Cont	Continue if stopped
	SIGEMT	-	Term	Emulator trap
	SIGFPE	P1990	Core	Floating-point exception
1	SIGHUP	P1990	Term	Hangup detected on controlling terminal or death of controlling process
	SIGILL	P1990	Core	Illegal Instruction
	SIGINFO	-		A synonym for SIGPWR
2	SIGINT	P1990	Term	Interrupt from keyboard
	SIGIO	-	Term	I/O now possible
	SIGIOT	-	Core	IOT trap. A synonym for SIGABRT
9	SIGKILL	P1990	Term	Kill signal
	SIGLOST	-	Term	File lock lost (unused)
	SIGPIPE	P1990	Term	Broken pipe: write to pipe with no readers; see pipe
	SIGPOLL	P2001	Term	Pollable event (Sys V); synonym for SIGIO
	SIGPROF	P2001	Term	Profiling timer expired
	SIGPWR	-	Term	Power failure (System V)
3	SIGQUIT	P1990	Core	Quit from keyboard
	SIGSEGV	P1990	Core	Invalid memory reference
	SIGSTKFLT	-	Term	Stack fault on coprocessor (unused)
19	SIGSTOP	P1990	Stop	Stop process
20	SIGTSTP	P1990	Stop	Stop typed at terminal
	SIGSYS	P2001	Core	Bad system call (SVr4); see also seccomp
15	SIGTERM	P1990	Term	Termination signal
	SIGTRAP	P2001	Core	Trace/breakpoint trap
	SIGTTIN	P1990	Stop	Terminal input for background process
	SIGTTOU	P1990	Stop	Terminal output for background process
	SIGUNUSED	-	Core	Synonymous with SIGSYS
	SIGURG	P2001	Ign	Urgent condition on socket (4.2BSD)
	SIGUSR1	P1990	Term	User-defined signal 1

SIGUSR2	P1990	Term	User-defined signal 2
SIGVTALRM	P2001	Term	Virtual alarm clock (4.2BSD)
SIGXCPU	P2001	Core	CPU time limit exceeded (4.2BSD); see setrlimit(2)
SIGXFSZ	P2001	Core	File size limit exceeded (4.2BSD); see setrlimit(2)
SIGWINCH	-	Ign	Window resize signal (4.3BSD, Sun)
		Term	Default action is to terminate the process.
		Ign	Default action is to ignore the signal.
		Core	Default action is to terminate the process and dump core
		Stop	Default action is to stop the process.
		Cont	Default action is to continue the process if it is currently stopped.

O aluno pode listar os sinais com o comando kill e com parâmetro -l, nunca gostei deste nome de comando, kill dá a idéia de morte ou finalização, mas um sinal pode ser de pausa ou de continuação.

```
usuario@debian:~$ kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
usuario@debian:~$
```

Os sinais SIGKILL e SIGSTOP não podem ser capturados, bloqueados ou ignorados pelos processos. Os sinais SIGHUP , SIGINT e SIGQUIT são gerados por uma entrada de teclado do terminal de controle (SIGINT e SIGHUP) ou se o terminal for desconectado (SIGHUP mas se o usuário utilizar o comando nohup ele torna os processos "imunes" de desligamentos).

Outros sinais de terminal relacionados a I/O incluem SIGSTOP, SIGTTIN, SIGTTOU e SIGTSTP. Para os sinais originados de um comando de teclado, a sequência de teclas real que gera os sinais, geralmente CTRL-C, é definida dentro dos parâmetros da sessão do terminal, normalmente via stty(1) que resulta no envio de um SIGINT para um processo. Exemplos de Sinais emitidos pelo teclado:

Ctrl+C - SIGINT

Ctrl+ - SIGQUIT

Ctrl+Z - SIGTSTP

7.5.1 Enviando um sinal para um processo com o comando kill

O comando kill está embutido na maioria dos shells derivados de Bourne, como Bash e Zsh. O comportamento do comando é ligeiramente diferente entre os shells e o executável é **/bin/kill**. A sintaxe do comando kill assume a seguinte forma:

```
1. kill [OPTIONS] [PID]...
```

O comando kill envia um sinal para processos ou grupos de processos especificados, fazendo com que eles ajam de acordo com o sinal. Quando o sinal não é especificado, o padrão é SIGTERM (15). Os sinais mais comumente usados são:

- **SIGHUP** (1): Finaliza o processo com o término da sessão.
- **SIGKILL** (9): Mata um processo.
- **SIGTERM** (15): Pare um processo normalmente.

Os PIDs fornecidos para o comando kill podem ser um dos seguintes:

- **Se PID for maior que zero**, o sinal é enviado ao processo com ID igual a PID;
- **Se PID for igual a zero**, o sinal é enviado a todos os processos do grupo de processos atual. Ou seja, o sinal é enviado a todos os processos pertencentes ao GID do shell que invocou o comando kill. Use o comando **ps -efj** para visualizar os IDs do grupo de processos (GIDs);
- **Se PID for igual a -1**, o sinal é enviado a todos os processos com o mesmo UID do usuário que está chamando o comando. Se o usuário de chamada for root, o sinal será enviado para todos os processos, exceto init e o próprio processo kill;
- **Se PID for menor que -1**, o sinal é enviado a todos os processos no grupo de processos com GID igual ao valor absoluto de PID.

Os usuários comuns podem enviar sinais para seus próprios processos, mas não aqueles que pertencem a outros usuários, enquanto o usuário root pode enviar sinais para os processos de outros usuários. Para encerrar ou encerrar um processo com o comando kill, primeiro você precisa encontrar o número de identificação do processo (PID). Você pode fazer isso usando comandos diferentes.

Digamos que o editor Gedit pare de responder e você precise interromper, para encontrar os PIDs do processo, use o comando ps e para filtrar use o grep, conforme exemplo:

```
1. ps aux | grep gedit | grep -v grep
```

A saída será:

```
well@wpo:~/projects/git/ccpp$ well@wpo:~/projects/git/ccpp$ ps aux | grep gedit | grep -v grep
well      6363  0.3  0.3 822560 60916 ?          Sl   04:17  1:24 /usr/bin/gedit
well@wpo:~/projects/git/ccpp$ well@wpo:~/projects/git/ccpp$
```

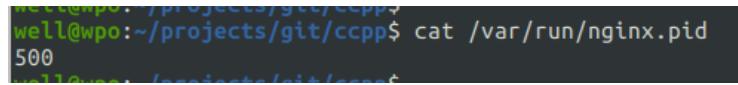
Depois de saber o número do processo, você pode matar enviando o sinal TERM:

```
1. kill -9 6363
```

Outro caso de uso comum de uso do kill é enviar o sinal HUP, que diz aos processos para recarregar suas configurações. Por exemplo, para recarregar o Nginx, você precisa enviar um sinal para o processo, o ID de processo do processo Nginx pode ser encontrado no arquivo `nginx.pid`, que normalmente está localizado no diretório `/var/run`. Use o comando `cat` para encontrar o PID:

```
1. cat /var/run/nginx.pid
```

Veja na imagem abaixo que foi retirado de uma prática deste livro sobre Nginx.



```
well@wpo:~/projects/git/ccpp$ cat /var/run/nginx.pid
500
well@wpo:~/projects/git/ccpp$
```

Sabendo qual o PID do processo é só enviar o sinal.

```
1. sudo kill -1 500
```

Para remover todos os processos que estão associados a um arquivo ou pelo nome do processo, utilize o grep associado ao awk e xargs conforme exemplo abaixo.

```
1. ps -ef | grep 'python' | grep -v grep | awk '{print $2}' | xargs -r kill -9
```

Usando o mesmo cenário de antes, você pode interromper os processos Gedit digitando:

```
1. killall -9 gedit
```

Para saber as opções de como enviar sinais para você pode obter uma lista de todas as opções digitando `killall` sem nenhum argumento. Por exemplo, para encerrar todos os processos em execução como um usuário `userlinux`, você executaria o seguinte comando:

```
1. sudo killall -u userlinux
```

O comando `pkill` encerra processos que correspondem ao padrão fornecido na linha de comando, exemplo:

```
1. pkill -9 gedit
```

O nome do processo não precisa ser uma correspondência exata. Com `pkill` você também pode enviar um sinal para processos que pertencem a um determinado usuário. Para eliminar o `gedit` do usuário `userlinux`, utilize:

```
1. pkill -9 -u userlinux gedit
```

7.5.2 Tratando sinais em processos

Funções que podem ser acionadas via programação:

- **raise()** Envia um sinal para o segmento de chamada, será demonstrado em C++ no tópico abaixo;
- **pidfd_send_signal()** Envia um sinal para um processo identificado por um arquivo descriptor PID;
- **killpg()** Envia um sinal para todos os membros de um determinado grupo de processos;
- **pthread_kill()** Envia um sinal para um thread POSIX especificado no mesmo processo como o chamador;
- **pausa()** Suspende a execução até que qualquer sinal seja detectado.
- **sigsuspend()** Altera temporariamente o sinal e suspende a execução até que um dos sinais seja lançado.

Neste exemplo de programa em C++, vamos capturar o sinal SIGINT e escrever no output que o sinal foi recebido e sair. Para isso vamos usar a biblioteca csignal que possui o código de tratativa de eventos void signal(int, void). Crie um arquivo chamado receivesignal.cpp com o nano e digite a seguinte listagem de código abaixo.

```

1. #include <iostream>
2. #include <csignal>
3. #include <unistd.h>
4. using namespace std;
5.
6. void signal_handler(int signum) {
7.     std::cout << "Processo será interrompido pelo sinal: (" << signum << ")" <<
8.         std::endl;
9.     exit(signum);
10.
11. int main () {
12.     signal(SIGINT, signal_handler);
13.
14.     while(1) {
15.         std::cout << "Dentro do laço de repetição infinito." << std::endl;
16.         sleep(1);
17.     }
18.
19.     return 0;
20. }
```



Na linha 12 registramos que no caso de um evento SIGINT o nosso programa deverá executar a função **void signal_handler(int signum)**, e dentro desta função podemos ou não sair da aplicação, ou seja, se comentar a linha 8 e executar seu programa verá que um Ctrl-C do teclado não irá informar ao seu programa que ele deverá sair. Para compilar utilize o comando abaixo (tem que existir um diretório chamado dist no mesmo diretório do arquivo .cpp).

```
1. g++ -o ./dist/receivesignal ./receivesignal.cpp
```

Então você deve executar o programa, chamando `./dist/receivesignal` pelo terminal, caso pressione Ctrl+C o aplicativo irá escrever uma mensagem e sair, mas caso pressione Ctrl+C e tenha comentado a linha 8, bom, não vai sair. Outra forma de cancelar o uso e ignorar o SIGINT é passando no void signal(int, *void) a constante SIG_IGN⁴⁷.

```

1. #include <iostream>
2. #include <csignal>
3. #include <unistd.h>
4.
5. using namespace std;
6.
7. int main () {
8.     signal(SIGINT, SIG_IGN);
9.
10.    while(1) {
11.        std::cout << "Dentro do laço de repetição infinito." << std::endl;
12.        sleep(1);
13.    }
14.
15.    return 0;
16. }
```



Faça seu programa rodar e tente novamente parar ele com Ctrl+C, verá que a aplicação não sai de execução com SIGINT. No próximo exemplo vou mostar como elevar um sinal para o próprio sistema, imagine que tem várias threads e que uma thread estoure o tempo de execução ou não complete seu trabalho, pode-se elevar um sinal interno para que tudo seja paralizado e a aplicação se feche.

```

1. #include <iostream>
2. #include <csignal>
3. #include <unistd.h>
4. using namespace std;
5.
6. void signal_handler(int signum) {
7.     std::cout << "Processo será interrompido pelo sinal: (" << signum << ")" . " <<
8.     std::endl;
9.     exit(signum);
10.
11. int main () {
12.     int i = 0;
13.     signal(SIGINT, signal_handler);
14.
15.     while(++i) {
16.         std::cout << "Dentro do laço de repetição infinito." << std::endl;
17.
18.         if( i == 5 ) {
19.             raise( SIGINT );
20.         }
21.     }
22. }
```

⁴⁷ Documentação oficial pode ser acessível pela URL:
<https://man7.org/linux/man-pages/man2/sigaction.2.html>

```

21.     sleep(1);
22. }
23. return 0;
24. }
25. }
```



Veja que na linha 19 foi enviado um sinal com void raise(int), neste caso mais simples caso variável inteira i alcance o valor de 5. No próximo exemplo será enviado um sinal semelhante ao SIGTERM para o processo corrente, trata-se de uma mesma forma de enviar um sinal para si mesmo.

```

1. #include <iostream>
2. #include <csignal>
3. #include <unistd.h>
4. using namespace std;
5.
6. void signal_handler(int signum) {
7.     std::cout << "Processo será interrompido pelo sinal: (" << signum << ")" <<
8.     std::endl;
9.     exit(signum);
10.
11. int main () {
12.     int pid, i = 0;
13.     signal(SIGINT, signal_handler);
14.
15.     while(++i) {
16.         std::cout << "Dentro do laço de repetição infinito." << std::endl;
17.
18.         if( i == 5 ) {
19.             pid = getpid();
20.             kill(pid, SIGUSR1);
21.             std::cout << "Tudo tem um fim, mas isso não será exposto, kkkk" <<
22.             std::endl;
23.         }
24.         sleep(1);
25.     }
26.     return 0;
27. }
```



Na linha 19 o int getpid() retorna o PID do processo atual e com o comando void kill(int, int) estou enviando um sinal SIGUSR1 que é semelhante⁴⁸ ao SIGTERM. No próximo exemplo vamos unir o aprendizado de processo filho com fork() e sinais POSIX. Neste exemplo, o processo pai enviará um sinal para finalizar o processo filho.

⁴⁸ Conforme documentação oficial: <https://man7.org/linux/man-pages/man7/signal.7.html>

```

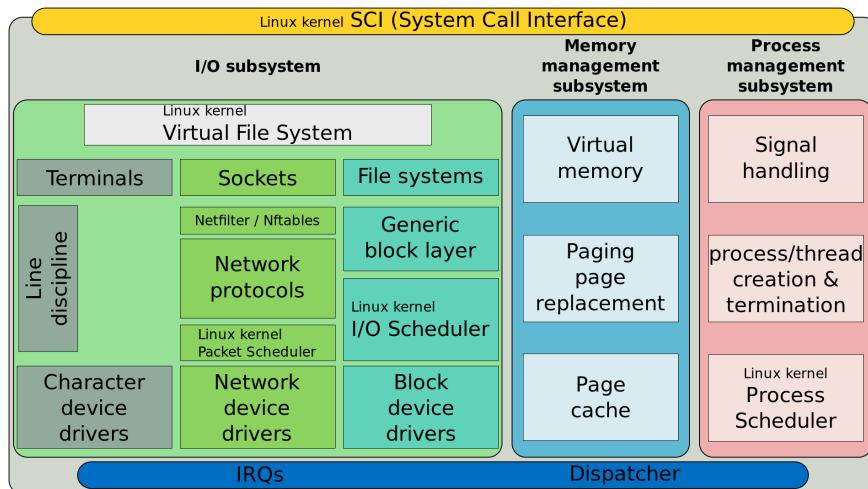
1. #include <iostream>
2. #include <csignal>
3. #include <unistd.h>
4.
5. using namespace std;
6.
7. void signal_parent_handler(int signum) {
8.     std::cout << "Processo (PAI) será interrompido pelo sinal: (" << signum << ")"
9.     << std::endl;
10. }
11. void signal_child_handler(int signum) {
12.     std::cout << "Processo (FILHO) será interrompido pelo sinal: (" << signum <<
13.     ")"
14.     << std::endl;
15.     sleep(1);
16.     kill( getppid(), SIGUSR1);
17. }
18. int main () {
19.     pid_t pid;
20.     pid = fork();
21.     if(pid < 0) {
22.         std::cout << "Falha!";
23.         exit(0);
24.     }
25.     // Processo filho o PID retornado pelo fork é Zero.
26.     else if(pid == 0) {
27.         signal(SIGUSR1, signal_child_handler);
28.         std::cout << "Processo filho aguardando sinal." << std::endl;
29.         pause();
30.     }
31.     // é o Pai
32.     else {
33.         signal(SIGUSR1, signal_parent_handler);
34.         sleep(2);
35.         std::cout << "Enviando sinal para o filho." << std::endl;
36.         kill(pid, SIGUSR1);
37.         std::cout << "Processo pai aguardando resposta." << std::endl;
38.         pause();
39.     }
40.
41.     return 0;
42. }

```



7.6 Escalonamento (Scheduling)

O scheduler é o componente do Kernel que seleciona o próximo processo a ser executado. O scheduler pode ser visto como o código que divide o recurso finito de tempo do processador entre os processos executáveis em um sistema. O scheduler é a base de um sistema operacional multitarefa como o Linux. Ao decidir qual processo pode ser executado, o scheduler é responsável por melhor utilizar o sistema e dar a impressão de que vários processos estão sendo executados simultaneamente.



A ideia por trás do scheduler é simples. Para melhor utilizar o tempo do processador, supondo que haja processos executáveis, um processo deve estar sempre em execução. Se houver mais processos do que processadores em um sistema, alguns processos nem sempre estarão em execução. Esses processos estão esperando para serem executados. Decidir qual processo será executado em seguida, dado um conjunto de processos executáveis, é uma decisão fundamental que o scheduler deve tomar.

Os sistemas operacionais multitarefa vêm em dois tipos:

- multitarefa cooperativa;
- multitarefa preemptiva.

O Linux, como todas as variantes do Unix e a maioria dos sistemas operacionais modernos, oferece multitarefa preemptiva. Na multitarefa preemptiva, o scheduler decide quando um processo deve interromper a execução e um novo processo deve retomar a execução. O ato de suspender involuntariamente um processo em execução é chamado de preempção. O tempo que um processo é executado antes de ser interrompido é predeterminado e é chamado de timeslice (no Tanenbaum chamado de Quantum) do processo.

O timeslice, com efeito, dá a cada processo uma fatia do tempo do processador. Gerenciar o timeslice permite que o scheduler tome decisões de agendamento global para o sistema. Também evita que qualquer processo monopolize o sistema.

Por outro lado, na multitarefa cooperativa, um processo não para de funcionar até que voluntariamente decida fazê-lo. O ato de um processo suspender-se voluntariamente é denominado cedência. As deficiências dessa abordagem são numerosas:

- o escalonador não pode tomar decisões globais sobre por quanto tempo os processos são executados;
- os processos podem monopolizar o processador por mais tempo do que o usuário deseja;
- um processo travado que nunca cede pode potencialmente derrubar todo o sistema.

Felizmente, a maioria dos sistemas operacionais projetados na última década forneceram multitarefa preemptiva, com o Mac OS 9 e anteriores sendo as exceções mais notáveis. Claro, o Unix foi preventivamente multitarefa desde o início.

Durante a série do kernel 2.5, o kernel Linux recebeu uma revisão do scheduler, onde um novo escalonador, comumente chamado de Schedule O⁴⁹ por causa de seu comportamento algorítmico, resolveu as deficiências do escalonador Linux anterior e introduziu novos recursos e características de desempenho poderosos.

7.6.2 Policy

Policy é o comportamento do scheduler que determina o que é executado quando. A política de um scheduler geralmente determina a sensação geral de um sistema e é responsável pela utilização ideal do tempo do processador.

Os processos podem ser classificados como:

- I/O-bound (vinculados ao I/O);
- processor-bound (vinculados aos processadores).

Os processos I/O-bound são caracterizados como processos que passam grande parte do tempo enviando e aguardando solicitações de I/O e consequentemente, esse processo geralmente pode ser executado, mas apenas por curtos períodos, porque eventualmente bloqueará a espera de mais I/O. Por outro lado, os processos vinculados ao processador gastam muito do seu tempo executando código, e estes tendem a ser executados até serem interrompidos porque não bloqueiam as solicitações de I/O com muita frequência. Como eles não são orientados por I/O, no entanto, a resposta do sistema não exige que o scheduler os execute com frequência. A política do scheduler para processos vinculados ao processador, portanto, tende a executar esses processos com menos frequência, mas por períodos mais longos.

A política de agendamento em um sistema deve tentar satisfazer dois objetivos conflitantes:

- **tempo de resposta rápido do processo e alto rendimento do processo:** Para satisfazer esses requisitos, os escalonadores costumam empregar algoritmos complexos para determinar o processo mais interessante a ser executado, sem comprometer a justiça para outros processos de menor prioridade;
- **favorecer os processos vinculados a I/O:** fornece tempo de resposta de processo aprimorado, porque os processos interativos são vinculados a I/O.

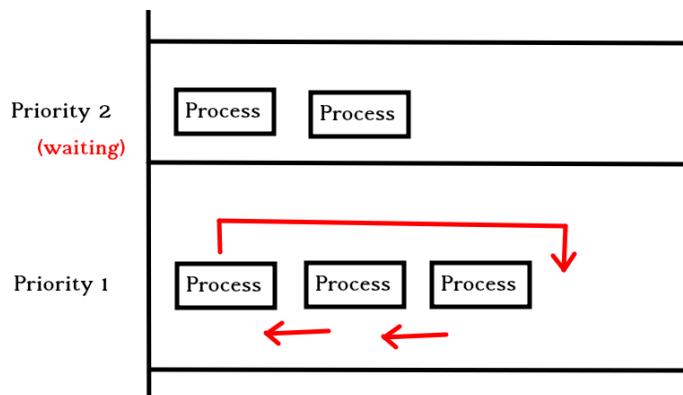
O Linux, para fornecer boa resposta interativa, otimiza a resposta do processo (baixa latência), favorecendo os processos vinculados a I/O em relação aos processadores vinculados ao processador. Como você verá, isso é feito de uma maneira que não negligencia os processos vinculados ao processador.

⁴⁹ Mais dados sobre este Schedule neste link

https://web.archive.org/web/20131231085709/http://joshaas.net/linux/linux_cpu_scheduler.pdf

7.6.3 Prioridade de Processo

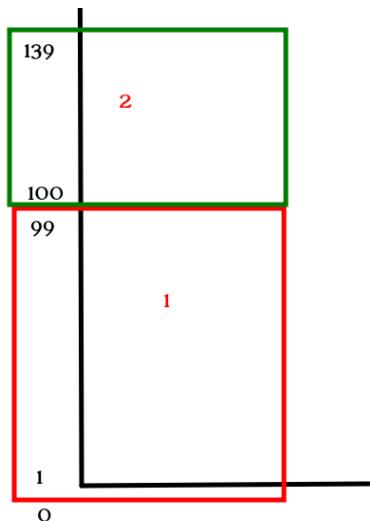
Um tipo comum de algoritmo de escalonamento é o escalonamento baseado em prioridade (caso do Kernel Linux). A ideia é classificar os processos com base em seu valor e necessidade de tempo de processador. Os processos com uma prioridade mais alta serão executados antes daqueles com uma prioridade mais baixa, enquanto os processos com a mesma prioridade são rodízio programado (um após o próximo, repetindo).



Em alguns sistemas, incluindo o Linux, os processos com uma prioridade mais alta também recebem um timeslice mais longo. O processo executável com timeslice restante e a prioridade mais alta sempre é executado. Tanto o usuário quanto o sistema podem definir uma prioridade de processos para influenciar o comportamento de agendamento do sistema.

O Linux se baseia nessa ideia e fornece agendamento dinâmico com base em prioridades e esse conceito começa com a prioridade básica inicial e, em seguida, permite que o agendador aumente ou diminua a prioridade dinamicamente para cumprir os objetivos de agendamento.

Por exemplo, um processo que está gastando mais tempo aguardando I/O do que em execução está claramente ligado à I/O. No Linux, ele recebe uma prioridade dinâmica elevada e como contra-exemplo, um processo que usa continuamente toda a sua fatia de tempo está vinculado ao processador, ele receberia uma prioridade dinâmica reduzida. O kernel do Linux implementa dois intervalos de prioridade separados, o primeiro é destinado aos aplicativos Real Time e o segundo aos aplicativos de usuário.



Onde:

1. Prioridade destinada aos aplicativos Real time;
2. Prioridade destinada aos aplicativos de usuário;

O intervalo de prioridade dos aplicativos Real Time⁵⁰, que será discutida mais tarde. Por padrão, ele varia de zero a 99. Todos os processos em tempo real têm uma prioridade mais alta do que os processos normais. O Linux implementa prioridades em tempo real de acordo com POSIX e a maioria dos sistemas Unix modernos implementam um esquema semelhante.

O outro grupo é o de aplicativos associados ao usuário e portanto não a execução do Kernel, **que por padrão assume a prioridade 120 (fixo)**. E para este grupo podemos "ajustar" a prioridade deduzindo em no máximo -20 ou acrescentando em no máximo 19, chamamos este **incremento/decremento de nice**.

Valores nice maiores correspondem a uma prioridade mais baixa naturalmente e os processos com um valor nice mais baixo (prioridade mais alta). Quem tem a prioridade mais alta são executados antes dos processos com uma prioridade mais baixa.

O valor nice também ajuda a determinar quanto tempo uma fatia de tempo o processo recebe. Um processo com um valor nice de -20 recebe a fatia de tempo máxima, enquanto um processo com um valor agradável de 19 recebe a fatia de tempo mínima. Valores agradáveis são a faixa de prioridade padrão usada em todos os sistemas Unix.

7.6.4 Timeslice (Quantum)

O timeslice é o valor numérico que representa por quanto tempo uma tarefa pode ser executada até ser interrompida. A política do scheduler deve ditar um timeslice padrão, o que não é simples. Uma fatia de tempo muito longa fará com que o sistema tenha um desempenho interativo insatisfatório, já o sistema não terá mais a sensação de que os aplicativos estão sendo executados simultaneamente. Uma fatia de tempo muito curta fará com que uma quantidade significativa de tempo do processador seja desperdiçada na

⁵⁰ Artigo REAL TIME POSIX acessível pela url
<https://www.cs.unc.edu/~anderson/teach/comp790/papers/posix-rt.pdf>

sobrecarga dos processos de comutação, pois uma porcentagem significativa do tempo do sistema será gasta na alternância de um processo com uma fatia de tempo curta para o próximo. Além disso, surgem novamente os objetivos conflitantes de processos vinculados a I/O versus processos vinculados ao processador. Os processos vinculados a I/O não precisam de períodos de tempo mais longos, enquanto os processos vinculados ao processador desejam longos períodos de tempo.

Com este argumento, pareceria que qualquer fatia de tempo longa resultaria em desempenho interativo insatisfatório. Em muitos sistemas operacionais, essa observação é levada a sério e o timeslice padrão é bastante baixo, por exemplo: 20ms.

O Linux, no entanto, aproveita o fato de que o processo de maior prioridade sempre é executado. O agendador do Linux elimina a prioridade das tarefas interativas, permitindo que sejam executadas com mais frequência. Consequentemente, o agendador do Linux oferece um timeslice padrão relativamente alto. Além disso, o escalonador do Linux determina dinamicamente o timeslice de um processo com base na prioridade. Isso permite que processos de maior prioridade, supostamente mais importantes, sejam executados por mais tempo e com mais frequência. A implementação de prazos e prioridades dinâmicas fornece desempenho de agendamento robusto.

Por exemplo, um processo com uma fatia de tempo de 100 milissegundos não precisa ser executado por 100 milissegundos de uma vez ou corre o risco de perder a fatia de tempo restante. Em vez disso, o processo pode ser executado em cinco trocas de contexto de diferentes programas de 20 milissegundos cada. Assim, uma grande fatia de tempo também beneficia as tarefas interativas, embora não precisem de uma fatia de tempo tão grande de uma só vez, ela garante que permaneçam executáveis pelo maior tempo possível.

Quando a fatia de tempo de um processo se esgota, o processo é considerado expirado. Um processo sem fatia de tempo não é elegível para execução até que todos os outros processos tenham esgotado sua fatia de tempo. Nesse ponto, os intervalos de tempo para todos os processos são recalculados. O agendador do Linux emprega um algoritmo interessante para lidar com o esgotamento do timeslice que é discutido em uma seção posterior.

7.6.5 Preempção de processo

Conforme mencionado, o sistema operacional Linux é preventivo. Quando um processo entra no estado **TASK_RUNNING**, o kernel verifica se sua prioridade é maior do que a prioridade do processo atualmente em execução. Se for, o scheduler é chamado para escolher um novo processo a ser executado. Além disso, quando a fatia de tempo de um processo chega a zero, ela é interrompida e o scheduler é chamado para selecionar um novo processo.

7.6.6 A política de agendamento em ação

Considere um sistema com duas tarefas executáveis:

- um editor de texto;
- um codificador de vídeo.

O editor de texto é vinculado a I/O porque gasta quase todo o seu tempo esperando os pressionamentos de tecla do usuário. Apesar disso, quando recebe um pressionamento de tecla, o usuário espera que o editor responda imediatamente. Por outro lado, o codificador de vídeo é vinculado ao processador. Além de ler o fluxo de dados brutos do disco e, posteriormente, gravar o vídeo resultante, o codificador passa o tempo todo aplicando o codec de vídeo aos dados brutos. Ele não tem nenhuma restrição de tempo forte para quando é executado - se começou a ser executado agora ou em meio segundo, o usuário não saberia dizer. Claro, quanto mais cedo terminar, melhor.

Nesse sistema, o escalonador dá ao editor de texto uma prioridade mais alta e um timeslice maior do que o codificador de vídeo, porque o editor de texto é interativo. O editor de texto tem vários timeslice disponíveis. Além disso, como o editor de texto tem uma prioridade mais alta, ele é capaz de antecipar o codificador de vídeo quando necessário. Isso garante que o editor de texto seja capaz de responder aos pressionamentos de tecla do usuário imediatamente. Isso prejudica o codificador de vídeo, mas como o editor de texto é executado apenas de forma intermitente, o codificador de vídeo pode monopolizar o tempo restante. Isso otimiza o desempenho de ambos os aplicativos.

7.6.7 O Algoritmo do Schedule

O schedule do Linux é definido em **kernel/sched.c**. O algoritmo do schedule e o código de suporte passaram por uma grande reescrita no início da série de desenvolvimento do Kernel 2.5. Consequentemente, o código do agendador é inteiramente novo e diferente do agendador dos kernels anteriores. O novo **sched.c** foi projetado para cumprir objetivos específicos:

Implementar O(1) schedule: Cada algoritmo no novo escalonador é concluído em tempo constante, independentemente do número de processos em execução ou de qualquer outra entrada.

Implemente escalabilidade SMP perfeita. Cada processador tem seu próprio bloqueio e fila de execução individual.

Implementar afinidade SMP aprimorada. Naturalmente, tente agrupar tarefas em uma CPU específica e continue a executá-las lá. Apenas migre tarefas de uma CPU para outra para resolver desequilíbrios nos tamanhos da fila de execução.

Fornece bom desempenho interativo. Mesmo durante uma carga considerável do sistema, o sistema deve reagir e agendar tarefas interativas imediatamente.

Ofereça justiça. Nenhum processo deve ficar sem divisão de tempo por um período de tempo razoável. Da mesma forma, nenhum processo deve receber uma quantidade injustamente alta de timeslice.

7.6.8 Runqueues

A estrutura de dados básica no schedule é uma fila de execução, o runqueue é definido em **kernel/sched.c** como uma struct runqueue. A fila de execução é a lista de processos executáveis em um determinado processador, há uma fila de execução por processador. Cada processo executável está em exatamente uma fila de execução, além disso, a fila de execução contém informações de agendamento por processador. Consequentemente, a fila de execução é a estrutura de dados de agendamento primária para cada processador. Por

que **kernel/sched.c** e não inclui **/linux/sched.h** porque é desejado abstrair o código do escalonador e fornecer apenas algumas interfaces para o resto do kernel.

Vejamos a estrutura, da struct:

```

1. struct runqueue {
2.     spinlock_t      lock;                      /* spin lock which protects this
   runqueue */
3.     unsigned long   nr_running;                 /* number of runnable tasks */
4.     unsigned long   nr_switches;                /* number of contextswitches */
5.     unsigned long   expired_timestamp;          /* time of last array swap */
6.     unsigned long   nr_uninterruptible;         /* number of tasks in uninterruptible
   sleep */
7.     struct task_struct *curr;                  /* this processor's currently
   running task */
8.     struct task_struct *idle;                  /* this processor's idle task */
9.     struct mm_struct *prev_mm;                /* mm_struct of last running task */
10.    struct prio_array *active;                /* pointer to the active priority
   array */
11.    struct prio_array *expired;                /* pointer to the expired priority
   array */
12.    struct prio_array arrays[2];              /* the actual priority arrays */
13.    int            prev_cpu_load[NR_CPUS]; /* load on each processor */
14.    struct task_struct *migration_thread;    /* the migration thread on this
   processor */
15.    struct list_head migration_queue;        /* the migration queue for this
   processor */
16.    atomic_t        nr_iowait;                /* number of tasks waiting on
   I/O */
17. }
```

Como as filas de execução são a estrutura de dados central no schedule, um grupo de macros é usado para obter filas de execução específicas. A macro **cpu_rq(processor)** retorna um ponteiro para a fila de execução associada ao processador fornecido. Da mesma forma, a macro **this_rq()** retorna a fila de execução do processador atual. Finalmente, a macro **task_rq(task)** retorna um ponteiro para a fila de execução na qual a tarefa fornecida está enfileirada.

Antes que uma fila de execução possa ser manipulada, ela deve ser bloqueada. Como cada fila de execução é única para o processador atual, é raro quando um processador deseja bloquear a fila de execução de um processador diferente (no entanto, isso acontece, como veremos). O bloqueio da fila de execução proíbe quaisquer alterações nela, enquanto o portador de bloqueio está lendo ou gravando os membros da fila de execução. A maneira mais comum de bloquear uma fila de execução é quando você deseja bloquear a fila de execução na qual uma tarefa específica é executada. Nesse caso, as funções **task_rq_lock()** e **task_rq_unlock()** são usadas:

```

1. struct runqueue *rq;
2. unsigned long flags;
3.
4. rq = task_rq_lock(task, &flags);
5. /* manipulate the task's runqueue */
6. task_rq_unlock(rq, &flags);
```

Alternativamente, o método **this_rq_lock()** bloqueia a fila de execução atual e **rq_unlock(struct runqueue * rq)** desbloqueia a fila de execução fornecida. Para evitar deadlock, o código que deseja bloquear várias filas de execução precisa sempre obter os bloqueios na mesma ordem: aumentando o endereço da fila de execução.

```

1. /* to lock ... */
2. if (rq1 < rq2) {
3.     spin_lock(&rq1->lock);
4.     spin_lock(&rq2->lock);
5. } else {
6.     spin_lock(&rq2->lock);
7.     spin_lock(&rq1->lock);
8. }
9.
10./* manipulate both runqueues ... */
11.
12./* to unlock ... */
13. spin_unlock(&rq1->lock);
14. spin_unlock(&rq2->lock);

```

Essas etapas são automatizadas pelas funções **double_rq_lock()** e **double_rq_unlock()**. As etapas acima se tornaram:

```

1. double_rq_lock(rq1, rq2);
2. /* manipulate both runqueues ... */
3. double_rq_unlock(rq1, rq2);

```

7.6.9 Matrizes de prioridade

Cada fila de execução contém duas matrizes de prioridade, a matriz ativa e a expirada. Matrizes de prioridade são definidas em kernel/sched.c como struct prio_array. Matrizes de prioridade são a estrutura de dados que fornece O(1) schedule. Cada array de prioridade contém uma fila de processadores executáveis por nível de prioridade. Essas filas contêm listas dos processos executáveis em cada nível de prioridade. As matrizes de prioridade também contêm um bitmap de prioridade usado para descobrir com eficiência a tarefa executável de mais alta prioridade no sistema.

```

1. struct prio_array {
2.     int          nr_active;      /* number of tasks */
3.     unsigned long bitmap[BITMAP_SIZE]; /* priority bitmap */
4.     struct list_head queue[MAX_PRIO]; /* priority queues */
5. };

```

MAX_PRIO é o número de níveis de prioridade no sistema. Por padrão, é 140. Portanto, há uma estrutura **list_head** para cada prioridade. **BITMAP_SIZE** é o tamanho que uma matriz de variáveis de tipo longo sem sinal deveria ter para fornecer um bit para cada nível de prioridade válido. Com 140 prioridades e palavras de 32 bits, são cinco. Assim, o bitmap é uma matriz com cinco elementos e um total de 160 bits.

Cada matriz de prioridade contém um campo de bitmap que possui pelo menos um bit para cada prioridade no sistema. Inicialmente, todos os bits são zero. Quando uma tarefa de determinada prioridade se torna executável (ou seja, seu estado se torna `TASK_RUNNING`), o bit correspondente no bitmap é definido como um. Por exemplo, se uma tarefa com prioridade sete puder ser executada, o bit sete será definido. Encontrar a tarefa de maior prioridade no sistema é, portanto, apenas uma questão de encontrar o primeiro bit definido no bitmap. Como o número de prioridades é estático, o tempo para concluir essa pesquisa é constante e não é afetado pelo número de processos em execução no sistema. Além disso, cada arquitetura com suporte no Linux implementa um algoritmo de localização rápida primeiro conjunto para pesquisar rapidamente o bitmap. Este método é chamado `ched_find_first_bit()`.

Cada matriz de prioridade também contém uma matriz chamada fila de filas de struct `list_head`, uma fila para cada prioridade. Cada lista corresponde a uma determinada prioridade e, de fato, contém todos os processos executáveis dessa prioridade que estão na fila de execução deste processador. Encontrar a próxima tarefa a ser executada é tão simples quanto selecionar o próximo elemento na lista. Dentro de uma determinada prioridade, as tarefas são agendadas Round Robin.

A matriz de prioridade também contém um contador, `nr_active`. Este é o número de tarefas executáveis nesta matriz de prioridade.

7.6.10 Recalculando timeslice

Muitos sistemas operacionais têm um método explícito para recalcular a fatia de tempo de cada tarefa quando todas atingem zero. Normalmente, isso é implementado como um loop em cada tarefa, como:

```

1. for (each task on the system) {
2.     recalculate priority
3.     recalculate timeslice
4. }
```

O novo agendador do Linux alivia a necessidade de um loop de recálculo. Em vez disso, ele mantém dois arrays de prioridade para cada processador: um array ativo e um array expirado. A matriz ativa contém todas as tarefas na fila de execução associada que ainda possuem timeslice. A matriz expirada contém todas as tarefas na fila de execução associada que esgotaram sua fatia de tempo. Quando a fatia de tempo de cada tarefa chega a zero, sua fatia de tempo é recalculada antes de ser movida para a matriz expirada. O recálculo de todos os intervalos de tempo é tão simples quanto alternar os arrays ativos e expirados. Como os arrays são acessados apenas por meio de um ponteiro, trocá-los é tão rápido quanto trocar dois ponteiros.

```

1. struct prio_array array = rq->active;
2. if (!array->nr_active) {
3.     rq->active = rq->expired;
4.     rq->expired = array;
5. }
```

Esta troca é um recurso chave do O(1) schedule, em vez de recalcular a prioridade e a fatia de tempo de cada processo o tempo todo, o O(1) schedule executa uma troca de array simples de duas etapas.

7.6.11 Schedule

O ato de escolher a próxima tarefa a ser executada e alternar para ela é implementado por meio da função **schedule()**. Essa função é chamada explicitamente pelo código do kernel que deseja dormir e também é chamada sempre que uma tarefa deve ser interrompida.

A função **schedule()** é relativamente simples para tudo que deve realizar. O código a seguir determina a tarefa de maior prioridade:

```

1. struct task_struct *prev, *next;
2. struct list_head *queue;
3. struct prio_array array;
4. int idx;
5.
6. prev = current;
7. array = rq->active;
8. idx = sched_find_first_bit(array->bitmap);
9. queue = array->queue + idx;
10. next = list_entry(queue->next, struct task_struct, run_list);

```

Primeiro, a matriz de prioridade ativa é pesquisada para encontrar o primeiro bit definido. Este bit corresponde à tarefa de maior prioridade que pode ser executada. Em seguida, o agendador seleciona a primeira tarefa na lista com essa prioridade.

Dois pontos importantes devem ser observados no código anterior. Em primeiro lugar, é muito simples e, consequentemente, bastante rápido. Em segundo lugar, o número de processos no sistema não tem efeito sobre o tempo que esse código leva para ser executado. Não há nenhum loop em qualquer lista para encontrar o processo mais adequado. Na verdade, nada afeta o tempo que o código **schedule()** leva para encontrar uma nova tarefa. É constante no tempo de execução.

7.6.12 Calculando a Prioridade e o timeslice

Os processos têm uma prioridade inicial chamada de valor nice. Este valor varia de -20 a 19 com um padrão de zero. Dezenove é a prioridade mais baixa e -20 é a prioridade mais alta. Este valor é armazenado no **static_prio** membro do processo **task_struct**. O valor é chamado de prioridade estática porque não muda do que o usuário especifica. O escalonador, por sua vez, baseia suas decisões na prioridade dinâmica que é armazenada no **priority**. A prioridade dinâmica é calculada em função da prioridade estática e da interatividade da tarefa.

O método **effective_prio()** retorna a prioridade dinâmica de uma tarefa. O método começa com o valor legal da tarefa e calcula um bônus ou penalidade na faixa de +5 com base na interatividade da tarefa. Por exemplo, uma tarefa altamente interativa com um bom valor de dez pode ter uma prioridade dinâmica de cinco. Por outro lado, um processador moderado com um bom valor de dez pode ter uma prioridade dinâmica de 12. Tarefas que são apenas

moderadamente interativas não recebem nenhum bônus ou penalidade e sua prioridade dinâmica é igual ao seu bom valor.

Obviamente, o escalonador não sabe mágicamente se um processo é interativo, requer alguma heurística que seja capaz de refletir com precisão se uma tarefa é limitada por I/O ou pelo processador. A métrica mais indicativa é quanto tempo a tarefa dorme. Se uma tarefa passa a maior parte do tempo adormecida, ela é limitada por I/O. Se uma tarefa passar mais tempo em execução do que em repouso, ela não será interativa. Isso se estende ao extremo; uma tarefa que passa quase todo o tempo em hibernação é totalmente limitada por I/O, enquanto uma tarefa que gasta quase todo o seu tempo em execução é totalmente limitada pelo processador.

Para implementar essa heurística, o Linux mantém um registro em execução de quanto tempo um processo é gasto em repouso versus quanto tempo o processo passa em um estado executável. Este valor é armazenado no membro `sleep_avg` da `task_struct`. Ele varia de zero a `MAX_SLEEP_AVG`, cujo padrão é 10 milissegundos. Quando uma tarefa se torna executável após dormir, `sleep_avg` é incrementado por quanto tempo ela dormiu, até que o valor alcance `MAX_SLEEP_AVG`. Para cada tique do cronômetro que a tarefa executa, `sleep_avg` é diminuído até chegar a zero.

Essa métrica é surpreendentemente precisa, ela é calculada com base não apenas em quanto tempo a tarefa dorme, mas também em quão pouco ela é executada, e portanto, uma tarefa que gasta muito tempo dormindo, mas também esgota continuamente sua fatia de tempo, não receberá um grande bônus. Também não é vulnerável a abusos, uma tarefa que recebe uma prioridade aumentada e fração de tempo rapidamente perde o bônus se virar e monopolizar o processador. Finalmente, a métrica fornece uma resposta rápida. Um processo interativo recém-criado recebe rapidamente um grande `sleep_avg`. Apesar disso, como o bônus ou penalidade é aplicado contra o valor legal inicial, o usuário ainda pode influenciar as decisões de agendamento do sistema alterando o valor legal do processo.

Por outro lado, o timeslice é um cálculo muito mais simples porque a prioridade dinâmica já é baseada em valor nice. Portanto, o timeslice pode simplesmente ser baseado na prioridade dinâmica. Quando um processo é criado pela primeira vez, o novo filho e o pai dividem a fatia de tempo restante do pai. Isso fornece justiça e evita que os usuários façam novos filhos para obterem timeslice ilimitados. Depois que a fatia de tempo de uma tarefa se esgota, no entanto, ela é recalculada com base na prioridade dinâmica da tarefa. A função `task_timeslice()` retorna um novo timeslice para a tarefa fornecida. O cálculo é uma escala simples da prioridade em um intervalo de intervalos de tempo. Quanto mais alta a prioridade de uma tarefa, mais fatias de tempo ela recebe por rodada de execução. O timeslice máximo, dado às tarefas de prioridade mais alta, é `MAX_TIMESLICE`, que por padrão é 200 milissegundos. Mesmo as tarefas de prioridade mais baixa recebem pelo menos o timeslice mínimo, `MIN_TIMESLICE`, que é de 10 milissegundos.

Timeslice	Duration	Interactivity	Nice Value
Initial	half of parent's	N/A	parent's
Minimum	10ms	low	high
Default	100ms	average	zero
Maximum	3200ms	high	low

O agendador fornece um auxílio adicional para tarefas interativas: se uma tarefa for suficientemente interativa, quando ela esgotar sua fatia de tempo, ela não será inserida na matriz expirada, mas, em vez disso, reinserida de volta na matriz ativa. Lembre-se de que o recálculo do timeslice é fornecido por meio da alternância das matrizes ativa e expirada. Normalmente, à medida que os processos esgotam sua fatia de tempo, eles são movidos da matriz ativa para a expirada. Quando não há mais processos no array ativo, os dois arrays são trocados; o ativo se torna o expirado e o expirado se torna o ativo.

Isso fornece O(1) Schedule. Ele também oferece a possibilidade de que uma tarefa interativa possa se tornar executável, mas não consiga ser executada novamente até que ocorra a troca do array, porque a tarefa está presa no array expirado. A reinserção de tarefas interativas de volta na matriz ativa alivia esse problema. Observe que a tarefa não será executada imediatamente, mas será agendada round robin com as outras tarefas em sua prioridade. A lógica para fornecer esse recurso é implementada em **scheduler_tick()**, que é chamado por meio da interrupção do temporizador:

```

1. struct task_struct *task = current;
2. struct runqueue *rq = this_rq();
3.
4. if (!task->time_slice) {
5.     if (!TASK_INTERACTIVE(task) || EXPIRED_STARVING(rq))
6.         enqueue_task(task, rq->expired);
7.     else
8.         enqueue_task(task, rq->active);
9. }
```

7.6.13 Dormir e acordar

As tarefas que estão suspensas (bloqueadas) estão em um estado especial de não execução. Isso é importante porque, caso contrário, o agendador selecionaria tarefas que não deveriam ser executadas ou, pior, a suspensão teria que ser implementada como um loop ocupado. Uma tarefa dorme por vários motivos, mas sempre esperando por algum evento. O evento pode ser um período de tempo especificado, mais dados de um arquivo de I/O ou outro evento de hardware.

Uma tarefa também pode adormecer involuntariamente ao tentar obter um semáforo contido no kernel. Um motivo comum para dormir é a I/O de arquivo - por exemplo, a tarefa emitiu um **read()** pedido em um arquivo que precisa ser lido do disco. Como outro exemplo, a tarefa pode estar aguardando uma entrada do teclado. Seja qual for o caso, o

1. comportamento do kernel é o mesmo;
2. a tarefa marca a si mesma como inativa;
3. se coloca em uma fila de espera;
4. se remove da fila de execução;
5. chama `schedule()` para selecionar um novo processo a ser executado.

Despertar é o inverso; a tarefa é definida como executável, removida da fila de espera e adicionada de volta à fila de execução.

Conforme discutido no capítulo anterior, dois estados estão associados ao repouso, **TASK_INTERRUPTIBLE** e **TASK_UNINTERRUPTIBLE**. Eles diferem apenas no fato de que as tarefas no estado **TASK_UNINTERRUPTIBLE** ignoram os sinais, enquanto as tarefas no estado **TASK_INTERRUPTIBLE** serão ativadas prematuramente e responderão a um sinal se um for emitido. Ambos os tipos de tarefas adormecidas ficam em uma fila de espera, aguardando a ocorrência de um evento, e não podem ser executados.

O dormir é feito por meio de filas de espera, uma fila de espera é uma lista simples de processos que aguardam a ocorrência de um evento. As filas de espera são representadas no kernel por `wake_queue_head_t`. As filas de espera são criadas estaticamente via **DECLARE_WAIT_QUEUE_HEAD()** ou dinamicamente via **init_waitqueue_head()**. Os processos se colocam em uma fila de espera e se marcam como não executáveis. Quando o evento associado à fila de espera ocorre, os processos na fila são ativados. É importante implementar dormir e acordar corretamente, para evitar condições de corrida.

Algumas interfaces simples para dormir costumavam ser amplamente utilizadas. Essas interfaces, no entanto, têm raças; é possível dormir depois que a condição se torna verdadeira. Nesse caso, a tarefa pode dormir indefinidamente. Portanto, o método recomendado para dormir no kernel é um pouco mais complicado:

```

1. /* 'q' is the wait queue we wish to sleep on */
2. DECLARE_WAITQUEUE(wait, current);
3. add_wait_queue(q, &wait);
4. set_current_state(TASK_INTERRUPTIBLE); /* or TASK_UNINTERRUPTIBLE */
5. while (!condition) /* 'condition' is the event we are waiting for */
6.     schedule();
7. set_current_state(TASK_RUNNING);
8. remove_wait_queue(q, &wait);

```

Se a condição ocorrer antes de a tarefa entrar em hibernação, o loop será encerrado e a tarefa não entrará em hibernação erroneamente. Observe que o código do kernel geralmente precisa realizar várias outras tarefas no corpo do loop. Por exemplo, pode ser necessário liberar bloqueios antes de chamar **schedule()** e readquiri-los depois, verificar se um sinal foi entregue e retornar -ERESTARTSYS ou reagir a outros eventos.

O despertar é feito por meio de **wake_up()**, que desperta todas as tarefas em espera na fila de espera fornecida. Ele chama **try_to_wake_up()**, que define o estado da tarefa para **TASK_RUNNING**, chama **activate_task()** para adicionar a tarefa a uma fila de execução e define **need_resched** se a prioridade da tarefa acordada for maior do que a prioridade da tarefa atual. O código que faz com que o evento ocorra normalmente chama **wake_up()**

depois. Por exemplo, quando os dados chegam do disco rígido, o VFS chama **wake_up()** na fila de espera que mantém os processos que aguardam os dados.

Uma observação importante sobre o sono é que existem despertares espúrios. Só porque uma tarefa foi ativada não significa que o evento pelo qual ela estava esperando ocorreu; dormir deve sempre ser tratado em um loop que garanta que a condição pela qual a tarefa está esperando realmente ocorreu.

7.6.14 Preempção e troca de contexto

A troca de contexto, a troca de uma tarefa executável para outra, é tratada pela função `context_switch()` definida em `kernel/sched.c`. É chamado por `schedule()` quando um novo processo é selecionado para execução. Ele faz duas tarefas básicas:

Chama `switch_mm()`, que é definido em `include /asm/mmu_context.h`, para alternar o mapeamento da memória virtual do processo anterior para o do novo processo;

Chama `switch_to()`, definido em `include /asm/system.h`, para mudar o estado do processador do processo anterior para o atual. Isso envolve salvar e restaurar as informações da pilha e os registros do processador.

O kernel, entretanto, deve saber quando chamar o `schedule()`. Se ele apenas chamassem `schedule()` quando o código o fazia explicitamente, os programas de espaço do usuário poderiam ser executados indefinidamente. Em vez disso, o kernel fornece o sinalizador **need_resched** para indicar se um reescalonamento deve ser executado. Este sinalizador é definido por `scheduler_tick()` quando um processo fica sem timeslice e por `try_to_wake_up()` quando um processo que tem uma prioridade mais alta do que o processo atualmente em execução é ativado. O kernel irá verificar o sinalizador, ver se ele está definido e chamar `schedule()` para mudar para um novo processo. O sinalizador é uma mensagem para o kernel de que o `schedule` deve ser chamado o mais rápido possível porque outro processo merece ser executado.

Ao retornar ao espaço do usuário ou retornar de uma interrupção, o sinalizador **need_resched** é verificado. Se estiver definido, o kernel invoca o `schedule` antes de continuar.

O sinalizador é por processo, e não simplesmente global, porque é mais rápido acessar um valor no descritor de processo do que uma variável global. Historicamente, a flag era global antes do kernel 2.2. Em 2.2 e 2.4, o sinalizador era um int dentro de **task_struct**. No 2.6, ele foi movido para um único bit de uma variável de sinalização especial dentro da estrutura `thread_info`. Como você pode ver, os desenvolvedores do kernel nunca estão satisfeitos.

7.6.15 Preempção do usuário

A preempção do usuário ocorre quando o kernel está prestes a retornar ao espaço do usuário, **need_resched** é definido e, portanto, o `schedule` é chamado. Se o kernel está retornando ao espaço do usuário, ele sabe que está em um estado quiescente seguro. Em outras palavras, se é seguro continuar executando a tarefa atual, também é seguro escolher uma nova tarefa para executar. Consequentemente, sempre que o kernel está se preparando para retornar ao espaço do usuário, seja no retorno de uma interrupção ou após

uma chamada de sistema, o valor de **need_resched** é verificado. Se estiver definido, o `schedule` é chamado para selecionar um novo processo a ser executado. Ambos os caminhos de retorno para retorno da interrupção e retorno da chamada do sistema são dependentes da arquitetura e normalmente implementados no assembly na entrada.

Resumindo, a preempção do usuário pode ocorrer:

- Ao retornar ao espaço do usuário de uma chamada de sistema;
- Ao retornar ao espaço do usuário de um manipulador de interrupção.

7.6.16 Preempção de kernel

O kernel do Linux, ao contrário da maioria das outras variantes do Unix e muitos outros sistemas operacionais, é um kernel totalmente preemptivo. Em kernels não preemptivos, o código do kernel é executado até a conclusão. Ou seja, o agendador não é capaz de reprogramar uma tarefa enquanto ela está no kernel - o código do kernel é agendado cooperativamente, não preventivamente. O código do kernel é executado até terminar ou ser bloqueado explicitamente. No kernel 2.6, entretanto, o kernel Linux tornou-se preemptivo; agora é possível antecipar uma tarefa a qualquer momento, desde que o kernel esteja em um estado seguro para reprogramar.

Então, quando é seguro reagendar? O kernel é capaz de antecipar uma tarefa em execução no kernel, desde que não tenha um bloqueio. Ou seja, os bloqueios são usados como marcadores de regiões de não preempção. Como o kernel é seguro para SMP, se um bloqueio não for mantido, o código atual é reentrante e pode ser antecipado.

A primeira mudança no apoio preempção núcleo foi o acréscimo de um contador de preempção, **preempt_count**, para cada processo de **task_struct**. Este contador começa em zero e aumenta para cada bloqueio adquirido e diminui para cada bloqueio liberado. Quando o contador é zero, o kernel é preemptivo. Ao retornar da interrupção, se retornar ao espaço do kernel, o kernel verifica os valores de **need_resched** e **preempt_count**. Se **need_resched** estiver definido e **preempt_count** for zero, uma tarefa mais importante pode ser executada e é seguro antecipar. Assim, o `schedule` é chamado. Se **preempt_count** é diferente de zero, um bloqueio está suspenso e não é seguro reagendar. Nesse caso, a interrupção retorna normalmente para a tarefa atualmente em execução. Quando todos os bloqueios que a tarefa atual está retendo são liberados, **preempt_count** retorna a zero. Nesse momento, o código de desbloqueio verifica se **need_resched** está definido. Nesse caso, o `schedule` será chamado. Habilitar e desabilitar a preempção do kernel às vezes é necessário no código do kernel.

A preempção do kernel também pode ocorrer explicitamente, quando uma tarefa no kernel bloqueia ou chama explicitamente **schedule()**. Essa forma de preempção do kernel sempre foi suportada porque nenhuma lógica adicional é necessária para garantir que o kernel esteja em um estado seguro para a preempção. Presume-se que o código que chama explicitamente o **schedule()** sabe que é seguro reprogramar.

A preempção do kernel pode ocorrer:

- Ao retornar ao espaço do kernel de um manipulador de interrupção;
- Quando o código do kernel se torna preemptivo novamente;

- Se uma tarefa no kernel chama explicitamente `schedule()`;
- Se uma tarefa no kernel bloqueia (o que resulta em uma chamada para `schedule()`).

7.6.17 Tempo real (real-time)

O Linux oferece duas políticas de agendamento em tempo real, **SCHED_FF** e **SCHED_RR**. A política de agendamento normal para processos que não são de tempo real é **SCHED_OTHER**. **SCHED_FIFO** implementa um algoritmo de escalonamento simples de fila, sem intervalos de tempo. Uma tarefa em **SCHED_FIFO** executável sempre será agendada em qualquer tarefa **SCHED_OTHER**. Quando uma tarefa **SCHED_FIFO** se torna executável, ela continuará a ser executada até bloquear ou fornecer explicitamente o processador; ele não tem timeslice e pode ser executado indefinidamente. Duas ou mais tarefas **SCHED_FIFO** na mesma prioridade são executadas em rodízio. Se um **SCHED_FIFO** a tarefa pode ser executada, todas as tarefas com prioridade mais baixa não podem ser executadas até que sejam concluídas.

SCHED_RR é idêntico a **SCHED_FIFO**, exceto que cada processo só pode ser executado até esgotar uma fatia de tempo predeterminada. Ou seja, **SCHED_RR** é **SCHED_FIFO** com timeslices é um algoritmo de agendamento round-robin em tempo real.

Ambas as políticas de agendamento em tempo real implementam prioridades estáticas. O kernel não calcula valores de prioridade dinâmica para tarefas em tempo real. Isso garante que um processo em tempo real em uma determinada prioridade sempre substituirá um processo em uma prioridade mais baixa.

As políticas de agendamento em tempo real no Linux fornecem comportamento suave em tempo real. Soft real-time refere-se à noção de que o kernel tenta agendar aplicativos dentro dos prazos, mas o kernel nem sempre promete ser capaz de cumpri-los. Por outro lado, os sistemas de tempo real hard têm garantia de atender a todos os requisitos de programação dentro de certos limites. O Linux não oferece garantias sobre a capacidade de agendar tarefas em tempo real. A política de agendamento do Linux, no entanto, garante que as tarefas em tempo real sejam executadas sempre que puderem ser executadas. Apesar de não ter um design que garanta um comportamento rígido em tempo real, o desempenho do escalonamento em tempo real no Linux é muito bom. O kernel 2.6 é capaz de atender a requisitos de tempo muito rigorosos.

As prioridades em tempo real variam inclusive de um a **MAX_RT_PRIO** menos um. Por padrão, **MAX_RT_PRIO** é 100, portanto, o intervalo de prioridade em tempo real padrão é de 1 a 99. Este espaço de prioridade é compartilhado com os bons valores de tarefas **SCHED_OTHER**; eles usam o espaço de **MAX_RT_PRIO** a $(\text{MAX_RT_PRIO} + 40)$. Por padrão, isso significa que o intervalo agradável de -20 a +19 mapeia diretamente no intervalo de prioridade de 100 a 140.

7.6.18 Chamadas do sistema relacionadas ao agendador

O Linux fornece uma família de chamadas de sistema para o gerenciamento de parâmetros do `schedule`. Essas chamadas de sistema permitem a manipulação da prioridade do

processo, política de agendamento e afinidade do processador, além de fornecer um mecanismo explícito para submeter o processador a outras tarefas.

As chamadas de sistema **sched_setscheduler()** e **sched_getscheduler()** definem e obtêm a política de agendamento de um determinado processo e a prioridade em tempo real, respectivamente. Sua implementação, como a maioria das chamadas de sistema, envolve muita verificação de argumento, configuração e limpeza. O trabalho importante, porém, é apenas para ler ou escrever as políticas e **rt_priority** valores do processo **task_struct**.

As chamadas de sistema **sched_setparam()** e **sched_getparam()** são definidas e obtêm a prioridade em tempo real de um processo. Esta chamada simplesmente retorna **rt_priority** codificada em uma estrutura **sched_param** especial.

As chamadas **sched_get_priority_max()** e **sched_get_priority_min()** retornam as prioridades máximas e mínimas, respectivamente, para uma determinada política de agendamento. A prioridade máxima para as políticas em tempo real é **MAX_USER_RT_PRIO** menos um, o mínimo é um.

Para tarefas normais, a função **nice()** incrementa a prioridade estática de determinado processo em um determinado valor. Apenas root pode fornecer um valor negativo, diminuindo assim o valor agradável e aumentando a prioridade. O **nice()** chama a função do kernel **set_user_nice()**, que define o **static_prio** e **Prio** valores na tarefa **task_struct**.

O schedule do Linux reforça a afinidade do processador, ou seja, embora tente fornecer afinidade de software ou natural ao tentar manter os processos no mesmo processador, o escalonador também permite que um usuário diga "esta tarefa deve permanecer neste subconjunto dos processadores disponíveis, não importa o que aconteça." Essa afinidade rígida é armazenada como uma máscara de bits na **task_struct** da tarefa como **cpus_allowed**.

O kernel reforça a afinidade de uma maneira muito simples. Primeiro, quando um processo é criado pela primeira vez, ele herda a máscara de afinidade de seu pai. Como o pai está executando em um processador permitido, o filho, portanto, executa em um processador permitido. Em segundo lugar, quando a afinidade de um processador é alterada, o kernel usa os threads de migração para enviar a tarefa para um processador legal. Por fim, o平衡ador de carga apenas extrai tarefas para um processador permitido. Portanto, um processo só é executado em um processador cujo bit é definido no campo **cpus_allowed** de seu descritor de processo.

7.7 Comunicação Inter-Processo

Comunicação entre processos (IPC) refere-se especificamente aos mecanismos de um sistema operacional que fornece para permitir que os processos para gerenciar dados compartilhados. Normalmente, os aplicativos podem usar IPC, categorizados como clientes e servidores, onde o cliente solicita dados e o servidor responde às solicitações do cliente. Muitos aplicativos são clientes e servidores, como comumente visto na computação distribuída.

O IPC é muito importante para o processo de design de microkernels e nanokernels, que reduzem o número de funcionalidades fornecidas pelo kernel. Essas funcionalidades são então obtidas através da comunicação com servidores via IPC, levando a um grande aumento na comunicação quando comparado a um kernel monolítico regular. As interfaces IPC geralmente abrangem estruturas de framework analítico variável. Esses processos garantem a compatibilidade entre os protocolos multivetoriais dos quais os modelos IPC dependem. São formas de comunicação entre processos:

- Arquivo;
- Sinal;
- Socket (cliente-servidor);
- Fila de mensagens;
- Pipe;
- Memória compartilhada (**fork()**);

7.8 Exibindo dados de processos com top e htop

O comando top é provavelmente o mais básico e é o mais usado para acompanhar e exibir os processos atuais que estão em nossa máquina. Quando executarmos o comando top em um terminal, vemos uma lista de processo que é atualizada em tempo real.

top - 00:31:58 up 7:07, 1 user, load average: 1,19, 1,56, 1,33										
Tasks: 411 total, 2 running, 409 sleeping, 0 stopped, 0 zombie										
%Cpu(s): 18,5 us, 3,3 sy, 0,0 ni, 78,1 id, 0,0 wa, 0,0 hi, 0,1 si, 0,0 st										
MiB Mem : 15946,6 total, 3946,1 free, 5701,1 used, 6299,5 buff/cache										
MiB Swap: 2048,0 total, 2048,0 free, 0,0 used. 9325,6 avail Mem										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
54108	well	20	0	5180912	732872	11844	R	105,3	4,5	14:17.21 kdenlive
2016	well	20	0	5062684	499732	126532	S	11,0	3,1	34:16.33 gnome-shell
1233	root	-51	0	0	0	0	S	6,6	0,0	6:59.98 irq/128-nvidia
1886	root	20	0	269164	106228	66192	S	4,0	0,7	41:47.99 Xorg
1796	well	9	-11	4064344	21532	15952	S	1,3	0,1	42:21.65 pulseaudio
14012	well	20	0	40,7g	330672	163788	S	1,3	2,0	4:26.92 brave
55649	well	20	0	40,5g	280732	122732	S	1,0	1,7	0:33.46 chrome
56450	well	20	0	12128	4396	3356	R	0,7	0,0	0:00.25 top
1236	root	20	0	0	0	0	S	0,3	0,0	0:46.58 nv_queue
5575	well	20	0	2516540	338380	119740	S	0,3	2,1	5:58.11 telegram-deskt
5968	well	20	0	735984	233948	85012	S	0,3	1,4	4:31.58 brave
14361	well	20	0	36,6g	542996	183736	S	0,3	3,3	2:52.04 brave
52997	well	20	0	946744	322328	171756	S	0,3	2,0	0:48.85 chrome
53038	well	20	0	394148	110224	67760	S	0,3	0,7	0:19.29 chrome
56421	well	20	0	813344	50072	38888	S	0,3	0,3	0:00.54 gnome-terminal-
1	root	20	0	168364	12360	8564	S	0,0	0,1	0:03.23 systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00 kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00 rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00 rcu_par_gp
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00 kworker/0:0H-kblockd
9	root	0	-20	0	0	0	I	0,0	0,0	0:00.00 mm_percpu_wq
10	root	20	0	0	0	0	S	0,0	0,0	0:00.10 ksoftirqd/0

Este novo layout é realmente interativo através de botões de teclado, parâmetros:

- **h ou ?** Mostrar uma janela de ajuda com todos os comandos e outras informações úteis;
- **space** pressionando espaço em seu teclado irá atualizar a tabela de processo imediatamente em vez de esperar os poucos segundos;
- **f** Adicionar campos para exibir no layout ou remover certos campos para que não sejam.

- **q** sair da aplicação top ou uma janela adicionalmente aberta que pertence à aplicação top;
- **I** Alterna a exibição da média de carga e informações de tempo de atividade;
- **m** Alterna exibição de informações de memória;

A ordenação:

shift+p Classificar os processos por uso da CPU;

shift+m Classificar os processos por uso da Memória;

shift+t Classificar os processos por uso do Tempo;

Veja a saída:

```
top - 00:42:45 up 7:18, 1 user,  load average: 2,11, 1,87, 1,62
Tasks: 422 total, 1 running, 421 sleeping, 0 stopped, 0 zombie
%Cpu(s): 19,3 us, 3,3 sy, 0,2 ni, 77,1 id, 0,0 wa, 0,0 hi, 0,1 si, 0,0 st
MiB Mem : 15946,6 total, 3353,9 free, 6187,7 used, 6405,0 buff/cache
MiB Swap: 2048,0 total, 2048,0 free, 0,0 used. 8804,9 avail Mem
```

Onde:

users: Número de usuários concorrentes no Linux;

load average: Número de processos que estão em execução/fila (1 minuto, 5 minutos, 15 minutos);

tasks: Número de tarefas que estão ativas no computador;

running: que está em execução ou presentes em uma fila;

sleeping: aguardam uma interrupção ou IO;

stopped: processos que receberam sinal de parada;

zombie: Número de processos que perderam referência do pai;

%CPU: porcentagem de tempo de CPU (processo de usuário);

sys: Porcentagem de CPU consumido por processos do sistema;

nice: processo com nice definido (não pelo sistema);

id: tempo em que a CPU permanece ociosa;

wa: Processos que aguardam I/O;

hi: Hardware que solicitam interrupção;

si: Software que solicita interrupção;

st: quando a máquina é virtual informa tempo para trocar contexto guest-host;

Mem total: Memória total da máquina;

Mem free: Memória não utilizada e apta para uso imediato;

Mem used: memória em uso e que não pode ser desvinculada;

Mem cache: memória que pode ser utilizada mas requer procedimento de desvinculação;

Swap total: Total de área para swap;

Swap free: Total da área de swap não utilizada;

Swap used: Total da área de swap utilizada;

Lista de processos:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
54108	well	20	0	5183368	740292	11844	S	114,6	4,5	26:18.25	kdenlive
2016	well	20	0	5090532	541256	126536	R	10,3	3,3	35:42.72	gnome-shell
1233	root	-51	0	0	0	0	S	7,0	0,0	7:44.82	irq/128-nvidia
1886	root	20	0	304924	142768	76780	S	6,0	0,9	43:11.00	Xorg
994	root	20	0	393852	14260	10296	S	3,3	0,1	0:05.83	udisksd
1170	gdm	20	0	314216	9800	7672	S	1,7	0,1	0:03.64	gvfs-udisks2-vo
1796	well	9	-11	3802200	21752	16172	R	1,7	0,1	42:30.59	pulseaudio
1824	well	20	0	387024	10280	7802	S	1,7	0,1	0:02.60	gvfs-udisks2-vo

Onde:

PID: Process ID, identificação numérica;

User: Usuário dono do processo;

PR: Prioridade do processo no sistema;

NI: Nice do processo;

Mem Virt: Toda memória utilizada, do processo, das bibliotecas que este utiliza, arquivos etc.. ou seja, tudo que está relacionado com o processo;

Mem Res: Memória real utilizada na RAM;

Mem Shr: Memória do processo que pode ser compartilhada;

Use o comando top com a opção **-u** para exibir detalhes específicos dos processos de um determinado usuário.

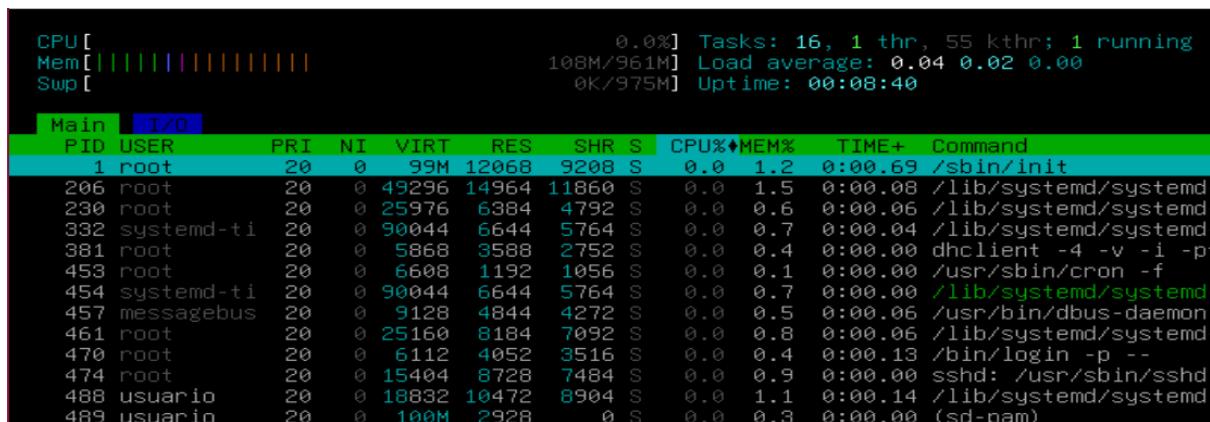
```
usuario@debian:~$  
usuario@debian:~$  
usuario@debian:~$ top -u usuario
```

Para salvar a saída dos resultados do comando superior em execução em um arquivo /tmp/top.out, use o seguinte comando.

```
usuario@debian:~$  
usuario@debian:~$ top -n 1 > /tmp/top.out  
usuario@debian:~$  
usuario@debian:~$ cat /tmp/top.out
```

O comando htop é um comando GNU/Linux para exibir informações cruciais sobre os processos do sistema. Ele pode ser considerado como uma contraparte GNU/Linux do Windows Task Manager. O comando htop é mais um programa interativo, pois suporta operações de mouse e teclado para alternar entre valores e guias.

Depois de executar o comando, o terminal é transformado em uma representação colorida de dados, conforme figura abaixo.



O painel superior esquerdo na tela corresponde ao uso de CPU e memória.



Aparece uma contagem com o número de CPUs, neste caso aparece apenas 1 CPU, representa os núcleos/CPUs do sistema. A barra descreve a quantidade e o tipo de processos usando cada núcleo. O valor em relação à barra denota a porcentagem que cada núcleo está sendo consumido.

Como é bastante visível para o usuário, existem várias cores usadas para descrever a barra. Cada cor tem um significado específico:

Green – Quantidade de CPU consumida pelos processos do usuário.

Red – Quantidade de CPU usada por processos do sistema.

Grey – Quantidade de CPU usada para processos baseados em entrada/saída.

O painel também contém informações relacionadas à quantidade de memória usada em cada instante. A barra “mem” representa a memória principal (ou RAM), enquanto “swp” refere-se à memória swap.

Green – Porcentagem de RAM sendo usada para executar processos no sistema.

Blue – Percentual de RAM sendo consumida por páginas de buffer.

Orange – Porcentagem de RAM sendo usada para memória cache.

7.9 Exibindo dados de processos com ps e pstree

Como todos sabemos, o GNU/Linux é um sistema multitarefa e multiusuário, e portanto, permite que vários processos operem simultaneamente sem interferir um no outro. O processo é um dos conceitos fundamentais e mais importante da teoria de Sistemas Operacionais, tanto que Tanenbaum se dedica 50% de seu tempo descrevendo diretamente este conceito, mas basicamente e sendo muito simplista neste momento, um processo é uma instância de execução de um programa e executa tarefas diferentes dentro do sistema operacional.

O GNU/Linux nos fornece um comando chamado ps para visualizar informações relacionadas aos processos em um sistema que se destaca como abreviação de Process Status. O comando ps é usado para listar os processos em execução no momento e seus PIDs, juntamente com algumas outras informações, dependem de diferentes opções.

Ele lê as informações do processo dos arquivos virtuais em /proc e neste diretório encontramos arquivos virtuais.

1. ps [opções]

Veja exemplo do output do processo ps.

```
usuario@debian:~$ ps
usuario@debian:~$ ps
  PID TTY      TIME CMD
 477 tty1    00:00:00 bash
 533 tty1    00:00:00 ps
usuario@debian:~$ _
```

Onde,

PID o ID de processo;

TTY tipo de terminal em que o usuário está conectado;

TIME quantidade de CPU em minutos e segundos que o processo está em execução;

CMD nome do comando que iniciou o processo.

Às vezes, quando executamos comando ps, mostra TIME como 00:00:00. Não passa de um tempo total acumulado de utilização da CPU para qualquer processo e 00:00:00 indica que nenhum tempo da CPU foi dado pelo kernel até agora. No exemplo acima, descobrimos que, para o bash, nenhum tempo da CPU foi concedido. Isso ocorre porque o bash é apenas um processo pai para diferentes processos que precisam de bash para sua execução e o próprio bash não está utilizando nenhum tempo da CPU até agora.

```
usuario@debian:~$ pstree 1
systemd-cron
|-dbus-daemon
|-dhclient
|-login-bash
|-login-bash-pstree
|sshd
|-systemd-(sd-pam)
|-systemd-journal
|-systemd-logind
|-systemd-timesyncd-{systemd-timesync}
|-systemd-udevd
usuario@debian:~$ _
```

O comando **pstree** no Linux mostra os processos em execução como uma árvore, que é uma maneira mais conveniente de exibir a hierarquia de processos e torna a saída mais atraente visualmente. A raiz da árvore é o process 1 em um GNU/Linux.

1. pstree [opções] [usuário, pid]

Algumas opções de argumentos:

- c** Desative a compactação de subárvores idênticas;
- G** Use caracteres de desenho de linha VT100.;
- h** Destaque o processo atual e seus ancestrais;
- H** Como -h, mas destaque o processo especificado;
- g** Mostrar PGIDs. Os IDs do grupo de processos são mostrados como números decimais entre parênteses após cada nome de processo;
- n** Classifique processos com o mesmo ancestral por PID em vez de por nome;
- p** Mostrar PIDs. Os PIDs são mostrados como números decimais entre parênteses após cada nome de processo;
- s** Mostrar processos pai do processo especificado;
- V** Exibir informações da versão.

7.10 Comando free

Por vezes, no entanto, o comando top pode ser um pouco demais para as suas necessidades básicas, tal como precisar ver a quantidade de memória livre e usada em seu sistema. Para isso, há o comando free supre suas necessidades.

1. `free [options]`

A saída do comando abaixo é exibida na figura abaixo.

```
usuario@debian:~$ free
              total        used        free      shared  buff/cache   available
Mem:      983768      251056      658100          496      214536      732712
Swap:      998396          0      998396
usuario@debian:~$ _
```

O comando free sem opções retorna resultados para memória:

- total (memória física);
- usada
- disponível

Ele também exibe categorias para:

- Shared (compartilhado)
- buff/cache
- Disponível (memória física + swap).

O comando free também permite que você especifique a unidade de medida da memória. As opções válidas são:

- b, --bytes** Exibir saída em bytes;
- kilo** Exibir a saída em kilobytes;
- mega** Exibição de saída em megabytes;
- giga** Exibir saída em gigabytes;
- tera** Exibir saída em terabytes;
- k, --kibi** Exibir saída em kibibytes;
- m, --mebi** Exibir saída em mebibytes;

-g, --gibi	Exibir saída em gibbytes;
--tebi	Exibir saída em tebibytes;
--peti	Exibir saída em pebibytes;
--si	Em vez de 1024, use 1000.

7.11 Trabalhos em background com comando bg e fg

O comando `bg` retomará os trabalhos suspensos no ambiente atual, executando-os como trabalhos de Background. Se o trabalho especificado já estiver em execução em background, o comando `bg` não tem efeito e sai com sucesso. Se nenhum parâmetro JobID for fornecido, o comando `bg` usa o trabalho suspenso mais recente.

Usar o comando `bg` para colocar um trabalho em segundo plano faz com que o ID do processo do trabalho seja conhecido no ambiente de shell atual. A saída de comando `bg` exibe o número de trabalho e o comando associado a essa tarefa.

1. `bg [JobID ...]`

O parâmetro JobID pode ser um número de ID de processo ou você pode usar uma das seguintes combinações de símbolos:

%Number	Refere-se a um trabalho pelo número de trabalho.
%String	Refere-se a uma tarefa cujo nome começa com a cadeia especificada.
%?String	Refere-se a uma tarefa cujo nome contenha a cadeia especificada.
%+ OR %%	Refere-se ao trabalho atual.
%-	Refere-se ao trabalho anterior.

Se o controle de trabalho estiver ativado, o comando `fg` move um trabalho do background para o ambiente atual (primeiro plano). Use o parâmetro JobID para indicar um trabalho específico a ser executado em primeiro plano. Se esse parâmetro não for fornecido, o comando `fg` usa o trabalho mais recentemente suspenso, colocado em segundo plano ou executado como um trabalho de fundo.

1. `fg [JobID ...]`

O parâmetro JobID pode ser um número de ID de processo ou você pode usar uma das seguintes combinações de símbolos:

%Number	Refere-se a um emprego pelo número de trabalho;
%String	Refere-se a uma tarefa cujo nome começa com a cadeia especificada;
%?String	Refere-se a uma tarefa cujo nome contém a cadeia de caracteres especificada;
%+OR%	refere-se ao trabalho atual;
%-	Refere ao trabalho anterior.

7.12 Alterando a prioridade de um processo com nice e renice

O comando `nice` executa um outro comando com uma prioridade inferior ou superior se comparado com a prioridade padrão 120. O parâmetro `Command` é o nome de qualquer arquivo executável no sistema. Se você não especificar um valor de incremento, o comando `nice` é padrão para um incremento de 10.

O valor de `nice` pode variar de -20 a 19, sendo 19 a menor prioridade ($120 + 19$). Por exemplo, se um comando normalmente é executado com uma prioridade de 10, especificando um incremento de 5 executa o comando com uma prioridade inferior, 15, e o comando é executado mais lentamente. O comando `nice` não retorna uma mensagem de erro se você tentar aumentar a prioridade de um comando sem a autoridade apropriada, e em vez disso, a prioridade do comando não é alterada e o sistema inicia o comando como normalmente faria.

O valor `nice` é usado pelo sistema para calcular a prioridade atual de um processo em execução. Use o comando `ps` com o sinalizador `-l` para ver o valor `nice` de um comando.

```
1. nice [ -Increment | -n Increment ] Command [ Argument ... ]
```

Onde:

-Increment Incrementos a prioridade de um comando para cima ou para baixo.

Você pode especificar um número positivo ou negativo;

-n Increment Esta forma é equivalente à `-Increment`.

O comando `renice` altera valor `nice` dos processos em execução.

```
1. renice [-n Increment] [-g | -p | -u] ID ...
```

O comando `renice` altera o valor `nice` de um ou mais processos em execução. Por padrão, os processos afetados são especificados por seus IDs de processo e quando você especifica um grupo de processos, a solicitação se aplica a todos os processos no grupo de processo.

Se você não tiver autoridade de usuário `root`, você só pode redefinir a prioridade dos processos que você possui e só pode aumentar sua prioridade dentro do intervalo de 0 a 20, com 20 sendo a prioridade mais baixa. Se você tiver autoridade de usuário `root`, poderá alterar a prioridade de qualquer processo e definir a prioridade para qualquer valor no intervalo -20 a 20.

O incremento especificado altera a prioridade de um processo das seguintes maneiras:

-g Interpreta todos os IDs como IDs de grupo de processo inteiro decimal não assinado;

-n Increment Especifica o número para adicionar ao bom valor do processo;

-p Interpreta todos os IDs como IDs de processo inteiro não assinados. O sinalizador `-p` é o padrão se você especificar nenhuma outra flag;

-u Interpreta todos os IDs como nome de usuário ou IDs de usuário numérico.

7.13 Sensors

Um programa muito útil em servidores que são utilizados para processamento pesado é o lm-sensors, um pacote que não vem instalado no Debian mas que é muito útil. Você pode rapidamente com este comando obter dados de temperatura de núcleos de processamento e outros sensores de temperatura da placa mãe e de vídeo. Para instalar basta executar o comando abaixo:

```
1. sudo apt install lm-sensors -y
```

O comando sensors irá ler arquivos em /proc, então para executar utilize o comando sudo, conforme listagem abaixo.

```
1. sudo sensors
```

Veja o output de um servidor Xeon 2680, repare que cada core tem uma temperatura diferente pois a distribuição de processos não é bem igualitária e fisicamente não estão na mesma posição, conforme figura abaixo.

```
well@big1:~$ sudo sensors
[sudo] password for well:
coretemp-isa-0000
Adapter: ISA adapter
Package id 0: +59.0°C  (high = +82.0°C, crit = +92.0°C)
Core 0:      +54.0°C  (high = +82.0°C, crit = +92.0°C)
Core 1:      +57.0°C  (high = +82.0°C, crit = +92.0°C)
Core 2:      +52.0°C  (high = +82.0°C, crit = +92.0°C)
Core 3:      +56.0°C  (high = +82.0°C, crit = +92.0°C)
Core 4:      +59.0°C  (high = +82.0°C, crit = +92.0°C)
Core 5:      +56.0°C  (high = +82.0°C, crit = +92.0°C)
Core 6:      +58.0°C  (high = +82.0°C, crit = +92.0°C)
Core 8:      +56.0°C  (high = +82.0°C, crit = +92.0°C)
Core 9:      +56.0°C  (high = +82.0°C, crit = +92.0°C)
Core 10:     +55.0°C  (high = +82.0°C, crit = +92.0°C)
Core 11:     +55.0°C  (high = +82.0°C, crit = +92.0°C)
Core 12:     +52.0°C  (high = +82.0°C, crit = +92.0°C)
Core 13:     +54.0°C  (high = +82.0°C, crit = +92.0°C)
Core 14:     +54.0°C  (high = +82.0°C, crit = +92.0°C)

nvme-pci-0200
Adapter: PCI adapter
Composite:  +39.9°C  (low = -40.1°C, high = +83.8°C)
                  (crit = +87.8°C)
Sensor 1:    +59.9°C  (low = -273.1°C, high = +65261.8°C)
Sensor 2:    +37.9°C  (low = -273.1°C, high = +65261.8°C)

radeon-pci-0300
Adapter: PCI adapter
temp1:        +46.0°C  (crit = +120.0°C, hyst = +90.0°C)
```

7.14 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

7.14.1 Prática prc0001 01: Listar processos python

Nesta prática o aluno deve ter a expertise para saber localizar processos python;

1. execute o comando **cd ~**
2. Digite o comando **sudo locale-gen "en_US.UTF-8"**
3. Execute o comando **script**
4. Com o comando PS liste os processos e filtre por python;
5. Execute o comando **exit**

Valide a prática com o comando:

```
sudo aied validar prc0001 checkpoint01
```

A saída do output será:

```
usuario@debian:~$ sudo aied validar prc0001 checkpoint01
Ação que será executada: validar

Prática: Pratica da aula de processos - Procurando processos python
Será enviado o OUTPUT do comando: cat /home/usuario/typescript

Deseja continuar? (s para SIM) s
s1 - Comando: cat /home/usuario/typescript
1.1      Validar por regex      /ps s+(-)*aux  | s+grep s+python/

Total de acertos: 1 do total de 1 validações equivalente a 100%.
Identificação: ad2c5925c2
AIED v(8)
```

8 Questões de memória, Linux e C/C++

A gestão da memória (Memory Management) é um aspecto crítico da programação em C/C++, começa pela alocação de memória que é o processo de reservar um bloco de memória para uso por um programa. Em C/C++, existem dois tipos principais de alocação de memória, são estas:

Static memory allocation: é realizada em tempo de compilação. A quantidade de memória alocada é fixa e não pode ser alterada em tempo de execução, esta alocação de memória estática é usada para variáveis que são declaradas no escopo global ou no escopo local de uma função;

Dynamic memory allocation: é realizado em tempo de execução. A quantidade de memória alocada pode ser alterada em tempo de execução.

Uma questão muito importante que ao longo dos anos foi deixada de lado nos cursos de Ciência da Computação é o uso correto da memória, acredita-se hoje que a memória é infinita, mas não é. Este uso eficiente da memória não apenas melhora o desempenho de seus aplicativos, mas também ajuda a evitar problemas comuns em softwares, como por exemplo o **Memory Leak**.

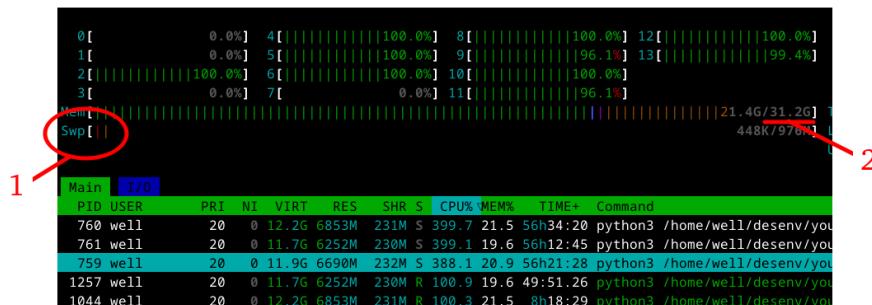
Em Sistemas Operacionais Linux e em especial softwares feitos em C/C++ temos duas áreas de memórias, uma para alocação de variáveis e outra chamadas de funções, conforme visto no começo do capítulo de Gerenciamento de Processos.

A **Dynamic Memory Allocation** permite alocar memória durante o tempo de execução usando ponteiros. O operador **new** é usado para alocação, e o operador **delete** é usado para a desalocação áreas da memória.

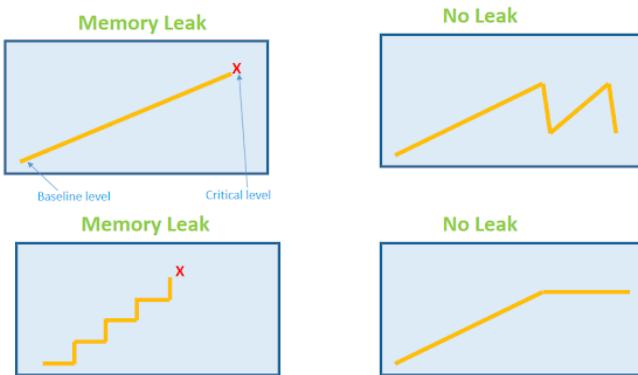
```
1. int* dynamicInt = new int;
2. *dynamicInt = 42;
3. delete dynamicInt;
```

Na linha 1 temos a alocação de uma nova memória de forma dinâmica, com o operador **new**, para adicionar um valor ou usar temos a linha 2, repare o uso do asterisco. Já na linha 3 temos o momento em que desalocamos a variável liberando espaço.

No entanto, o uso indevido de alocação de memória dinâmica pode levar a **Memory Leak**, isso é um problema principalmente quando o aprendizado de base não é bem executado, programadores porcos acabam ignorando o fato de que uma boa gestão de memória também potencializa seus sistemas, hoje o programador acredita que a memória é infinita e que o processamento é quântico, uma infeliz realidade. Na figura abaixo vemos um Memory Leak no **Open AI**, após 72 horas de processamento.

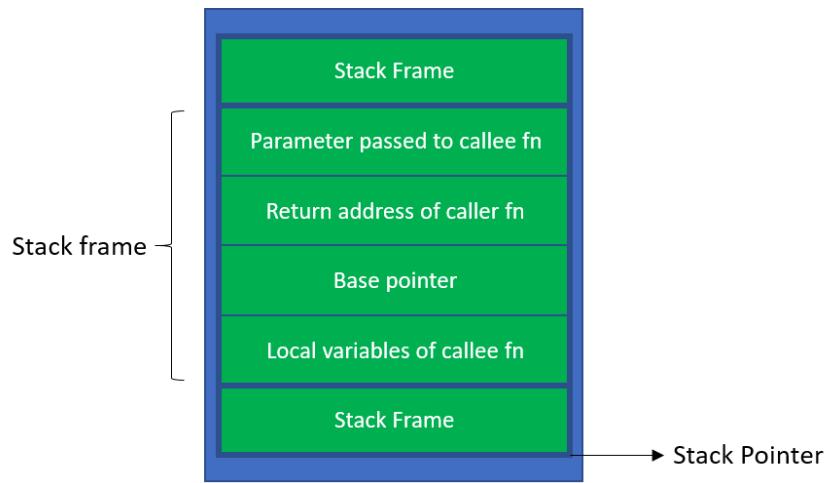


Como o Memory Leak é visto em alto nível? Simples, programas que só crescem na memória e mesmo que se feche telas, o programa só cresce, cresce, cresce até o computador entrar em Swap. Então o Memory Leak ocorre quando a memória alocada não é desalocada adequadamente, levando a uma perda gradual da memória disponível. Para evitar vazamentos de memória, certifique-se de desalocar memória usando o delete operador, conforme visto. Veja na figura abaixo comportamento de consumo de memória de programas com e sem Memory Leak.

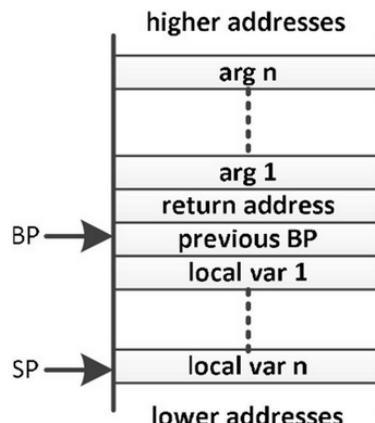


Conforme dito o processo possui duas áreas, a área Heap e a área Stack, e conforme já dito na Stack encontramos as chamadas das funções. Isso distingue as variáveis de escopo do programa das variáveis de escopo da função, quando criamos uma nova variável na função esta variável é criada dentro da área da função na stack e quando a função é removida da stack as variáveis deixam de existir. Já as variáveis criadas na memória Heap persistem enquanto não forem removidas com delete.

Quando invocamos uma função criamos na Stack uma estrutura chamada **Stack Frame**, que contém várias informações, tal como a função, parâmetros e variáveis. Tudo criado aqui possui o escopo da função que está sendo executada.



Cada chamada de cada função monta essa estrutura, que tem como início o **Stack Pointer** pois cada frame deste é um objeto na memória e é tratado como uma variável (chega a ser complicado dizer isso). Dois endereços são importantes, o endereço **BP** que marca onde se inicia a área de variáveis criadas dentro da função, por isso quando se cria uma variável dentro da função ela só é acessível quando a função estiver no topo da pilha sendo executada, e está garantido que ao término da execução da função estas variáveis serão destruídas (ver figura abaixo). O término da área de variáveis da função se dá pelo endereço de memória **SP**.



O término da área de variáveis da função se dá pelo endereço de memória **SP**. Ainda temos no **Stack Frame** outras informações importantes para a execução da função, além dos argumentos de entrada que ficam no topo da estrutura (abaixo do endereço **higher address**) e acima do endereço **BP**. Também temos algo muito importante, o endereço de memória de onde estará o retorno da função, então mesmo que temos um VOID temos um retorno que é o ponteiro. Criar uma variável na área do Stack Frame é mais rápido que criar uma variável na Heap do processo, enquanto a memória de Heap fornece maior flexibilidade na alocação de memória.

8.1 Endereços de Memória em Linux

O subsistema de gerenciamento de memória Linux é responsável, como o nome indica para gerenciar a memória no sistema, e isso inclui a implementação de memória virtual e paginação sob demanda com a área de swap, alocação de memória tanto para as

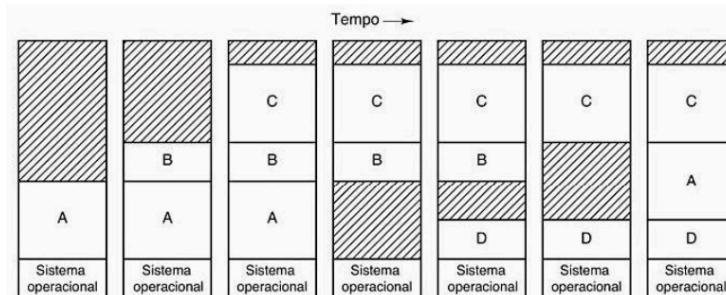
estruturas internas do Kernel como programas especiais do usuário, mapeamento de arquivos e processos.

O gerenciamento de memória Linux é um sistema complexo com muitos arquivos configuráveis, sendo a maioria desses arquivos de configuração estão disponíveis no próprio sistema de arquivos (que é virtual) no diretório `/proc`, e podem ser consultados e ajustados usando `sysctl`.

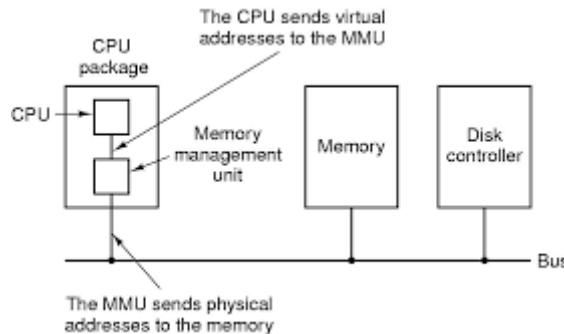
Documentar melhor `/proc/sys/vm/`

O mecanismo de gerenciamento de memória do Linux não surgiu do dia para a noite, trata-se de um longo período de desenvolvimento e adaptação e é muito complexo, é estranho dizer mas é um tipo de sistema que não se faz e já tem sucesso, ou seja, é um código resiliente baseado em sua história de falhas.

Em um cenário de acesso direto à memória por endereçamento físico, inúmeros problemas podem acontecer, imagine que o mais grave seja a fragmentação da memória, com o entra e sai de processos e com tamanho variado é natural haja **buracos sem uso** e que não são suficientes para armazenar um processo. Outro problema é que em caso de necessidade de mais espaço o processo tenha que ser totalmente copiado para outra área de memória bit a bit, isso é um processo complicado visto que a memória não processa, estaria tudo a cargo do CPU. Tudo isso faz com que lidar diretamente com a memória física seja bastante complexo, para evitar essa complexidade foi desenvolvido um conceito de memória virtual. Veja no exemplo abaixo (de Tanenbaum), com a saída do processo A e a entrada do processo D, um espaço vago ficará.



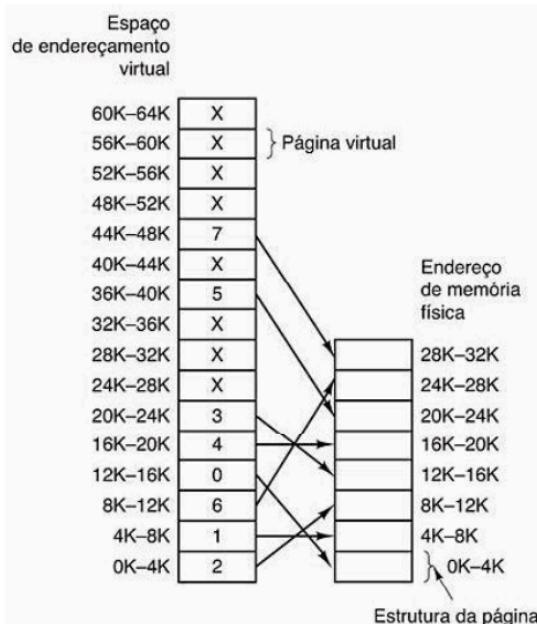
A memória virtual é uma abstração de software sobre a memória física, sendo abstrato podemos ter mais flexibilidade, mas acaba criando uma nova camada entre o CPU e a memória física (conforme figura abaixo obtida de Tanenbaum).



Além destes problemas serem resolvidos por memória virtual ainda há a abstração dos detalhes via software que permite por exemplo uma paginação sob demanda e ainda

fornecer um mecanismo mínimo de segurança de acesso a memória, como pretendo explicar neste capítulo⁵¹.

É natural que sendo virtual não necessariamente tem que ter endereços físicos e sequenciais, então uma grande diferença diz respeito ao endereçamento, que com o uso da MMU é virtual, mas a MMU tem que fazer essa tradução, por isso na imagem acima a MMU está entre o CPU e a memória física. Assim como no sistema de arquivos as memórias são divididas em pequenas porções, neste caso páginas, estas páginas podem ter tamanho fixo ou tamanho variado, vai depender do subsistema, no Linux o tamanho da página é uma seleção feita no processo de compilação do Kernel, definindo uma opção de configuração. Lembre-se que você pode re-compilar um Kernel, e essa era uma prática comum há alguns anos atrás.



Um ponto importante que deve ser destacado é que com uso de memória virtual o tamanho virtual não é correspondente ao tamanho real, podemos virtualizar uma memória muito maior.

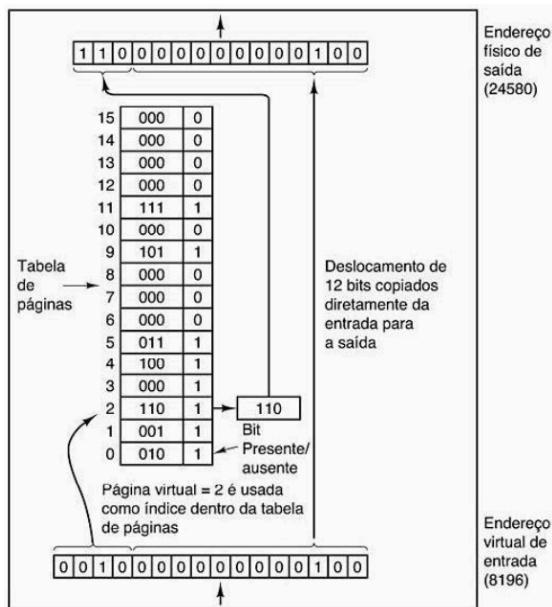
Lembre-se que segundo a teoria da computação um processo pode ser processado apenas se estiver na memória principal⁵², então esta grande memória virtual que não existe pode usar o sistema secundário de armazenamento.

Se faltar uma página então a MMU irá usar o sistema secundário de armazenamento para fazer **troca de páginas**.

Como os endereços virtuais são virtuais, cada página física na memória (possui um endereço) pode ser mapeada como uma ou mais entradas na tabela de endereços virtuais do sistema, então há a necessidade do MMU ter uma tabela de tradução e realizar este trabalho. Geralmente isso consome tempo e por isso há queda no desempenho se comparado ao acesso direto à memória física.

⁵¹ Muito se fala hoje sobre as inseguranças da linguagem C/C++ sobre acesso às memórias, mas lembre-se que há muito mais de política do que realmente problemas técnicos, sabe ver isso quem tem conhecimento, por isso, estude sempre;

⁵² Deve-se recorrer a explicação de **John von neumann** que você estudou em Arquitetura e Organização de Computadores;

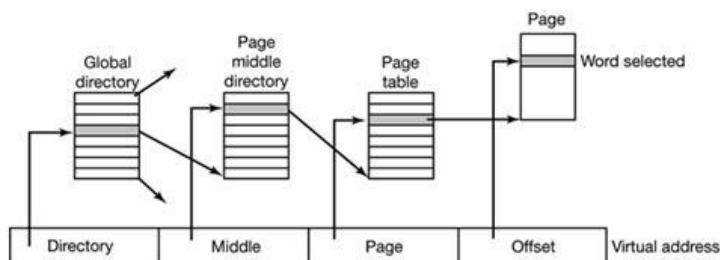


No livro do tanenbaum estas tabelas são simples, ou seja, possuem apenas um nível, conforme figura ao lado.

Veja que 4 bits para endereçamento da página é na verdade mapeado em 3 bits em uma memória real.

A segunda coluna que possui somente 1 bit indica se está presente na memória física principal (quando 1) ou se está no sistema secundário de armazenamento (quando está em 0). Esta imagem foi obtida de Tanenbaum.

No Linux não é tão simples assim, encontramos aqui níveis de tabelas em uma espécie de hierarquia, na verdade são três níveis. No primeiro nível temos Page Global Directory (PGD) referente ao processo. Cada entrada ativa na tabela Page Global Directory (PGD) aponta para um quadro de página contendo uma matriz de entradas do Page Middle Directory (PMD) do tipo **pmd_t** que por sua vez aponta para quadros de página contendo entradas de Page Table Entries (PTE) do tipo **pte_t**, que finalmente aponta para quadros de página contendo os dados reais do usuário. No caso de a página ter sido trocada para o armazenamento secundário, a entrada de troca é armazenada no PTE e usada por **do_swap_page()** durante a falha de página para encontrar a entrada de swap contendo os dados da página. O layout da tabela de páginas é ilustrado na Figura abaixo.



Qualquer endereço linear dado pode ser dividido em partes para produzir deslocamentos dentro desses três níveis de tabela de página e um deslocamento dentro da página real. Para ajudar a dividir o endereço linear em suas partes componentes, uma série de macros são fornecidas para cada nível de tabela de página, uma macro **SHIFT**, uma **SIZE** e uma **MASK**. As macros SHIFT especificam o comprimento em bits que são mapeados por cada nível das tabelas de página, conforme ilustrado na figura abaixo.

Em caso de Page Fault será executada a troca de página com swap. O algoritmo usado pelo Linux para remover páginas da memória é o do algoritmo Least Recently Used (LRU)⁵³. O LRU remove da memória a página que não foi utilizada por mais tempo. Isso baseia-se

⁵³ Recomendo obter mais detalhes em <https://linux-mm.org/LRU>

na suposição de que páginas que não foram recentemente utilizadas também não o serão nas próximas instruções, enquanto páginas bastante utilizadas agora tendem a ser bastante utilizadas nas instruções seguintes também. **Segundo Tanenbaum é uma implementação difícil de ser realizada e ter um resultado exato, talvez ainda esteja ressentido com Linus Torvalds.**

Para implementar esse algoritmo, o Linux tem uma lista duplamente encadeadas, uma lista de páginas ativas e outra com páginas inativas. As páginas ativas correspondem às páginas acessadas recentemente. A segunda lista contém as páginas não acessadas há algum tempo.

8.2 Como é tratada a memória em programas C e C++

Quando o programador cria uma variável ele simplesmente informa o tipo, e este tipo define o tamanho em bits de uso de memória, um programador moderno hoje não sabe dizer qual o tamanho de uma variável do tipo int na sua linguagem preferida. Detalhes que são omitidos pois para um programador moderno não importa, afinal a memória hoje é infinita e o processador é quântico (lógico que foi uma ironia, comum para o público brasileiro).

Na verdade a memória é uma sequência contínua de bytes, olhamos bytes e cada byte é endereçável, estes bytes são chamados de células de memória. Vou exemplificar com um int no C++ de 4 bytes, quando é feito uma operação de atribuição ou leitura é realizada a leitura de 4 bytes a partir do endereço do primeiro byte, incluindo lógico este primeiro byte. A leitura ou alteração é feita em uma única operação de 4 bytes, pois é um int signed ou unsigned. Para isso repare que é necessário armazenar o endereço do primeiro byte e saber o tamanho, além disso devem estar na sequência ambos os bytes.

Então quando uma variável é declarada, a memória necessária para armazenar seu valor é atribuída a um local específico na memória. Geralmente, os programas C++ não decidemativamente os endereços de memória exatos onde suas variáveis são armazenadas. Felizmente, essa tarefa é deixada para o ambiente onde o programa é executado geralmente, um sistema operacional que decide os locais de memória específicos em tempo de execução. No entanto, pode ser útil para um programa para ser capaz de obter o endereço de uma variável durante o tempo de execução, a fim de acessar as células de dados que estão em uma determinada posição em relação a ele.

O endereço de uma variável pode ser obtido precedendo o nome de uma variável com um sinal de ampersand (&), conhecido como endereço do operador. Por exemplo:

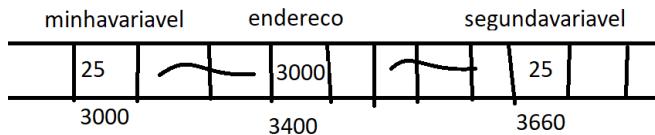
```
1. endereco = &minhavariavel;
```

O endereço real de uma variável na memória não pode ser conhecido antes do tempo de execução, mas vamos supor, para ajudar a esclarecer alguns conceitos, que minhavariavel é colocada durante o tempo de execução no endereço de memória 3000. Neste caso, considere o seguinte fragmento de código:

```
1. minhavariavel = 25;
```

```
2. endereco = &minhavariavel;
3. segundavariavel = minhavariavel ;
```

Os valores contidos em cada variável após a execução desta são mostrados no seguinte diagrama:

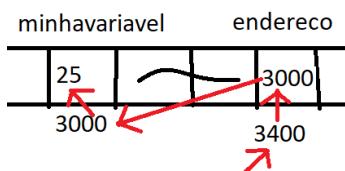


A variável que armazena o endereço de outra variável é o que em C++ é chamado de **ponteiro**. Os ponteiros são um recurso muito poderoso da linguagem que tem muitos usos na programação de nível inferior. Um pouco mais tarde, veremos como declarar e usar ponteiros.

Uma propriedade interessante dos ponteiros é que eles podem ser usados para acessar a variável que eles apontam diretamente. Isso é feito precedendo o nome do ponteiro com o operador de de-referência asterisco (*). Portanto, seguindo com os valores do exemplo anterior, a seguinte declaração:

```
1. terceiravariavel = *endereco;
```

Então a terceira variável terá o valor contido no endereço de memória que é apontado pelo endereco, ou seja, 25. Veja na imagem abaixo.



Assim, eles têm um tipo de significados opostos, onde um endereço obtido com & pode ser desreferenciado com *.

Devido à capacidade de um ponteiro para se referir diretamente ao valor que aponta para, um ponteiro tem propriedades diferentes quando aponta para um char do que quando aponta para um int ou um float. Uma vez desreferenciado, o tipo precisa ser conhecido. E para isso, a declaração de um ponteiro precisa incluir o tipo de dados para o qual o ponteiro vai apontar. Exemplos:

```
1. int* point_for_int;
2. char* point_for_char;
3. double* point_for_decimal;
```

Estas são três declarações de ponteiros. Cada um tem a intenção de apontar para um tipo de dados diferente, mas, na verdade, todos eles são ponteiros e todos eles provavelmente vão ocupar a mesma quantidade de espaço na memória⁵⁴. No entanto, os dados aos quais eles apontam não ocupam a mesma quantidade de espaço nem são do mesmo tipo.

⁵⁴ O tamanho na memória de um ponteiro depende da plataforma onde o programa é executado;

Portanto, embora essas três variáveis de exemplo sejam todas ponteiros, elas realmente têm tipos diferentes: **int***, **char*** e **double*** respectivamente, dependendo do tipo que apontam. Vamos ver um exemplo em ponteiros:

```

1. #include <iostream>
2.
3. int main ()
4. {
5.     int firstvar, secondvar;
6.     int * pointervar;
7.
8.     pointervar = &firstvar;
9.     *pointervar = 10;
10.    pointervar = &secondvar;
11.    *pointervar = 20;
12.
13.    std::cout << "First Var is      " << firstvar      << std::endl;
14.    std::cout << "Second Var is " << secondvar << std::endl;
15.    return 0;
16. }
```

Na linha 8 temos a atribuição do valor de **pointvar** com o endereço de memória de **firstvar**, para isso foi utilizado o caracter **&**, então ao atribuir ao ***pointvar** um valor estamos diretamente atribuindo em **firstvar**, e realmente modifica a área de memória representada por **firstvar**. O mesmo depois é feito por **secondvar**.

Para continuar desbravando, recomendo o livro **C++ How to Program**⁵⁵ do autor **Deitel & Deitel**.

8.3 Segmentação de áreas de memória

A proteção de memória é uma característica essencial dos sistemas operacionais modernos que garante que diferentes processos em execução em um computador não possam acessar o espaço de memória entre si sem permissão. A proteção de memória é aplicada pela MMU usando muitas técnicas.

Conforme vimos, uma tabela de páginas é uma estrutura de dados usada para traduzir endereços de memória virtual para endereços de memória física. Os endereços de memória virtual são os endereços que um programa usa para acessar a memória, enquanto os endereços de memória física são os locais reais da memória na RAM do computador. A MMU usa tabelas de página para mapear endereços de memória virtual e os traduz para endereços de memória física.

A MMU divide a memória em pequenos blocos chamados páginas que por padrão no Linux é de 4 KB. Cada página recebe um endereço de memória virtual exclusivo e a MMU usa uma tabela de páginas para mapear o endereço de memória virtual para o endereço de memória física correspondente. A tabela de páginas também contém informações sobre as permissões para cada página, como se a página pode ser lida, escrita ou executada.

⁵⁵ Recomendo que obtenha informações sobre o livro no site oficial:
<https://deitel.com/c-plus-plus-how-to-program-10-e/>

A MMU consegue isso incluindo informações sobre as permissões para cada página na tabela de páginas. Por exemplo, uma página pode ser marcada como somente leitura, o que significa que o processo pode ler dados da página, mas não pode gravar dados nela. Da mesma forma, uma página pode ser marcada como somente execução, o que significa que o processo pode executar código da página, mas não pode ler ou gravar dados nela. Isso permite que o sistema restrinja o acesso a páginas específicas com base no tipo de acesso, o que é importante para manter a segurança do sistema.

Quando um programa acessa a memória, a MMU primeiro verifica a tabela de páginas para determinar as permissões para a página que o programa está tentando acessar. Se o programa não tiver as permissões apropriadas, a MMU gera uma falha de página, o que faz com que o sistema operacional lide com o erro. O sistema operacional pode então tomar as medidas apropriadas, como encerrar o programa ou conceder ao programa as permissões necessárias.

A MMU consegue isso atribuindo uma tabela de página exclusiva a cada processo, o que permite que cada processo tenha seu próprio espaço de memória virtual. Isto significa que um processo só pode aceder às páginas atribuídas à sua tabela de páginas e não pode aceder às páginas atribuídas a outras tabelas de páginas do processo. Isso garante que um processo não possa acessar ou interferir no espaço de memória de outros processos, o que é importante para manter a estabilidade e a segurança do sistema.

A MMU usa diferentes tipos de tabelas de página para reforçar a proteção de memória. O tipo mais simples de tabela de página é a tabela de página linear, que é uma matriz unidimensional que mapeia endereços de memória virtual para endereços de memória física. Cada entrada na tabela de página linear contém o endereço de memória física e as permissões para o endereço de memória virtual correspondente.

A MMU também usa tabelas de página hierárquicas, que são mais complexas do que as tabelas de página lineares. As tabelas de página hierárquicas são matrizes multidimensionais que são usadas para dividir a memória em blocos menores chamados quadros de página. Cada quadro de página recebe um endereço de memória virtual exclusivo e a MMU usa a tabela de páginas para mapear o endereço de memória virtual para o endereço de memória física correspondente. A tabela de páginas também contém informações sobre as permissões para cada quadro de página.

A MMU usa um buffer Translation lookaside buffer (TLB) para acelerar o processo de tradução de endereços de memória virtual para endereços de memória física. O TLB é um cache que armazena entradas de tabela de página usadas recentemente e é usado para reduzir o número de acessos de memória necessários para traduzir endereços de memória virtual para endereços de memória física.

A proteção no nível do Kernel é uma técnica usada pela Unidade de Gerenciamento de Memória (MMU) para proteger os processos no núcleo e no sistema dos processos no nível do usuário. O Kernel é a parte central do sistema operacional que gerencia os recursos do computador, como memória, CPU e dispositivos de entrada/saída. Os processos no núcleo e no sistema têm acesso privilegiado aos recursos do sistema, o que significa que eles

podem executar ações que não são permitidas para processos no nível do usuário. Para garantir que os processos ao nível do utilizador não podem aceder ao espaço de memória processo ao nível do kernel ou do sistema, a MMU utiliza proteção ao nível do kernel.

A proteção no nível do Kernel envolve a criação de uma tabela de página separada para os processos no núcleo e no sistema. Esta tabela de páginas é chamada de tabela de páginas do kernel e é usada para mapear o espaço de memória virtual do kernel para a memória física. A tabela de páginas do kernel tem permissões diferentes das tabelas de páginas no nível do usuário e só é acessível pelos processos no núcleo e no nível do sistema. Isso significa que os processos no nível do usuário não podem acessar o espaço de memória do kernel, o que os impede de interferir nos processos no núcleo ou no nível do sistema.

A MMU usa um bit específico nas entradas da tabela de páginas para identificar as páginas que pertencem à tabela de páginas do kernel. Esse bit é chamado de bit supervisor e é definido como 1 para páginas na tabela de páginas do kernel e 0 para páginas nas tabelas de páginas no nível do usuário. A MMU usa esse bit para determinar se um processo tem os privilégios para acessar uma página específica e, se não, gera uma falha de página.

Quando um processo de nível de usuário tenta acessar uma página na tabela de páginas do kernel, a MMU verifica o bit supervisor da entrada da tabela de páginas e, se estiver definida como 1, gera uma falha de página. A falha da página é então tratada pelo sistema operacional, que pode tomar as medidas apropriadas, como encerrar o processo ou negar o acesso à página.

A proteção em nível de kernel é uma técnica importante para manter a segurança do sistema. Ao impedir que os processos ao nível do utilizador acedam ao espaço de memória processos' ao nível do kernel ou do sistema, a proteção ao nível do kernel garante que os processos ao nível do utilizador não podem interferir com os processos ao nível do kernel ou do sistema, o que é importante para manter a estabilidade e segurança do sistema.

Vale ressaltar que a proteção no nível do kernel não é uma técnica infalível e pode ser ignorada por invasores sofisticados. No entanto, torna o processo de exploração mais difícil, demorado e menos previsível.

8.4 Address Space Layout Randomization (ASLR) e bit NX

Address Space Layout Randomization (ASLR) é uma técnica usada pela MMU para proteger o sistema do computador contra ataques baseados em memória. O ASLR é usado para randomizar a localização de dados no sistema e no nível do usuário na memória, o que torna mais difícil para um invasor prever a localização de dados específicos. Isso torna mais difícil para um invasor explorar vulnerabilidades baseadas em memória, como estouros de buffer ou ataques baseados em heap.

O ASLR funciona aleatorizando o endereço base do espaço de memória virtual do processo. O endereço base é o ponto de partida do espaço de memória virtual do processo e é utilizado como ponto de referência para todos os endereços de memória virtual do mesmo. Ao randomizar o endereço base, o ASLR torna difícil para um invasor prever a

localização de dados específicos na memória, como a localização da pilha ou do Heap, tendo então que vasculhar byte a byte a memória realizando exaustivos testes de tipo de dados.

O ASLR também randomiza a localização de outros dados do sistema, como bibliotecas compartilhadas e estruturas de dados do Kernel. Ao randomizar a localização dessas estruturas de dados, o ASLR torna mais difícil para um invasor explorar vulnerabilidades baseadas em memória que dependem do invasor saber a localização de estruturas de dados específicas.

Além de randomizar a localização dos dados na memória, o ASLR também usa outras técnicas para proteger o sistema contra ataques baseados em memória. Por exemplo, o ASLR pode randomizar a localização da Stack e do Heap dentro de um espaço de memória virtual process', o que torna mais difícil para um invasor explorar vulnerabilidades baseadas em pilha ou em heap. O ASLR também pode randomizar o layout da tabela de páginas, o que torna mais difícil para um invasor explorar vulnerabilidades baseadas em tabelas de páginas.

Vale a pena mencionar que o ASLR não é uma técnica infalível, e pode ser contornado por atacantes sofisticados. No entanto, torna o processo de exploração mais difícil, demorado e menos previsível.

Existem dois mecanismos principais para proteger o acesso à memória que são ativados por padrão na maioria dos sistemas x86-64 Linux. O primeiro é o chamado NX bit que é um cenário que dá permissões mais detalhadas para regiões de memória mapeadas. O segundo é randomização de layout de espaço de endereço (ASLR) que randomiza onde certas partes de um programa são carregadas na memória. Eu vou discutir esses dois tópicos separadamente, pois eles são complementares, mas completamente ortogonais um para o outro.

Arquiteturas de computadores tradicionais que implementaram mecanismos de proteção de memória normalmente possuíam dois estados em que uma região de memória mapeada poderia estar:

- Somente leitura;
- Gravável.

Os detalhes exatos dos recursos de proteção de memória dependem da Arquitetura de CPU, mas o que descrevi se aplica à maioria dos sistemas x86 de 32 bits. Além dessa alternância de leitura/gravação, o sistema também implementaria algumas outras proteções básicas de memória, como garantir que diferentes processos não possam ler ou escrever para as regiões de memória uns dos outros.

A forma como este sistema é implementado é que cada processo tem dados esparsos estrutura alocada no Kernel chamada tabela de páginas. A tabela de páginas contém um mapeamento de regiões de memória virtual para a memória física associada. Para cada página na tabela de páginas há também um bit de leitura/gravação que é usado pela MMU para impor permissões para áreas que devem ser somente leitura. Proteção da memória entre os processos ocorre através do fato de que o kernel irá organizar a MMU para usar

uma tabela de página de um determinado processo quando esse processo está programado para ser executado, o que significa que, na ausência de bugs do kernel, um processo não pode entrar em um estado onde está usando a tabela de página de outro processo.

Existem alguns casos de uso diferentes para regiões de memória somente leitura, mas provavelmente o caso de uso mais importante são os segmentos de texto somente leitura. Um segmento de texto é a parte da memória de um processo que contém o código de máquina real para o processo. A razão pela qual é desejável mapear essa área somente leitura é que ela ajuda a atenuar ataques que tentam injetar código malicioso em um processo. Se um invasor pode injetar código malicioso na área de código de um processo executar código arbitrário com as permissões do próprio processo. Em muitos casos essas permissões são consideráveis, mesmo que o processo não esteja sendo executado como root. Por exemplo, um invasor que pode injetar código pode executar um sistema de arquivos arbitrário de operações como o UID efetivo do processo. Quando a área de texto é somente leitura pode ter certeza de que o código na área de texto que é executado não foi adulterado.

Uma grande limitação aqui é que, embora possamos pensar na memória como pertencente para código somente leitura e dados gravados, na verdade algumas informações sobre o que é código a execução é armazenada na área de dados. Em particular, a convenção convocatória sobre x86 funciona armazenando o endereço de retorno para uma chamada de função na pilha, e a pilha deve ser mapeada com permissões de gravação. Assim, um atacante que pode pisar na memória na pilha pode alterar o endereço de retorno para uma função. Isto é um problema muito conhecido e conhecido como um estouro do buffer de pilha. Normalmente, quando as pessoas falam sobre "buffer overflows", elas são especificamente referindo-se a estouros de buffer de pilha, uma vez que os estouros de pilha são geralmente os mais perigosos de estouro de buffer (embora não seja o único tipo).

O bit NX, que está presente em todos os sistemas x86 de 64 bits, é uma medida importante que ajuda a mitigar este ataque. Sistemas que implementam esta funcionalidade têm três permissões bits: **ler**, **escrever** e **executar**. Um atacante ainda pode organizar um endereço de retorno arbitrário para ser usado como o endereço de retorno da função, mas a menos que esse endereço de retorno realmente aponte para um deslocamento válido na área de texto o processo será segfault (ou seja, terminará com SIGSEGV). O processo também pode terminar com SIGILL se um deslocamento inválido em a área de texto é usada, já que qualquer deslocamento que não esteja alinhado em um real o deslocamento de instruções provavelmente se tornará uma sequência inválida rapidamente.

O ataque mais perigoso que ainda é possível com estouros de buffer de pilha quando o bit NX está ativado é chamado de ataque de retorno à libc. A maneira como isso funciona é que, se as condições estiverem corretas, um invasor pode organizar para que o endereço de retorno de uma função seja o endereço de uma sequência de instruções na libc que farão com que o programa faça algo perigoso (por exemplo, executar um shell). A razão pela qual esse ataque é perigoso é porque a libc tem um ajudante rotinas para quase todas as chamadas do sistema, além de inúmeras outras funções de alto nível isso pode ser bastante poderoso. A razão pela qual este ataque é muito mais difícil de executar do que um buffer overflow de pilha típico é que não só o atacante deve saiba exatamente para qual endereço na libc retornar, o atacante também deve organizar para que os registros e a

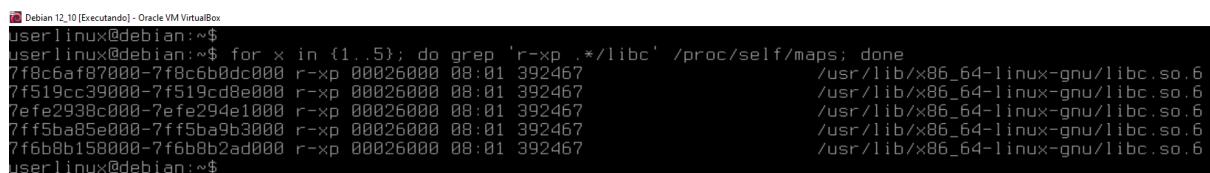
sequência de instruções de retorno estejam corretos, então a libc o código está no estado correto para realmente executar as instruções maliciosas.

ASLR é um mecanismo que é tecnicamente complementar ao bit NX, mas é feito muito mais poderoso quando o bit NX está ativado. A razão é que o ASLR é projetado precisamente para fazer o retorno à libc muito mais difícil de executar.

A maneira como o ASLR funciona é que, quando bibliotecas compartilhadas são mapeadas em um processo de memória, elas são carregadas em locais aleatórios. Por exemplo:

```
1. for x in {1..5}; do grep 'r-xp .*/libc' /proc/self/maps; done
```

Veja a saída abaixo em uma máquina virtual, lembrando que os dados diferem.



```
Debian 12.10 [Executando] - Oracle VM VirtualBox
userlinux@debian:~$ for x in {1..5}; do grep 'r-xp .*/libc' /proc/self/maps; done
7fb86af87000-7f8c6b0dc000 r-xp 00026000 08:01 392467
7f519cc39000-7f519cd8e000 r-xp 00026000 08:01 392467
7e7e2938c000-7efe294e1000 r-xp 00026000 08:01 392467
7ff5ba85e000-7ff5ba9b3000 r-xp 00026000 08:01 392467
7f6b8b158000-7f6b8b2ad000 r-xp 00026000 08:01 392467
userlinux@debian:~$ _
```

A sequência hexadecimal mais à esquerda nesta saída mostra o deslocamento em que a libc é carregada para cinco instâncias bash aleatórias diferentes. Como você pode ver, em cada invocação libc é carregado em um deslocamento de memória diferente. Isso significa que o endereço de um dado o método em libc será diferente em cada invocação de processo. Se um atacante quiser executar um método como **unlink** eles não podem facilmente saber exatamente para onde o código **unlink** será realmente carregado na memória.

Outra maneira de tornar isso mais aparente é escrever um programa⁵⁶ C que imprime o endereço de uma função libc cada vez que ela é executada. Considere o seguinte listagem de programas:

```
1. #include <dlfcn.h>
2. #include <stdio.h>
3.
4. const char libc_name[] = "libc.so.6";
5.
6. int main(int argc, char **argv) {
7.     void *handle = dlopen(libc_name, RTLD_LAZY | RTLD_NOLOAD);
8.     if (handle == NULL) {
9.         printf("libc not found\n");
10.        return 1;
11.    }
12.    printf("unlink is load at %p\n", dlsym(handle, "unlink"));
13.    return 0;
14. }
```

⁵⁶ Baseado no exemplo: <https://www.ibm.com/docs/pt-br/aix/7.3?topic=d-dlopen-subroutine>

A rotina **dlopen** é uma forma portátil de carregar dinamicamente bibliotecas compartilhadas. Ele realiza a inicialização estática C/C++ dos módulos que ele carrega. O valor retornado pelo **dlopen** pode ser usado em chamadas subsequentes para **dlsym** e **dclose**. Se ocorrer um erro durante a operação, **dlopen** retorna **NULL**. Para compilar use o comando abaixo:

```
1. gcc -o lslibc lslibc.c
```

Em uma sequência de várias invocações diferentes observa-se endereços diferentes para a biblioteca compartilhada:

```
userlinux@debian:~$ for x in {1..5}; do ./lslibc ; done
unlink is load at 0x7ff9c1ab8ac0
unlink is load at 0x7f4600b76ac0
unlink is load at 0x7f2a4e402ac0
unlink is load at 0x7f9354a42ac0
unlink is load at 0x7ff53aa1dac0
userlinux@debian:~$
```

Mecanismos podem ser ignorados, tanto bit NX quanto ASLR e há inúmeras maneiras de se fazer isso e são essas maneiras que são exploradas pelo hacker. O bit NX pode ser ignorado por um chamador usando a chamada do sistema **mprotect**. A primitiva **mprotect()**⁵⁷ altera as proteções de acesso para a chamada de páginas da memória do processo, e pode conter qualquer parte do intervalo de endereços em [**addr** , **addr** + **tamanho** -1]. Se o processo de chamada tentar acessar a memória de uma maneira que viola as proteções, então o kernel gera um **SIGSEGV** sinal para o processo.

No exemplo abaixo⁵⁸, será forçado o acesso em uma área da memória fora do processo e levará a execução de um sinal de **SIGSEGV**. Crie um arquivo chamado **useprotect.c**, e com o nano edite o seguinte código.

```
1. #include <malloc.h>
2. #include <signal.h>
3. #include <stdio.h>
4. #include <stdlib.h>
5. #include <sys/mman.h>
6. #include <unistd.h>
7.
8. #define handle_error(msg) \
9.         do { perror(msg); exit(EXIT_FAILURE); } while (0)
10.
11. static char *buffer;
12.
13. static void handler(int sig, siginfo_t *si, void *unused) {
14.     printf("Got SIGSEGV at address: %p\n", si->si_addr);
15.     exit(1);
16. }
17.
18. int main(int argc, char **argv) {
19.     int pagesize;
```

⁵⁷ Documentação acessível pela URL: <https://man7.org/linux/man-pages/man2/mprotect.2.html>

⁵⁸ Exemplo baseado no exemplo oficial: <https://man7.org/linux/man-pages/man2/sigaction.2.html>

```

20.     struct sigaction  sa;
21.
22.     sa.sa_flags = SA_SIGINFO;
23.     sigemptyset(&sa.sa_mask);
24.     sa.sa_sigaction = handler;
25.     if (sigaction(SIGSEGV, &sa, NULL) == -1)
26.         handle_error("sigaction");
27.
28.     pagesize = sysconf(_SC_PAGE_SIZE);
29.     if (pagesize == -1)
30.         handle_error("sysconf");
31.
32.     buffer = memalign(pagesize, 4 * pagesize);
33.     if (buffer == NULL)
34.         handle_error("memalign");
35.
36.     printf("Start of region:      %p\n", buffer);
37.
38.     if (mprotect(buffer + pagesize * 2, pagesize, PROT_READ) == -1)
39.         handle_error("mprotect");
40.
41.     for (char *p = buffer ; ; )
42.         *(p++) = 'a';
43.
44.     printf("Loop completed\n");
45.     exit( 0 );
46. }
```

A definição de **handle_error(msg)** é feito para que se possa capturar todos os reports de erros lançados pelo programa, um exemplo de uso é feito na linha 34 e 39. A função estática **handler(int, siginfo_t, void)** é uma função que recebe de forma assíncrona sinais enviados por outros processos, a questão de sinal POSIX será tratado mais adiante. Basicamente vamos forçar o processo ao erro de acesso de memória, e vamos capturar o evento/sinal SIGSEGV. Para compilar é fácil, com o gcc edite o comando.

```
1. gcc -o useprotect useprotect.c
```

Quando executar o laço infinito da linha 41 irá forçar a atribuição de 1 CHAR no endereço de memória por uso de ponteiro (linha 42), se passar da área protegida então uma execução do **handler()** será executada e o **exit()** da linha 15 irá parar o processo. Veja na imagem abaixo o resultado final.

```

userlinux@debian:~$ ./useprotect
Start of region:      0x55d767575000
Got SIGSEGV at address: 0x55d767577000
userlinux@debian:~$
```

Uma coisa divertida que um chamador pode fazer é definir as regiões de texto para um processo seja gravável. Se isso for feito, ele ignora completamente o estado somente leitura que as áreas de código são geralmente mapeadas em. O chamador também pode mapear regiões de dados (como a pilha) para ser executável, que também efetivamente ignora o habitual mecanismo de proteção. Para realmente fazer qualquer uma dessas coisas, um

chamador deve saber o endereço exato para o qual eles desejam alterar as permissões. Fazer isso é um pouco dificultado pelo fato de que muitas partes da memória vivem ao acaso compensações devido a ASLR.

Uma vez que o ASLR é o principal mecanismo de proteção que impede um ataque de remapeamento trivial de regiões de memória com `mprotect(2)` o resto desta seção discutirá maneiras que ASLR pode ser contornado.

Como demonstrei brevemente anteriormente, um processo pode ver seu layout de memória exato por olhando para `/proc/self/maps`. Este arquivo é a maneira mais fácil de contornar o ASLR proteções. Uma vez que o arquivo tem um nome bem conhecido e é facilmente analisável, o acesso ao formato facilita saber exatamente onde tudo está carregado e quais permissões são impostas em cada região de memória. Em um ambiente "cooperativo" onde você está lendo intencionalmente seu próprio arquivo de mapas, basta um par de linhas de código C ou C++ para encontrar coisas e remapear.

Na prática, seria difícil para um invasor usar esse arquivo porque ele seria necessário realmente injetar código para abrir/parse/interpretar o arquivo. Normalmente os próprios mecanismos NX + ASLR devem ser suficientes para proteger contra isto. Em alguns casos, um invasor pode usar outro processo em execução como o mesmo UID para analisar o arquivo de mapas de um determinado processo, mas mesmo isso parece distante buscado porque o invasor ainda precisa de outro canal lateral para obter os dados do processo. Há também várias maneiras de isolar os diretórios de processo em `/proc` para proteger contra este tipo de ataque **Side Channel**.

Processar o `maps` arquivo em `/proc` não é a única maneira de contornar o ASLR. Os processos também podem localizar funções na memória olhando para o símbolo da função tabelas. O arquivo C de demonstração que listei mostra como fazer isso com `dlopen(3)` e `dlsym(3)` o que torna as coisas mais fáceis, desde então `libdl` implementa o baixo real bits de nível que sabem como fazer essa análise de símbolo de função. No entanto, vale a pena observando isso `libdl` não está fazendo nada mágico aqui, e no uso que eu mostrei não olha para `/proc/self/maps`.

O caminho `libdl` funciona com `glibc` é que ele contém o código que sabe como analisar um arquivo ELF real. Quando você liga `dlopen(3)` e `dlsym(3)` da maneira Eu demonstrei, `libdl` localizará `libc-2.19.so` pesquisando na biblioteca caminho de pesquisa e, em seguida, ele irá analisar a tabela de símbolos para ver o que compensar o `unlink(2)` o método está localizado em. Ele pode então usar isso para encontrar o método na tabela de símbolos do processo. Portanto, é possível, em princípio, que um atacante use esse tipo de mecanismo mesmo se um executável não estiver vinculado a `libdl`. Novamente, em um cenário malicioso real, é improvável que um invasor possa na verdade, faça isso desde a lógica para analisar um arquivo ELF e o símbolo a tabela é considerável e teria que ser injetada em um processo.

Há outro mecanismo que não é bem compreendido por muitas pessoas; este é também o que eu acho que seria a maneira mais provável que um atacante tentaria ignorar o ASLR em cenários reais. Para executáveis típicos, apenas bibliotecas compartilhadas tenha suas localizações randomizadas. A área de texto para o próprio executável será não use locais

de memória aleatórios. Portanto, se um atacante tiver uma cópia de um executável, eles podem analisar o código do executável e determinar a eficácia lugares para voltar. Por exemplo, suponha que um invasor queira desvincular um arquivo. Se o próprio executável tiver código que chama `unlink(2)` em vez de tentar para realmente voltar a `unlink(2)`, o invasor pode retornar ao código isso chama `unlink(2)`, que existirá em um local bem conhecido. Para muitos isso ainda é difícil, pois o invasor precisa obter registros e/ou memória no estado certo para passar os argumentos certos para a função de destino. Ainda assim, esta é uma maneira eficaz de contornar o elemento de randomização que o ASLR introduz. Em alguns casos, pode-se também usar esse mecanismo, mesmo que o executável não chama a função C pretendida: o invasor precisa apenas para encontrar uma sequência de bytes executáveis que podem ser interpretados como o chamada de função desejada. Assim, pode-se intencionalmente saltar para um deslocamento de bytes inválido pode causar um acidente com SIGILL se alguém sabe disso antes do ilegal as instruções são atingidas, haverá uma breve execução de instruções que pode ser analisada o caminho desejado.

Se você está preocupado com este último ataque, você pode compilar com o `-PIE` bandeira, que significa posição executável independente. Isso causa toda uma símbolos do executável a serem carregados em locais aleatórios usando ASLR. O executável ainda terá um ponto de entrada fixo e bem conhecido, mas esse ponto de entrada apenas providenciará para o tempo de execução C (a.k.a. CRT) funções de inicialização para carregar e a partir daí tudo terá um endereço de memória imprevisível. Compilando com `-PIE` tem sobrecarga não negligenciável, particularmente em 32 bits sistemas, mas é indispensável para executáveis sensíveis à segurança, como ssh.

Uma última maneira de contornar as proteções de memória, que mencionarei apenas para completude, é usando o `ptrace(2)` chamada do sistema. Esta chamada do sistema permite a processo remoto ler e gravar a memória de um processo remoto sem restrição (ou seja. não é necessário `mprotect(2)` qualquer coisa). Esta chamada de sistema é o que GDB usos para sondar processos remotos. O `ptrace(2)` a chamada do sistema é muito poderosa mas porque é tão poderoso seu uso é tipicamente bastante restrito. Em um no mínimo, você precisa ter o mesmo UID efetivo que o processo remoto em ordem para rastreá-lo. Além disso, há um arquivo chamado `/proc/sys/kernel/yama/ptrace_scope` que pode ser usado para neutralizar o `ptrace(2)` chamada do sistema. Por padrão Debian e O Ubuntu envia kernels que têm o escopo do Yama `ptrace` configurado para que apenas o superusuário pode rastrear outros processos (e somente o superusuário pode alterar isso configuração). Em geral, se você pode `ptrace` outro processo que você tem ilimitado capacidade de fazer qualquer coisa como esse processo, enquanto o `ptrace` pode ser usado para ignorar proteções de memória, qualquer situação em que você pode rastrear outro processo é um situação em que você já é capaz de comprometer o processo.

9 Configurando interface de Redes em Linux

Uma interface de rede ou NIC é um dispositivo de I/O a responsável por enviar e receber dados da rede, e um Sistema Operacional pode possuir várias interfaces de rede. Trata-se de uma importante fonte de dados para processamento e computadores, para se ter uma idéia a capacidade de processamento de computadores evoluiu 16 vezes por década e a capacidade de transmissão de dados das LANs evoluíram 15.8 vezes por década, mostrando o quanto é importante um dispositivo de rede para um computador moderno.



O principal comando utilizado nesta sessão é o comando **ip**, naturalmente não é o único e neste tópico este comando será amplamente utilizado, a distribuição Debian utilizada neste material já possui este comando e não é necessário a instalação.

9.1 Faixas de IPs reservados para LAN

Quando se configura um endereço IP em uma rede não se pode utilizar a qualquer faixa de endereços segundo a IANA⁵⁹ e a IEEE o administrador de rede deve utilizar as faixas adequadas destinadas para LAN e os fabricantes de equipamentos já trabalham utilizando estas regras em seus equipamentos.

Além das faixas de IPs reservadas para as LANs, o administrador de rede deve estar atento às classes de IPs, para LANs pequenas utiliza-se a Classe C, mas, para grandes redes utiliza-se a Classe A.

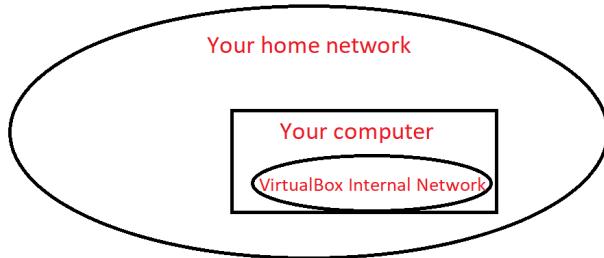
Nome	Faixa de endereços IP	Número de IPs	<i>classful</i> Descrição
8-bit block	10.0.0.0 – 10.255.255.255	16,777,216	Uma classe A
12-bit block	172.16.0.0 – 172.31.255.255	1,048,576	16 classes B
16-bit block	192.168.0.0 – 192.168.255.255	65,536	256 classes C

Então em uma LAN em uma empresa ou em uma casa deve-se usar os endereços acima. Nada impede de se alterar em uma necessidade futura, por exemplo, se está usando um endereço classe C você poderá ter no máximo 254 equipamentos na rede, se um dia alcançar este limite poderá mudar naturalmente para um endereço classe B que poderá ter 65534 endereços para equipamentos. Mas se chegar neste ponto acredito que terá implantado em sua infra estrutura um serviço DHCP, algo que é abordado neste livro. Neste caso seu trabalho será realizado com menos esforço.

⁵⁹ Acessível pela url <https://www.iana.org/>

9.1.1 Definindo uma configuração para a LAN

Qualquer faixa de IP definida para LAN fora do intervalo da figura acima é um erro, pois está definido para WAN ou reservada para outros fins que não a LAN. Os endereços classe C são os clássicos nas LANs, principalmente em ambiente Home, já as classes B e A são mais clássicas em ambientes Office, pois admite-se que em sua casa a pessoa não possui mais de 254 equipamentos em rede ao mesmo tempo.



Um endereço de rede muito usado em ambiente home é o endereço 192.168.0.0/24, ou seja, rede 192.168.0.0 com máscara de rede 255.255.255.0. No Windows é possível ver este endereço com o comando **ipconfig**, já no GNU/Linux com o comando **ip address**. Esta informação é muito importante, pois em todas as práticas em caso de se utilizar uma rede Interna (no VirtualBox) em Adaptadores, vamos definir como rede 192.168.200.0/24, se por coincidência sua rede real da sua casa for 192.168.200.0/24 então deve trocar para 192.168.201.0/24.

Repare que está sendo definida a rede informando o endereço 192.168.200.0, isso ocorre pois sempre o primeiro endereço é utilizado para definir a Rede ou a Sub-rede, por isso o endereço 192.168.200.0 não é usado em interfaces NIC em Hosts ou Routers. O aluno deve compreender que o último endereço 192.168.200.255 é utilizado pelo Broadcast da rede, e também não pode ser utilizado em interfaces NIC.

9.2 Definição dos nomes das Interfaces de Rede

Tanenbaum no capítulo de dispositivos I/O explica uma tal nomeação uniforme dos dispositivos, e já foi dito neste livro que há apenas dois tipos de dispositivos, que são de carácter e de bloco. Se pegar todos os dispositivos de bloco que estão em SATA o Sistema Operacional não conseguiria identificar unicamente cada um, na verdade ele identifica baseado em sua posição no barramento, ou seja, qual slot físico foi ligado o dispositivo ou pela sequência de descoberta.

Então o nome é sempre semelhante baseado no que o dispositivo está ligado, para SSD em SATA usa-se **sda**, **sdb**, **sdc**, e assim por diante. No caso de interfaces de rede alguns linux fazem assim: **eth0**, **eth1**, **eth2** e assim por diante. Com a introdução do Systemd no Debian GNU/Linux este esquema foi alterado e a regra foi definida como **NamePolicy** está descrita no arquivo **/lib/systemd/network/99-default.link**⁶⁰.

No caso de placa de rede⁶¹ com RJ45 Ethernet 802.3 o nome começa com **en** já para interfaces de rede Wi-Fi 802.11 o nome começa com **wl**. Segue-se em caso de placa de

⁶⁰ Documentação acessível pela URL: <https://man7.org/linux/man-pages/man5/systemd.link.5.html>

⁶¹ A nomenclatura completa para placa de redes está descrita na URL:

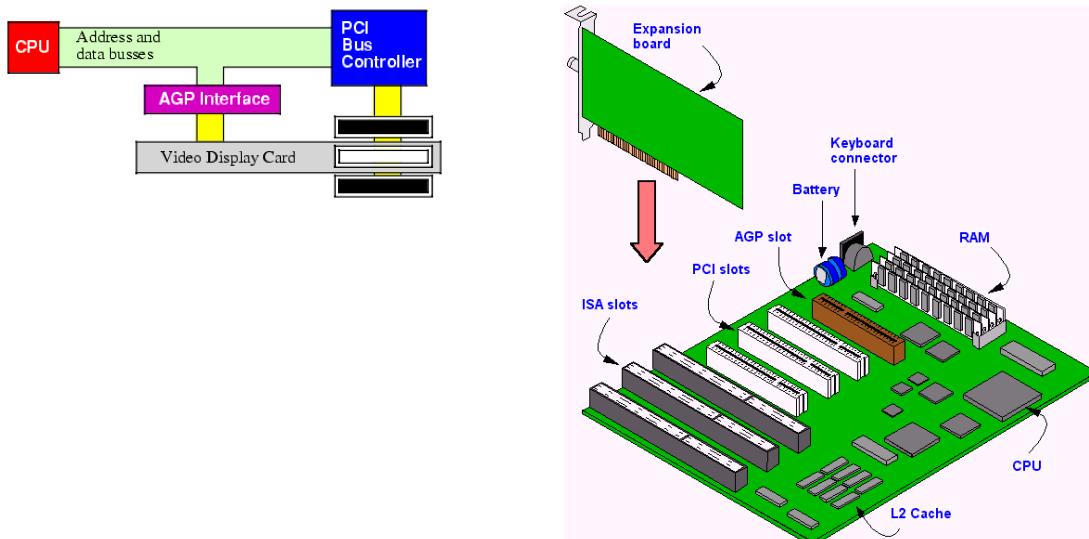
<https://man7.org/linux/man-pages/man7/systemd.net-naming-scheme.7.html>

rede 802.3 a letra **p** para PCI. Depois tanto para 802.3 e 802.11 adiciona-se o número do barramento (bus number), vamos imaginar que para nosso exemplo seja o barramento **0**. Segue-se depois o número do slot e se o device tiver mais de uma placa, o número da função (da porta no hardware). Vamos exemplificar, olhe a imagem abaixo.

```
00:01.1 IDE interface: intel
00:02.0 VGA compatible contr
00:03.0 Ethernet controller:
00:04.0 System peripheral: ]
```



Trata-se de uma interface 802.3 em uma PC, o barramento é **00** seguido do slot **03** e o device só possui uma porta (porta **0**), ou seja, uma posição para se pôr o cabo (ver figura acima). Então o nome da placa é: **enp0s3**. Nas figuras abaixo temos detalhes de uma arquitetura e o esquema de uma placa mãe, a placa Expansion Board pode ser usada no nosso exemplo de placa de rede.



Em servidores reais é comum ter uma placa de rede com inúmeras portas, conforme figura abaixo, e esta porta é numerada com **f** e seguido do número da porta. Vamos imaginar que a placa abaixo esteja no barramento 0 e no slot 3. Teremos as placas **enp0s3f0**, **enp0s3f1**, **enp0s3f2** e **enp0s3f3**. **Eu odeio isso!!!!**



9.2.1 Exibindo dados de interfaces de rede PCI com lspci

Antes de se configurar a interface de rede no GNU/Linux, deve-se ter certeza que as interfaces foram carregadas no Kernel, para isso utiliza-se o comando **lspci** conforme figura abaixo.

```

aluno@aluno-desktop:~$ lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:02.0 VGA compatible controller: InnoTek Systemberatung GmbH VirtualBox Graphics Adapter
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)
00:04.0 System peripheral: InnoTek Systemberatung GmbH VirtualBox Guest Service
00:05.0 Multimedia audio controller: Intel Corporation 82801AA AC'97 Audio Controller (rev 01)
00:06.0 USB Controller: Apple Computer Inc. KeyLargo/Intrepid USB
00:07.0 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:0d.0 SATA controller: Intel Corporation 82801HBM/HEM (ICH8M/ICH8M-E) SATA AHCI Controller (rev 02)
aluno@aluno-desktop:~$ 

```

No exemplo acima existe somente 1 interface de rede Ethernet, caso tenha muitos dispositivos PCI então é possível filtrar usando Pipe e Grep como exemplo abaixo.

```
1. lspci | grep Ethernet
```

9.2.2 Exibindo interfaces de rede USB com lsusb

Existe a possibilidade de uso de uma interface de rede Plug and Play por USB, neste caso o comando lspci não irá localizar a interface, ela é uma opção comum em:

- Notebooks modernos que não possuem a interface Ethernet;
- Em uma TI para solucionar um problema urgente de falha de interface PCI;

Este elemento é semelhante a imagem abaixo.



O comando é muito simples, conforme listagem abaixo.

```
1. sudo lsusb
```

Caso não esteja instalado, instala-se esse comando com o comando apt, conforme listagem abaixo.

```
1. sudo apt update -y
2. sudo apt install usbus -y
```

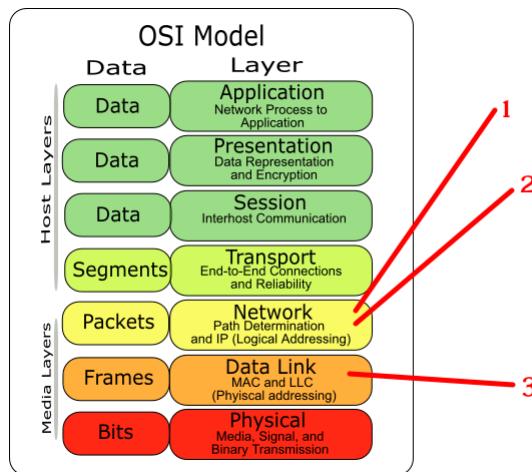
9.3 Comando ip

Uma forma de realizar a configuração de uma interface de rede é alterando um arquivo no sistema, mas é considerado por muitos administradores modernos como uma forma arcaica, pois, quando se altera o layout dos arquivos a documentação deve ser refeita. Comandos é mais fácil de se documentar, pois estamos falando de requisitos funcionais que pouco mudam, mas o requisito não-funcional que escreve, lê e comprehende os dados pode ser alterado e naturalmente o Sistema Operacional tem mais liberdade de evolução interna.

O comando `ip` é usado para executar várias tarefas de administração de rede. Este comando é usado para mostrar ou manipular roteamento, dispositivos e túneis. O comando `ip` é usado para executar várias tarefas, como atribuir um endereço a uma interface de rede ou configurar parâmetros da interface de rede. Com ele podemos atuar nas seguintes camadas do modelo OSI:

- Camada de enlace de dados;
- Camada de rede;

Na figura abaixo temos a pilha de camadas OSI e as opções principais do comando `ip` para manutenção da configuração lógica.



Onde:

1. **ip address** para configurar a camada de Rede (Modelo OSI);
2. **ip route** para configurar rotas de saída do computador na camada de Rede (Modelo OSI);
3. **ip link** para configurar a interface e interação com a parte física, pela camada de Enlace (Modelo OSI).

A sintaxe do comando é simples e bem estruturada, conforme listagem abaixo.

```
1. ip [OPTION] SUBCOMMAND {COMMAND | help}
```

Subcomandos que você usará com mais frequência incluem:

- **link** usado para exibir e modificar interfaces de rede.
- **address** usado para exibir e modificar endereços de protocolo (IP, IPv6).
- **route** usada para exibir e alterar a tabela de roteamento.
- **neigh** usado para exibir e manipular objetos vizinhos (tabela ARP).

Existem muitos outros objetos e comandos disponíveis. Para ver uma lista completa digite o comando `ip` com a opção “**help**”, conforme figura abaixo.

```
usuario@debian:~$ ip help
usuario@debian:~$ ip help
Usage: ip [OPTIONS] OBJECT {COMMAND | help}
      ip [-force] -batch filename
where  OBJECT := { address | addrlabel | amt | fou | help | ila | ioam | l2tp |
               link | macsec | maddress | monitor | mptcp | mroute | mrule |
               neighbor | neighbour | netconf | netns | nexthop | ntable |
               ntbl | route | rule | sr | tap | tcpmetrics |
               token | tunnel | tuntap | vrf | xfrm }
OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
             -h[uman-readable] | -i[ec] | -j[son] | -p[retty] |
             -f[amily] { inet | inet6 | mpls | bridge | link } |
             -4 | -6 | -M | -B | -9 | -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
             -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename] |
             -rc[vbuf] [size] | -n[etns] name | -N[umeric] | -a[ll] |
             -c[olor] }
```

9.3.1 Obtendo informações de interfaces de rede

Para obter informações das interfaces de rede, tal como endereço IP, MAC, etc., basta digitar o comando:

1. ip address

Onde,

address é a opção que será usada pelo comando IP para exibir os endereços;

```
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group 0
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group 0
    link/ether 08:00:27:51:1d:06 brd ff:ff:ff:ff:ff:ff
        inet 192.168.0.6/24 brd 192.168.0.255 scope global dynamic enp0s3
            valid_lft 35978sec preferred_lft 35978sec
        inet6 2804:14c:bf41:9956:a00:27ff:fe51:1d06/64 scope global dynamic
            valid_lft 86400sec preferred_lft 72000sec
        inet6 fe80::a00:27ff:fe51:1d06/64 scope link
            valid_lft forever preferred_lft forever
usuario@debian:~$ _
```

No output acima o equipamento possui duas interfaces de rede, sendo:

1. **lo**: uma interface virtual chamada loopback no qual todos os computadores possuem, e está com ip 127.0.0.1;
2. **enp0s3**: uma interface de rede conectada na rede 192.168.0.0 com máscara 255.255.255.0.

A interface virtual “**lo**” é utilizada para os programas locais desta máquina se conectarem com serviços locais nesta máquina, ou seja, cliente e servidor na mesma máquina. O endereço 127.0.0.1 é um endereço universal para esta interface, em muitas máquinas este endereço ip é resolvido pelo apelido localhost ou local.

9.3.2 Alterando/Atribuindo endereços para uma interface

Uma placa de rede possui um endereço de camada de rede chamado endereço IP, este endereço tem duas versões:

- IPv4 com 32 bits, geralmente representado em 4 octetos;
- IPv6 com 128 bits, geralmente representado em hexadecimal;

Está evidente que um endereço de 32 bits não será capaz de dar um endereço de rede para cada dispositivo na terra e no espaço. IPv6 é a versão mais atual do Protocolo de Internet. Originalmente oficializada em 6 de junho de 2012, é fruto do esforço do IETF para criar a "nova geração do IP", cujas linhas mestras foram descritas por Scott Bradner e Allison Marken, em 1994, na RFC 1752.

O protocolo está sendo implantado gradativamente na Internet e deve funcionar lado a lado com o IPv4, numa situação tecnicamente chamada de "dual stack", por algum tempo. A longo prazo, o IPv6 tem como objetivo substituir o IPv4. No exemplo anterior a interface de rede possui o endereço 192.168.0.6, e neste próximo exemplo este endereço será removido, então para remover um IP basta utilizar o comando **ip address** com o parâmetro "**del**", conforme exemplo abaixo.

```
1. sudo ip address del 192.168.0.6/24 dev enp0s3
```

Onde,

del remover um endereço inet;

192.168.0.6/24 é um endereço Classe C de LAN com máscara padrão;

dev informa que o próximo parâmetro é o Devide;

enp0s3 é o device alvo da configuração.

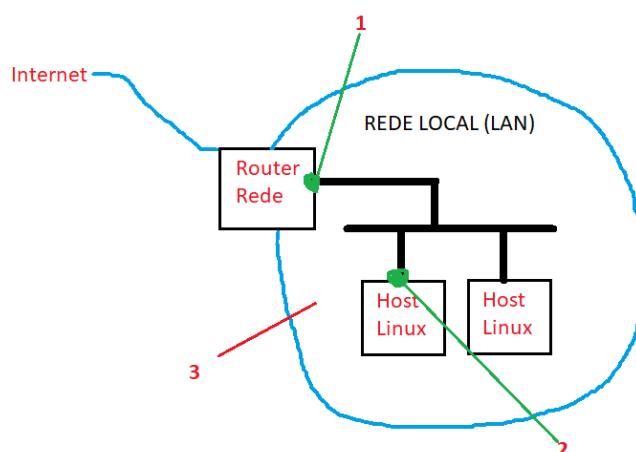
Já para adicionar um novo endereço, torque o "**del**" pelo "**add**" conforme exemplo abaixo.

```
1. sudo ip address add 192.168.0.101/24 dev enp0s3
```

No exemplo acima o endereço 192.168.0.101 com a máscara 255.255.255.0 foi adicionado ao device **enp0s3**, estes dados são calculados baseado na configuração da rede, ou seja, no projeto lógico.

9.3.3 Criando e removendo rotas

Explicar o que é uma Rede Local com poucas palavras é uma tarefa complicada, digamos que um agrupamento de máquinas interconectadas por elementos de infra-estrutura e cabeamento, e nesta rede temos dispositivos classificados como End-Device⁶².



⁶² Computador, servidor, notebook, celular, impressora, voip, etc..

Onde na imagem acima:

1. Interface de rede de um Host, IP 10.68.76.41/23
2. Interface de rede do Router Gateway, IP 10.68.76.1/23
3. Rede local com endereço 10.68.76.0/23

Dentro desta rede de computadores temos dois projetos:

- **Projeto físico:** no qual encontramos equipamentos e cabeamento;
- **Projeto lógico:** a configuração dos equipamentos e serviços;

O comando **ip route** ajuda você a ver a rota que os pacotes adotarão conforme definido na sua tabela de roteamento. Um computador possui por padrão duas opções de rota:

1. Rotear os pacotes diretamente para um outro host na mesma rede;
2. Rotear os pacotes para um Router Gateway que envia para a Internet.

No exemplo abaixo, o computador está na rede 10.68.76.0/23. Qualquer computador no intervalo 10.68.76.0 até 10.68.77.255 está na mesma rede local, utilizando o esquema de rota 1 (listagem acima). Caso um pacote tenha que ser enviado para um host que está em uma rede diferente de 10.68.76.0/23, então o pacote é enviado para o endereço do router gateway na borda da rede, no caso rota 2 da listagem acima.

```
usuario@debian:~$ ip route
default via 10.68.76.1 dev enp0s3
10.68.76.0/23 dev enp0s3 proto kernel scope link src 10.68.76.41
usuario@debian:~$ _
```

No caso do exemplo acima a regra **default via 10.68.76.1 dev enp0s3**, corresponde a rota 2 da listagem acima, usado por pacotes que vão sair da rede local. Para os pacotes destinados aos hosts da rede local estes vão utilizar a regra **10.68.76.0/23 dev enp0s3 proto kernel scope link src 10.68.76.41**.

Então para um pacote que deverá sair da rede este deverá ser roteado até o host 10.68.76.1/23 (ver esquema da rede acima) e caso o pacote tenha como destino um host dentro da rede local, este deve sair pela interface 10.68.76.41/23. Para adicionar um Router Gateway padrão utilize o comando abaixo:

```
1. sudo ip route add default via 192.168.0.1
```

Onde,

default indica que é a regra padrão que aponta para o gateway da rede;
via informa que para alcançar o gateway deve ser usado o IP que vem a seguir;
192.168.0.1 é o IP do Roteador gateway da rede onde a máquina está.

Caso queira remover, da mesma forma como demonstrado no **ip address**, substitua o “**add**” por “**del**”.

9.3.4 Ligando e desligando uma interface de rede

Conforme já descrito, o comando `ip` atua tanto na camada de Network quanto na camada de Data Link, esta última destina-se a controlar o dispositivo I/O. Quando ligamos ou desligamos uma interface de rede atuamos nesta camada, então o comando é `ip link`.

```
1. sudo ip link set dev enp0s3 down
```

No comando acima a interface alvo da ação `down` é a interface `enp0s3`. A palavra `set` refere-se a alteração do `device`. Para ligar a interface de rede, basta trocar de `down` para `up`.

9.4 Comando ifconfig

O comando `ifconfig` é usado para configurar as interfaces de rede. É usado no momento da inicialização para configurar as interfaces conforme necessário e depois disso, geralmente é usado quando necessário durante a depuração ou quando você precisa de ajuste do sistema. Além disso, este comando é usado para atribuir o endereço IP e a máscara de rede a uma interface ou para ativar ou desativar uma determinada interface.

Versões mais recentes de algumas distribuições GNU/Linux não têm o comando `ifconfig` pré-instalado. Então, no caso, há um erro “`ifconfig: command not found`”. Como este comando pertence ao pacote `net-tools`.

```
1. ifconfig [...OPTIONS] [INTERFACE]
```

Onde as opções:

- a** Imprime a configuração para todas as interfaces, não apenas as ativas.
- s** Exibe apenas uma lista de todas as interfaces.
- v** Imprime uma configuração mais detalhada para todas as interfaces.
- [IP_address_number]** Atribua um endereço IP a uma interface especificada.
- netmask [endereço]** Altera a máscara de rede para uma interface específica.
- broadcast [endereço]** Define o endereço de transmissão para uma interface específica.
- mtu [número]** Define a Unidade de Transferência Máxima (MTU) de uma interface.
- arp** Permite o protocolo ARP em uma interface específica.

Caso precise instalar:

```
1. sudo apt install net-tools -y
```

Alguns exemplos:

```
1. sudo ifconfig
2. sudo ifconfig enp0s3 down
3. sudo ifconfig enp0s3 up
4. sudo ifconfig enp0s3 192.168.0.2 netmask 255.255.255.0 broadcast 192.168.0.255
```

Onde na linha 1 o comando está listando todos os dados de todas as interfaces de rede. Na linha 2 estamos desligando a interface de rede `enp0s3` e na linha 3 estamos ligando a interface. Na linha 4 estamos alterando o endereço IPv4 para `192.168.0.2` com netmask `255.255.255.0` e broadcast `192.168.0.255`.

9.5 Arquivo `/etc/network/interfaces`

Outra possibilidade é realizar a configuração no arquivo `/etc/network/interface`, mas o leitor deve estar atento que nem toda distribuição GNU/Linux possui este arquivo e que ainda está no mesmo layout.

```
GNU nano 3.2                               /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp
```

A linha **allow-hotplug enp0s3** ativa a interface de rede **enp0s3**, pode ser substituída por **auto enp0s3**. A diferença é que o allow-hotplug inicia a interface de rede junto com o Sistema Operacional, se algum problema ocorrer o Linux vai demorar um minuto a mais para carregar. Já o auto, a placa de rede é carregada após o carregamento do sistema operacional.

A linha **iface enp0s3 inet dhcp** informa ao `networking.service` que a configuração da interface de rede será feita por serviço DHCP onde esta máquina, em especial, esta interface é cliente de tal serviço. Para configurar um IP estático nesta máquina de forma estática, altere o arquivo conforme imagem abaixo em vez de usar os comandos acima.

```
GNU nano 3.2                               /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto enp0s3
iface enp0s3 inet static
    address 192.168.0.101
    netmask 255.255.255.0
    network 192.168.0.0
    broadcast 192.168.0.255
    gateway 192.168.0.1
    dns-nameservers 192.168.0.1
```

Onde,

iface enp0s3 inet static informa para o `networking.service` que o IP será atribuído de forma estática;

address informa o IP da interface de rede enp0s3;
netmask informa a máscara, que neste caso é máscara padrão;
network informa a rede que este computador faz parte, é opcional;
broadcast informa o endereço de comunicação de Broadcast;
gateway é o IP do router gateway da rede, se não for configurado os PDUs não são capazes de ir para uma rede distante, mas se comunica na rede local;
dns-nameservers informa o endereço da máquina que possui o DNS Server configurado para resolução de nomes.

Após fazer a configuração por arquivo de configuração, é fundamental que se reinicie o Networking, para isso execute o comando:

```
1. sudo systemctl restart networking.service
```

Mas atenção, se mesmo assim o IP não aparecer no comando ip address então terá que reiniciar o servidor GNU/Linux com o comando reboot.

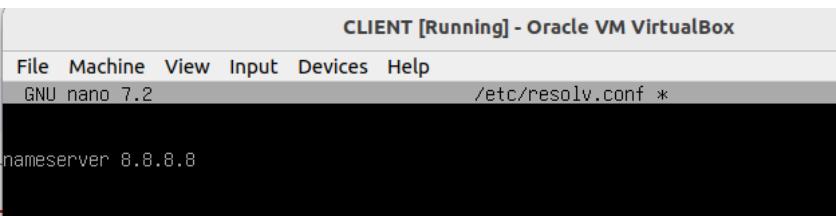
9.6 Domain Name System com resolver, host, nslookup, dig, e whois

O serviço de tradução de nomes é fundamental para se manter um cenário humano, pois as conexões de tudo que usamos precisam de endereços IPs que são intragáveis para o cérebro humano.

Por exemplo, no dado momento que escrevo este livro o domínio **google.com** está no ip **172.217.28.14**, e para o autor é mais fácil decorar nomes do que números. A teoria de DNS é descrita com detalhes no capítulo de [Serviço Domain Name System](#). A configuração do DNS (IP do serviço de tradução de nomes) pode ser realizado em dois arquivos, são estes:

1. /etc/network/interfaces
2. /etc/resolv.conf

No caso do arquivo 1, o atributo que deve ser adicionado é o atributo dns-nameservers (conforme visto no tópico [Arquivo /etc/network/interfaces](#)). Já no arquivo 2, é complicado. Para um usuário comum, ou seja, para um aluno mediano é apenas um arquivo, conforme exemplo abaixo.



```
CLIENT [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2
/etc/resolv.conf *
nameserver 8.8.8.8
```

O resolver é um conjunto de rotinas em biblioteca C/C++ que fornece acesso ao Domain Name System (DNS) da Internet. O sistema operacional além de ter um programa que de tempos em tempos atualiza o arquivo /etc/resolv.conf possui também a implementação

necessária para qualquer programador ou programa utilizar os recursos da biblioteca `resolv.h`.

O arquivo `/etc/resolv.conf` foi projetado para ser legível por humanos e contém uma lista de palavras-chave com valores que fornecem várias formas de se resolver um domínio. As principais opções de configuração são:

nameserver IP Endereço de Internet de um servidor de nomes que o resolver deve consultar, seja um endereço IPv4 ou um endereço IPv6 em notação de dois pontos conforme RFC 2373, esta lista de IPs tem um limite de até MAXNS (atualmente 3, consulte `<resolv.h>`). Se houver vários servidores, a biblioteca resolver os consulta na ordem listada. Se nenhuma entrada de servidor de nomes estiver presente, o padrão é usar o servidor de nomes na máquina local;

search domain Por padrão, a lista de pesquisa contém uma entrada, o nome de domínio local. É determinado a partir do nome do host local retornado por `gethostname(2)`; o nome de domínio local é considerado tudo após o primeiro '.', exemplo `debian.aied.com.br`. Finalmente, se o nome do host não contiver um '.', o domínio raiz será assumido como o nome do domínio local, por exemplo `aied.com.br`;

sortlist IP/MASK Esta opção permite que os endereços retornados por `gethostbyname(3)` sejam classificados. Uma lista de classificação é especificada por pares endereço IP-máscara de rede. A máscara de rede é opcional e o padrão é a máscara de rede natural da rede. O endereço IP e os pares de rede opcionais são separados por barras. Até 10 pares podem ser especificados. Um exemplo: `sortlist 130.155.160.0/255.255.240.0 130.155.0.0`

Um grande problema que temos no cotidiano de um programador é resolver um domínio e obter um endereço IPv4. No próximo exemplo de código C++, vamos utilizar o header `arpa/inet.h` para utilizar a função `inet_ntoa()`, esta função converte endereços IP de uma string no padrão de octetos para uma `struct in_addr`. Já o header `netdb.h` é importado para ter acesso a função `gethostbyname()` que retorna dados em formato de endereço a partir de um nome de máquina ou domínio. Como resultado final vamos ter um endereço IPv4 do endereço `www.aied.com.br`. Escreva o código abaixo em um arquivo `resolвеaied.cpp`.

```

1. #include <netdb.h>
2. #include <string.h>
3. #include <arpa/inet.h>
4. #include <iostream>
5.
6. struct hostent *he;
7. struct in_addr a;
8.
9. int main () {
10.     he = gethostbyname("www.aied.com.br");
11.     if ( he ) {
12.         std::cout << "Name: " << he->h_name << std::endl;
13.         while ( *he->h_aliases ) {
14.             std::cout << "Alias: " << *he->h_aliases++ << std::endl;
15.         }
16.
17.         while ( *he->h_addr_list ) {

```

```

18.         bcopy( *he->h_addr_list++, (char *) &a, sizeof(a) );
19.         std::cout << "Address: " << inet_ntoa(a) << std::endl;
20.     }
21. } else {
22.     std::cerr << "error" << std::endl;
23. }
24. return 0;
25. }
```

Como estamos utilizando as bibliotecas padrões do GNU/Linux o processo de compilação é simples, conforme visto abaixo.

```

1. g++ -o resoliveaied resoliveaied.cpp
2. ./resoliveaied
```

No exemplo acima além do endereço IPv4 também exibido o alias, caso queira somente o endereço IPv4 a versão abaixo é uma versão reduzida.

```

1. #include <netdb.h>
2. #include <arpa/inet.h>
3. #include <iostream>
4.
5. int main()
6. {
7.     struct hostent *he = gethostbyname("www.aied.com.br");
8.     char *ip = inet_ntoa( *(struct in_addr*)he->h_addr_list[0] );
9.     std::cout << ip;
10.    return 0;
11. }
```

O comando **nslookup** é um programa utilizado para consultar servidores de nomes de domínio da Internet, ele possui dois modos:

- interativo;
- não interativo.

O modo interativo permite ao usuário consultar servidores de nomes para obter informações sobre vários hosts e domínios ou imprimir uma lista de hosts em um domínio. O modo não interativo imprime apenas o nome e as informações solicitadas para um host ou domínio. Para entrar no modo interativo, basta digitar no terminal o comando **nslookup** e pressionar **ENTER**, o programa irá solicitar parâmetros e domínio, conforme listagem abaixo. Para sair basta digitar **exit**.

```

1. nslookup
2. > set type=ns
3. > aied.com.br
4. >
```

Na linha 2 temos a opção SET seguido de um parâmetro, temos as seguintes opções para o comando nslookup:

- domain=[domínio] Alterar o nome DNS padrão.
- debug Mostra informações de depuração.

- port=[número porta]** Especifique a porta para consultas. O número da porta padrão é 53.
- timeout=[segundos]** Especifique o tempo permitido para o servidor responder.
- type=a** Visualiza informações sobre os registros de endereço DNS A.
- type=mx** Os registros MX armazenam todos os dados relevantes do servidor Mail Exchange. Essas informações são usadas para rotear todas as solicitações de email do domínio para o servidor de email apropriado;
- type=soa** Os registros de início de autoridade (SOA) fornecem informações oficiais sobre o domínio e o servidor, como endereço de e-mail do administrador, número de série, intervalo de atualização, tempo de expiração da consulta, etc.ç
- type=txt** Os registros TXT contêm informações importantes para usuários fora do domínio. Por exemplo, Google e Facebook usam registros TXT para verificar a propriedade do domínio;

O modo não interativo é direto, basta chamar o comando, passar as opções e por fim o domínio, conforme exemplo abaixo.

```
1. nslookup -type=ns aied.com.br
```

Conforme mencionado, há uma grande base de dados descentralizada de domínios. Esta base de dados é descentralizada pois cada país ou região é responsável pela gestão de seus domínios/subdomínios. No Brasil temos a CertBr⁶³ com o projeto RegistroBr⁶⁴. Quando uma pessoa Física ou Jurídica adquire um domínio, então ela obtém um registro DNS nos serviços de NS destas organizações, e por isso em alguns países há um cadastro.

No passado estes dados eram utilizados para reportar mau funcionamento dos serviços WEB, quem lembra que nos primórdios de 2000 os serviços eram instáveis. Hoje a WEB é tão profissional que só grandes organizações ou datacenters conseguem manter um serviço de tanta qualidade como os que vemos hoje. Então no passado, não era de se estranhar que qualquer pessoa tivesse acesso a todos os dados de cadastro do domínio para reclamar de qualidade.

Hoje é um grande problema, principalmente em países como o Brasil que as leis de privacidade não são aplicadas a órgãos públicos, e a irresponsabilidade destes órgãos recaem sobre a insegurança dos clientes de domínios dos serviços CertBr. Tais informações são usadas tanto na fase de fingerprint quanto na fase de footprint.

Um comando muito usado pelos hackers, na verdade, não é um comando feito para os hackers, trata-se do comando whois. Caso o whois não esteja instalado, instale o módulo whois, para isso use o comando apt.

```
1. whois [-h HOST] [-p PORT] [-aCFH1LMmrRSVx] [-g FONTE : PRIMEIRO ÚLTIMO ]Objeto
[-i ATTR] [-S SOURCE] [-T TYPE]
```

Onde:

⁶³ Acessível pela url <https://www.cert.br/>

⁶⁴ Acessível pela url <https://registro.br/>

-h HOST	Conecte-se ao host de banco de dados WHOIS HOST;
-H	Suprime a exibição de isenções de responsabilidade legais;
-p PORT	Ao conectar, conecte à porta de rede PORT;
-verbose	Todos os detalhes são expostos no outupt;
-Help	Exiba uma mensagem de ajuda.

Mas atenção, para evitar raspagem de dados, há limitações do servidor contra IPs de clientes, então este comando pode não funcionar no local onde o aluno está.

O comando **host** no sistema GNU/Linux é usado para operações de pesquisa DNS (Domain Name System). Em palavras simples, este comando é usado para encontrar o endereço IP de um determinado nome de domínio ou se você quiser descobrir o nome de domínio de um determinado endereço IP. Você também pode encontrar detalhes mais específicos de um domínio especificando a opção correspondente junto com o nome de domínio.

1. host [-aCdlriTWV] [-c class] [-N ndots] [-t type] [-W time] [-R number] [-m flag] hostname [server]

Veja o exemplo básico da busca por dados sobre o domínio aied.com.br:

```
usuario@debian:~$ host aied.com.br
aied.com.br has address 212.1.209.207
aied.com.br has IPv6 address 2a02:4780:1:793:0:3848:1b21:2
aied.com.br mail is handled by 5 mx1.hostinger.com.br.
aied.com.br mail is handled by 10 mx2.hostinger.com.br.
usuario@debian:~$ _
```

Pode-se filtrar, digamos que queria somente a classe mail, pode utilizar **-c mail**.

O comando **dig** é mais um comando que exibe informações sobre um domínio, mas uma grande vantagem que vejo é que consigo com ele obter dados de um domínio na "na visão de um servidor DNS específico". Imagine que ao realizar uma tradução de domínio do ponto em que estou no planeta eu recebo um IP, agora, em outro ponto do planeta retornaria outro IP, seja por delay de replicação de DNS ou até mesmo por distribuição de serviços, algo comum em CDN.

Na figura abaixo estou mostrando uma forma mais humana em um site, ou seja, não é um comando. O objetivo desta figura é demonstrar isso para o leitor.

DNS CHECK

CHECK DNS PROPAGATION

Whether you have recently changed your DNS records, switched web host, or started a new website, checking whether the DNS records are propagated globally is essential. DNS Checker provides a check service to verify Domain Name System records against a selected list of DNS servers worldwide. Perform a quick DNS propagation lookup for any hostname or domain, and check from all available DNS Servers to confirm that the DNS records are fully propagated.

DNS Propagation Map by DNSChecker.org

Map showing propagation status across the world. Most locations are marked as 'Available' (blue), with a few yellow markers indicating propagation status.

Server	IP Address	Status
OpenDNS	212.1.209.207	✓
Mountain View CA, United States (Google)	212.1.209.207	✓
Berkeley, US (Quad9)	212.1.209.207	✓
Miami, United States (AT&T Services)	212.1.209.207	✓
Virginia, United States (VeriSign Global Registry Services)	212.1.209.207	✓
San Francisco, US (Quad9)	212.1.209.207	✓
Ashburn, United States (NeuStar)	212.1.209.207	✓
Burnaby, Canada (Fortinet Inc.)	212.1.209.207	✓
Ramenskoye, Russia (IONICA LLC)	212.1.209.207	✓
Cullinan, South Africa (Liquid Telecommunications Ltd)	212.1.209.207	✓

Primeiro vou mostrar um exemplo simples, no qual de onde estou utilizando os dados de nameserver em /etc/resolv.conf vou obter dados de um domínio. Veja na figura abaixo que o servidor que respondeu foi o 10.0.2.3.

```
usuario@debian:~$ dig aied.com.br
; <>> DiG 9.18.19-1~deb12u1-Debian <>> aied.com.br
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53092
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;aied.com.br.           IN      A
;; ANSWER SECTION:
aied.com.br.       6992    IN      A      212.1.209.207
;; Query time: 3 msec
;; SERVER: 10.0.2.3#53(10.0.2.3) (UDP)
;; WHEN: Fri Dec 08 16:10:02 -03 2023
;; MSG SIZE  rcvd: 56
```

Se olhar o arquivo resolv.conf verá que este IP está cadastrado como nameserver.

```
usuario@debian:~$ cat /etc/resolv.conf
nameserver 10.0.2.3
usuario@debian:~$
```

Agora se quiser ver o IP retornado para o domínio aied.com.br a partir de outro servidor de DNS, basta usar @ seguido do IP do servidor com DNS.

```
usuario@debian:~$ dig @8.8.8.8 aied.com.br
; <>> DiG 9.18.19-1~deb12u1-Debian <>> @8.8.8.8 aied.com.br
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52995
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;aied.com.br.           IN      A
;; ANSWER SECTION:
aied.com.br.        14400   IN      A      212.1.209.207
;; Query time: 263 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Fri Dec 08 16:16:25 -03 2023
;; MSG SIZE  rcvd: 56
```

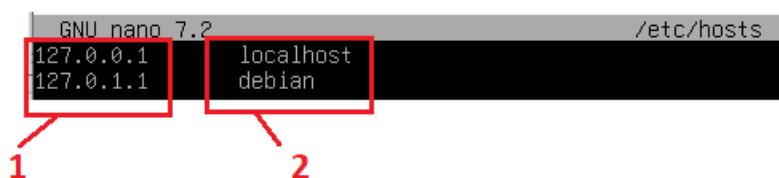
Esse recurso é muito útil para estudar a replicação de um domínio.

9.7 Arquivo /etc/hosts

Um dos arquivos mais antigos é o arquivo `/etc/hosts`, tanto que é comum em ambientes Unix, Linux e Windows. Sua ideia provém dos tempos da DARPA como um arquivo de resolução de nomes para todos os hosts conectados à internet, em um cenário que nem existia os serviços de DNS. Então naquela época esse arquivo era replicado manualmente.

É de se imaginar que este arquivo ficou muito grande, e com isso a manutenção passou a ser mais difícil. Lembre-se de que nessa alvorecer da internet (com i minúsculo) tínhamos um cenário volátil. A necessidade de serviços DNS vem desta dificuldade enfrentada anteriormente.

Embora seja uma ideia antiga, ainda persistem nos sistemas operacionais, e é um importante arquivo que pouco modificamos, mas que tem um grande impacto nos sistemas. Antes de um host questionar os serviços de DNS sobre a tradução de um domínio, o host consulta o arquivo `/etc/hosts`, então se não encontra o domínio ali, ele realiza a requisição para o serviço DNS.



```
GNU nano 7.2          /etc/hosts
127.0.0.1      localhost
127.0.1.1      debian
```

Onde:

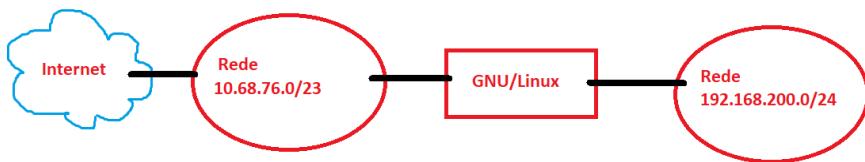
1. Primeira coluna ficam os endereços IP dos domínios ou hosts;
2. Segunda coluna ficam os nomes dos domínios ou hosts.

Então, se algum processo no computador precisar resolver localhost, o próprio arquivo tem a definição para isso e o computador retornará 127.0.0.1. Repare na opção `debian`, ou seja, o nome do host. Muitos serviços não endereçam ao localhost suas requisições, acabam

endereçando ao nome do computador, então se trocar o nome do computador deve também trocar aqui o nome do computador.

9.8 Regras de roteamento do Linux

Sempre que o GNU/Linux é inicializado um complexo esquema de carregamento de processos iniciará, um destes processos é um serviço chamado Network Service, este irá carregar as interfaces de rede e iniciar o processo de conexão de rede. Para cada rede que um computador conseguir carregar ele irá criar uma regra de roteamento. Então se temos um computador em 2 redes ele terá no mínimo 3 regras. Uma para cada rede e uma terceira que seria a regra default. Na figura abaixo o Linux está entre 2 redes, mas só a rede 10.68.76.0/23 tem acesso para a Internet.



Então quando o Linux for iniciado, ele irá criar as 3 regras, conforme figura abaixo.

```

1
2
3
userlinux@debian:~$ ip route
default via 10.68.76.1 dev enp0s3
10.68.76.0/23 dev enp0s3 proto kernel scope link src 10.68.76.254
192.168.200.0/24 dev enp0s8 proto kernel scope link src 192.168.200.1
userlinux@debian:~$ _

```

Onde:

1. O Debian cria uma regra default na qual será usada para enviar todas as mensagens que não são para rede 192.168.200.0/24 e nem para a rede 10.68.76.0/23;
2. O Debian irá usar essa regra para enviar mensagens para a rede 10.68.76.0/23;
3. O Debian irá usar essa regra para enviar mensagens para a rede 192.168.200.0/24;

Vejo muitas soluções para Kill Switch, que é o momento que temos que desligar todas as conexões, talvez por estar sendo atacado ou até mesmo já estar invadido. **ISSO É NORMAL**. Mas a melhor de todas as soluções é dropar a regra **default**. Com isso nada vai sair para a Internet e nem entrar, poderá fazer uma auditoria ou mesmo uma forense. Veja abaixo o processo.

```

userlinux@debian:~$ sudo ip route del default
[sudo] password for userlinux:
userlinux@debian:~$ ip route
10.68.76.0/23 dev enp0s3 proto kernel scope link src 10.68.76.254
192.168.200.0/24 dev enp0s8 proto kernel scope link src 192.168.200.1
userlinux@debian:~$ 

```

Se você se sentir confiante para restabelecer a comunicação com o mundo exterior, basta adicionar novamente, conforme sequência de comandos abaixo.

```
userlinux@debian:~$ sudo ip route add default via 10.68.76.1
userlinux@debian:~$ ip route
default via 10.68.76.1 dev enp0s3
10.68.76.0/23 dev enp0s3 proto kernel scope link src 10.68.76.254
192.168.200.0/24 dev enp0s8 proto kernel scope link src 192.168.200.1
userlinux@debian:~$
```

Só tem que lembrar qual é o IP do gateway da rede.

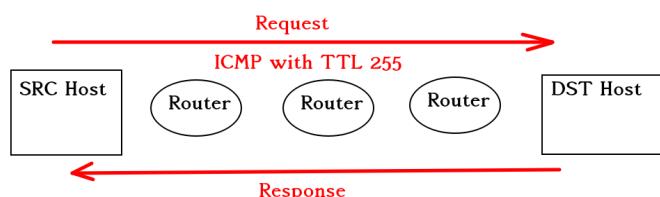
9.9 Realizando testes com ping, traceroute, tracepath e mtr

O protocolo ICMP é um protocolo de camada de rede, é utilizado por equipamentos para troca de informações, na verdade, diversos tipos de informações. Este protocolo se utiliza do protocolo IP mesmo estando na mesma camada, e força uma conexão por datagrama. Na visão humana é uma das ferramentas de testes de disponibilidade de um host, embora segundo a CISCO deve ser a segunda opção pois alguns routers podem negar tais protocolos para evitar ataques.

Provavelmente o aluno já ouviu falar em PING, ping é um programa de computador que envia protocolos ICMP para um alvo, e naturalmente se obtém resposta computa não só a disponibilidade do host alvo como também o tempo de resposta. No Debian GNU/Linux se não encontrar tanto o Ping quanto o Traceroute, instale o pacote **iputils-ping** conforme listagem abaixo.

```
1. sudo apt install iputils-ping -y
```

O que o Ping no GNU/Linux faz é enviar vários protocolos ICMP com TTL (tempo total de vida) bem alto contra um alvo remoto, e então ao receber a mensagem o alvo remoto retorna a resposta.



Veja na imagem acima que há vários routers no meio do caminho, mas como o destino aceita ICMP passará ileso o protocolo, chegando até o alvo. Por cada router que o ICMP passar ele abate o TTL em -1, e para parâmetro, do Estado de São Paulo no Brasil até a Califórnia USA, o número de salto médio é 18.

```
usuario@debian:~$ 
usuario@debian:~$ ping -c 1 google.com 1
PING google.com (142.250.219.142) 56(84) bytes of data.
64 bytes from [REDACTED] -in-f14.1e120.net (142.250.219.142): icmp_seq=1 ttl=118 time=4.69 ms
3
--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.686/4.686/4.686/0.000 ms
usuario@debian:~$ _
```

Onde:

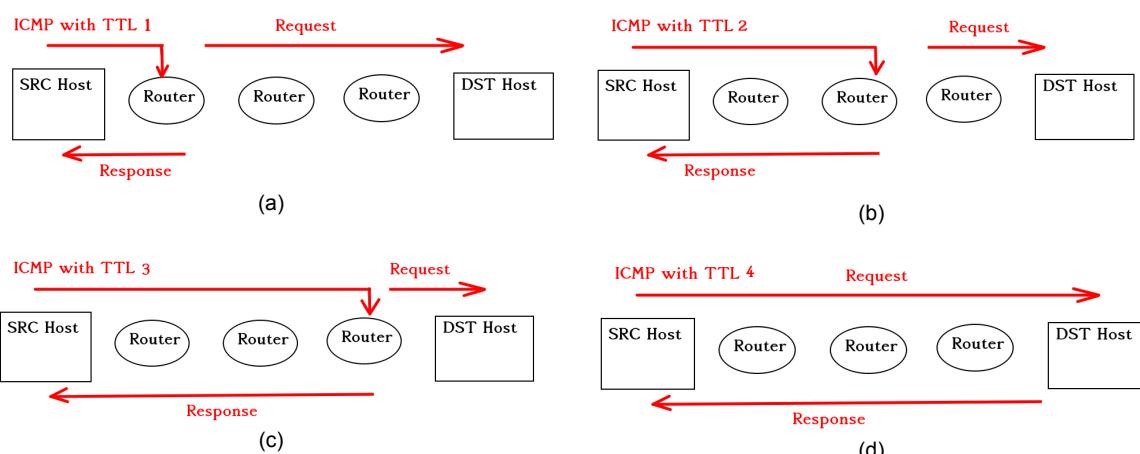
1. Comando ping com limite de 1 requisição contra o alvo google.com;
2. IP do alvo google.com, o ping traduz o domínio em um IP;
3. TTL do pacote ICMP e tempo de resposta.

Já o Traceroute é utilizado para:

- Analisar topologia;
- Fazer um diagnóstico do caminho;

Quando não conhecemos a topologia então tentamos utilizar ICMP para aprender sobre o caminho tomado por um pacote da origem até o destino, ou, podemos analisar um ponto de gargalo em um caminho, gerando assim um relatório de tempos de resposta.

Ao invés de mandar um único ICMP com TTL muito alto (conforme Ping), o programa Traceroute em um laço de repetição envia ICMP para o alvo mas com TTL iniciando em 1. Assim que o primeiro equipamento responde (ver em "a" na figura abaixo), refazemos o laço. E assim o algoritmo vai incrementando o TTL em +1, a cada resposta que chega, até que a resposta venha justamente do nosso alvo.



Na figura acima o laço de repetições foi executado 4 vezes. Para exemplificar será executado o comando traceroute contra o domínio google.com, tendo como partida uma máquina na província de São Paulo no Brasil.

```

1  usuario@debian:~$ traceroute google.com
traceroute to google.com (216.58.222.14), 30 hops max, 60 byte packets
 1  192.168.3.1 (192.168.3.1)  0.976 ms  0.862 ms  0.789 ms
 2  192.168.100.1 (192.168.100.1)  1.747 ms  2.018 ms  2.208 ms
 3  10.255.255.7 (10.255.255.7)  11.021 ms  11.098 ms  10.857 ms
 4  100.127.249.1 (100.127.249.1)  11.196 ms  11.129 ms  11.155 ms
 5  as15169.saopaulo.sp.ix.br (187.16.218.58)  11.080 ms  11.368 ms  11.182 ms
 6  108.170.245.129 (108.170.245.129)  11.195 ms  108.170.245.161 (108.170.245.161) ms
 7  142.251.76.111 (142.251.76.111)  9.593 ms  142.251.76.113 (142.251.76.113)
 8  gru06s25-in-f14.1e100.net (216.58.222.14)  5.774 ms  10.167 ms  10.367 ms
2  usuario@debian:~$
```

Onde,

1. Temos o comando traceroute tendo como alvo o google.com;
2. Uma lista de saltos, com tempo de respostas;

Na figura acima está claro que o salto 1 e 2 estão com bom tempo de resposta, mas o grande peso vem entre o salto 2 e 3. Mas não é um link ruim, pois o tempo de latência está na casa de **ms** e foram apenas 8 saltos até a Califórnia.

O comando `tracepath` no GNU/Linux é usado para rastrear o caminho até o destino descobrindo o MTU de cada dispositivo no caminho. Ele usa porta UDP ou alguma porta aleatória. É semelhante a `traceroute`, mas não requer privilégios de superusuário e não tem opções sofisticadas. `tracepath6` é um bom substituto para `traceroute6` e exemplo clássico da aplicação de filas de erros do Linux. A situação com o IPv4 é pior porque os roteadores IP comerciais não retornam informações suficientes em mensagens de erro ICMP. Talvez precise ser instalado, caso precise, digite o comando abaixo.

```
1. sudo apt install iputils-tracepath -y
```

Para executar basta apontar para um domínio conforme `traceroute`.

```
usuario@debian:~$ tracepath google.com
 1?: [LOCALHOST]                                pmtu 1500
 1: 192.168.3.1                                1.201ms
 1: 192.168.3.1                                1.213ms
 2: 192.168.100.1                               1.903ms
 3: 192.168.100.1                               2.155ms pmtu 1492
 3: 10.255.255.7                                4.081ms
 4: 100.127.249.1                               4.247ms
 5: as15169.saopaulo.sp.rix.br                  5.765ms asymm 6
29: no reply
30: no reply
      Too many hops: pmtu 1492
      Resume: pmtu 1492
usuario@debian:~$
```

Como é IPv4, não retornasse muita informação, na figura abaixo em outro ambiente é possível ver outros elementos como `_gateway`.

```
2022-11-08 14:46:58 |ruby-2.7.0| NamLabs-ThinkPad-E14 in ~
o ~ tracepath www.google.com
 1?: [LOCALHOST]                                pmtu 1500
 1: _gateway                                    3.911ms
 1: _gateway                                    2.811ms
 2: 49.207.184.1.actcorp.in                  4.407ms
 3: no reply
 4: no reply
 5: 72.14.242.244                               8.685ms asymm 9
 6: no reply
 7: no reply
 8: no reply
 9: no reply
10: no reply
```

O comando `mtr` é a combinação das funcionalidades encontradas no `ping` e no `traceroute`, ou seja, podemos avaliar a estabilidade de cada salto com base em vários ICMPs enviados para cada salto. Para instalar este comando é preciso executar o comando apt conforme listagem abaixo.

```
1. sudo apt install mtr -y
```

Trata-se de uma ferramenta de alto nível, que vale a pena ser utilizada em dúvida sobre a estabilidade de sua rota. Para exemplificar vamos validar o comando disparando ICMP contra o domínio aied.com.br, conforme listagem abaixo.

```
1. mtr aied.com.br
```

Repare na imagem abaixo que há uma alta taxa de erro no alto 12, pode ser que realmente haja instabilidade ou alguma regra automática foi aplicada, esta última é rara.

My traceroute [v0.95]							2023-12-08T18:33:06-0300		
debian (10.0.2.15) -> aied.com.br (212.1.209.207)		Keys: Help Display mode Restart statistics		Order of fields		quit			
Host		Packets			Pings				
		Loss%	Snt	Last	Avg	Best	Wrst	StDev	
1. 10.0.2.2		0.0%	6	0.5	0.6	0.5	0.6	0.1	
2. _gateway		0.0%	6	1.1	1.6	1.1	3.2	0.8	
3. 201-61-34-85.bbone.telesp.net.br		0.0%	6	1.8	1.8	1.7	1.8	0.0	
4. 186.201.241.153		0.0%	6	3.3	3.4	3.3	3.5	0.1	
5. 186.201.241.154		0.0%	6	3.8	3.9	3.8	4.0	0.1	
6. 192.168.5.2		0.0%	6	4.0	3.8	3.6	4.0	0.1	
7. 192.168.20.12		0.0%	6	3.5	3.8	3.4	5.5	0.8	
8. 192.168.40.2		0.0%	6	4.6	4.5	4.3	4.7	0.1	
9. 192.168.40.4		0.0%	6	4.6	4.9	4.3	6.6	0.8	
10. 187-51-216-237.customer.tdatabrasil.net.br		0.0%	6	4.7	4.6	4.4	4.9	0.2	
11. (waiting for reply)									
12. ae29-grisanem3.net.telefonicaglobalsolutions.com		33.3%	6	6.2	5.9	5.4	6.2	0.4	
13. 94.142.118.1		0.0%	6	116.8	116.8	116.4	117.3	0.3	
14. (waiting for reply)									
15. 4.69.219.150		0.0%	6	132.3	138.2	132.3	150.9	7.3	
16. IMMEDION-LL.edge6.Atlanta2.Level3.net		0.0%	6	130.2	130.3	129.8	130.6	0.3	
17. ip.dartpoints.com		0.0%	6	137.7	137.3	137.0	137.7	0.2	
18. ip.dartpoints.com		0.0%	5	136.2	136.6	136.2	136.9	0.3	
19. ip.dartpoints.com		0.0%	5	137.1	137.0	136.7	137.4	0.3	
20. dial-186.r24.tnmmrl.blomand.net.95.74.206.in-addr.ar		0.0%	5	137.2	137.2	137.1	137.3	0.1	
21. 153.92.2.254		0.0%	5	136.6	136.7	136.6	136.9	0.1	
22. 153.92.2.221		0.0%	5	137.2	137.2	137.0	137.3	0.1	
23. 212.1.209.207		0.0%	5	137.3	137.2	137.0	137.3	0.1	

9.10 Comando Netstat e ss

Netstat é uma ferramenta de linha de comando que exibe informações úteis, como conexões de rede, tabelas de roteamento, estatísticas de interface e muito mais, relacionadas ao subsistema de rede Linux. É útil para solução de problemas de rede e análise de desempenho. Se este comando não está instalado no seu Debian GNU/Linux, execute a instalação conforme a listagem abaixo.

```
1. sudo apt install net-tools -y
```

O comando é simples, precisa ter privilégios e basta usar **netstat** seguido das opções. As principais opções do comando são:

- t: que filtra os resultados para incluir apenas conexões TCP;
- u: que funciona de maneira similar para conexões UDP;
- a: para também listar soquetes ativos (esperando por conexões de entrada);
- n: para exibir os resultados com números: endereço IP, números de porta e ids de usuários;

- p**: para listar os processos envolvidos; essa opção só é útil quando o netstat é executado como root, já que usuários normais apenas verão seus próprios processos;
- c**: para atualizar continuamente a lista de conexões.

Além disso, também é uma ferramenta fundamental de depuração de serviços de rede usada para verificar quais programas estão escutando em quais portas. Por exemplo, o comando a seguir mostrará todas as portas TCP no modo de escuta e quais programas estão escutando nelas.

A exibição padrão para os soquete ativos mostra os itens a seguir:

- Endereços local e remoto
- Tamanhos das filas de envio e recebimento (em bytes)
- Protocolo
- Estado interno do protocolo;

Os formatos de endereço na Internet serão host.porta ou rede.porta se o endereço de um soquete especificar uma rede, mas nenhum endereço de host específico. Se o endereço puder ser resolvido como um nome de host simbólico, o endereço de host, bem como os endereços de rede serão exibidos simbolicamente.

Active Internet connections (w/o servers)				Foreign Address	State
Proto	Recv-Q	Send-Q	Local Address		
tcp	0	0	10.0.2.3:39702	199.232.114.132:http	TIME_WAIT
Active UNIX domain sockets (w/o servers)					
Proto	RefCnt	Flags	Type	State	I-Node Path
unix	3	[]	DGRAM		10011 /run/systemd/notify
unix	7	[]	DGRAM		10031 /run/systemd/journal/dev-log
unix	7	[]	DGRAM		10045 /run/systemd/journal/socket
unix	2	[]	DGRAM		10141 /run/systemd/journal/syslog
unix	2	[]	DGRAM		13535 /run/user/1000/systemd/notify
unix	3	[]	DGRAM		12534
unix	3	[]	STREAM	CONNECTED	13037
unix	2	[]	DGRAM		12479
unix	3	[]	DGRAM		12536
unix	2	[]	DGRAM		13029
unix	3	[]	STREAM	CONNECTED	12366 /run/systemd/journal/stdout
unix	3	[]	STREAM	CONNECTED	13038 /var/run/dbus/system_bus_socket
unix	3	[]	DGRAM		12537
unix	3	[]	STREAM	CONNECTED	12745 /run/systemd/journal/stdout
unix	3	[]	DGRAM		12535
unix	3	[]	DGRAM		10014
unix	3	[]	DGRAM		10013
unix	3	[]	STREAM	CONNECTED	12683 /run/systemd/journal/stdout
unix	3	[]	STREAM	CONNECTED	13012

A primeira lista na saída exibe conexões de internet estabelecidas ativas no computador. Os seguintes detalhes estão nas colunas:

Proto - Protocolo da conexão (TCP, UDP);

Recv-Q - Fila de recebimento de bytes recebidos ou prontos para serem recebidos;

Send-Q - Envia fila de bytes prontos para serem enviados.

Local Address - Detalhes do endereço e porta da conexão local. Um asterisco (*) no host indica que o servidor está escutando e se uma porta ainda não foi estabelecida;

Foreign Address - Detalhes do endereço e porta da extremidade remota da conexão. Um asterisco (*) aparece se uma porta ainda não foi estabelecida.

State - Estado do socket local , mais comumente ESTABLISHED, LISTENING, CLOSED ou blank.

A segunda lista mostra todos os soquetes abertos " Domínio Unix " ativos com os seguintes detalhes:

- Proto** - Protocolo usado pelo socket;
- RefCnt** - Contagem de referência do número de processos anexados a este soquete;
- Flags** - Geralmente ACC ou em branco;
- Type** - O tipo de soquete;
- State** - Estado do soquete, na maioria das vezes CONNECTED, LISTENING ou em branco;
- I-Node** - inode do sistema de arquivos (nó de índice) associado a este soquete;
- Path** - caminho do sistema para o soquete.

O formato de exibição da interface fornece uma tabela de estatísticas cumulativas para os itens a seguir:

- Erros
- Colisões

```
usuario@debian:~$ sudo netstat -i
Kernel Interface table
Iface      MTU     RX-OK RX-ERR RX-DRP RX-OVR     TX-OK TX-ERR TX-DRP TX-OVR Flg
enp0s3     1500      523     0     0     0      528     0     0     0     BMRU
lo        65536      0     0     0     0          0     0     0     0     LRU
usuario@debian:~$
```

Nota: A contagem de colisões para interfaces Ethernet não é aplicável.

A exibição da interface fornece também o nome, número e endereço da interface, bem como as unidades máximas de transmissão (MTUs). A exibição da tabela de roteamento indica as rotas disponíveis e seus status. Cada rota consiste em um host ou rede de destino e um gateway para uso no encaminhamento de pacotes.

```
usuario@debian:~$ sudo netstat -r
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
default         10.0.2.2      0.0.0.0        UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0        255.255.255.0  U        0 0          0 enp0s3
usuario@debian:~$ _
```

Uma rota é fornecida no formato A.B.C.D/XX, que apresenta duas partes de informações. A.B.C.D indica o endereço de destino e XX indica a máscara de rede associada à rota. A máscara de rede é representada pelo número de bits que são configurados. Por exemplo, a rota 9.3.252.192/26 tem uma máscara de rede 255.255.255.192, com um conjunto de 26 bits. Para listar todas as conexões ativas, basta usar -a e para filtrar somente as TCP, use -t, conforme exemplo abaixo.

```
usuario@debian:~$ sudo netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 10.0.2.15:55868          199.232.114.132:http    TIME_WAIT
usuario@debian:~$
```

Uma atividade muito comum é analisar serviços ativos, para procurar entender se algum programa está realizando um shell reverso (por exemplo), para isso use -l.

```
usuario@debian:~$ sudo netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type      State      I-Node      Path
unix     2 [ACC]      STREAM    LISTENING  10249      /run/systemd/fsck.progress
unix     2 [ACC]      STREAM    LISTENING  10015      /run/systemd/private
unix     2 [ACC]      STREAM    LISTENING  10039      /run/systemd/journal/stdout
unix     2 [ACC]      STREAM    LISTENING  12609      /var/run/dbus/system_bus_socket
unix     2 [ACC]      SEQPACKET LISTENING  10335      /run/udev/control
unix     2 [ACC]      STREAM    LISTENING  13445      /run/user/1000/systemd/private
usuario@debian:~$
```

Outro comando de rede interessante é o nmap, que permite avaliar portas abertas bem como que tipo de serviço se encontra em cada porta, mas vou deixar este conteúdo para o livro [Hacker, entre a luz e as trevas](#).

Mas vou implementar um testador de portas, que avalia se uma porta está aberta ou fechada, sem todo o poder de um nmap. Vamos precisar da biblioteca **SFML/Network**⁶⁵, por isso vamos importar tal header conforme linha 2 da listagem abaixo. Nele já temos um teste para **TcpSocket()**. Esta biblioteca possui inúmeros facilitadores interessantes para programadores, recomendo o estudo dela. Com o nano crie um arquivo com o nome **testport.cpp**, e codifique a listagem abaixo.

```
1. #include <iostream>
2. #include <SFML/Network.hpp>
3. #include <string>
4.
5. static bool port_is_open(const std::string& address, int port)
6. {
7.     return (sf::TcpSocket().connect(address, port) == sf::Socket::Done);
8. }
9.
10.int main()
11.{
12.    std::cout << "Port 80 : aied.com.br" << std::endl;
13.    if ( port_is_open("aied.com.br", 80) )
14.        std::cout << "Port is OPEN" << std::endl;
15.    else
16.        std::cout << "Port is CLOSED" << std::endl;
17.    return 0;
18.}
```

O teste será contra o domínio aied.com.br e o teste será contra a porta 80. Segundo a CISCO o teste ideal é este, pois o ICMP pode ser barrado em regras de roteadores por ser um protocolo mal compreendido pelos administradores de rede. Agora no terminal, instale a biblioteca sfml (ver linha 1), compile informando o diretório de bibliotecas do usuário (ver parâmetro **-L/usr/local/lib**) e informe as bibliotecas **network** e **system** conforme linha 3. Para executar é simples, conforme linha 5.

```
1. sudo apt-get install libsfml-dev -y
2.
3. g++ -L/usr/local/lib -o testport testport.cpp -lsfml-network -lsfml-system
```

⁶⁵ Documentação oficial do pacote disponível na URL: <https://packages.debian.org/sid/libsfml-dev>

```
4.
5. ./testport
```

O comando `ss` (socket statistics) é usado para mostrar estatísticas de rede. O comando `ss` é uma versão mais simples e rápida do agora obsoleto comando `netstat`. Juntamente com o comando `ip`, `ss` é essencial para coletar informações de rede e solucionar problemas de rede.

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port	Process
u_str	ESTAB	0	0	*	13379	* 13412
u_dgr	ESTAB	0	0	*	14020	* 12759
u_dgr	ESTAB	0	0	/run/systemd/notify	12738	* 0
u_dgr	ESTAB	0	0	*	13982	* 13983
u_dgr	ESTAB	0	0	*	13424	* 13423
u_dgr	ESTAB	0	0	/run/systemd/journal/dev-log	12759	* 0
u_str	ESTAB	0	0	*	14134	* 14135
u_dgr	ESTAB	0	0	/run/systemd/journal/socket	12761	* 0
u_dgr	ESTAB	0	0	*	12739	* 12740
u_dgr	ESTAB	0	0	*	13983	* 13982
u_dgr	ESTAB	0	0	*	13423	* 13424
u_dgr	ESTAB	0	0	*	13379	* 13412

As colunas mostram os seguintes detalhes:

Netid Tipo de soquete. Os tipos comuns são TCP, UDP, `u_str` (fluxo Unix) e `u_seq` (sequência Unix).

State – State of the socket. Mais comumente ESTAB (estabelecido), UNCONN (não conectado), LISTEN (ouvir).

Recv-Q Número de pacotes recebidos na fila.

Send-Q Número de pacotes enviados na fila.

Local address:port Endereço da máquina local e porta.

Peer address:port Endereço da máquina remota e da porta.

9.11 Endereço MAC Address e o comando arp

Na década de 70, dentro da DIX⁶⁶ foi projetado um protocolo de comunicação para pequenas redes, o que no futuro ficou conhecido como LAN. Em Fevereiro de 1980 a IEEE e outros órgãos regulamentaram o padrão de comunicação Ethernet que mais tarde veio a se tornar o padrão para as LANs. Desde o projeto inicial, nos laboratórios da DIX, este endereço continha 6 bytes, ou seja, 48 bits. A ideia é que cada placa de rede no mundo tenha um endereço MAC único, por isso atuando com 6 bytes. Com a regulamentação, este endereço foi dividido em duas partes:

OUI: Que representa a identificação da organização;

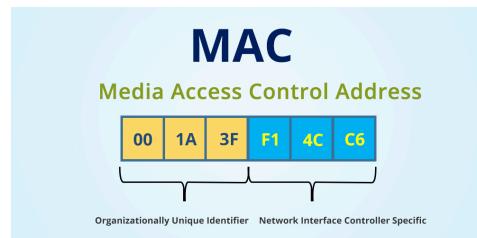
NICS: Que é um número incremental, no qual começa com 00:00:00 e termina com FF:FF:FF.

Veja a imagem abaixo, que a parte mais significativa é destinada ao OUI, e a parte menos significativa ao incremento por parte do fabricante⁶⁷. É dessa forma que os Hackers

⁶⁶ DIX foi o maior centro de pesquisa bancado pela Xerox, recomendo conhecer esta brilhante história;

⁶⁷ A lista completa de OUI de fabricantes pode ser obtida neste link: <https://standards-oui.ieee.org/>

conseguem dirigir ataques a dispositivos específicos, e saiba, hardware é vulnerável da mesma forma que softwares;



9.11.1 Origem do MAC Address

Não posso dizer que todo dispositivo de rede possui um MAC Address gravado em uma memória ROM, há casos em que se solicita tal endereço ou tem permissão para se alterar. Mas falando de interfaces de rede de computadores, em especial servidores, posso dizer que o MAC Address é gravado em uma memória ROM. O que acontece é que a placa de rede é mapeada na memória, no capítulo de dispositivos de I/O no livro de Sistemas Operacionais Modernos do autor Tanenbaum ou autor deixa claro que muitos dispositivos mapeiam seus dados de operação não em registradores, mas sim em memória principal diretamente. E é por isso que alteramos o MAC Address em nosso Linux e após a reinicialização voltamos ao MAC Address original de fábrica.

Não estou lhe ensinando isso tudo para ser um simples usuário Linux, acredito que estou o capacitando para ser o melhor profissional que sua empresa terá no quesito Linux e Redes de computadores, então vou demonstrar um parâmetro que encontramos no Debian, na verdade não sei em quais Linux é possível localizar isso pois é algo que achei em uma tarde de domingo. Com o nano abra o arquivo: **/lib/systemd/network/99-default.link**

```
Debian 12 64 bits [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
GNU nano 7.2
/lib/systemd/network/99-default.link
# SPDX-License-Identifier: MIT-0
#
# This config file is installed as part of systemd.
# It may be freely copied and edited (following the MIT No Attribution license).
#
# To make local modifications, one of the following methods may be used:
# 1. add a drop-in file that extends this file by creating the
#    /etc/systemd/network/99-default.link.d/ directory and creating a
#    new .conf file there.
# 2. copy this file into /etc/systemd/network or one of the other paths checked
#    by systemd-udevd and edit it there.
# This file should not be edited in place, because it'll be overwritten on upgrades.

[Match]
OriginalName=*

[Link]
NamePolicy=keep kernel database onboard slot path
AlternativeNamesPolicy=database onboard slot path
MACAddressPolicy=random
```

O campo **[LINK]** é destinado a parametrização da placa de rede com relação a camada Data Link do modelo OSI, veja o atributo **MACAddressPolicy**⁶⁸, este campo aceita 3 possíveis valores:

⁶⁸ Arquivo acessível na url: <https://man7.org/linux/man-pages/man5/systemd.link.5.html>

persistent: Conforme dito se o hardware tiver em uma memória ROM um valor para MAC Address, sempre que o dispositivo for carregado os valores da ROM serão usados. Mas caso não tenha MAC Address na ROM então o Kernel irá gerar um automaticamente;

random: Um novo endereço é gerado aleatoriamente a cada vez que o computador for carregado, independente. Inclusive todos os 48 bits, incluindo a parte **OUI** do fabricante;

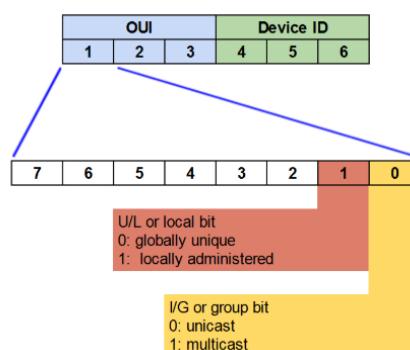
none: O Kernel irá gerar um novo MAC Address e utilizar este endereço sempre;

O conhecimento irá fazer de você um alvo para invejosos e pessoas de má fé, eu já fui atacado por colegas de trabalho e já respondo a processo judicial por ter muito conhecimento, então use modo **random**, criptografe todo o seu notebook e sempre mantenha sigilo de suas senhas, o seu inimigo está literalmente ao seu lado.

9.11.2 Endereço Universal e Local

Endereços podem ser Universally Administered Addresses (UAA) ou Locally Administered Addresses (LAA). Um endereço administrado universalmente é atribuído exclusivamente a um dispositivo por seu fabricante. Os três primeiros octetos identificam a organização que emitiu o identificador e são conhecidos como OUI conforme já mencionado. O restante do endereço são atribuídos por essa organização de quase qualquer maneira que desejarem, sujeitos à restrição de exclusividade

Já um LAA é atribuído a um dispositivo por software ou um administrador de rede, substituindo o endereço gravado de um dispositivo físico, e isso é muito comum em routers de rede. LAA são diferenciados de UAA pela configuração, atribuindo o valor de 1 ao segundo bit menos significativo do primeiro octeto do endereço. Na imagem abaixo veja o segundo bit em laranja.



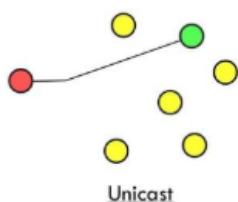
Este bit também é conhecido como bit U/L, abreviação de Universal/Local, que identifica como o endereço é administrado. Se o bit for 0, o endereço é UAA. Se for 1, o endereço é LAA. No endereço de exemplo **06-00-00-00-00-00**, o primeiro octeto é 06 (hexadecimal), cuja forma binária é 00000110, onde o segundo bit menos significativo é 1. Portanto, é um endereço LAA.

9.11.3 Comunicação Unicast, Multicast e Broadcast

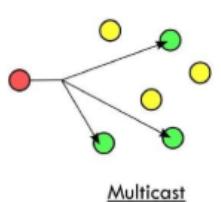
O bit menos significativo do primeiro octeto de um endereço é chamado de bit I/G, ou Individual/Grupo. Quando esse bit é 0, o quadro deve atingir apenas uma interface de rede

receptora. Esse tipo de transmissão é chamado unicast. Um quadro unicast é transmitido para todos os nós dentro do domínio de colisão. Em uma configuração com fio moderna, o domínio de colisão geralmente é o comprimento do cabeamento Ethernet entre duas interfaces de rede. Em uma configuração sem fio, o domínio de colisão são todos os receptores que podem detectar um determinado sinal sem fio. Se um switch não sabe qual porta leva a um determinado endereço MAC, o switch encaminhará um quadro unicast para todas as suas portas, uma ação conhecida como inundação unicast. Somente o nó com o endereço MAC de hardware correspondente (normalmente) aceitará o quadro; interfaces de rede com endereços MAC não correspondentes ignoram o quadro, a menos que estejam no modo promíscuo.

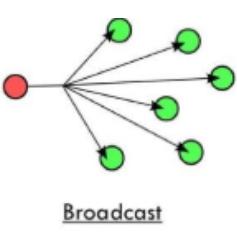
Se o bit menos significativo do primeiro octeto for definido como 1, o quadro ainda será enviado apenas uma vez; no entanto, os controladores de interface de rede escolherão aceitá-lo ou ignorá-lo com base em critérios diferentes da correspondência de seus endereços MAC individuais: por exemplo, com base em uma lista configurável de endereços MAC multicast aceitos. Isso é chamado de endereçamento multicast.



Um quadro endereçado em Unicast é enviado apenas para a interface que leva a uma placa de rede específica. Conforme dito o bit IG do primeiro octeto de um endereço for definido como zero, o quadro deverá alcançar apenas uma placa de rede receptora.



O endereço multicast permite que a fonte envie um quadro para um grupo de dispositivos. O IEEE alocou o bloco de endereços 01-80-C2-xx-xx-xx (01-80-C2-00-00-00 a 01-80-C2-FF-FF-FF) para endereços de grupo, para uso por protocolos padrão.



Semelhante à Camada de Rede, o Broadcast também é possível na camada subjacente⁶⁹. Quadros Ethernet com 1s em todos os bits do endereço de destino (FF-FF-FF-FF-FF-FF) são chamados de endereços de broadcast. Quadros destinados com endereço MAC FF-FF-FF-FF-FF-FF chegarão a todos os computadores pertencentes a esse segmento de LAN.

Mas lembre-se que o endereço MAC Address está ligado ao hardware, na verdade a Subcamada MAC da camada de Data Link no modelo OSI, então ainda é físico. O comando ip com a opção link permite operar sobre este endereço, veja na figura abaixo.

```
usuario@debian:~$ ip link show enp0s3 1
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    link/ether 08:00:27:1c:31:77 brd ff:ff:ff:ff:ff:ff 3
    usuario@debian:~$ _
```

Onde,

⁶⁹ Baseado no Modelo OSI;

1. Comando ip com a opção link;
2. Nome da interface de rede;
3. Endereço de enlace e Broadcast;

Para alterar o endereço, basta executar um set, conforme exemplo.

```
1. sudo ip link set enp0s3 down
2. sudo ip link set enp0s3 address 64:1C:67:DE:A8:53
3. sudo ip link set enp0s3 up
```

Repare que para alterar o endereço, foi necessário desligar a interface. Sempre que um computador se comunica com outro, há uma troca natural de mensagens, algumas delas para execução da operação de troca de dados, já outras para obter informações de elementos.

9.11.4 Tradução de IP e MAC Address com ARP e RARP

Na camada de Network do modelo OSI temos um endereço de 32 bits chamado Endereço IP, e a ideia desse endereço é ter uma planificação uniforme de endereços não importando onde estejam tanto origem quanto destino. Mas localmente em uma LAN as mensagens são trocadas por MAC Address.

O protocolo de rede para traduzir endereço IP em endereço MAC Address é chamado de protocolo ARP enquanto o protocolo para traduzir endereço MAC Address em endereço IP é o protocolo RARP.

Umas das informações trocadas é o endereço ARP, pois para que a mensagem navegue pela rede local ela vai precisar deste endereço. O computador armazena uma lista de endereços e com isso podemos saber quais endereços já nos comunicamos. O comando **arp** manipula exatamente essa informação que vulgarmente chamamos de Cache ARP. Mas para isso utilizamos um endereço IP, ou seja, usamos o endereço de camada 3 do modelo OSI para obter um endereço de camada 2 do modelo OSI. A sintaxe é simples:

```
1. arp [-v] [-i if] [-H type] -a [hostname]
```

Uma boa opção é o uso de **verbose** e **all**, ou seja, **-va** como parâmetro. No exemplo a seguir, uma máquina virtual está ligada com uma interface de rede em modo NAT.

```
usuario@debian:~$ ip address show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    link/ether 08:00:27:1c:31:77 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 82895sec preferred_lft 82895sec
    inet6 fe80::a00:27ff:fe1c:3177/64 scope link
        valid_lft forever preferred_lft forever
usuario@debian:~$
```

Essa máquina até então se comunicou com 10.0.2.3 e 10.0.2.2, por isso o comando que exibe todo o cache **arp** mostra apenas estas entradas.

```
usuario@debian:~$ sudo arp -va
? (10.0.2.3) at 52:54:00:12:35:03 [ether] on enp0s3
? (10.0.2.2) at 52:54:00:12:35:02 [ether] on enp0s3
Entries: 2      Skipped: 0      Found: 2
usuario@debian:~$
```

Na coluna que aparece nas interrogações aparecerá o nome das máquinas somente se foi configurado o arquivo **/etc/hosts**, como não existe mais o motivo desta configuração o arp não sabe o nome da máquina e exibe interrogação.

Repare que temos uma coluna com 48 bits em formato hexadecimal, o que indica que é o endereço MAC Address de camada 2 do modelo OSI. Também encontramos o protocolo de camada 2, que neste caso é Ether de Ethernet 802, e que está acessível pela interface de rede **enp0s3**. Na verdade para a máquina, somente importa o endereço MAC e a interface de saída, é por estas informações que ele sabe realizar a comutação de portas para o quadro ethernet.

9.12 Realizando download com cURL ou WGET

Comumente é necessário o download de arquivos para o servidor ou testar algum serviço HTTP. No passado o wget era utilizado para fazer download das bibliotecas para se fazer a instalação manual, tudo manual. Pode-se baixar um arquivo de texto como um ASCII ou até mesmo arquivos binários com wget.

Os dois comandos (wget e cURL) são comandos úteis para isso, o wget é um comando que vai ser encontrado em todas as distribuições, realiza o download e retorna o HTML status code. Já o cURL é um comando sofisticado, que requer que se instale e naturalmente aumenta a superfície de ataque em um sistema operacional e tem muita aderência com programadores C++, pos cURL é na verdade uma biblioteca chamada **libcURL**⁷⁰.

Além de trabalhar com HTTP, também pode-se acessar os protocolos FTP, FTPS, LDAP, POP3, SFTP, SMB, SMTP entre outros. O cURL suporta certificado TLS e SSL e proporciona tanto um meio para se obter dados quanto enviar dados (conforme visto na lista de protocolos). Também possui uma fácil configuração de Proxy incluindo SOCKS4 e SOCKS5. Para exemplificar, no próximo exemplo será realizado uma requisição HTTP com método GET para se obter um arquivo .iso.

```
1. wget -O /tmp/projeto.iso http://www.aied.com.br/linux/download/output_image.iso
```

Veja o resultado da execução:

⁷⁰ Documentação oficial da biblioteca no link: <https://curl.se/libcurl/>

```

1  usuario@debian:~$ wget -O /tmp/projeto.iso http://www.aied.com.br/linux/download/output_image.iso
2  --2023-11-25 17:37:42-- http://www.aied.com.br/linux/download/output_image.iso
Resolving www.aied.com.br (www.aied.com.br)... 212.1.209.207, 2a02:4780:1:793:0:3848:1b21:2
Connecting to www.aied.com.br (www.aied.com.br)|212.1.209.207|:80... connected.
HTTP request sent, awaiting response... 200 OK
3  Length: 372736 (364K) [application/x-cd-image]
Saving to: '/tmp/projeto.iso'

/tmp/projeto.iso      100%[=====] 364.00K  570KB/s    in 0.6s
2023-11-25 17:37:43 (570 KB/s) - '/tmp/projeto.iso' saved [372736/372736]

```

Onde:

1. O parâmetro `-O` define o local que será salvo o arquivos, se já existir ele apaga o anterior e salva;
2. A URL, o `wget` então acessar essa URL e traz o arquivo;
3. A resposta HTTP com código⁷¹ 200, que significa sucesso.

O mesmo exemplo será realizado com o `cURL`.

```

1  usuario@debian:~$ 
2  usuario@debian:~$ 
3  usuario@debian:~$ curl -o /tmp/projeto.iso http://www.aied.com.br/linux/download/output_image.iso
  % Total    % Received % Xferd  Average Speed   Time   Time   Current
     0     0     0      0      0      0      0      0      0      0      0      0
  100  364k  100  364k    0     0  396k    0  --:--:-- --:--:-- --:--:--  396k
4  usuario@debian:~$ 
5  
```

Onde:

1. O parâmetro `-o` define o path do output, no qual será utilizado para salvar o arquivo;
2. A URL, o `cURL` então acessar essa URL e traz o arquivo;
3. Resposta do `cURL` com o tempo e tamanho do download.

O `cURL` precisa ser instalado em algumas distribuições, para isso caso precise fazer a instalação, com `apt` conforme listagem abaixo.

```
1. sudo apt install curl -l
```

No próximo exemplo vou demonstrar como usar o `cURL` no seu código C++, da mesma forma que foi feito no `wget` e no `cURL`, neste programa será realizado o download do mesmo arquivo e será salvo em `/tmp`. O header `curl/curl.h` está sendo importado para o uso do **CURL**, o output é salvo de forma binária, por isso é utilizado o parâmetro **"wb"**. Com o nano crie um arquivo chamado `getcurl.cpp` e codifique a listagem abaixo.

```

1. #include <stdio.h>
2. #include <curl/curl.h>
3.
4. int main( ) {
5.     CURL *curl;
6.     FILE *fp;
7.     CURLcode res;
8.     char *url = "http://www.aied.com.br/linux/download/output_image.iso";
9.     char outfilename[FILENAME_MAX] = "/tmp/output_image.iso";

```

⁷¹ Veja a lista completa de status code do protocolo HTTP na url:
https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

```

10. curl = curl_easy_init();
11. if ( curl ) {
12.     fp = fopen( outfile, "wb" );
13.     curl_easy_setopt(curl, CURLOPT_URL, url);
14.     curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, NULL);
15.     curl_easy_setopt(curl, CURLOPT_WRITEDATA, fp);
16.     res = curl_easy_perform(curl);
17.     curl_easy_cleanup(curl);
18.     fclose(fp);
19. }
20. return 0;
21. }
```

Para compilar será preciso adicionar a biblioteca `-lcurl`, conforme listagem abaixo.

```

1. g++ -o getcurl getcurl -lcurl
2. ./getcurl
3. ls -l /tmp | grep output
```

Onde,

1. Compilação, repare que foi adicionada a biblioteca `-lcurl`;
2. Execução do programa, ele fica parado por alguns segundos;
3. Listando o arquivo baixado.

Até o momento foi demonstrado como trazer dados ou arquivos até a máquina, no próximo exemplo será demonstrado como enviar dados para um servidor, dá para utilizar tal solução para integrar algo no servidor Linux com alguma API privada por meio de JSON.

No exemplo abaixo não será utilizada a biblioteca `jsoncpp`, será enviado com cURL e por método POST um json simples para uma página PHP chamada `echo`. A página recebe o json e retorna. Crie um novo arquivo chamado **postjson.cpp** e codifique conforme listagem abaixo.

```

1. #include <stdio.h>
2. #include <curl/curl.h>
3. #include <string>
4.
5. int main(void)
6. {
7.     CURLcode ret;
8.     CURL *hnd;
9.     struct curl_slist *slist1;
10.    std::string jsonstr = "{\"username\":\"aied\", \"password\":\"123456\"}";
11.
12.    slist1 = NULL;
13.    slist1 = curl_slist_append(slist1, "Content-Type: application/json");
14.
15.    curl = curl_easy_init();
16.    if( curl ) {
17.        curl_easy_setopt(curl, CURLOPT_URL,
18.                        "http://www.aied.com.br/linux/download/echo.php");
19.        curl_easy_setopt(curl, CURLOPT_NOPROGRESS, 1L);
20.        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, jsonstr.c_str());
```

```

20.     curl_easy_setopt(hnd, CURLOPT_USERAGENT, "curl/7.38.0");
21.     curl_easy_setopt(hnd, CURLOPT_HTTPHEADER, slist1);
22.     curl_easy_setopt(hnd, CURLOPT_MAXREDIRS, 50L);
23.     curl_easy_setopt(hnd, CURLOPT_CUSTOMREQUEST, "POST");
24.     curl_easy_setopt(hnd, CURLOPT_TCP_KEEPALIVE, 1L);
25.
26.     ret = curl_easy_perform(hnd);
27.
28.     curl_easy_cleanup(hnd);
29.     hnd = NULL;
30.     curl_slist_free_all(slist1);
31.     slist1 = NULL;
32. }
33. }
```

Grifei na listagem acima o POST, eu sempre achei estranha a forma de input de parâmetros no cURL, mas devo admitir que é uma biblioteca fantástica quanto a sua funcionalidade.

```

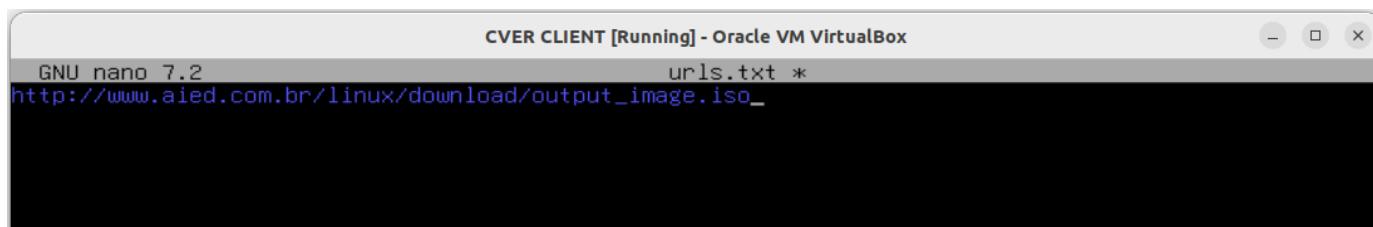
1. g++ -o postjson postjson.cpp -lcurl
2. ./postjson
```

Quando executar o programa (linha 2) será impresso no output um json de retorno, você agora está apto a enviar dados de seu servidor para alguma API.

O próximo exemplo é um exemplo da pesada, na verdade é um exemplo de como realizar download de enormes arquivos até em uma rede instável. Sempre que o link falhar será reiniciado o download do ponto em que parou. Inicie o procedimento criando um diretório chamado Downloads, conforme comando abaixo.

```
1. mkdir /home/userlinux/Downloads
```

Dentro do arquivo downloads com o nano crie um arquivo chamado urls.txt, crie com o nano e já escreva no arquivo a url: http://www.aied.com.br/linux/download/output_image.iso conforme figura abaixo. Neste arquivo você pode digitar quantas urls quiser digitar, o mecanismo vai baixar todas.



Crie o arquivo download.sh dentro do diretório /home/**userlinux**/Downloads/ e codifique o código da listagem abaixo.

```

1. #!/bin/bash
2. cd /home/userlinux/Downloads
3. while true
4. do
```

Na linha 5, foi grifado um trecho do comando curl, digite esse parâmetro -x somente se você tem TOR instalado na máquina e quer ficar anônimo, caso contrário, pode remover esse trecho. Dê permissão de execução e execute o script, conforme listagem abaixo.

```
1. chmod +x /home/userlinux/download.sh  
2. /home/userlinux/download.sh
```

O download irá iniciar, como o arquivo é pequeno será muito rápido, na imagem abaixo é o download de um outro arquivo muito grande, usando a rede TOR. Veja que sempre que reinicia o download (devido ao While True) ele resume o download de onde parou.

```
% Total % Received % Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
100 1971k 100 1971k 0 0 10132 0 0:03:19 0:03:19 ----- 19257
% Total % Received % Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
0 495G 0 10.9M 0 0 28561 0 215d 11h 0:06:41 215d 10h 33238
curl: (18) transfer closed with 531675075837 bytes remaining to read
** Resuming transfer from byte position 2018589
% Total % Received % Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
100 190 100 190 0 0 42 0 0:00:04 0:00:04 ----- 42
** Resuming transfer from byte position 11460108
% Total % Received % Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
0 495G 0 11.7M 0 0 38236 0 160d 22h 0:05:22 160d 22h 23524
```

9.13 Práticas do capítulo

Nesta sessão o aluno deve realizar a prática, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para resolver, isto já é levado em consideração.

ATENÇÃO: Antes de rodar o comando AIED teste se tem conectividade, com comando: ping -c 1 8.8.8.8

9.13.1 Prática 0002 checkpoint03: Configurando o arquivo de Interfaces

Nesta prática o aluno deve configurar uma máquina Virtual com um adaptador em modo NAT com IP estático em sua rede.

1. Configure o Adaptador de Rede para modo NAT;
 2. Ligue a Virtual Machine;
 3. Edite o arquivo **/etc/network/interfaces**:
 - a. Endereço: 10.0.2.3;
 - b. Máscara: 24 bits;
 - c. Gateway: 10.0.2.2;

- d. DNS server: 8.8.8.8;
- 4. **Reinicie** a máquina virtual;

Valide a prática com o comando:

```
1. sudo aied validar 0002 checkpoint03
```

A saída do output será:

```
usuario@debian:~$ sudo aied validar 0002 checkpoint03
Ação que será executada: validar

Prática: Prática de lab redes ip address em arquivo /etc/network/interfaces
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: ip route
Será enviado o OUTPUT do comando: ping -c 1 8.8.8.8
Será enviado o OUTPUT do comando: ping -c 1 terra.com.br
Será enviado o OUTPUT do comando: cat /etc/network/interfaces

Deseja continuar? (s para SIM) s
s1 - Comando: ip address
  1.1  Validar por text          10.0.2.3
  1.2  Validar por regex        /brd s+10.0.2.255/
  2 - Comando: ip route
  2.1  Validar por regex        /default s+via s+10.0.2.2 s+dev/
  3 - Comando: ping -c 1 8.8.8.8
  3.1  Validar por regex        / s+0% s+packet s+loss/
  4 - Comando: ping -c 1 terra.com.br
  4.1  Validar por regex        / s+0% s+packet s+loss/
  5 - Comando: cat /etc/network/interfaces
  5.1  Validar por regex        /auto(.*) w+/
  5.2  Validar por regex        /iface(.*)inet s+static/
  5.3  Validar por regex        /address(.*)10.0.2.3/
  5.4  Validar por regex        /broadcast(.*)10.0.2.255/
  5.5  Validar por regex        /netmask(.*)255.255.255.0/
  5.6  Validar por regex        /gateway(.*)10.0.2.2/

Total de acertos: 11 do total de 11 validações equivalente a 100%.
Identificação: b9e0eec03a
AIED v(8)
```

9.13.2 Prática 0002 checkpoint04: Configurando por comandos

Nesta prática o aluno deve configurar uma máquina Virtual com um adaptador em modo NAT com IP estático em sua rede.

1. Configure o Adaptador de Rede para modo NAT;
2. Ligue a Virtual Machine;
3. Apague os campos abaixo da interface enp0s3:
 - a. address
 - b. netmask
 - c. network
 - d. broadcast
 - e. gateway
 - f. dns-nameservers
4. Configure a interface enp0s3 para usar auto dhcp;
5. Reinicie a máquina virtual;
6. Configure por comando IP:
 - a. Endereço: 10.0.2.3;
 - b. Máscara: 24 bits;
 - c. Gateway: 10.0.2.2;
 - d. DNS server: 8.8.8.8;

```
1. sudo aied validar 0002 checkpoint04
```

O output do validador será:

```

Prática: Pratica de lab redes ip address
Será enviado o OUTPUT do comando: ping -c 1 terra.com.br
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: ip route
Será enviado o OUTPUT do comando: ping -c 1 8.8.8.8
Será enviado o OUTPUT do comando: cat /etc/network/interfaces

Deseja continuar? (s para SIM) s
s1 - Comando: ping -c 1 terra.com.br
1.1   Validar por regex      / s+0% s+packet s+loss/
2 - Comando: ip address
2.1   Validar por regex      /inet s+ d+. d+. d+. d+ / d+/
2.2   Validar por regex      /brd s+ d+. d+. d+. 255/
3 - Comando: ip route
3.1   Validar por regex      /default s+via s+ d+. d+. d+. d+ s+dev/
4 - Comando: ping -c 1 8.8.8.8
4.1   Validar por regex      / s+0% s+packet s+loss/
5 - Comando: cat /etc/network/interfaces
5.1   Validar por regex      /auto s+enp0s3/
5.2   Validar por regex      /iface s+enp0s3 s+inet s+dhcp/
5.3   Validar por text       address
5.4   Validar por text       gateway
5.5   Validar por text       netmask
5.6   Validar por text       broadcast

Total de acertos: 11 do total de 11 validações equivalente a 100%.
Identificação: b9e0eec03a
AIED v(8)

```

9.13.3 Prática 0002 checkpoint05: Baixando arquivos com wget

Nesta prática o aluno deve baixar um arquivo com wget e salvar em /tmp

1. Faça download do arquivo <http://www.aied.com.br/linux/download/install.py> com wget, salve o arquivo com o nome /tmp/install.py

```
1. sudo aied validar 0002 checkpoint05
```

O output do validador será:

```

Prática: Pratica de Linux
Será enviado o OUTPUT do comando: ping -c 1 terra.com.br
Será enviado o OUTPUT do comando: cat /tmp/install.py

Deseja continuar? (s para SIM) s
s1 - Comando: ping -c 1 terra.com.br
1.1   Validar por regex      / s+0% s+packet s+loss/
2 - Comando: cat /tmp/install.py
2.1   Validar por text       aied.com.br

Total de acertos: 2 do total de 2 validações equivalente a 100%.
Identificação: b9e0eec03a
AIED v(8)

```

10 Instalação de Programas e pacotes

No GNU/Linux o usuário encontra uma ampla gama de soluções de software para seu cotidiano, então um usuário do Debian pode instalar e naturalmente customizar sua distribuição para seu nicho de atuação, ou para suas necessidades específicas, de editores de texto a poderosas ferramentas de gestão de redes o GNU/Linux é aderente.

Este ambiente versátil possibilita que desenvolvedores distribuam suas soluções de diferente formas, são estas:

- Disponibilizar um simples binário que pode ser executado de qualquer ponto ([ver este script](#));
- Disponibilizar o código fonte para ser compilado na máquina do cliente, e pode até disponibilizar um script de instalação pronto;
- Disponibilizar um conjunto de pacotes .deb;
- Participar de um repositório de distribuições oficial ou ter seu próprio repositório online para instalação e atualização;

Neste capítulo será descrito do processo de codificação a distribuição de artefatos que podem ser executados ou incorporados como bibliotecas. Também será demonstrado como realizar a instalação de softwares.

10.1 O processo de desenvolvimento

O processo de desenvolvimento é complexo, o que podemos dizer é que nasce de uma ideia e há um processo de engenharia de software antes de sua codificação, estou sendo simplista. Após a codificação, o aplicativo passa por um processo complexo de homologação e então é realizado o deploy e a distribuição. Em um livro comum apenas aprenderia a instalar um artefato pronto, mas neste livro terá que fazer seu próprio software para que aprenda a instalar, é por isso que quem termina este livro está apto a assumir qualquer cargo na TI de qualquer grande empresa.

Mas o processo de compilação não é trivial, é complexo. Quando a empresa desenvolve uma solução muito grande, com muitos artefatos de código o trabalho é árduo, a compilação do Debian por exemplo é muito complexa, pois depende de inúmeros artefatos devidamente linkados, isso é o que chamamos de dependência.

Já falamos sobre bibliotecas, você já programou em C++, sabe que há dependências principalmente em compilados dinamicamente e linkados. Agora imagine uma solução tão complexa quanto um Apache, Mysql ou até mesmo o Debian.

10.2 Comando make

O comando make no GNU/Linux é um dos comandos mais usados pelos administradores de sistema e programadores. Embora ajude os administradores a compilar e instalar muitos utilitários de código aberto por meio da linha de comando, os programadores o usam para gerenciar a compilação de seus projetos grandes e complicados.

O comando make aceita como argumentos de linha de comando o alvo que está descrito no arquivo de configuração chamado Makefile, neste arquivo de configuração encontramos comandos de execução para o alvo definido na linha de comando.

Quando o comando make é executado pela primeira vez, ele verifica o Makefile para encontrar o destino alvo e então lê suas dependências. Se essas dependências forem os próprios destinos, ele verifica o Makefile para esses destinos e constrói suas dependências e, em seguida, as constrói. Uma vez que as dependências principais são construídas, ele constrói o destino principal. Vamos a um exemplo, mas antes, faça a instalação dos pacotes build-essential e g++ conforme listagem abaixo:

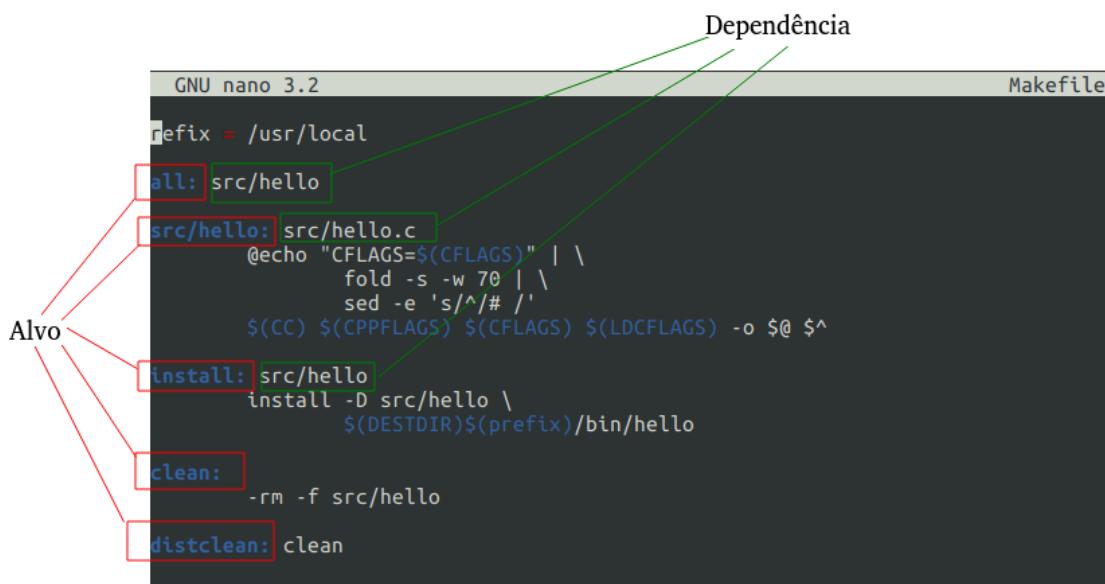
```
1. sudo apt install g++ -y
2. sudo apt install gcc -y
3. sudo apt install build-essential -y
```

10.2.1 Arquivo Makefile

O comando make por padrão consome um arquivo de configuração chamado Makefile que está localizado no mesmo diretório do projeto, é comum navegar o cursor do sistema de arquivos até a raiz do projeto onde se encontra o arquivo Makefile. Cada alvo no arquivo Makefile pode ter associado dependências que são outros alvos ou artefatos, e cada alvo possui um ou mais comandos.

```
alvo: dependência
      comando
      comando
      comando
```

Veja o exemplo:



<esta imagem acima tem um erro, está faltando um p de prefix na primeira linha>

O alvo **install** depende do alvo **src/hello** e o alvo **src/hello** depende do arquivo **src/hello.c**, caso na linha de comando seja invocado o comando **make** passando o alvo **install** (ver linha abaixo), existindo o arquivo **src/hello.c** será executado o alvo **src/hello** e só após o **src/hello** sendo executado que o alvo **install** será executado.

```
1. make install
```

Para exercitar será demonstrado o exemplo clássico de **debian.org**, trata-se de um projeto com um hello básico em **printf**, crie um diretório **debhello-0.2** no diretório do usuário corrente⁷², e dentro deste diretório crie um outro diretório chamado **src**, veja na figura abaixo em azul os diretórios descritos.

```
usuario@debian:~$ tree /home/
/home/
└── usuario
    └── debhello-0.2
        ├── Makefile
        └── src
            └── hello.c
3 directories, 2 files
usuario@debian:~$ _
```

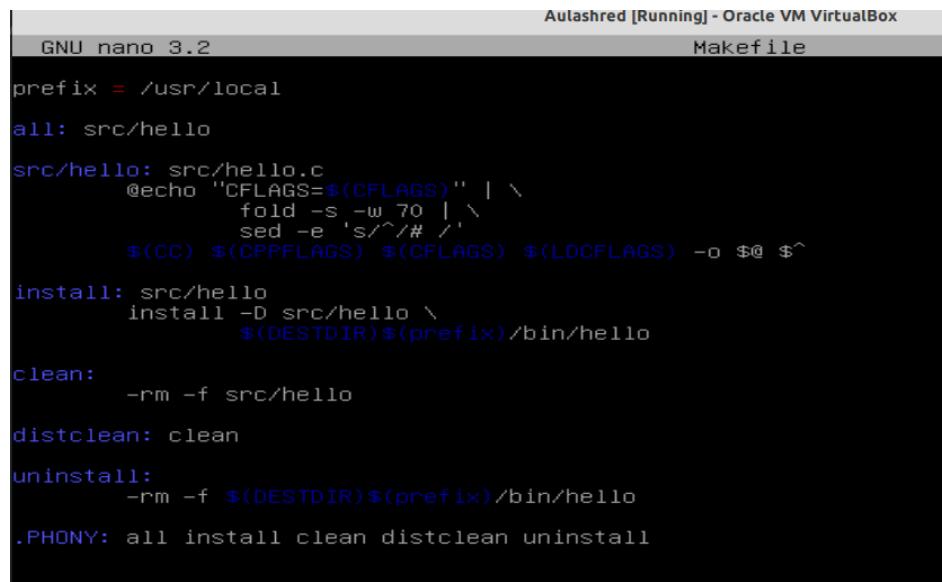
Com o nano crie o arquivo **/home/usuario/debhhello-0.2/src/hello.c** e digite o código abaixo:

```
GNU nano 3.2                               /home/usuario/debhhello-0.2/src/hello.c
#include <stdio.h>

int main() {
    printf("Hello, visit https://www.youtube.com/user/aiedonline\n");
    return 0;
}
```

Trata-se de um **printf** escrevendo um texto padrão, veja que **main** retorna um inteiro e por isso o **return 0**. Agora salve o arquivo, o programa está pronto mas para usar o **make** precisamos criar um outro arquivo chamado **Makefile** conforme código abaixo.

⁷² Assume-se que o leitor esteja utilizando uma distribuição configurada conforme Capítulo 1.



```

GNU nano 3.2
Makefile

prefix = /usr/local

all: src/hello

src/hello: src/hello.c
    @echo "CFLAGS=$(CFLAGS)" | \
        fold -s -w 70 | \
        sed -e 's/^/# /'
    $(CC) $(CPPFLAGS) $(CFLAGS) $(LDFLAGS) -o $@ $^

install: src/hello
    install -D src/hello \
        $(DESTDIR)$(prefix)/bin/hello

clean:
    -rm -f src/hello

distclean: clean

uninstall:
    -rm -f $(DESTDIR)$(prefix)/bin/hello

.PHONY: all install clean distclean uninstall

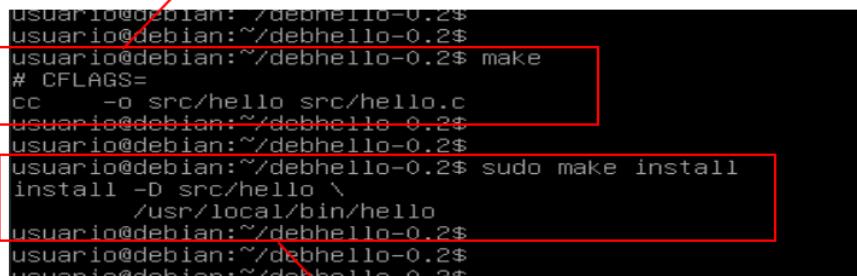
```

CFLAGS permite que switches sejam adicionados para o compilador de linguagem C, enquanto que **CPPFLAGS** deve ser usada para trabalhar com switches ao invocar um compilador de linguagem C++. Estas variáveis estão implícitas no make e podem ser consultadas no projeto oficial neste tópico: [Variables Used by Implicit Rules](#)

\$(CC) é uma variável que possui o compilador C/C++, ou seja, é nesta linha é invocado o compilador informando as variáveis CFLAGS E CPPFLAGS bem como o caminho do projeto. O alvo install tem como comando único em seu bloco de código o comando install, isso, o alvo se chama install e o comando utilizado dentro é install também (o que pode confundir um pouco o leitor, mas mantém-se este padrão).

Este projeto é básico e demonstra uma estrutura mínima de uma distribuição de uma solução (produto), é comum que uma empresa ou grupo entregue o projeto neste ponto para que os usuários instalem usando o make, veja que é simples fazer uma instalação make pois tanto os arquivos de código bem como o arquivo Makefile já são escritos pela empresa/grupo que disponibiliza o produto. O usuário apenas precisa executar o **make** e depois o **make install**.

Executando o comando make



```

usuario@debian:~/debhello-0.2$ make
# CFLAGS=
cc -o src/hello src/hello.c
usuario@debian:~/debhello-0.2$ 
usuario@debian:~/debhello-0.2$ sudo make install
install -D src/hello \
        /usr/local/bin/hello
usuario@debian:~/debhello-0.2$ 
usuario@debian:~/debhello-0.2$ 

```

Executando com sudo o comando make e passando como parâmetro o alvo install definido no arquivo Makefile

Executa-se o make antes do make install para que se compile o projeto e naturalmente se valide as dependências, quando invocamos o make automaticamente será carregado o alvo all no arquivo Makefile.

Veja na imagem do Makefile demonstrado neste documento o alvo all apenas faz a compilação, isso é uma prática saudável pois antes de iniciar o bloco de instalação (usando make install) vamos ter certeza que foi possível compilar. Quando o make foi invocado (ver figura acima) nenhum erro foi liberado e um arquivo binário compilado foi gerado (em verde na figura abaixo), sucesso, só após isso que se executa o make install para fazer a instalação do binário recém compilado.

```
usuario@debian:~$ tree
.
└── debhello-0.2
    ├── Makefile
    └── src
        └── hello
            └── hello.c

2 directories, 3 files
usuario@debian:~$ _
```

O comando install que foi descrito no alvo install do arquivo Makefile copia arquivos (geralmente apenas compilados) em locais de destino que foram definidos. Se você deseja baixar e instalar um pacote pronto para usar em um sistema GNU/Linux, você deve usar um gerenciador de pacotes apt ou wget.

```
usuario@debian:~$ hello
Hello, visit https://www.youtube.com/user/aiedonline
usuario@debian:~$
```

Este procedimento de executar o comando make sobre as fontes de uma solução é uma forma de instalação e é a que gera um produto mais aderente ao computador pois trata-se de uma compilação justa para as características da máquina, usando assim 100% do potencial da mesma, mas naturalmente não é uma boa opção para o fabricante que pretende manter seu produto.

10.3 Instalação de baixo nível

No Debian GNU/Linux a instalação de baixo nível é realizada pelo utilitário dpkg, o dpkg possui várias utilidades, de configuração por exemplo a instalação;

- **dpkg-reconfigure locales** para reconfigurar Locales, fundamental para o funcionamento do ambiente.
- **dpkg**, invocado com vários argumentos de linha de comando. Se você encontrar um pacote para o debian que não está acessível através das ferramentas de alto nível, você pode baixar os pacotes utilizando o wget ou o curl e instalá-los diretamente com o dpkg. Note que este método não resolverá automaticamente as dependências entre pacotes etc, portanto, use uma ferramenta de alto nível preferencialmente;
- **dpkg --list-files PACOTE** para listar todos os arquivos fornecidos pelo pacote PACOTE. Isso só funciona para pacotes que já estão instalados. Isso pode ser útil para descobrir em que forma a documentação de um pacote foi fornecida;

- **dpkg -S /path/to/some/file** para descobrir a qual pacote um arquivo pertence;

10.3.1 Criando um pacote .deb com dpkg-deb

Utilizamos o dpkg para fazer a instalação de pacotes .deb no Debian, este é o seu uso extensivo, os desenvolvedores de produtos para Debian utilizam o dpkg-deb para gerar os arquivos .deb, é um procedimento relativamente simples. Para este exemplo será utilizado o hello.c, compile usando o gcc ou g++ criando o arquivo binário hello. Crie um diretório chamado pacotes no diretório do usuário, conforme listagem abaixo.

```
1. cd ~
2. mkdir pacotes
3. mkdir pacotes/hello_0_0_amd64
4. mkdir pacotes/hello_0_0_amd64/DEBIAN
5. mkdir pacotes/hello_0_0_amd64/usr
6. mkdir pacotes/hello_0_0_amd64/usr/bin
```

Crie um arquivo chamado **control** em **DEBIAN** utilizando o nano, este arquivo possui características que serão usadas pelo dpkg -i para realizar a instalação, edite o arquivo conforme imagem abaixo.

```
GNU nano 3.2          /home/usuario/pacotes/hello_0_0_amd64/DEBIAN/control

Package: Hello
Version: 0.0
Architecture: all
Essential: no
Priority: optional
Maintainer: Aied
Description: Um hello
```

Onde ⁷³,

Package é o nome do pacote, que se chamará Hello;

Version é a versão

Architecture é a arquitetura, é uma extensa lista que pode ser obtida com o comando **dpkg-architecture -L**, neste exemplo vou utilizar all que indica que pode ser utilizada em qualquer arquitetura;

Essential: Se definido como yes, o sistema de gerenciamento de pacotes se recusará a remover o pacote;

Priority: Este campo representa a importância de o usuário ter o pacote instalado, são valores válidos:

required: Pacotes que são necessários para o funcionamento adequado do sistema;

important: Programas importantes, incluindo aqueles que se esperaria encontrar em qualquer sistema semelhante ao Unix;

standard: Esses pacotes fornecem um sistema de modo de caractere razoavelmente pequeno, mas não muito limitado;

optional: Esta é a prioridade padrão para a maioria do arquivo;

⁷³ Todas as possibilidades neste tópico:

<https://www.debian.org/doc/debian-policy/ch-controlfields.html#s-f-description>

Maintainer: O nome e endereço de e-mail do mantenedor do pacote;
Description: Descrição do produto.

O próximo arquivo que será editado será executado antes da instalação do cliente, use o nano informando o caminho conforme figura abaixo.

```
GNU nano 3.2                               /home/usuario/pacotes/hello_0_0_amd64/DEBIAN/preinst
#!/bin/bash
if [ -f "/usr/bin/hello" ]; then
    sudo rm -f /usr/bin/hello
    echo "removido"
fi
```

Neste script caso já exista uma instalação do hello em /usr/bin o script vai providenciar sua exclusão, evitando assim um erro de instalação. Cuidado, respeite os espaços e as quebras de linha, nos próximos capítulos a explicação sobre scripts bash será dada.

Após criar este arquivo altere a permissão de acesso para 775 conforme comando abaixo.

```
1. chmod 775 /home/userlinux/pacotes/hello_0_0_amd64/DEBIAN/preinst
```

Com o comando cp, copie o **hello** compilado binário para dentro de /home/**userlinux**/pacotes/hello_0_0_amd64/usr/bin conforme figura abaixo.

```
usuario@debian:~/pacotes$ pwd
/home/usuario/pacotes
usuario@debian:~/pacotes$ usuario@debian:~/pacotes$ tree
.
└── hello_0_0_amd64
    ├── DEBIAN
    │   ├── control
    │   └── preinst
    └── usr
        └── bin
            └── hello
4 directories, 3 files
usuario@debian:~/pacotes$
```

Os arquivos estão prontos para serem empacotados, para isso será usado o dpkg-deb conforme listagem abaixo.

```
1. cd /home/userlinux/pacotes/
2. dpkg-deb --build hello_0_0_amd64/
```

Veja a saída do comando na figura abaixo.

```
usuario@debian:~/pacotes$ usuario@debian:~/pacotes$ dpkg-deb --build hello_0_0_amd64/
dpkg-deb: building package 'hello' in 'hello_0_0_amd64.deb'.
usuario@debian:~/pacotes$ usuario@debian:~/pacotes$
```

Será gerado um arquivo .deb conforme figura abaixo.

```
usuario@debian:~/pacotes$ tree
.
+-- hello_0_0_amd64
|   +-- DEBIAN
|   |   +-- control
|   |   +-- preinst
|   +-- usr
|       +-- bin
|           +-- hello
|           +-- hello_0_0_amd64.deb
.
4 directories, 4 files
usuario@debian:~/pacotes$ _
```

Agora você pode distribuir o arquivo `/home/userlinux/pacotes/hello_0_0_amd64.deb`

10.3.2 Instalando um pacote com dpkg -i

O comando dpkg atua em instalações de baixo nível, já vimos como utilizar o utilitário dpkg-deb para gerar um pacote .deb, agora vamos usar o dpkg para instalar este pacote. Este procedimento de instalação é semelhante para todas as instalações com dpkg. Execute o comando da listagem abaixo, neste exemplo o pacote será hello_0_0_amd64:

1. `sudo dpkg -i hello_0_0_amd64.deb`

O dpkg irá:

- Desempacotar o arquivo .deb;
- Executa o script hello_0_0_amd64/DEBIAN/preinst (caso exista)
- Instalará o conteúdo, que neste exemplo é apenas um arquivo hello
- Executa o script hello_0_0_amd64/DEBIAN/postinst (caso exista)

```
usuario@debian:~$ ls
debs hello_0_0_amd64.deb pacotes
usuario@debian:~$ sudo dpkg -i hello_0_0_amd64.deb
[sudo] password for usuario:
(Reading database ... 49496 files and directories currently installed.)
Preparing to unpack hello_0_0_amd64.deb ...
removido
Unpacking hello (0.0) over (0.0) ...
Setting up hello (0.0) ...
usuario@debian:~$
```

Agora execute o comando hello que está em /usr/bin, conforme imagem abaixo.

```
usuario@debian:~$ file /usr/bin/hello
/usr/bin/hello: ELF 64-bit LSB pie executable, x86-64, vers
eter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, Buil
b1e59f3de, not stripped
usuario@debian:~$ 
usuario@debian:~$ 
usuario@debian:~$ hello
Hello, visit https://www.youtube.com/user/aiedonline
usuario@debian:~$
```

10.3.3 Obtendo informações de pacotes instalados

Para se obter informações do pacote, antes da instalação utiliza-se o parâmetro `--info` conforme exemplo.

```
usuario@debian:~$ ls
debhello-0.2 hello_0_0_amd64.deb pacotes
usuario@debian:~$ 
usuario@debian:~$ dpkg --info hello_0_0_amd64.deb
new Debian package, version 2.0.
size 2840 bytes: control archive=396 bytes.
  118 bytes,    7 lines  * control
  101 bytes,    8 lines  * preinst          #!/bin/bash
Package: Hello
Version: 0.0
Architecture: all
Essential: no
Priority: optional
Maintainer: Aied
Description: Um hello
usuario@debian:~$
```

Para cada pacote instalado é criado um diretório em **/var/lib/dpkg/info/** contendo alguns arquivos, na figura abaixo existem 3 arquivos apenas, mas este número pode ser maior.

```
usuario@debian:~$ ls -l /var/lib/dpkg/info/hello*
-rw-r--r-- 1 root root 32 Apr 29 22:50 /var/lib/dpkg/info/hello.list
-rw-r--r-- 1 root root 48 Apr 29 22:50 /var/lib/dpkg/info/hello.md5sums
-rwxr-xr-x 1 root root 101 Apr 29 21:06 /var/lib/dpkg/info/hello.preinst
usuario@debian:~$
```

O arquivo **.md5sums** contém o MD5 do arquivo original de instalação, ou seja, é possível validar a integridade do arquivo pois executáveis podem ser deliberadamente modificados por terceiros. O arquivo **.list** possui uma lista de diretórios e arquivos instalados, já o arquivo **.preinst** é o arquivo editado pelo responsável pelo pacote, ver tópico anterior. Para listar todos os pacotes instalados utilize o **dpkg-query** conforme listagem abaixo.

1. sudo dpkg-query -l

Uma imensa lista será exibida, mas caso queira filtrar utilize o **grep**, conforme exemplo abaixo.

1. sudo dpkg-query -l | grep hello

```
usuario@debian:~$ sudo dpkg-query -l | grep hello
[sudo] password for usuario:
ii  hello                                0.0          all      Um hello
usuario@debian:~$
```

10.3.4 Removendo pacotes com dpkg

Com o **dpkg** para remover o pacote utilize a opção **-r** seguido do nome do pacote que deseja desinstalar.

- Remova um pacote (mas não seus arquivos de configuração): **dpkg --remove hello**
- Remova um pacote (incluindo seus arquivos de configuração): **dpkg --purge hello**

1. sudo dpkg --purge hello

10.4.5 Log de operações do dpkg

Todas as operações sobre os pacotes realizadas pelos usuários bem como as operações automáticas sobre os pacotes são registradas no arquivo **/var/log/dpkg.log**, é um arquivo

contínuo em forma de lista que está em constante edição, pode usar um CAT ou o TAIL para o consultar. Na figura abaixo temos um recorte deste arquivo.

Operações realizadas no pacote hello

```

2021-04-30 00:51:25 startup archives install
2021-04-30 00:51:25 upgrade hello:all 0.0 0.0
2021-04-30 00:51:25 status half-configured hello:all 0.0
2021-04-30 00:51:25 status unpacked hello:all 0.0
2021-04-30 00:51:25 status half-installed hello:all 0.0
2021-04-30 00:51:25 status unpacked hello:all 0.0
2021-04-30 00:51:25 configure hello:all 0.0 0.0
2021-04-30 00:51:25 status half-configured hello:all 0.0
2021-04-30 00:51:25 status installed hello:all 0.0
2021-04-30 10:29:03 startup packages remove
2021-04-30 10:29:03 status installed hello:all 0.0
2021-04-30 10:29:03 remove hello:all 0.0 <none>
2021-04-30 10:29:03 status half-configured hello:all 0.0
2021-04-30 10:29:03 status half-installed hello:all 0.0
2021-04-30 10:29:03 status config-files hello:all 0.0
2021-04-30 10:29:03 status not-installed hello:all <none>

```

10.4.6 Procurar nos metadados do pacote

Embora hoje em dia visite o site Debian <https://packages.debian.org/> facilite a busca nos meta-dados dos pacotes, vamos ver modos mais tradicionais.

Os comandos grep-dctrl, grep-status e grep-available podem ser utilizados para procurar qualquer arquivo que tenha o formato geral de um arquivo de **control** de pacotes .deb. O comando **dpkg -S NOME_DO_PACOTE.DEB** pode ser utilizado para procurar nomes de pacotes instalados pelo dpkg que contenham arquivos com o nome coincidente.

Se necessitar de fazer uma busca mais elaborada nos meta-dados do dpkg, necessita executar o comando **grep -e padrão_de_expressão_regular *** no diretório **/var/lib/dpkg/info/**, isto deverá procurar as palavras mencionadas nos scripts dos pacotes e nos textos de questões de instalação.

10.4 Instalação de alto nível

O Debian GNU/Linux é uma organização voluntária que constrói distribuições consistentes de pacotes binários pré-compilados de software livre e distribui-os a partir de repositórios de arquivos.

Conforme já visto o comando dpkg é um comando para instalação de baixo nível, é considerado de baixo nível pois o dpkg não resolve problemas de dependências o que pode gerar um grande transtorno para os administradores, na prática os administradores optam por instalações de alto nível com algum recurso que resolve dependência e garante a atualização segura dos pacotes, são estes apt, aptitude e apt-get/apt-cache.

O comando apt vem sendo mais utilizado em documentos, trata-se de uma interface de linha de comandos de alto nível para gestão de pacotes. É basicamente um invólucro dos apt-get, apt-cache e comandos, originalmente destinada a ser um comando para o usuário final.

Sempre que iniciar um processo de instalação execute o **comando apt** com parâmetro update conforme listagem abaixo.

```
1. sudo apt update -y
```

Aqui estão algumas operações básicas de gestão de pacotes com a linha de comandos a usar apt:

apt update atualiza os metadados do arquivo de pacotes;
apt install XXXXX instala a versão candidata do pacote "xxxx" com as suas dependências;
apt upgrade instala as versões candidatas dos pacotes instalados sem remover quaisquer outros pacotes;
apt full-upgrade instala as versões candidatas dos pacotes instalados a remover outros pacotes caso necessário;
apt remove xxxx remove o pacote "xxxx" a deixar os seus ficheiros de configuração;
apt autoremove remove os pacotes auto instaláveis que já não sejam necessários;
apt purge xxxx purga o pacote "xxxx" com os seus ficheiros de configuração;
apt clean limpa completamente o repositório local de ficheiros de pacotes obtidos;
apt autoclean limpa os pacotes desatualizados do repositório local dos ficheiros de pacotes recebidos;
apt show xxxx mostra informação detalhada acerca do pacote "xxxx";
apt search <regex> procura pacotes que correspondem à <expressão-regular>;

10.4.1 Arquivo de repositórios sources.list

O **comando apt** baixa pacotes de um ou mais repositórios de software e os instala em seu computador. Um repositório é geralmente um servidor de rede com um serviço HTTP ou FTP configurado, mas CD e DVD também são aceitos. A lista destes serviços está definida em um arquivo chamado **sources.list** que está em **/etc/apt/**, trata-se de um arquivo de linhas corridas com uma entrada (linha) para cada serviço.

```
usuario@debian:~$ sudo cat /etc/apt/sources.list
[sudo] password for usuario:
#
# deb cdrom:[Debian GNU/Linux 10.6.0 _Buster_ - Official i386 NETINST 20200926-11:50]/ buster main
#deb cdrom:[Debian GNU/Linux 10.6.0 _Buster_ - Official i386 NETINST 20200926-11:50]/ buster main
deb http://deb.debian.org/debian/ buster main
deb-src http://deb.debian.org/debian/ buster main

deb http://security.debian.org/debian-security buster/updates main
deb-src http://security.debian.org/debian-security buster/updates main

# buster-updates, previously known as 'volatile'
deb http://deb.debian.org/debian/ buster-updates main
deb-src http://deb.debian.org/debian/ buster-updates main
```

Os repositórios específicos (fontes de pacote) configurados em sua máquina afetam:

- Quais pacotes de software estão disponíveis para download;
- Quais versões de pacotes estão disponíveis;
- Quem empacota o software;

Fontes de pacotes comumente usados são:

- DebianStable**: repositório Debian oficial para a versão atual;
- StableProposedUpdates**: repositório Debian oficial para os próximos lançamentos;
- StableUpdates**: repositório Debian oficial para mudanças que não podem esperar pelo próximo lançamento pontual;
- DebianSecurity**: repositório Debian oficial para atualizações de segurança;
- DebianBackports**: versões mais recentes de alguns pacotes, compatíveis com DebianStable;
- DebianTesting**: estado de desenvolvimento atual da próxima distribuição Debian estável;
- DebianUnstable**: versão de desenvolvimento contínuo contendo os pacotes mais recentes;
- DebianExperimental**: versão de desenvolvimento contendo os pacotes experimentais;

As entradas neste arquivo normalmente seguem este formato:

- ```
1. deb http://site.example.com/debian distribuição componente1 componente2 componente3
2. deb-src http://site.example.com/debian distribuição componente1 componente2
```

**Tipo de arquivo**: A primeira palavra em cada linha, deb ou deb-src, indica o tipo de arquivo. Deb indica que o arquivo contém pacotes binários (.deb), os pacotes pré-compilados que normalmente usamos. Deb-src indica pacotes fonte, que são os originais do programa fontes mais o arquivo de controle Debian (.dsc) e o diff.gz contendo as modificações necessárias para o empacotamento do programa.

**URL do repositório**: A próxima entrada na linha é um URL para o repositório do qual você deseja baixar os pacotes. A lista principal de mirrors do repositório Debian está localizada neste link: <https://www.debian.org/mirror/list>.

**Distribuição**: A 'distribuição' pode ser o nome ou alias do código de lançamento (jessie, stretch, buster, sid) ou a classe de lançamento (oldstable, stable, testing, unstable) respectivamente. Se você pretende rastrear uma classe de lançamento, use o nome da classe; se quiser rastrear um lançamento pontual do Debian, use o nome do código.

#### **Componente**:

main consiste em pacotes compatíveis com DFSG, que não dependem de software fora desta área para operar. Estes são os únicos pacotes considerados parte da distribuição Debian.

Os pacotes contrib contêm software compatível com DFSG, mas não possuem dependências no principal (possivelmente empacotados para o Debian em não-livre).

non-free contém software que não está em conformidade com a DFSG.

#### 10.4.2 Adicionando PPA com add-apt-repository

O Personal Package Archive (PPA) permite que desenvolvedores de aplicativos para Debian GNU/Linux possam empacotar seus produtos em seus próprios repositórios e simplificando o processo de atualização e distribuição.

| Name                                  | Last modified    | Size |
|---------------------------------------|------------------|------|
| Parent Directory                      | -                | -    |
| <a href="#">README</a>                | 2023-12-10 17:10 | 1.2K |
| <a href="#">README.CD-manufacture</a> | 2010-06-26 09:52 | 1.3K |
| <a href="#">README.html</a>           | 2023-12-10 17:10 | 2.8K |
| <a href="#">README.mirrors.html</a>   | 2017-03-04 20:08 | 291  |
| <a href="#">README.mirrors.txt</a>    | 2017-03-04 20:08 | 86   |
| <a href="#">dists/</a>                | 2023-12-10 17:11 | -    |
| <a href="#">doc/</a>                  | 2023-12-26 19:56 | -    |
| <a href="#">extrafiles</a>            | 2023-12-26 20:28 | 211K |
| <a href="#">indices/</a>              | 2023-12-26 20:27 | -    |
| <a href="#">ls-lR.gz</a>              | 2023-12-26 20:21 | 15M  |
| <a href="#">pool/</a>                 | 2022-10-05 17:09 | -    |
| <a href="#">project/</a>              | 2008-11-17 23:05 | -    |
| <a href="#">tools/</a>                | 2012-10-10 16:29 | -    |
| <a href="#">zzz-dists/</a>            | 2023-10-07 11:07 | -    |

**Debian Archive**

See <https://www.debian.org/> for information about Debian GNU/Linux.

Repare, no GNU/Linux temos uma infinidade de soluções e seria muito complexo a centralização de tudo que se produz em um repositório oficial, então esta distribuição de responsabilidade reduz este peso central. Mas traz problemas, imagine que um hacker mal intencionado pode criar e distribuir um repositório com pacotes contaminados. Fica a cargo da equipe de TI pesquisar sobre a veracidade dos PPAs.

Um repositório é uma coleção de arquivos que tem informações sobre vários softwares, suas versões e alguns outros detalhes como a soma de verificação. Toda a árvore de dependência está lá descrita em arquivos. Também encontramos nestes repositórios arquivos binários e scripts.

O computador do usuário então acessa este repositório, pois possui informações de como chegar até estes repositórios. Essas informações são armazenadas no arquivo **sources.list** no diretório **/etc/apt** conforme já visto.

Sempre que utiliza o **comando apt**, tal como **apt update**, seu sistema usa a ferramenta APT para verificar o repositório e armazena as informações sobre o software e sua versão em um cache. Quando você usa o comando **sudo apt install package\_name**, ele usa as informações para obter esse pacote na URL em que o software real está armazenado.

O comando **add-apt-repository** é um script Python que permite adicionar um repositório a lista de repositórios **/etc/apt/sources.list** ou para um arquivo separado no caso de ter um específico em **/etc/apt/sources.list.d/**. O comando também pode ser usado para remover um repositório já existente, caso necessário.

Se o **add-apt-repository** não está disponível no seu sistema receberá uma mensagem de erro a dizer "add-apt-repository command not found", então para instalar use os comandos abaixo:

```
1. sudo apt update -y
2. sudo apt install software-properties-common -y
```

A sintaxe básica do add-apt-repository comando é o seguinte:

```
1. add-apt-repository [options] repository
```

Onde:

**repository** pode ser um repositório que pode ser adicionada ao sources.list exemplo: deb http://repo.tld/ubuntu distro component ou um repositório PPA no formato ppa:<user>/<ppa-name>. Veja um exemplo para instalação do Mariadb.

```
1. sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el]
https://mariadb.mirror.liquidtelecom.com/repo/10.6/ubuntu focal main'
```

Para ver todas as opções disponíveis do add-apt-repository tipo de comando man add-apt-repository no seu terminal.

#### 10.4.3 Listando todos os pacotes de um repositório

Para listar os pacotes em um repositório, primeiro o administrador deve localizar todos os arquivos que o nome terminam com **\_Packages** e que estão no diretório **/var/lib/apt/lists/**, um simples **ls /var/lib/apt/lists/\*\_Packages** pode resolver isso.

```
usuario@debian:~$ ls /var/lib/apt/lists/*_Packages
/var/lib/apt/lists/deb.debian.org_debian_dists_buster_main_binary-amd64_Packages
/var/lib/apt/lists/deb.debian.org_debian_dists_buster-updates_main_binary-amd64_Packages
/var/lib/apt/lists/security.debian.org_debian-security_dists_buster_updates_main_binary-amd64_Packages
usuario@debian:~$
```

Como exemplo será utilizado o arquivo:

**deb.debian.org\_debian\_dists\_buster-updates\_main\_binary-amd64\_Packages**

Este arquivo possui uma estrutura onde cada pacote está descrito em sequência, semelhante a uma coleção de arquivos **control**.

```
1. cat
/var/lib/apt/lists/deb.debian.org_debian_dists_buster-updates_main_binary-amd64_Packages
```

O arquivo em geral é muito grande, cada pacote começa com uma entrada chamada **Package**, conforme visto na figura abaixo.

```
Section: net
Priority: optional
Filename: pool/main/p/pagekite/pagekite_0.5.9.3-2+deb10u1_all.deb
Size: 138448
SHA256: baadfb6639efe2e01c6d2f06ef7a8d36544134505eb9caf5fcf2666f0dc0c8c

Package: tzdata
Version: 2021a-0+deb10u1
Installed-Size: 3040
Maintainer: GNU Libc Maintainers <debian-glibc@lists.debian.org>
Architecture: all
Replaces: libc0.1, libc0.3, libc6, libc6.1
Provides: tzdata-buster
```

Com o GREP pode-se filtrar os nomes dos pacotes, conforme exemplo.

```
2. cat
/var/lib/apt/lists/deb.debian.org_debian_dists_buster-updates_main_binary-amd64_Pac
kages | grep Package:
```

Pronto, agora se tem uma lista de pacotes disponíveis neste repositório.

```
Package: nx-x11proto-damage-dev
Package: nx-x11proto-randr-dev
Package: nx-x11proto-render-dev
Package: nx-x11proto-scrnsaver-dev
Package: nx-x11proto-xext-dev
Package: nx-x11proto-xfixes-dev
Package: nx-x11proto-xinerama-dev
Package: nxagent
Package: nxproxy
Package: libafsauthent2
Package: libafsrpc2
Package: libkopenafs2
Package: libopenafs-dev
```

O comando apt com o parâmetro search localiza pacotes, é um processo simples, no exemplo abaixo será realizada a busca por pacotes com o termo **mariadb**.

```
well@usb:~$ apt search mariadb
Sorting... Done
Full Text Search... Done
apophenia-bin/jammy 1.0+ds-8build2 amd64
 Apophenia Statistical C Library -- binary package
apophenia-doc/jammy,jammy 1.0+ds-8build2 all
 Apophenia Statistical C Library -- reference manual
dbconfig-common/jammy,jammy 2.0.21 all
 framework that helps packages to manage databases
dbconfig-mysql/jammy,jammy 2.0.21 all
 dbconfig-common MySQL/MariaDB support
mariadb-server/jammy-updates,jammy-updates,jammy-security
12-0ubuntu0.22.04.1 all
 MariaDB database server (metapackage depending on the l
mariadb-server-10.6/jammy-updates,jammy-security 1:10.6.1
4 all
 MariaDB database server binaries
mariadb-server-core-10.6/jammy-updates,jammy-security 1:1
amd64
 MariaDB database core server files
mariadb-test/jammy-updates,jammy-security 1:10.6.12-0ubun
 MariaDB database regression test suite
```

Quando localizar o nome do pacote, pode então solicitar mais informações sobre o mesmo, neste caso será usado como exemplo o pacote mariadb-server.

```
well@usb:~$ apt info mariadb-server
Package: mariadb-server
Version: 1:10.6.12-0ubuntu0.22.04.1
Priority: optional
Section: universe/database
Source: mariadb-10.6
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Debian MySQL Maintainers <pkg-mysql-maint@lists.alioth.debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 69,6 kB
Depends: mariadb-server-10.6 (>= 1:10.6.12-0ubuntu0.22.04.1)
Homepage: https://mariadb.org/
Download-Size: 11,8 kB
APT-Sources: http://br.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Pa
ckages
Description: MariaDB database server (metapackage depending on the latest versio
n)
 This is an empty package that depends on the current "best" version of
 mariadb-server (currently mariadb-server-10.6), as determined by the MariaDB
 maintainers. Install this package if in doubt about which MariaDB
 version you need. That will install the version recommended by the
 package maintainers.
 .
 MariaDB is a fast, stable and true multi-user, multi-threaded SQL database
 server. SQL (Structured Query Language) is the most popular database query
 language in the world. The main goals of MariaDB are speed, robustness and
 ease of use.
N: There is 1 additional record. Please use the '-a' switch to see it
```

#### 10.4.4 Instalando e reinstalando um pacote com APT

Para adicionar um pacote, deve-se localizar o nome correto do pacote, por exemplo o pacote `net-tools`<sup>74</sup>, para isso utilize o site: <https://packages.debian.org/pt-br/sid/> A instalação é simples, basta usar o parâmetro `install` seguido do nome do pacote. Temos as seguintes sintaxes:

1. `sudo apt install PACOTE -y`
2. `sudo apt install PACOTE --no-upgrade`
3. `sudo apt install PACOTE --only-upgrade`
4. `sudo apt install PACOTE=NUMERO_VERSAO`
5. `sudo apt reinstall PACOTE -y`

Onde,

- y** aceita tudo, sem questionamento;
- no-upgrade** não realizar o upgrade do pacote;
- only-upgrade** realizar o upgrade do pacote;
- pacote=numero\_versao** instala uma versão específica do pacote.

Vamos instalar o pacote `net-tools`, como exemplo:

1. `sudo apt install net-tools -y`

O próximo comando não é tão simples de ser decorado, até recomendo que não faça mas saiba que ele existe. Digamos que há muita alteração e que requer muitos pacotes de dependência, o código abaixo irá instalar tudo.

1. `sudo apt reinstall PACOTE $(apt-cache depends --recurse --installed PACOTE || grep '[ ]')`

#### 10.4.5 Atualizando pacotes e realizando upgrade com APT

O primeiro procedimento que se faz em um GNU/Linux é atualizar os pacotes, no Debian e provenientes do Debian se utiliza o comando `apt` com o parâmetro `update` conforme demonstrado na listagem abaixo.

1. `sudo apt update -y`

A saída do comando é semelhante à imagem abaixo.

```
usuario@debian:~$ sudo apt update
[sudo] password for usuario:
Get:1 http://security.debian.org/debian-security buster/updates InRelease [65.4 kB]
Get:2 http://security.debian.org/debian-security buster/updates/main Sources [183 kB]
Hit:3 http://deb.debian.org/debian buster InRelease
Get:4 http://deb.debian.org/debian buster-updates InRelease [51.9 kB]
Get:5 http://security.debian.org/debian-security buster/updates/main amd64 Packages [285 kB]
Get:6 http://security.debian.org/debian-security buster/updates/main Translation-en [147 kB]
Fetched 732 kB in 1s (735 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
usuario@debian:~$
```

<sup>74</sup> Url: <https://packages.debian.org/pt-br/sid/net-tools>

O administrador deve percorrer a lista de atualizações e se certificar que todos os repositórios foram acessados e que obteve sucesso no update. **Esta operação também é recomendada antes de qualquer install.**

Diferente do Microsoft Windows, o GNU/Linux é atualizado e nada para de funcionar, então atualizar um servidor Linux é uma tarefa agradável. É comum que eu faça um script para ser executado 2 ou 3 vezes por dia atualizando. Mas às vezes as versões ficam muito depreciadas, algo em torno de 10 meses. Então recomenda-se analisar a lista de pacotes que serão alterados com um upgrade.

```
1. sudo apt list --upgradable
2. sudo dpkg --get-selections | grep -v deinstall > ~/tmp/packages.txt
3. sudo apt upgrade
4. sudo apt dist-upgrade
```

A linha 1 do exemplo acima, retorna uma lista de pacotes que serão trocados/atualizados caso seja realizado um upgrade do sistema, isso é importante pois um upgrade pode ser traumático embora que há mais de 10 anos não enfrentei nenhuma situação de reverter um sistema após um upgrade, provavelmente devido a introdução do systemd. Em um processo de atualização muitos pacotes podem ser marcados para desinstalação, na linha 2 temos um exemplo de comando que cria um txt com todos os pacotes que serão desinstalados.

```
Get:10 http://deb.debian.org/debian bookworm-updates/main InRelease [1,111 B]
Get:17 http://deb.debian.org/debian bookworm-updates/main Sources T-2023-11-06-2008.27-F-2023-11-06-2008.27.pdiff [1,393 B]
Get:17 http://deb.debian.org/debian bookworm-updates/main Sources T-2023-11-06-2008.27-F-2023-11-06-2008.27.pdiff [1,393 B]
Get:18 http://deb.debian.org/debian bookworm-updates/main amd64 Packages T-2023-11-06-2008.27-F-2023-11-06-2008.27.pdiff [475 B]
Get:18 http://deb.debian.org/debian bookworm-updates/main amd64 Packages T-2023-11-06-2008.27-F-2023-11-06-2008.27.pdiff [475 B]
Get:19 http://deb.debian.org/debian bookworm-updates/main Translation-en T-2023-11-06-2008.27-F-2023-11-06-2008.27.pdiff [320 B]
Get:19 http://deb.debian.org/debian bookworm-updates/main Translation-en T-2023-11-06-2008.27-F-2023-11-06-2008.27.pdiff [320 B]
Fetched 24.9 MB in 4s (7,068 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
57 packages can be upgraded. Run 'apt list --upgradable' to see them.
N: Repository 'http://deb.debian.org/debian bookworm InRelease' changed its 'Version' value from '12.1' to '12.2'
usuario@debian:~$ sudo apt upgrade -y
```

Na terceira linha temos um upgrade da versão corrente, se estou em um Debian 12 irá fazer upgrade dos pacotes e me manterá no Debian 12, já na linha 4 temos um upgrade geral, se tiver uma nova versão, ou seja, um Debian 13 então iremos atualizar para a versão mais recente.

#### 10.4.7 Removendo pacotes com AP

Para remover o pacote bem como todos os seus arquivos de configuração do seu sistema, deve-se passar como parâmetro purge e o nome do pacote.

```
1. sudo apt purge net-tools
```

A opção PURGE irá apagar tudo, apaga os binários e os arquivos de configuração gerados pelo programa, caso queira apagar somente os binários, utilize REMOVE.

#### 10.4.8 Limpando cache do comando APT

Toda a atualização e instalação de pacotes é armazenado no computador, mesmo que não tenha sido feito e é comum que o computador fique cheio de arquivos cache. Esses arquivos cache podem dar problemas em uma atualização, então recomenda-se remover caso tenha problemas com o comando APT. Segue comandos que devem executar para isso:

```
1. sudo apt clean
2. sudo apt autoclean
```

### 10.5 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para resolver, isto já é levado em consideração.

#### 10.5.1 Prática 092900 01: Criando e instalando um pacote

Nesta prática o aluno deverá criar seu próprio pacote .deb e instalar no GNU/Linux utilizando dpkg, este pacote terá um programa escrito em C que retorna uma mensagem.

1. Criar em /home/userlinux/ o diretório **pacotes**;
2. Criar em /home/userlinux/pacotes/ o diretório **aluno\_0\_0\_amd64**;
3. Criar em /home/userlinux/pacotes/aluno\_0\_0\_amd64/ os diretórios **DEBIAN** e **usr/bin/**;
4. Escreva um arquivo em linguagem C (/home/userlinux/aluno.c) que escreve com printf a seguinte mensagem no output: “**sou o aluno XXXXXXX**” (atenção, se é aluno de algum curso presencial, utilize seu Registro Acadêmico no lugar dos X);
5. Compile com
6. /home/userlinux/pacotes/aluno\_0\_0\_amd64/usr/bin/ (exemplo: **gcc /home/userlinux/aluno.c -o /home/userlinux/pacotes/aluno\_0\_0\_amd64/usr/bin/aluno**)
7. Crie o arquivo **control** (ver figura abaixo) e o **preinst** no diretório **DEBIAN**;
8. Execute o comando chmod dando a permissão **775** ao arquivo **preinst**;
9. Execute o comando dpkg-deb e crie seu novo pacote .deb;
10. Execute o comando dpkg e instale o seu novo pacote .deb;

Dados que eu quero no arquivo control:

```
GNU nano 3.2 DEBIAN/control
Package: Aluno
Version: 0.0
Architecture: all
Essential: no
Priority: optional
Maintainer: Ated
Description: Pratica do aluno
```

Depois de instalar execute o comando aied para validar a prática, passando como parâmetro o código da prática e o checkpoint01 conforme listagem abaixo.

1. sudo aied validar 092900 checkpoint01

O resultado será próximo ao da imagem abaixo pois o mecanismo de validação está sempre em evolução.

```
usuario@debian:~/pacotes$ sudo aied validar 092900 checkpoint01
Ação que será executada: validar

Prática: Prática da aula de instalação de pacotes - O próprio pacote do aluno
Será enviado o OUTPUT do comando: aluno
Será enviado o OUTPUT do comando: dpkg -l | grep aluno

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: aluno
1.1 Validar por text sou o aluno

2 - Comando: dpkg -l | grep aluno
2.1 Validar por text aluno
2.2 Validar por text 0.0
2.3 Validar por text all

Total de acertos: 4 do total de 4 validações equivalente a 100%.
Identificação: ce224acacc
AIED v(7)
```

### 10.5.2 Prática 092900 02: Removendo um pacote com dpkg

Nesta prática o aluno deverá remover seu próprio pacote .deb utilizando dpkg.

1. Remova o pacote aluno completamente;

Agora execute o comando aied para validar a prática, passando como parâmetro o código da prática e o checkpoint02 conforme listagem abaixo.

1. sudo aied validar 092900 checkpoint02

O resultado será próximo ao da imagem abaixo pois o mecanismo de validação está sempre em evolução.

```
usuario@debian:/tmp$ sudo aied validar 092900 checkpoint02
Ação que será executada: validar

Prática: Prática da aula de instalação de pacotes - remover o pacote Aluno
Será enviado o OUTPUT do comando: dpkg -l | grep aluno

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: dpkg -l | grep aluno
1.1 Validar por text aluno
1.2 Validar por text 0.0
1.3 Validar por text all

Total de acertos: 3 do total de 3 validações equivalente a 100%.
Identificação: ce224acacc
AIED v(7)
```

#### 10.5.4 Prática 092900 03: Instalando o pacote net-tools com apt

Nesta prática o aluno deverá utilizar o comando apt para instalar o pacote net-tools.

1. Utilizando o comando apt, instale o pacote chamado **net-tools**;

Agora execute o comando aied para validar a prática, passando como parâmetro o código da prática e o checkpoint03 conforme listagem abaixo.

```
1. sudo aied validar 092900 checkpoint03
```

O resultado será próximo ao da imagem abaixo pois o mecanismo de validação está sempre em evolução.

```
usuario@debian:/tmp$ sudo aied validar 092900 checkpoint03
Ação que será executada: validar

Prática: Prática da aula de instalação de pacotes - Adicionando net-tools com apt
Será enviado o OUTPUT do comando: dpkg -l | grep net-tools

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: dpkg -l | grep net-tools
1.1 Validação por texto net-tools

Total de acertos: 1 do total de 1 validações equivalente a 100%.
Identificação: ce224acacc
AIED v(7)
```

## 11 Shell Script e Bash

No GNU/Linux existe uma grande adesão a scripts de execução que são interpretados, é uma prática tão comum que inúmeros interpretadores e inúmeras linguagens são encontradas neste ecossistemas de scripts. São interpretadores encontrados:

**sh** Conhecido como **Borne Shell** e é o shell original, no Debian é um link simbólico para dash;

**dash** Debian Almquist Shell, o Debian usa o dash como seu shell não interativo padrão. A falta de recursos interativos o torna aproximadamente **4x** mais rápido que o bash;

**csh, tcsh** São derivados bem conhecidos e amplamente usados do shell Borne;

**bash** O **Shell Borne Again** é o shell mais popular usado para linux e desenvolvido pelo projeto GNU;

**python** Implementado como uma ferramenta de desenvolvimento de sistemas se adaptou na área de Sistemas Operacionais, presente em todas as distribuições;

**perl** Originalmente implementado para atuar sobre arquivos de texto assim como o Python passou a ser utilizado como ferramenta para interpretar scripts com objetivo de administração de sistemas Linux;

**awk\*** Script que é executado como um comando e naturalmente é escrito em uma única linha;

**Atenção:** São considerados **SHELL SCRIPT** apenas sh, dash, bash, csh e tcsh na listagem acima, as demais são apenas scripts. Neste livro serão utilizados os interpretadores clássicos dash, sh e bash (que mantém uma mesma sintaxe) e o Python para ser mais aderente a uma linguagem moderna.

### 11.2 Como criar um script no Linux

A vantagem de se usar scripts até em operações básicas é que pode-se testar um roteiro de manutenção em um ambiente de homologação e também tira um pouco do ser humano do momento da manutenção em produção. O primeiro passo é compreender o problema e a solução, embora trivial é observado que inúmeros administradores abrem um editor de script antes de pensar nisso. O segundo passo é localizar um diretório para armazenar seu script.

Não existe um diretório específico com um **X** vermelho dizendo “salve aqui seu script”, não, o que existe é a perfeita organização citada no capítulo de Sistema de Arquivos que até então o aluno já estudou. Mas os principais são:

**/bin** e **/usr/bin** Diretórios reservados para scripts e binários do sistema, não se adicionam nada aqui que não seja para o sistema como um todo, este diretório sempre está no PATH do sistema;

**/usr/local/bin** e **/usr/local/sbin** Diretório reservado para programas e scripts do usuários, este diretório está no PATH do sistema;

**/opt** Diretório de customização e instalações de extensões;

**/etc/cron.d** Diretório de scripts agendados no cron, cron é um sistema de agendamento de execuções clássico;

**/etc/** Diretório de instalação de programas mais complexos, e está cheio de diretórios de programas cheio de binários e scripts.

**~/.local/bin** ;

### 11.2.1 Variável de ambiente PATH

A variável de ambiente PATH é uma variável do sistema que é utilizado pelos programas para acessarem um rol de programas ou scripts sem o uso de todo o caminho dos mesmos, ou seja, em vez de usar no terminal **/bin/cp** usamos simplesmente **cp** pois este comando está em um determinado diretório mapeando em PATH.

```
usuario@debian:~$ echo $PATH;
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
usuario@debian:~$
```

A montagem da variável PATH pode ser feita em vários pontos do sistema, pode-se ter parte do PATH genérico para todos definido em **/etc/environment**, **/etc/profile** e outra parte específica por usuário em **~/.bashrc**.

## 11.3 O Interpretador /bin/bash

Bash é um shell Unix(terminal) que interpreta uma linguagem de comando escrita por Brian Fox para o Projeto GNU como um substituto do shell Bourne. Lançado pela primeira vez em 1989, e foi usado como o shell de login padrão para a maioria das distribuições GNU/Linux.

Bash é um interpretador de comando que normalmente é executado em uma janela de texto onde o usuário digita comandos e os programas são executados realizando as ações. O Bash também pode ler e executar comandos de um arquivo, chamado de script.

**ATENÇÃO:** Uma falha de segurança no Bash datada da versão 1.03 (agosto de 1989), apelidada de Shellshock, foi descoberta no início de setembro de 2014 e rapidamente levou a uma série de ataques pela Internet.

## 11.4 Como criar um script

O cabeçalho define que tipo de script e quem o escreveu, que versão é e quais suposições ou opções de shell o Bash usa.

A primeira linha de um script é a linha do cabeçalho, esta linha começa com **#!** na parte superior do script. Essa combinação de caracteres identifica o tipo de script e naturalmente qual o interpretador será utilizado pelo GNU/Linux.

Se for escrever um script em python, geralmente se utiliza (cada GNU/Linux o python pode estar instalado em um determinado diretório) **/bin/python**, caso precise de uma versão específica pode-se utilizar:

|                       |                                              |
|-----------------------|----------------------------------------------|
| <b>/bin/python</b>    | A versão corrente do python;                 |
| <b>/bin/python2</b>   | A subversão corrente do Python versão 2;     |
| <b>/bin/python2.7</b> | Uma subversão específica do Python versão 2; |

Exemplo de uso do Python versão 2:

```
1. #!/usr/bin/python2
2. # nome do script
3. # para que serve o script
4.
```

As linhas que começam com **#** são comentários e são notas para o leitor e não afetam a execução de um script. Os comentários devem ser sempre informativos, descrevendo o que o script está tentando realizar, não precisa descrever comando por comando, mas preferencialmente trechos, em um ambiente de negócios, comentários claros e informativos ajudam a solucionar e depurar problemas obscuros.

Caso seja um script Bash clássico, basta informar `/bin/bash` na primeira linha conforme listagem abaixo.

```
1. #!/bin/bash
2. # nome do script
3. # para que serve o script
4.
5.
```

Esta forma simples permite que inúmeros interpretadores possam ser utilizadas mas naturalmente que o script deve manter a sintaxe do interpretador adequado, não adianta escrever um script em Python e invocar o `/bin/bash`.

Um script pode ser editado em qualquer editor de texto ASCII ou UTF-8, porém os dois mais recomendados são o vim e o nano, e o processo de edição é trivial. Após salvar o script e tendo certeza que está tudo correto, então deve-se invocar o script, existe 2 formas de se fazer isso:

**Forma 1:** Dar poder de execução para o arquivo e simplesmente o invocar logo em seguida informando o caminho do arquivo;

**Forma 2:** Invocar o interpretador (quem vai ser executado é o interpretador) pelo nome e passar como parâmetro o caminho do script que acabou de escrever;

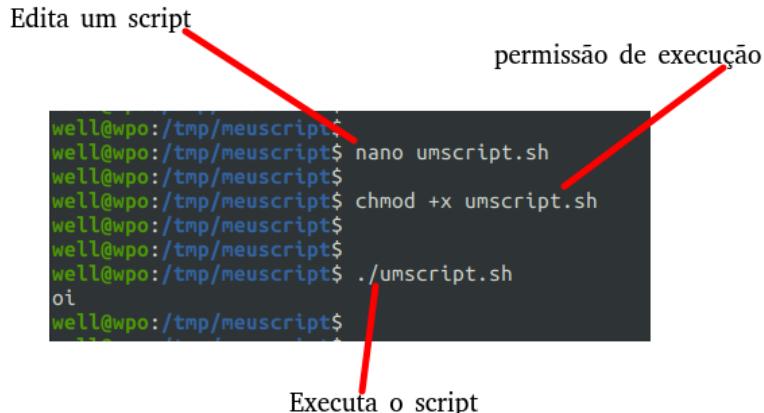
Veja o exemplo de script:

```
1. #!/bin/bash
2. # umscript.sh
3. # Apenas um exemplo para demonstração
4.
5. echo "oi"
6. exit 0
```

Para exemplificar<sup>75</sup> a execução da forma 1 veja a figura abaixo.

---

<sup>75</sup> Estou usando **chmod +x** ao invés de **chmod u+x** para que fique mais fácil para o aluno;



Por simplificação a permissão de execução foi dada para todos, quando o script é executado na linha de comando, nesta Forma 1 a primeira linha será lida e o interpretador definido dentro do script deverá (ver linha 1 do script acima).

Para exemplificar a Forma 2 vamos criar um segundo script com o mesmo código da listagem acima, para executar o interpretador e passando como parâmetro o caminho do script não será necessário dar permissão de execução para o script pois quem está sendo executado é o interpretador.

```

1. # doiscript.sh
2. # Apenas um exemplo para demonstração
3.
4. echo "oi"
5. exit 0
```

Escreve o script

```

well@wpo:/tmp/meuscript$ nano doiscript.sh
well@wpo:/tmp/meuscript$ /bin/bash ./doiscript.sh
oi
well@wpo:/tmp/meuscript$
```

Executa o interpretador

Neste segundo caso, a linha 1 da listagem acima será ignorada e o interpretador definido na execução do shell será utilizado.

## 11.5 Shell Script BASH /bin/bash

Por convenção, os scripts de shell Bash têm nomes que terminam com .sh. A Listagem abaixo mostra um script chamado ola.sh, um script muito curto que grava uma mensagem na tela de saída.

```

1. # não vou usar aqui o interpretador pois vou demonstrar como invocar o script
 usando o interpretador na linha de comando!!!!
2. # ola.sh
3. # Este script apenas escrever uma mensagem na saída
4.
```

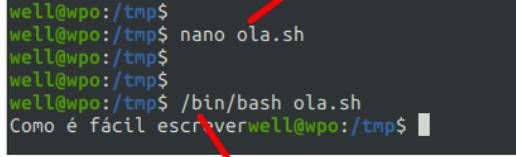
```

5. printf "Como é fácil escrever\n"
6. #echo "Como é fácil escrever"
7.
8. exit 0;

```

A saída do comando é mostrada na imagem abaixo.

Edito o script da  
listagem acima



Executo o script com  
/bin/bash

O comando `exit`, seguido por um zero, informa ao programa que está executando o script de shell que o script foi executado com êxito, qualquer número diferente de zero é um erro definido por quem escreveu o script e é uma forma de localizar o problema em scripts que estão em produção.

### 11.5.1 Declarações Globais

Todas as declarações que se aplicam à totalidade do script devem ocorrer na parte superior do script, abaixo do cabeçalho. Ao colocar as declarações globais em um só lugar, você torna mais fácil para alguém se referir a elas enquanto lê o script.

`declare [-a] [-f] [-g] [-F] [-x] [-p] [-i] [name[=value]] [name[=value]] ...`

- p** É usado para exibir as opções e atributos de cada nome de variável se for usado com argumentos de nome;
- f** Cada nome é tratado como o nome da função, não como uma variável. Portanto, você pode dizer que esta opção restringe a ação ou exibição a nomes e definições de funções.
- F** Ao mostrar informações sobre uma função, mostra apenas o nome e os atributos da função. Portanto, o conteúdo da função não será exibido.
- g** Faz com que todas as operações em variáveis tenham efeito no escopo global.
- i** Força uma variável inteira;
- x keyseq** shell-command Faz com que o comando shell seja executado sempre que keyseq for inserido;
- X** Liste todas as sequências de teclas vinculadas aos comandos do shell e os comandos associados em um formato que pode ser reutilizado como entrada.

Exemplo:

```

1.#!/bin/bash
2.
3. # Global Declarations
4. declare SCRIPT=${0##*/}

```

```
5. declare wc=/usr/bin/wc
6.
```

### 11.5.2 Parando um Script

O comando `logout`, que encerra uma sessão de login interativa, não pode ser usado para interromper um script. Em vez disso, o Bash fornece dois comandos integrados para encerrar um script. Como visto anteriormente, o comando `exit` interrompe incondicionalmente um script. A saída pode incluir um código de status para retornar ao chamador do script. Um código de status de 0 indica nenhum erro. Se o código de status for omitido, o status do último comando executado pelo script é retornado pois é sempre melhor fornecer um status de saída. Já o comando `quit()` finaliza a execução do script saindo.

Um script para automaticamente quando chega ao seu final, como se houvesse uma saída implícita digitada lá, mas o status de saída é o status do último comando executado.

O comando `suspend` também interrompe incondicionalmente um script, no entanto, ao contrário de `exit`, a execução é suspensa até que o script seja sinalizado para despertar e retomar a próxima instrução após o comando `suspend`.

```
1. suspend
```

Também existe um utilitário Linux chamado `sleep`. `Sleep` suspende o script por um número específico de segundos, após o qual ele é ativado e retoma a próxima instrução após o comando `sleep`.

```
1. sleep 5
```

O `sleep` é útil para colocar pausas no script, permitindo que o usuário leia o que foi exibido até agora. A suspensão não é adequada para sincronizar eventos, no entanto, porque quanto tempo um determinado programa é executado no computador muitas vezes depende da carga do sistema, número de usuários, atualizações de hardware e outros fatores fora do controle do script.

### 11.5.3 Descritores Input, Output e Error no Linux

Cada processo no Linux é fornecido com três arquivos abertos (geralmente chamados de descriptor de arquivos). Esses arquivos são os arquivos de entrada (input), saída (output) e erros padrão (err).

**Descriptor input:** nos processos em terminal o teclado é abstraído como um arquivo para facilitar a escrita de scripts de shell;

**Descriptor output:** é a janela do shell ou o terminal a partir do qual o script é executado, abstraído como um arquivo para tornar a escrita de scripts e programas mais fácil;

**Descriptor err:** é igual à saída padrão: a janela do shell ou terminal a partir do qual o script é executado.

Um descritor de arquivo é simplesmente um número que se refere a um arquivo aberto. Por padrão, o descritor de arquivo **0 (zero) se refere ao input (stdin)** e geralmente abreviado como stdin. O descritor de arquivo **1 se refere ao output (stdout)** e o descritor de arquivo **2 se refere ao err (stderr)**.

Esses números são importantes quando você precisa acessar um arquivo específico, especialmente quando deseja redirecionar esses arquivos para outros locais.

#### 11.5.3.1 Redirecionando o output

Sintaxe para redirecionar a saída de um comando para um arquivo.

```
1. comando > /tmp/output_file
```

#### 11.5.3.2 Redirecionando o input

Sintaxe para redirecionar a entrada de um comando para vir de um arquivo.

```
1. comando < /tmp/input_file
```

Use o operador < para redirecionar a entrada de um comando, o exemplo é mostrado abaixo:

```
1. wc -l < umarquivo.txt
```

Neste exemplo, a entrada para o comando 'wc' vem do arquivo denominado umarquivo.txt. O shell envia o conteúdo do arquivo como uma entrada padrão para o comando wc.

Pode-se combinar os dois redirecionamentos com a seguinte sintaxe:

```
1. comando < /tmp/input_file > /tmp/output_file
```

#### 11.5.3.3 Redirecionando error

Além de redirecionar a I/O padrão para um script ou comando, também podemos redirecionar os erros do comando. Existem boas razões pelas quais stdout e stderr são tratados separadamente. O principal motivo é que podemos redirecionar a saída de um comando ou comandos para um arquivo, mas você não tem como saber se ocorreu um erro. Separar stderr de stdout permite que a mensagem de erro apareça em sua tela enquanto a saída ainda vai para um arquivo.

Sintaxe para redirecionar stderr de um comando para um arquivo.

```
1. comando 2> /tmp/error_file
```

O 2 em 2> refere-se ao descritor de arquivo 2, o número do descritor para stderr.

#### 11.5.3.4 Redirecionando o output e o error

Use a sintaxe `2>&1` para redirecionar o erro padrão para o mesmo local da saída padrão.

```
1. ls /usr/bin > /tmp/output_file 2>&1
```

Onde,

`ls /usr/bin` é o comando executado

`> command.txt` redireciona a saída do comando `ls`

`2>&1` envia a saída do descriptor de arquivo 2, `stderr`, para o mesmo local que o descriptor de arquivo 1, `stdout`.

#### 11.5.3.5 Anexando em arquivos existentes

Use o operador `>>` para redirecionar a saída de um comando, mas anexe ao arquivo, se ele existir. A sintaxe é fornecida a seguir:

```
1. comando >> /tmp/output_file
```

#### 11.5.3.6 Suprimindo o output ou error

Existem alguns cenários em que você não deseja apenas redirecionar a saída de um comando, mas também descarta a saída. Você pode fazer isso redirecionando a saída de um comando para o arquivo nulo `/dev/null`

```
1. ls /usr/bin > /dev/null
```

### 11.5.4 Uso de comandos no script

O SH original foi projetado de forma que tudo o que não fosse uma parte essencial do shell fosse implementado como algum programa fora do próprio shell. Esse design tornava o shell Bourne muito flexível, mas também apresentava algumas desvantagens.

Primeiro, era lento porque os programas eram carregados e reiniciados constantemente para realizar até as tarefas mais simples, e em segundo lugar, tornava os scripts de shell difíceis de transportar porque não havia garantias de que um comando em um sistema operacional fosse implementado da mesma maneira em outro.

Para lidar com esses problemas, o Bash tem muitos de seus comandos fundamentais integrados. Mas para compatibilidade básica com o shell Bourne mais antigo, o Linux ainda implementa sua própria versão dos comandos do Bash.

Por exemplo, `test` é um comando embutido no Bash, mas o Linux também fornece seu próprio programa `/usr/bin/test` para shells que não fornecem uma versão embutida de `test`. Se você não sabe se um comando está embutido, o comando `type` no Bash irá informá-lo, se for um comando interno deverá então retornar “is a shell builtin”.

```
well@wpo:~$ type cd
cd is a shell builtin
well@wpo:~$ type ls
ls is aliased to `ls --color=auto'
well@wpo:~$ type id
id is /usr/bin/id
```

Estes comandos que não são builtin geralmente estão em /bin e /usr/bin mas podem estar em outros diretórios que estão definidos em PATH.

```
well@wpo:/tmp/meuscript$ file /bin/echo
/bin/echo: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=714b557112010bbcd04b0e5e6efc1b106166733c, for GNU/Linux 3.2.
0, stripped
well@wpo:/tmp/meuscript$
```

O Bash contém muitos recursos opcionais que podem ser ativados ou desativados. Essas opções referem-se a como o Bash responde interativamente com o usuário, como ele se comporta em scripts de shell e como o Bash é compatível com outros shells e padrões.

### 11.5.5 Variáveis

A saída dos comandos podem ser armazenados em um arquivo ou variáveis, como as variáveis são salvas na memória, elas tendem a ser mais rápidas. O Bash não coloca um limite superior no tamanho de uma variável: elas são grandes o suficiente para conter qualquer coisa que você precise segurar.

As variáveis são declaradas usando o comando Bash declare. Para declarar uma variável chamada CUSTO, use o seguinte:

```
1. define CUSTO
```

Use a instrução typeset para compatibilidade com o shell Korn. Se você estiver usando o Bash, use declare. O comando declare tem todos os recursos do comando typeset mais antigo. Os nomes das variáveis começam com um caractere alfabético ou sublinhado, seguido por caracteres alfanuméricos adicionais ou sublinhados.

Embora as variáveis possam estar em maiúsculas ou minúsculas, a prática é que as variáveis sejam nomeadas em maiúsculas para não serem confundidas com os comandos do shell, que quase sempre estão em minúsculas. TOTAL, ORDERS\_EUROPE e \_W3C são todos nomes de variáveis legítimos. As variáveis recebem novos valores com um sinal de igual (=). Para atribuir uma string vazia a uma variável, não forneça nenhum valor.

```
1. CUSTO=0
2. exemplo=
```

Embora `printf %d` imprima um zero para um valor igual a zero e uma string vazia, Bash considera os dois valores diferentes. Uma variável sem valor não é o mesmo que uma variável com valor zero.

O Bash diferencia entre o nome de uma variável e o valor que a variável representa. Para se referir ao valor de uma variável, você deve preceder o nome com um cífrão (\$).

```
1. printf "%d\n" $custo
```

```
well@wpo:~$ printf "%d\n" $custo
0
well@wpo:~$
```

O cífrão significa “substituto”. O shell substitui \$custo pelo valor que custo representa. Nesse caso, o valor de custo é zero. Após substituir o valor de custo, o comando se torna:

```
1. printf "%d\n" 0
```

Com o comando declare as variáveis são criadas e elas permanecem em existência até que o script termine ou até que a variável seja destruída com o comando interno chamado unset.

O comando unset é uma conveniência de linha de comando. Em scripts bem estruturados, as variáveis são declaradas no início de um script e não são eliminadas com unset. Isso evita confusão porque o programador pode ter certeza de que as variáveis sempre existem enquanto o script está sendo executado.

Os resultados de um comando também podem ser atribuídos a uma variável. Se um comando está contido entre CRASE (`), tudo escrito na saída padrão é armazenado na variável que está sendo atribuída.

```
1.#!/bin/bash
2. # exemplo_comando.sh
3. # Exemplo de armazenamento de output de comando em uma script
4.
5. declare saida=`ls /`
```

As pessoas familiarizadas com outras linguagens de computador podem se confundir com a maneira como Bash usa aspas. As aspas simples e duplas não são usadas para delinear strings ou caracteres, mas para controlar como o shell agrupa caracteres e interpreta caracteres especiais em uma string. Bash chama esse processo de divisão de palavras.

```
1. custo=0
2. custo="0"
```

Essas duas atribuições produzem o mesmo resultado: custo recebe o valor 0. As aspas duplas mostram explicitamente que o valor a ser atribuído a custo inclui o caractere zero.

Valores alfanuméricos curtos podem ser atribuídos a variáveis sem aspas. No entanto, quando a string contém espaços, não é mais aparente qual valor deve ser atribuído. Veja um trecho de um script:

```
1. cidade=Linhares
2. printf "%s\n" $cidade
3.
4. estado="Espírito Santo"
5. printf "%s\n" $estado
6.
```

Como as aspas não são delimitadoras, mas dicas sobre como interpretar caracteres especiais, elas podem ser usadas consecutivamente. Esta é uma maneira conveniente de separar variáveis do texto ao redor. Veja um trecho de um script:

```
1.
2. taxa=10
3. taxa_mensagem="A taxa é """$taxa""%
4. printf "%s\n" "$taxa_mensagem"
5.
```

Separar cada uma das peças citadas com um espaço resultaria no mesmo problema que você viu anteriormente: o Bash as trataria como três valores individuais. Alternativamente, o nome da variável de substituição de uma variável pode ser colocado entre chaves para deixar claro onde o nome da variável começa e termina.

```
1.
2. taxa=10
3. taxa_mensagem="A taxa é ${taxa}%
4. printf "%s\n" "$taxa_mensagem"
5.
```

Todas as variáveis Bash são armazenadas como strings simples. Cada variável tem certas opções chamadas atributos, que podem ser ativadas ou desativadas usando o comando declare.

Se uma variável for declarada com a opção -i, o Bash ativa o atributo inteiro para essa variável o shell se lembrará de que a string deve ser tratada como um valor inteiro. Se um valor não numérico for atribuído a uma variável inteira, o Bash não relatará um erro, mas, em vez disso, atribui um valor zero.

### 11.5.6 Variáveis Predefinidas

O Bash tem mais de 50 variáveis predefinidas. Essas variáveis, criadas quando o Bash é iniciado, fornecem informações sobre a sessão do Bash e podem ser usadas para controlar alguns dos recursos do shell.

Algumas dessas variáveis têm propriedades especiais que podem ser perdidas se você desconfigurar a variável e, em seguida, criar uma nova com o mesmo nome. Por exemplo, a variável RANDOM contém um número aleatório.

Se você excluir RANDOM com unset e declarar uma nova variável chamada RANDOM, essa nova variável é uma variável de shell normal e não contém números aleatórios. Portanto, é melhor evitar a criação de variáveis com o mesmo nome das variáveis predefinidas.

**BASH** O caminho completo do Bash.

**BASH\_ENV** Em um script de shell, o nome do arquivo de perfil executado antes do script ser iniciado.

**BASH\_VERSION** A versão do Bash.

**COLUMNS** O número de caracteres por linha em sua tela.

**FUNCNAME** Se estiver dentro de uma função, o nome da função.

**HOSTNAME** O nome do computador. Em algumas versões do Linux, pode ser o nome da máquina. Em outros, pode ser um nome de domínio totalmente qualificado.

**HOSTTYPE** Tipo de computador.

**HOME** O nome do seu diretório inicial.

**IFS** O separador de campo interno, uma lista de caracteres usados para dividir uma linha em palavras shell.

**LINENO** O número da linha atual em um script ou função.

**LINES** O número de linhas horizontais em sua exibição.

**OSTYPE** O nome do sistema operacional.

**PATH** Lista separada por dois pontos de caminhos de pesquisa para encontrar um comando a ser executado.

**PPID** O ID do processo do processo pai do shell.

**PROMPT\_COMMAND** Comando a ser executado antes da configuração da string do prompt primário PS1.

**PS1** A string de prompt principal.

**PS2** A string de prompt secundária.

**PS3** A string do prompt de comando de seleção.

**PS4** A string de prefixo de saída do comando trace.

**PWD** O diretório de trabalho atual (conforme definido pelo comando cd).

**RANDOM** Retorna um número aleatório entre 0 e 32767 cada vez que é referenciado.

**SHELL** O shell preferido para usar; para programas que iniciam um shell para você.

**TERM** O tipo de emulação de terminal (por exemplo, console).

### 11.5.7 Exports

Quando o terminal é aberto ou um script é executado por algum mecanismo como o Cron ou Systemd é criado uma “caixa de areia” chamada environment, este environment é um contexto no qual os processos ali estão encapsulados. Este environment possui por padrão variáveis e dados que os programas podem utilizar, isso é o comum.

Mas digamos que um processo pai abra um novo processo filho por fork(), o que tem em comum em ambos? Simples, as variáveis criadas pelo processo pai e que está em Pai e também o environment.

Então um processo pai ou filho podem criar ou consultar variáveis comuns do environment criado para o pai quando o pai foi instanciado, saiba que o Terminal é um processo e que

quando ele invoca um script o script é executado como processo filho deste Terminal, por isso quando você fecha um Terminal todos os scripts são parados, lembre-se da aula de processos.

No exemplo abaixo será criada uma variável no Environment do Terminal chamada UMAVARIABLE e esta variável será acessada de dentro de um script, no qual será executada como processo filho.

No diretório **/tmp** crie um arquivo chamado **script.sh** e edite o seguinte código:

```
1.#!/bin/bash
2. # script.sh
3. # Um exemplo de acesso as variáveis Environment
4.
5. echo $UMAVARIABLE;
```

Agora pelo terminal é dada a permissão de execução para o script em questão (linha 1 da listagem abaixo), veja que na listagem acima não existe a criação da variável UMAVARIABLE, ela será criada no environment do Terminal.

```
1. chmod +x /tmp/script.sh
2.
3. export UMAVARIABLE="Compartilhe aied, ajude o canal"
4.
5. echo $UMAVARIABLE
6.
7. /tmp/script.sh
```

Na linha 3 é criada uma variável no environment utilizando o comando **export** e esta variável tem uma importante solicitação do autor para o leitor, plz....

Na linha 5 é testada a variável, e veja que no terminal ela existe e possui um valor, na linha 7 o script é executado e conforme visto na figura abaixo UMAVARIABLE existe no contexto do script que é um processo filho do Terminal.

```
well@wpo:~/tmp$ well@wpo:~/tmp$ well@wpo:~/tmp$ export UMAVARIABLE='Compartilhe aied, ajude o canal' well@wpo:~/tmp$ well@wpo:~/tmp$ well@wpo:~/tmp$ echo $UMAVARIABLE Compartilhe aied, ajude o canal well@wpo:~/tmp$ well@wpo:~/tmp$ /tmp/script.sh Compartilhe aied, ajude o canal well@wpo:~/tmp$ well@wpo:~/tmp$
```

### 11.5.7 Arrays

Array são listas de valores criadas com o atributo **-a** (matriz), o mecanismo cria uma série de elementos e acessamos cada elemento por meio de um número denominado índice, que refere-se ao item de posição do array.

```
1. declare -a produto
```

Novos itens são atribuídos ao array usando colchetes para indicar a posição na lista, mas a primeira posição é a posição zero. Se um valor inicial for especificado, ele será atribuído à primeira posição. Atribuir um valor não é particularmente útil, mas é incluído para compatibilidade com outros shells. Alternativamente, os valores iniciais podem ser atribuídos a posições específicas incluindo uma posição entre colchetes conforme exemplo abaixo.

```
1. declare -a produto[0]="Baiao de 2" produto[1]="Suco de coco com Leite 750ml"
 produto[2]="Acarajé 4 unidades"
```

Para editar uma posição existente ou adicionar uma nova posição basta informar a posição e atribuir um valor.

```
1. produto[3]="Chopp de vinho 700ml"
```

Para obter o valor de um elemento do array utilize  `${NOMEARRAY[POSICAO]}`, se existir a posição um valor será retornado, mas caso informe uma posição que não existe então retornará vazio. Se nenhuma posição for informada  `${NOMEARRAY}` então o mecanismo irá supor que está referenciando a primeira posição do vetor e retorna o valor desta primeira posição.

o Array inteiro pode ser acessado usando um asterisco (\*) ou um sinal de arroba (@), estes dois símbolos diferem apenas quando aspas duplas são usadas: O asterisco retorna uma string, com cada item separado pelo primeiro caractere de a variável IFS (geralmente espaço) e o sinal de arroba retorna cada item como uma string separada sem nenhum caractere de separação.

```
1.#!/bin/bash
2. # /tmp/array.sh
3. # Exemplo de uso de array
4.
5. # Declarando um array
6. declare -a produto[0]="Baiao de 2" produto[1]="Suco de coco com Leite 750ml"
 produto[2]="Acarajé 4 unidades"
7.
8. # Adicionando mais um elemento
9. produto[3]="Chopp de vinho 700ml"
10.
11. # Imprimindo a posição 1 e a 2
12. printf "Elemento na posição 1: %s\n" "${produto[0]}"
13. printf "Elemento na posição 2: %s\n" "${produto[1]}"
14.
15. # imprimindo todas as posições
16. printf "Todos os elementos: %s\n" "${produto[*]}"
17. printf "Todos os elementos: %s\n" "${produto[@]}"
```

A saída do script será:

```
well@wpo:/tmp$./array.sh
Elemento na posição 1: Suco de coco com Leite 750ml
Elemento na posição 2: Acarajé 3 unidades
Todos os elementos: Baiao de 2 Suco de coco com Leite 750ml Acarajé 3 unidades chopp de vinho
Todos os elementos: Baiao de 2
Todos os elementos: Suco de coco com Leite 750ml
Todos os elementos: Acarajé 3 unidades
Todos os elementos: chopp de vinho
```

O número de itens no array é retornado quando # é usado na frente do nome da variável com uma posição de \* ou @, conforme exemplo abaixo.

```
1. printf "Total: %s\n" "${#produto[@]}"
```

### 11.5.8 Expressões e test

Uma expressão é uma fórmula que é utilizada para calcular um valor que pode ser verdadeiro ou falso. O Bash tem vários comandos e funções para calcular expressões, e nem todos têm a mesma sintaxe ou recursos, em alguns casos por exemplo, há mais de uma maneira de calcular a mesma expressão.

O comando **test** integrado contém uma ampla variedade de testes para arquivos, ele pode testar o tipo de arquivo, a acessibilidade do arquivo, comparar as idades dos arquivos ou outros atributos de arquivo.

```
1. test EXPRESSION
```

```
(EXPRESSION) EXPRESSION is true
! EXPRESSION EXPRESSION is false
EXPRESSION1 -a EXPRESSION2 both EXPRESSION1 and EXPRESSION2 are true
EXPRESSION1 -o EXPRESSION2 either EXPRESSION1 or EXPRESSION2 is true
```

A seguir está uma lista completa de testes de arquivo Bash:

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <b>-b arquivo</b> | verdadeiro se o arquivo for um arquivo de dispositivo de bloco;              |
| <b>-c arquivo</b> | verdadeiro se o arquivo for um arquivo de dispositivo de caracteres;         |
| <b>-d arquivo</b> | verdadeiro se o arquivo for um diretório;                                    |
| <b>-e arquivo</b> | verdadeiro se o arquivo existe;                                              |
| <b>-f arquivo</b> | verdadeiro se o arquivo existir e for um arquivo normal;                     |
| <b>-g arquivo</b> | verdadeiro se o arquivo tiver o conjunto de permissões <b>set-group-id</b> ; |
| <b>-h arquivo</b> | Verdadeiro se o arquivo for um link simbólico;                               |
| <b>-k arquivo</b> | verdadeiro se o arquivo tiver o conjunto de <b>sticky bit</b> ativado;       |
| <b>-p arquivo</b> | verdadeiro se o arquivo for um pipe;                                         |
| <b>-r arquivo</b> | verdadeiro se o arquivo pode ser lido por seu script;                        |
| <b>-s arquivo</b> | verdadeiro se o arquivo existir e não estiver vazio                          |
| <b>-S arquivo</b> | verdadeiro se o arquivo se estiver vazio;                                    |
| <b>-t fd</b>      | Verdadeiro se o descritor de arquivo está aberto em um terminal              |
| <b>-u arquivo</b> | verdadeiro se o arquivo tiver o conjunto de permissões <b>set-user-id</b>    |
| <b>-w arquivo</b> | verdadeiro se o arquivo pode ser gravável (pelo seu script)                  |
| <b>-x arquivo</b> | verdadeiro se o arquivo pode ser executável (por seu script)                 |
| <b>-O arquivo</b> | verdadeiro se o arquivo pode ser (efetivamente) de sua propriedade           |

|                   |                                                                                  |
|-------------------|----------------------------------------------------------------------------------|
| <b>arquivo -G</b> | verdadeiro se o arquivo pode ser (efetivamente) de propriedade do seu grupo      |
| <b>-N arquivo</b> | verdadeiro se o arquivo tiver novo conteúdo (desde a última vez em que foi lido) |
| <b>f1 -nt f2</b>  | (mais novo que) Verdadeiro se o arquivo f1 for mais novo que f2                  |
| <b>f1 -ot f2</b>  | (mais antigo que) Verdadeiro se o arquivo f1 for mais antigo que f2              |
| <b>f1 -ef f2</b>  | (arquivo equivalente) Verdadeiro se o arquivo f1 for um hard link para f2        |

No exemplo abaixo vou validar se /tmp/ e /tmpx/ são diretórios, estou realizando um teste em linha semelhante ao if que estamos acostumados nas linguagens de programação.

```
well@wpo:~$
well@wpo:~$ test -d /tmp/ && echo "Um diretório" || echo "Não é diretório"
Um diretório
well@wpo:~$ test -d /tmpx/ && echo "Um diretório" || echo "Não é diretório"
Não é diretório
well@wpo:~$
```

Já no exemplo abaixo, estou validando a existência do link aied, um ótimo programa e tão importante para o aluno, quando chegar no tópico das estruturas de código vou ensinar como usar no if.

```
well@wpo:~$
well@wpo:~$ test -h /usr/bin/aied && echo "O link para aied existe" || echo "Não existe link para aied"
O link para aied existe
well@wpo:~$
```

O comando de test pode comparar um par de strings:

|                   |                                              |
|-------------------|----------------------------------------------|
| <b>-z string</b>  | Verdadeiro se a string estiver vazia         |
| <b>-n string</b>  | Verdadeiro se a string não estiver vazia     |
| <b>s1 = s2</b>    | Verdadeiro se a string s1 for igual a s2     |
| <b>s1 != s2</b>   | Verdadeiro se a string s1 não for igual a s2 |
| <b>s1 &lt; s2</b> | Verdadeiro se a string s1 for menor que s2   |
| <b>s1 &gt; s2</b> | Verdadeiro se a string s1 for maior que s2   |

Fique atento que os comparadores acima são para testes em strings, veja exemplo ([veja o A é diferente de a](#)).

```
well@wpo:~$
well@wpo:~$ ["awesome" = "awesome"] && echo "Igual" || echo "Diferente";
Igual
well@wpo:~$
well@wpo:~$ ["awesome" = "Awesome"] && echo "Igual" || echo "Diferente";
Diferente
well@wpo:~$
```

Na figura abaixo, estou mostrando uma comparação, embora pareça que 1 é um numérico, tudo é texto, logo pode ser comparado como string, mas recomenda-se utilizar os operadores descritos abaixo.

```
well@wpo:~$ [1 = 1] && echo "Igual" || echo "Diferente";
Igual
well@wpo:~$ [1 = 2] && echo "Igual" || echo "Diferente";
Diferente
```

O comando de test pode comparar valores numéricos, mas usa operadores diferentes daqueles usados para comparar valores de string. Como todas as variáveis do shell são armazenadas como strings, as variáveis podem ser comparadas como números ou strings, dependendo da escolha do operador.

|                  |                                          |
|------------------|------------------------------------------|
| <b>n1 -eq n2</b> | Verdadeiro se n1 for igual a n2          |
| <b>n1 -ne n2</b> | Verdadeiro se n1 não for igual a n2      |
| <b>n1 -lt n2</b> | Verdadeiro se n1 for menor que n2        |
| <b>n1 -le n2</b> | Verdadeiro se n1 for menor ou igual a n2 |
| <b>n1 -gt n2</b> | Verdadeiro se n1 for maior que n2        |
| <b>n1 -ge n2</b> | Verdadeiro se n1 for maior ou igual a n2 |

Veja o exemplo correto para se testar números:

```
well@wpo:~$ [1 -eq 2] && echo "Igual" || echo "Diferente";
Diferente
well@wpo:~$ [1 -eq 1] && echo "Igual" || echo "Diferente";
Igual
```

O comando let integrado realiza cálculos matemáticos de expressões e o resultado é atribuído à variável.

```
well@wpo:~$ let "SOMA=1+1"
well@wpo:~$ echo $SOMA
2
```

Outra forma interessante é:

1. **valor=\$(( 10 % 2 ))**
2. **echo \$valor;**

Você não precisa usar \$ para concatenar ao nome de uma variável na string let entende que qualquer variável que apareça no lado direito do sinal de igual precisa ter seu valor substituído na expressão.

```
well@wpo:~$ declare -i x
well@wpo:~$ x=5
well@wpo:~$ x=x+1
well@wpo:~$ echo $x
6
well@wpo:~$
```

Aqui estão os operadores aritméticos em ordem de precedência. Muitos desses operadores serão familiares aos programadores C:

- , + Unário menos e mais
- !, ~ Negação lógica e bit a bit

```

*, /, % Multiplicação, divisão e resto
+, -- Adição e subtração
<<, >> Deslocamentos bitwize à esquerda e à direita
<=, >=, <, > Comparação
==, != Igualdade e desigualdade
& Bitwise AND
^ Bitwise XOR
| Bitwise OU
&& E lógico
|| Lógico OU
expr ? expr: Expressão condicional
=, *=, /=, %= Atribuição
+=, -=, <<=, >>=, &=, ^=, |= Operações auto-referenciais

```

Testes podem ser combinados utilizando **&&** para AND e **||** para OR, mas o correto é atuar com outros símbolos para os mesmos, dê preferência ao **-a** ao invés de **&&** e **-o** ao invés de **||**.

O operador **&&** executa o comando de test duas vezes mesmo que o resultado da esquerda tenha retornado falso e pode ser executada mais lentamente do que a versão que usa a opção **-a**, portanto, a opção **-a** é preferível ao uso de **&&** em uma expressão if.

Uma situação que tende a confundir os programadores de shell é quando se mistura o operador **not** com a opção **-a** ou **-o**. Na maioria das linguagens de computador, **not** tem precedência como um operador unário e é executado primeiro. No Bash, **-a** e **-o** têm precedência sobre o operador **not**.

### 11.5.9 Comando if

O comando interno if executa um comando e verifica os resultados, se o comando for bem-sucedido, ele executa outro conjunto de comandos (bloco de código).

O comando if executa uma de duas ou mais séries alternativas de comandos com base em um código de status retornado por outros comandos. Normalmente, a instrução if é usada em conjunto com o comando test.

```

1. if [expressão] ; then
2. bloco de código
3. fi

```

O ponto-e-vírgula antes do then é obrigatório pois then é tecnicamente um comando separador, embora funcione em conjunto com if. Por estarem em uma linha, o ponto-e-vírgula é necessário para separar os comandos do teste.

Os comandos if podem ter uma outra parte que só é executada quando a condição falha.

```

1. if [expressão] ; then
2. bloco que é executado quando se tem sucesso no teste

```

```

3. else
4. bloco que é executado quando falha o teste
5. fi

```

A incorporação de comandos complexos em um comando if pode tornar um script difícil de ler e depurar; você deve evitar fazer isso. Nesse caso, o comando rm não é tão proeminente quanto seria se aparecesse em uma linha própria.

Da mesma forma, é possível declarar variáveis dentro de um comando if, mas torna muito difícil determinar quais variáveis existem e quais não existem.

Veja um exemplo que utiliza o comando if para validar um número:

```

1.#!/bin/bash
2. # numero.sh
3. # Um script que faz a leitura de um número e testa se é maior que 10.
4.
5. printf "Informe um número: "
6. read VAR
7.
8. if [$VAR -gt 10] ; then
9. printf "O valor %s é maior que 10.\n" $VAR
10. fi

```

Veja a saída do script na figura abaixo.

```

well@wpo:/tmp$./numero.sh
Informe um número: 15
O valor 15 é maior que 10.
well@wpo:/tmp$ █

```

### 11.5.10 Comando case

O nome vem do fato de que a variável é testada caso a caso, então o comando case compara uma variável com uma série de valores ou padrões e se o valor ou padrão correspondente o bloco de código será executado.

Ao contrário dos comandos **elif** que testam o código de status para comandos individuais, o caso testa o valor de uma variável. O comando case é preferível a um longo conjunto de elifs ao testar valores de string.

Quando um bloco é executado sua execução é parada quando um par de ponto-e-virgula é localizado.

```

1. case VARIABEL in
2. OPCAO1)
3. bloco de código
4. ;;
5. OPCAO2 | OPCAO3)
6. bloco de código
7. ;;

```

```

8. *)
9. bloco de código
10. ;;
11. esac

```

O asterisco padrão (\*) é o caso geral; ele corresponde a todos os valores que não foram tratados por um caso anterior, embora este caso seja opcional, é bom design sempre incluir um caso pega-tudo, mesmo que contenha apenas uma instrução nula.

No script abaixo o case está sendo utilizado para validar a entrada do usuário em um simples exemplo de menu de opções.

```

1.#!/bin/bash
2. # menu.sh
3. # Um exemplo simples de menu em bash
4.
5. printf "Lista de Opções:\nA\tAdicionar arquivo;\nE\tExcluir arquivo;\nS\tSair;\n\n"
6.
7. echo -n "Informe a opção: "
8. read OPCAO
9.
10.case $OPCAO in
11. A)
12. echo "Adicionado :)"
13. ;;
14. E)
15. echo "Excluído :("
16. ;;
17. S)
18. echo "Sair"
19. ;;
20. *)
21. echo "Não entendi..."
22. ;;
23. esac

```

A execução do script exibe um menu de opções na qual o usuário informa a opção desejada e o case realiza o teste sobre o valor da variável.

```

well@wpo:/tmp$./menu.sh
Lista de Opções:
A Adicionar arquivo;
E Excluir arquivo;
S Sair;

Informe a opção: A
Adicionado :)

```

### 11.5.11 Comando while

Existem vários comandos para repetir um conjunto de comandos, o comando while repete os comandos incluídos enquanto o comando testado é bem-sucedido, se o comando falhar na primeira tentativa, os comandos incluídos nunca serão executados.

```

1. while [expressão] ; do
2. bloco de código

```

3. done

Um loop while é completado com o comando done, não elihw como você pode esperar kkkk, ver caso do if e case. Um loop infinito pode ser criado usando o comando true. Como true sempre é bem-sucedido, o loop continuará indefinidamente.

Um loop while pode ser interrompido prematuramente com o comando break, em contrapartida para parar a interação e continuar utiliza-se o comando continue, o que faz com que o restante das instruções incluídas sejam ignoradas; o loop é retomado no topo. continue pode ser seguido por um número que indica de quantas instruções delimitadoras interromper antes de continuar.

```
GNU nano 4.8 loop_exemplo.sh
#!/bin/bash
#/tmp/loop_exemplo.sh

n=1
while [$n -lt 5]
do
 printf "0 valor de N %s\n" $n;
 n=$(($n + 1))
done
```

O resultado da execução do script acima está exposto na figura abaixo.

```
well@wpo:/tmp$./loop_exemplo.sh
0 valor de N 1
0 valor de N 2
0 valor de N 3
0 valor de N 4
```

### 11.5.12 Comando until

Ao contrário do loop while é o loop until, o comando until é idêntico ao comando while, exceto que ele repete as instruções incluídas até que a condição seja bem-sucedida.

1. until [ expressão ] ; do  
 2. bloco de código  
 3. done

Para exemplificar, será desenvolvido um script que terá o mesmo efeito do while, mas com o until, crie um novo arquivo script em /tmp chamado loop\_until.sh e edite o seguinte código:

```
GNU nano 4.8 loop_until.sh
#!/bin/bash
#/tmp/loop_until.sh

n=1;
until [$n -eq 5]
do
 printf "Valor de N %s\n" $n;
 n=$(($n + 1))
done
```

O resultado é semelhante (ver figura abaixo) mas a expressão foi alterada de `-lt` para `-eq`, o autor deste conteúdo acha esta troca um risco para execução de laços complexos.

```
well@wpo:/tmp$./loop_until.sh
Valor de N 1
Valor de N 2
Valor de N 3
Valor de N 4
well@wpo:/tmp$
```

### 11.5.13 Comando for

O laço de repetição mais seguro, pois controla a variável `index` e garante o término. Muito útil quando se tem o número de interações já definidas ou um limite que se pode definir quantos laços serão realizados.

```
GNU nano 4.8 for.sh
#!/bin/bash
#/tmp/for.sh

for i in 1 2 3 4
do
 printf "Contador I %i\n" $i;
done
```

O resultado será semelhante ao obtido com `while` e `until`, conforme figura abaixo, mas com uma segurança muito maior.

```
well@wpo:/tmp$./for.sh
Contador I 1
Contador I 2
Contador I 3
Contador I 4
well@wpo:/tmp$
```

Um recurso muito útil é utilizar resultados de comandos nas estruturas, principalmente em laços de repetição, veja exemplo abaixo.

```
GNU nano 4.8 for_users.sh
#!/bin/bash
#/tmp/for_users.sh

for u in `getent passwd | awk -F: '{print $1}'`
do
 printf "O valor de u é: %s\n" $u
done
```

No script acima o comando `getent passwd | awk -F: '{print $1}'` lista a primeira coluna do arquivo `/etc/passwd`, mas repare que para usar comandos dentro de estruturas é obrigatório o uso de crase delimitando tais comandos.

## 11.6 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para resolver, isto já é levado em consideração.

### 11.6.1 Prática 8ab001 01: Imprimindo na tela

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

1. Criar um arquivo no diretório `~/` com o nome **atividadei.sh**;
2. Neste arquivo defina o **/bin/bash** como interpretador;
3. Faça a chamada do `printf`, o `printf` deve escrever seu nome;
4. Salve o arquivo;
5. Dê a permissão de execução;

Agora execute o comando aied para validar a atividade:

```
1. sudo aied validar 8ab001 checkpoint01
```

Veja abaixo a execução da validação da atividade.

```
usuario@debian:~$ sudo aied validar 8ab001 checkpoint01
Ação que será executada: validar

Prática: Prática do capítulo de Shell Script
Será enviado o OUTPUT do comando: /home/aluno/atividadei.sh
Será enviado o OUTPUT do comando: ls /home/usuario/atividadei.sh -l

Deseja continuar? (s para SIM) s
s1 - Comando: /home/aluno/atividadei.sh
1.1 Validar por regex / w+/
2 - Comando: ls /home/usuario/atividadei.sh -l
2.1 Validar por regex /-rwxr-xr-x(.*)atividadei.sh/

Total de acertos: 2 do total de 2 validações equivalente a 100%.
Identificação: 92a575f60c
AIED v(8)
```

### 11.6.2 Prática 8ab001 02: Uma variável

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

1. Criar um arquivo no diretório `~/` com o nome **atividadeii.sh**;
2. Neste arquivo defina o **/bin/bash** como interpretador;
3. Crie com declare a variável `x` (repare que o `x` é minúsculo) e utilize como valor o seu nome completo;
4. Faça a chamada ao `echo` e imprima o valor da variável;
5. Salve o arquivo;
6. Dê a permissão de execução;

Agora execute o comando aied para validar a atividade:

```
1. sudo aied validar 8ab001 checkpoint02
```

Veja abaixo a execução da validação da atividade.

```
usuario@debian:~$ sudo aied validar 8ab001 checkpoint02
Ação que será executada: validar

Prática: Prática do capítulo de Shell Script
Será enviado o OUTPUT do comando: /home/aluno/atividadeii.sh
Será enviado o OUTPUT do comando: cat /home/usuario/atividadeii.sh
Será enviado o OUTPUT do comando: ls /home/usuario/atividadeii.sh -l

Deseja continuar? (s para SIM) s
s1 - Comando: /home/aluno/atividadeii.sh
1.1 Validar por regex / w+/
2 - Comando: cat /home/usuario/atividadeii.sh
2.1 Validar por regex /declare s+x =/
3 - Comando: ls /home/usuario/atividadeii.sh -l
3.1 Validar por regex /-rwxr-xr-x(.*)?atividadeii.sh/

Total de acertos: 3 do total de 3 validações equivalente a 100%.
Identificação: 92a575f60c
AIED v(8)
```

### 11.6.3 Prática 8ab001 03: Imprimindo mas com python2

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

1. Criar um arquivo no diretório `~/` com o nome **atividadeiii.py**;
2. Defina o interpretador deste arquivo o Python2 (vai ter que achar no sistema);
3. Faça a chamada do `print`, o `print` deve escrever seu nome;
4. Salve o arquivo;
5. Dê a permissão de execução;

Agora execute o comando `aied` para validar a atividade:

```
1. sudo aied validar 8ab001 checkpoint03
```

Veja abaixo a execução da validação da atividade.

```
usuario@debian:~$ sudo aied validar 8ab001 checkpoint03
Ação que será executada: validar

Prática: Prática do capítulo de Shell Script
Será enviado o OUTPUT do comando: /home/aluno/atividadeiii.py
Será enviado o OUTPUT do comando: cat /home/usuario/atividadeiii.py
Será enviado o OUTPUT do comando: ls /home/usuario/atividadeiii.py -l

Deseja continuar? (s para SIM) s
s1 - Comando: /home/aluno/atividadeiii.py
1.1 Validar por regex / w+/
2 - Comando: cat /home/usuario/atividadeiii.py
2.1 Validar por regex /print s+ "/"
3 - Comando: ls /home/usuario/atividadeiii.py -l
3.1 Validar por regex /-rwxr-xr-x(.*)?atividadeiii.py/

Total de acertos: 3 do total de 3 validações equivalente a 100%.
Identificação: 92a575f60c
AIED v(8)
```

### 11.6.4 Prática 8ab001 04: Script que valida par ou ímpar com if

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

1. Criar um arquivo no diretório `~/` com o nome **atividadeiv.sh**;
2. Defina o interpretador deste arquivo o bash;
3. Receba o primeiro parâmetro passado para o script (para receber o primeiro parâmetro utilize `$1`);
4. Se o primeiro parâmetro for um número par, escreva “par”;
5. Se o primeiro parâmetro for um número ímpar, escreva “ímpar”;
6. Salve o arquivo;

7. Dê a permissão de execução;
8. Teste, veja abaixo o valor esperado no teste antes de invocar o comando aied;

```
usuario@debian:~$./atividadeiv.sh 1
impar
usuario@debian:~$./atividadeiv.sh 2
par
usuario@debian:~$
```

Agora execute o comando aied para validar a atividade:

1. sudo aied validar 8ab001 checkpoint04

Veja abaixo a execução da validação da atividade.

```
usuario@debian:~$ sudo aied validar 8ab001 checkpoint04
Ação que será executada: validar

Prática: Prática do capítulo de Shell Script
Será enviado o OUTPUT do comando: /home/usuario/atividadeiv.sh 1
Será enviado o OUTPUT do comando: /home/usuario/atividadeiv.sh 2
Será enviado o OUTPUT do comando: ls /home/usuario/atividadeiv.sh -1

Deseja continuar? (s para SIM) s
s1 - Comando: /home/usuario/atividadeiv.sh 1
1.1 Validar por text impar
2 - Comando: /home/usuario/atividadeiv.sh 2
2.1 Validar por text impar
3 - Comando: ls /home/usuario/atividadeiv.sh -1
3.1 Validar por regex /-rwxr-xr-x(.*)?atividadeiv.sh/

Total de acertos: 3 do total de 3 validações equivalente a 100%.
Identificação: 92a575f60c
AIED v(8)
```

### 11.6.5 Prática 8ab001 05: Script que valida se um número é maior que 10

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

1. Criar um arquivo no diretório `~/` com o nome **atividadev.sh**;
2. Defina o interpretador deste arquivo o bash;
3. Receba o primeiro parâmetro passado para o script, esse parâmetro será um número;
4. Se o primeiro parâmetro for um número maior que 10 então escreva “maior”;
  - a. Caso contrário, escreva “menor ou igual”;
5. Salve o arquivo;
6. Dê a permissão de execução;
7. Teste, veja abaixo o valor esperado no teste antes de invocar o comando aied;

```
usuario@debian:~$./atividadev.sh 5
menor ou igual
usuario@debian:~$./atividadev.sh 11
maior
usuario@debian:~$ _
```

Agora execute o comando aied para validar a atividade:

2. sudo aied validar 8ab001 checkpoint05

Veja abaixo a execução da validação da atividade.

```
usuario@debian:~$ sudo aied validar 8ab001 checkpoint05
Ação que será executada: validar

Prática: Prática do capítulo de Shell Script
Será enviado o OUTPUT do comando: /home/usuario/atividadev.sh 6
Será enviado o OUTPUT do comando: /home/usuario/atividadev.sh 12
Será enviado o OUTPUT do comando: ls /home/usuario/atividadev.sh -l

Deseja continuar? (s para SIM) s
s1 - Comando: /home/usuario/atividadev.sh 6
1.1 Validar por text menor
2 - Comando: /home/usuario/atividadev.sh 12
2.1 Validar por text maior
3 - Comando: ls /home/usuario/atividadev.sh -l
3.1 Validar por regex /-rwxr-xr-x(.*?)atividadev.sh/

Total de acertos: 3 do total de 3 validações equivalente a 100%.
Identificação: 92a575f60c
AIED v(8)
```

## 12 Python para Sistemas Operacionais

Python é uma das linguagens que mais avança em popularidade atacando proposta inicial do Java que era a portabilidade mas Python chega a esta marca mantendo a eficiência no uso de recursos. Em 2020 Python ultrapassa Java nas intenções (medição feita por TIOBE Index) medida por pesquisas nos mecanismos de busca.

Em novembro passado, o Python trocou de posição brevemente com o Java pela segunda posição no índice TIOBE e, neste mês, o Python atacou novamente. A diferença é de apenas 0,11%, mas é razoável supor que o Python manterá sua segunda posição por mais tempo agora. Pode até estar chegando ao primeiro lugar no índice TIOBE no próximo semestre, porque C (assim como Java) está perdendo popularidade. Em outra parte do índice, Rust está tentando voltar ao top 20 e Dart e Julia também estão subindo. *Paul Jansen*  
CEO TIOBE Software

Python está quase todos sistemas operacionais modernos (exceto no maior colaborador do projeto Python, o Windows), todas as distros, sem preocupação, seu script vai funcionar. A versão mais clássica é a versão 2, está a garantia é de 100% de que o usuário encontrará o interpretador para seu script sem necessidade de instalação.

O administrador terá acesso ao Python por 2 caminhos:

- Pelo terminal interativo;
- Por scripts em python;

### 12.1 Listando versões de Python no Linux

Python usa controle de versão semântico, as versões prontas para produção são versionadas no seguinte esquema:

**MAIOR<sup>76</sup>.MENOR.MICRO**

Por exemplo, no Python 3.6.8, 3 é uma versão principal, 6 é uma versão secundária e 8 é uma micro versão.

**MAIOR** Python tem duas versões principais que não são totalmente compatíveis: Python 2 e Python 3. Por exemplo, 3.5.7, 3.7.2 e 3.8.0 fazem parte da versão principal do Python 3.

**MENOR** Esses lançamentos estão trazendo novos recursos e funções. Por exemplo, 3.6.6, 3.6.7 e 3.6.8 fazem parte da versão secundária do Python 3.6.

**MICRO** As novas micro versões contém várias correções de bugs e melhorias.

As versões de desenvolvimento têm qualificadores adicionais. Para obter mais informações, leia a documentação do “Ciclo de Desenvolvimento” do Python.

Python é uma tecnologia que é resiliente a versões anteriores, ou seja, você pode ter várias versões de Python instalados no seu computador e estas versões não vão se conflitar

<sup>76</sup> A sintaxe pode mudar aqui, como foi visto da 2 para 3

gerando problemas, para saber quais versões do Python estão em seu servidor, liste os arquivos python conforme listagem abaixo.

```
1. ls -l /usr/bin/python*
```

Veja a saída abaixo.

```
well@wpo:~$ ls -l /usr/bin/python*
lrwxrwxrwx 1 root root 7 abr 15 2020 /usr/bin/python -> python2
lrwxrwxrwx 1 root root 9 mar 13 2020 /usr/bin/python2 -> python2.7
-rwxr-xr-x 1 root root 3674216 mar 8 10:02 /usr/bin/python2.7
lrwxrwxrwx 1 root root 9 jun 6 2020 /usr/bin/python3 -> python3.8
-rwxr-xr-x 1 root root 5486384 jan 27 12:41 /usr/bin/python3.8
lrwxrwxrwx 1 root root 33 jan 27 12:41 /usr/bin/python3.8-config -> x86_64-linux-gnu-python3.8-config
lrwxrwxrwx 1 root root 16 mar 13 2020 /usr/bin/python3-config -> python3.8-config
-rwxr-xr-x 1 root root 384 mar 27 2020 /usr/bin/python3-future
-rwxr-xr-x 1 root root 388 mar 27 2020 /usr/bin/python3-pasteurize
```

Na figura acima o servidor possui várias versões instaladas, conforme já dito, não há problemas nisso. Como o diretório **/usr/bin** está no **PATH** do sistema então de qualquer ponto do sistema de arquivos eu posso invocar as seguintes versões do Python:

**python2.7** Para invocar esta versão pode-se utilizar no terminal os comandos:

```
python
python2
python2.7
```

**python3.8** Para invocar esta versão pode-se utilizar no terminal os comandos:

```
python3
python3.8
```

Antes que se pergunte se há diferença na sintaxe entre as versões Python, a resposta é sim, mas calma, a diferença na sintaxe existente entre as versões MAIOR, por exemplo em todos as versões MENOR de python2 o comando print não precisa de parênteses, já no python3 print requer parênteses.

Sua decisão deve ser entre escrever scripts em Python 2 ou Python 3, em meu processo decisório uso Python 2 somente em caso onde o alvo pode ser qualquer Distribuição é independente da versão (recente ou passada, muito passada). Para este material será utilizado o Python 3 pois o conteúdo deve partir de algo mais recente e algumas particularidades podem existir nesta versão e neste material.

## 12.2 Python em modo interativo

Para invocar o python no modo interativo simplesmente digite o comando `python3` no terminal e pressione Enter.

```
usuario@debian:~$ python3
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

O python interativo no terminal será caracterizado por >>>, então pode-se digitar comandos python e pressionar Enter;

```
>>>
>>> print("Print é uma função usada para imprimir um texto no output");
Print é uma função usada para imprimir um texto no output
>>> -
```

O comando é finalizado com um ponto-e-vírgula. Para sair no modo interativo deve-se utilizar o comando **exit(CÓDIGO)** na qual passa-se um código e assim como foi visto anteriormente em bash o código 0 significa que tudo ocorreu bem e obteve-se sucesso, no caso do modo interativo qualquer código numérico pode ser utilizado.

## 12.3 Python em scripts

Todo o potencial do python em modo interativo será encontrado no python em script, trata-se de uma forma de automatizar tarefas utilizando arquivos de texto que podem ser interpretados, e a forma que é tratado pelo Sistema Operacional é idêntico ao um script shell clássico.

Quando um script é invocado no terminal shell o usuário pode enviar parâmetros, como qualquer outro comando GNU/Linux já estudado. Em python para obter os parâmetros enviados pelo sistema utiliza-se `sys.argv`, trata-se de um array de valores tendo na posição 0 o nome do script, e sucessivamente os parâmetros a partir deste valor. Vamos a um exemplo, crie um arquivo em `/tmp/` chamado **script.py** e codifique conforme listagem abaixo.

```
1.#!/usr/bin/python3
2.#/tmp/script.py
3.
4.import sys;
5.print("Parâmetro 1 passado no terminal", sys.argv[1]);
```

Para chamar este script por terminal, deve-se dar permissão de execução e invocar conforme listagem abaixo.

```
1. chmod +x /tmp/script.py
2. /tmp/script.py AIEDONLINE
```

Atenção: A sintaxe python, variáveis, estruturas pode ser consultada neste [conteúdo gratuito disponibilizado pela LUMAC](#).

## 12.4 Importando módulos Python

Se você estiver importando qualquer módulo, o interpretador Python verifica se esse módulo está disponível ou não, você pode fazer isso usando a instrução **import** conforme linha 5 do script abaixo.

---

```
1.#!/usr/bin/python3
```

```

2. # gerarsha1.py
3. # Além de um bom exemplo de import, é um script útil para gerar chaves sha1
4. import hashlib;
5.
6. umtexto = '123456';
7. hash_object = hashlib.sha1(umtexto.encode());
8. bHash = hash_object.hexdigest();
9. print(bHash);

```

---

No exemplo acima após importar a hashlib é possível utilizar os recursos desta biblioteca, conforme demonstrado na linha 6 com o uso da função sha1(). Além de ser um script curto é um script importante que será melhorado nos próximos tópicos.

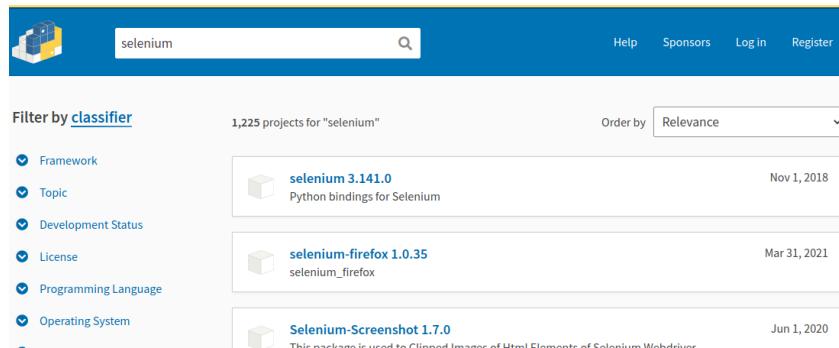
Uma característica muito comum em projetos de código aberto é o foro de engajamento do público, que neste caso são programadores dotados de grande capacidade de criação e que naturalmente almejam ser colaboradores de grandes projetos como o Python, esse é outro ponto forte o que distingue Python de Java e C#.

Este ambiente colaborativo permite aos usuários do Python o compartilhamento de soluções gerais para problemas geralmente comuns e estas contribuições vêm em formato de módulos, sim, então além dos módulos básicos do Python conforme já usado é possível que um desenvolvedor distribua módulos para outros programadores.

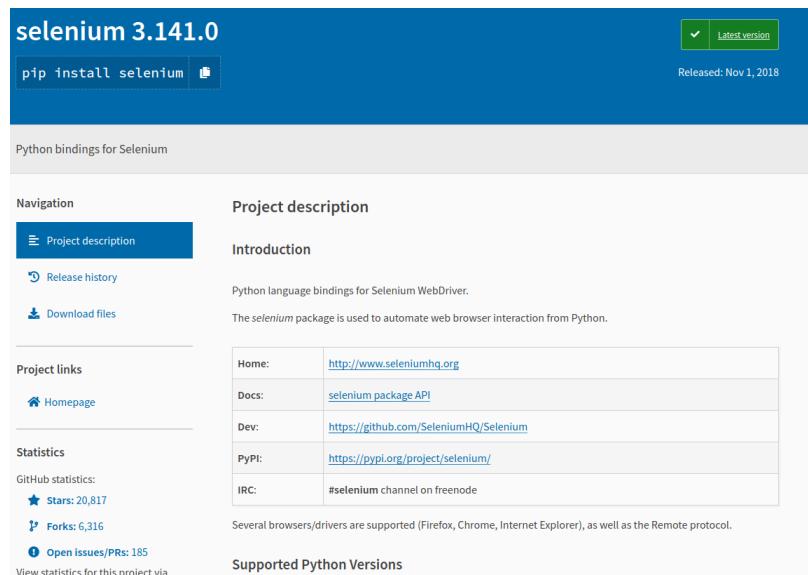
Tudo isso é facilitado como uso de um recurso chamado PIP, trata-se de um programa que é capaz de obter módulos empacotados de uma fonte oficial e faz a instalação inclusive de suas dependências caso seja possível.

Todos os módulos que não estão na instalação padrão do Python estão descritos em um repositório muito instrutivo localizado no site: <https://pypi.org/project/pip/>

Uma busca por “Selenium” podemos ver uma vasta coleção de módulos conforme a imagem abaixo que é um recorte do retorno.



Quando se seleciona um projeto, o usuário encontra uma página com dados conforme exemplo abaixo, inclusive o comando que se deve utilizar para fazer a instalação.



**ATENÇÃO:** Para python2 é pip, para python3 é pip3, mas tem que instalar este último.

Para instalar neste material será utilizado o pip3 para o Python3, para fazer a instalação use o comando apt conforme listagem abaixo.

- 
1. sudo apt update -y
  2. sudo apt install **python3-pip** -y
- 

É uma instalação grande mas não é complicada, ao término o pip3 estará disponível, mas não recomendo o uso direto do pip3 na linha de comando, isso pois é comum que o administrador utilize várias versões de python e é recomendado que se instale o módulo para uma versão específica que realmente irá utilizar o módulo. Então utilize assim:

```

usuario@debian:~$ ls /usr/bin/python*
/usr/bin/python /usr/bin/python3 /usr/bin/python3.7m
/usr/bin/python2 /usr/bin/python3.7 /usr/bin/python3.7m
/usr/bin/python2.7 /usr/bin/python3.7-config /usr/bin/python3-config
usuario@debian:~$ python3.7 -m pip install selenium
|
```

Em 1 verifiquei as versões do Python conforme já descrito no material, após decidir que será utilizado o Python 3.7 em 2 é informado a versão que será usada e em 3 a instalação do módulo selenium.

```
usuario@debian:~$ python3.7 -m pip install selenium
Collecting selenium
 Downloading https://files.pythonhosted.org/packages/80/d6/4294f0b4bce4de0abf13e17190289f9d0613b0a4
 4e5dd6a7f5ca98459853/selenium-3.141.0-py2.py3-none-any.whl (904kB)
 100% |████████████████████████████████| 911kB 632kB/s
Requirement already satisfied: urllib3 in /usr/lib/python3/dist-packages (from selenium) (1.24.1)
Installing collected packages: selenium
Successfully installed selenium-3.141.0
usuario@debian:~$ _
```

No output do processo acima é possível observar que uma dependência para o Selenium é o urllib3 que já se encontra instalado e também a informação que o pacote do Selenium foi instalado com sucesso. A instalação de pacotes por parte do usuário no Debian 10 é alocado em **`~/.local/lib/pythonMAIOR.MENOR/site-packages/`**, para saber é simples, importe o módulo e exiba a propriedade **`__path__`** conforme listagem abaixo.

---

1. `python3.7 -c "import selenium as modulo; print(modulo.__path__)"`

---

Veja o output do comando.

```
usuario@debian:~$ python3.7 -c "import selenium as modulo; print(modulo.__path__);"
['/home/usuario/.local/lib/python3.7/site-packages/selenium']
usuario@debian:~$ _
```

## 12.5 Módulo OS, SYS e SHUTIL

Este módulo fornece uma API com funcionalidades que auxiliam o desenvolvedor a interagir com o Sistema Operacional de forma simples e rápida. Por exemplo, se é preciso ler ou escrever um arquivo, de forma simples se utiliza o `open()` em 3 linhas você edita um arquivo. No módulo OS vamos encontrar uma api para trabalhar com diretórios, com `os.path` pode-se concatenar diretórios independente do Sistema Operacional, validar que diretórios e arquivos existem, tudo isso com 1 única linha.

O módulo SYS fornece uma API para acesso fácil a variáveis usadas e mantidas pelo interpretador que são reflexo do Sistema Operacional, é uma forma de interagir com o SO. Geralmente trabalhamos com OS e SYS em conjunto.

**Atenção:** Nem todas as funcionalidades são compatíveis com todos os sistemas operacionais, veja caso `os.stat()`.

### 12.5.1 Dados do Sistema Operacional e do Environment

Uma variável muito útil para desenvolvedores de scripts é o `os.name`, este atributo retorna qual o tipo de Sistema Operacional está sendo utilizado, veja a diferença entre um Linux para um Windows:

|           |         |
|-----------|---------|
| GNU/Linux | Windows |
|-----------|---------|

```
>>> import os;
>>>
>>> print(os.name);
posix
>>>
```

```
>>> import os;
>>>
>>> print(os.name);
nt
>>>
```

Com este atributo pode-se distinguir se o Sistema Operacional é um GNU/Linux (quando retorna **posix** de um Sistema Operacional Windows quando retorna **nt**). Uma boa tática para se definir o Sistema Operacional é utilizando o **os.uname()**, esta função retorna além do Sistema Operacional algumas informações que pode ser fundamental para se definir a Distribuição que está sendo utilizada.

Veja na figura abaixo o resultado de **os.uname()** em um GNU/Debian 10 terminal.

```
>>> import os;
>>>
>>> print(os.uname());
posix.uname_result(sysname='Linux', nodename='debian', release='4.19.0-16-amd64', version='#1 SMP Debian 4.19.181-1 (2021-03-19)', machine='x86_64')
>>>
>>>
```

Com este resultado dá para se saber se a máquina é 64bits ou 32bits, qual a versão do kernel, o nome do sistema e o nodename, com estas informações dá para se saber onde o script está sendo executado e com isso pode-se fazer adaptações no mesmo.

Na figura abaixo o comando **os.uname()** está sendo executado em um serviço chamado pythonAnywhere<sup>77</sup>, mesmo sendo um GNU/Linux veja que o nodename não é igual ao da imagem anterior, ou seja, não é um debian clássico.

```
Python 3.9.5 (default, May 27 2021, 19:45:35)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os;
>>>
>>> print(os.uname());
posix.uname_result(sysname='Linux', nodename='green-liveconsole9', release='5.8.0-1041-aws', version='#43~20.04.1 SMP Wed Aug 4 15:15:42 UTC 2021', machine='x86_64')
```

Dizer que o Windows não é posix seria injusto pois nem o GNU/Linux segue em sua totalidade o padrão posix, mas pode-se dizer que GNU/Linux é mais aderente ao padrão, e o **os.uname()** requer justamente a aderência do Sistema Operacional ao padrão posix, e por este motivo o comando não é compatível no mundo Windows.

```
>>> import os;
>>>
>>> print(os.uname());
Traceback (most recent call
 File "<stdin>", line 1, in
AttributeError: module 'os'
>>>
```

Uma boa alternativa para se extrair informações do Sistema Operacional é o uso das variáveis de Environment, tanto no GNU/Linux quanto no Windows. Para conhecer estas variáveis de ambiente utilize no Linux o comando **printenv**, conforme figura abaixo.

<sup>77</sup> Acessível pela url: <https://www.pythonanywhere.com/>

```
well@wpo:~$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/wpo:@/tmp/.ICE-unix/1000
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LC_ADDRESS=pt_BR.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=pt_BR.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=pt_BR.UTF-8
```

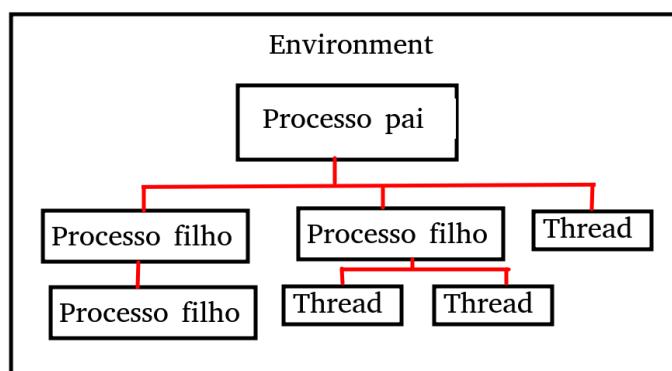
Muitas destas variáveis são universais no mundo Linux, tal como a variável PATH, SHELL, etc., outras variáveis são específicas, tal como GNOME\_SHELL\_SESSION\_MODE, DESKTOP\_SESSION, etc.. Para obter um valor de uma determinada variável para se definir qual sistema operacional está sendo utilizado, ou alguma variável interna de Environment, basta utilizar a função os.getenv() conforme figura abaixo.

```
>>> import os;
>>> print(os.getenv("LOGNAME"));
usuario
>>>
```

Outra forma de obter dados do Environment é utilizando o Dictionary os.environ, por este dicionário pode-se obter qualquer informação do Environment e ainda pode criar, editar e deletar uma variável.

```
>>> import os;
>>>
>>> print(os.environ['LOGNAME']);
usuario
>>>
```

Os processo filhos e threads estão contidas dentro do mesmo contexto “Environment” do processo original, ou seja, o processo pai. Qualquer alteração neste environment é perceptível por todos dentro deste contexto.



Estas duas funcionalidades também são acessíveis de um Microsoft Windows, se no mundo Linux não há um consenso sobre as variáveis de Environment é natural que não há um consenso entre Linux e Windows sobre os nome de variáveis em Environment. O comando para listar as variáveis de ambiente do Windows é **SET**, conforme figura abaixo.

```
C:\Users\wellington>SET
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\wellington\AppData\Roam
CommonProgramFiles=C:\Program Files\Comm
CommonProgramFiles(x86)=C:\Program Files\Comm
CommonProgramW6432=C:\Program Files\Comm
COMPUTERNAME=DESKTOP-OFADFKQ
ComSpec=C:\WINDOWS\system32\cmd.exe
DriverData=C:\Windows\System32\Drivers\D
```

A operação é semelhante a vista no mundo Linux, `os.getenv()` ou `os.environ` podem ajudar o desenvolvedor a obter dados do environment e atuar sobre estas variáveis.

```
>>> import os;
>>>
>>> print(os.getenv('COMPUTERNAME'));
DESKTOP-OFADFKQ
>>> -
```

### 12.5.2 Comandos clássicos GNU/Linux e Windows no python

O python oferece um conjunto de comandos clássicos do GNU/Linux e Windows, um exemplo é o comando `pwd` do Linux que pode ser acessível por `os.getcwd()`, e este comando é compatível com o mundo Windows.

| GNU/Linux                                                              | Windows                                                                        |
|------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| <pre>&gt;&gt;&gt; print(os.getcwd()); /home/usuario &gt;&gt;&gt;</pre> | <pre>&gt;&gt;&gt; print(os.getcwd()); C:\Users\wellington &gt;&gt;&gt; -</pre> |

É comum ser necessário a movimentação do cursor no sistema de arquivos, às vezes para executar scripts que foram desenvolvidos com caminhos relativos, para movimentar o cursor até um diretório específico basta usar o comando `os.chdir()`, conforme figura abaixo.

```
>>> print(os.getcwd());
/home/usuario
>>> os.chdir("/tmp/");
>>>
>>> print(os.getcwd());
/tmp
>>>
```

Neste diretório então o Python pode executar scripts relativos à sua posição que deixam arquivos desnecessários para o sistema após seu uso, `/tmp` é um diretório eliminado ao fim da execução do GNU/Linux.

O comando `os.chmod()` altera a permissão de acesso a um arquivo, as variáveis são adicionadas por meio das constantes, conforme listagem abaixo.

- `stat.S_ISUID` : Set user ID on execution
- `stat.S_ISGID` : Set group ID on execution
- `stat.S_ENFMT` : Record locking enforced
- `stat.S_ISVTX` : Save text image after execution
- `stat.S_IREAD` : Read by owner.
- `stat.S_IWRITE` : Write by owner.
- `stat.S_IEXEC` : Execute by owner.
- `stat.S_IRWXU` : Read, write, and execute by owner
- `stat.S_IRUSR` : Read by owner
- `stat.S_IWUSR` : Write by owner.
- `stat.S_IXUSR` : Execute by owner.
- `stat.S_IRWXG` : Read, write, and execute by group

- stat.S\_IRGRP : Read by group
- stat.S\_IWGRP : Write by group
- stat.S\_IXGRP : Execute by group
- stat.S\_IRWXO : Read, write, and execute by others.
- stat.S\_IROTH : Read by others
- stat.S\_IWOTH : Write by others
- stat.S\_IXOTH : Execute by others

Se há a necessidade de dar permissão de escrita e leitura de um arquivo para o dono do arquivo e leitura para o grupo dono do arquivo o comando é o seguinte:

```
1. import os, stat;
2.
3. os.chmod("/tmp/meuarquivo.txt", stat.S_IWRITE + stat.S_IREAD + stat.S_IRGRP);
```

Os arquivos pertencem a um usuário e a um grupo, todos os outros usuários são considerados outros, para alterar o dono e o grupo dono do arquivo pode-se utilizar a funcionalidade os.chown(), conforme exemplo abaixo:

```
1. import os;
2.
3. uid = 1000; # UID de um usuário do Linux, consulte /etc/passwd
4. gid = 1000; # GID de um grupo, consulte /etc/group
5. os.chown("/tmp/meuarquivo.txt", uid, gid);
```

### 12.5.3 Trabalhando com PATH

Este módulo contém algumas funções úteis para se trabalhar com PATH de arquivos e diretório, todo framework ou api que se propõe a ser multi-plataforma enfrenta o problema de prover um meio seguro para o desenvolvedor de informar diretórios independente da plataforma, saber dominar os.path é fundamental para isso no mundo python mesmo tendo este conteúdo focado em GNU/Linux.

Mas antes de atuar com métodos que manipulam path de arquivos, temos que entender que em inúmeras vezes é necessário se obter PATHs clássicos, como a raiz do sistema, diretório do usuário, diretório temporário, etc.

Para se obter o diretório do usuário deve-se utilizar expanduser, conforme exemplo abaixo.

```
1. import os
2. print(os.path.expanduser('~'))
```

Se for um Windows retornará C:\Users\Well já se for um GNU/Linux retornará /home/well, e é neste diretório que o programa sendo rodado pelo user pode salvar arquivos. No python3 há a possibilidade de se utilizar pathlib que já possui uma interface para definição deste path.

```
1. from pathlib import Path;
2. print(Path.home());
```

Obs. O retorno será o mesmo para ambos os códigos acima.

Outro diretório muito acionado é a raiz, que no caso do Windows é C:\ e em GNU/Linux é / (geralmente). Para isso basta pegar o path absoluto do separador, ou seja, a raiz, veja exemplo.

```
1. import os;
2. print(os.path.abspath(os.sep));
```

Todo programa deve armazenar arquivos do usuário no diretório Home, mas caso se queira registrar arquivos de cache então deve-se utilizar o diretório temporário do sistema, para se obter o diretório temporário do sistema basta utilizar tempfile conforme exemplo.

```
1. import tempfile;
2. print(tempfile.gettempdir());
```

Nosso próximo exemplo é mais complexo mas muito corriqueiro quando há um desenvolvimento de scripts, quando um script precisa obter seu path não importa onde esteja alojado no sistema de arquivos, utiliza-se o inspect (ver 1 na figura abaixo) para obter o caminho relativo de onde se está e o abspath (ver 2 na figura abaixo) para transformar este caminho relativo em um caminho absoluto no sistema.

```
GNU nano 4.8 script.py Modi
#!/usr/bin/python3
Exibe o path do script
/tmp/script.py

import os, inspect;
print(os.path.abspath(inspect.getfile(inspect.currentframe())));
```

O resultado do script acima é /etc/script.py, ou seja, o caminho absoluto do script, pegamos geralmente o caminho absoluto do script para se obter o diretório do script, pois provavelmente este script pode manipular outros artefatos relacionados a ele, tal como executar outro script que está em seu diretório, para pegar o diretório basta utilizar os.path.dirname conforme exemplo abaixo.

```
GNU nano 4.8 script.py Modi
#!/usr/bin/python3
Exibe o path do script
/tmp/script.py

import os, inspect;

SCRIPT_PATH = os.path.abspath(inspect.getfile(inspect.currentframe()));
SCRIPT_DIR = os.path.dirname(SCRIPT_PATH);

print(SCRIPT_PATH, SCRIPT_DIR);
```

O resultado pode ser visto na figura abaixo.

```
well@wpo:/tmp$ nano script.py
well@wpo:/tmp$./script.py
/tmp/script.py /tmp
well@wpo:/tmp$
```

Agora o objetivo é validar se um path é um caminho para um arquivo ou um direorio, para isso será utilizado no script abaixo **isdir** e **isfile**.

```
GNU nano 4.8 script.py
#!/usr/bin/python3
Exibe o path do script
/tmp/script.py

import os, inspect;

SCRIPT_PATH = os.path.abspath(inspect.getfile(inspect.currentframe())) ;
SCRIPT_DIR = os.path.dirname(SCRIPT_PATH);

print(SCRIPT_PATH , 'é um arquivo?' , os.path.isfile(SCRIPT_PATH));
print(SCRIPT_DIR , 'é um diretório?' , os.path.isdir(SCRIPT_DIR));
```

Quando atuamos com múltiplos sistemas operacionais é fundamental o acerto no path que é uma string que consumida pelos comandos que interpretam, e para interpretar o path tem que ser no mínimo aderente ao Sistema Operacional. Para concatenar path, é utilizado o `os.path.join` que garante que o path gerado será coerente com o Sistema Operacional.

```
GNU nano 4.8 script.py Modified
#!/usr/bin/python3
Exibe o path do script
/tmp/script.py

import os, inspect;

SCRIPT_PATH = os.path.abspath(inspect.getfile(inspect.currentframe())) ;
SCRIPT_DIR = os.path.dirname(SCRIPT_PATH);

SEGUNDO_SCRIPT_PATH = os.path.join(SCRIPT_DIR, "segundo.py");
print("O Path do segundo script é:", SEGUNDO_SCRIPT_PATH);
```

No exemplo acima, um path está sendo criado para um segundo script, possivelmente será utilizado para execução em subprocess (vamos ver no futuro), mas será que o arquivo existe? Para validar que o arquivo existe utiliza-se `os.path.exists` conforme exemplo abaixo.

```
GNU nano 4.8 script.py
#!/usr/bin/python3
Exibe o path do script
/tmp/script.py

import os, inspect;

SCRIPT_PATH = os.path.abspath(inspect.getfile(inspect.currentframe()));
SCRIPT_DIR = os.path.dirname(SCRIPT_PATH);

SEGUNDO_SCRIPT_PATH = os.path.join(SCRIPT_DIR, "segundo.py");

print("O Path do segundo script é:", SEGUNDO_SCRIPT_PATH);

if os.path.exists(SEGUNDO_SCRIPT_PATH):
 print("o arquivo existe");
else:
 print("o arquivo NAO existe");
```

O path é coerente para o sistema operacional, mas não é um arquivo que existe e naturalmente não poderá ser executado posteriormente com subprocess.

#### 12.5.4 Manipulando arquivos

Uma rotina muito comum é a cópia de arquivos, remoção de arquivos e renomeação de arquivos, começamos copiando um arquivo que já existe, o arquivo `/tmp/meuarquivo.txt` será copiado para `/home/userlinux/`, conforme exemplo abaixo.

```
1. import os, shutil;
2. shutil.copyfile("/tmp/meuarquivo.txt", "/home/userlinux/meuarquivo.txt");
```

Para renomear ou mover um arquivo é muito simples, basta usar o comando `shutil.move()` conforme exemplo abaixo.

```
1. import os, shutil;
2. shutil.move("/tmp/meuarquivo.txt", "/home/userlinux/comoutronome.txt");
```

Para remover um arquivo a função é `os.unlink()`, o arquivo continua a existir no disco mas sua relação com a tabela i-node é extinta.

```
1. import os;
2. os.unlink("/home/userlinux/comoutronome.txt");
```

Para ler o conteúdo de um arquivo utiliza-se a função `os.open(path, flags)`, onde é passado o caminho do arquivo seguido de alguns flags que alteram os requisitos não funcionais da função open.

Flags para manipular arquivos:

- `os.O_RDONLY` – open for reading only
- `os.O_WRONLY` – open for writing only
- `os.O_RDWR` – open for reading and writing
- `os.O_NONBLOCK` – do not block on open
- `os.O_APPEND` – append on each write

- os.O\_CREAT – create file if it does not exist
- os.O\_TRUNC – truncate size to 0
- os.O\_EXCL – error if create and file exists
- os.O\_SHLOCK – atomically obtain a shared lock
- os.O\_EXLOCK – atomically obtain an exclusive lock
- os.O\_DIRECT – eliminate or reduce cache effects
- os.O\_FSYNC – synchronous writes
- os.O\_NOFOLLOW – do not follow symlinks

Veja um exemplo de leitura de um arquivo que existe e contém dados.

```
1.#!/usr/bin/python3
2.import os;
3. # /tmp/abrirarquivo.py
4.
5.fd = os.open("/tmp/exemplo.txt", os.O_RDONLY | os.O_CREAT);
6.os.lseek(fd, 0, 0);
7.bytes_arquivo = os.read(fd, os.path.getsize(fd));
8.print(bytes_arquivo.decode());
9.os.close(fd);
```

Para escrever um arquivo é simples, com um os.open() e alterar os flags, informando o flag de escrita os.O\_RDWR. Após isso é só executar o write.

```
1.#!/usr/bin/python3
2.import os;
3. # /tmp/criararquivo.py
4.
5.fd = os.open("/tmp/exemplo.txt", os.O_RDWR | os.O_CREAT);
6.os.write(fd, "Aied é top".encode());
7.os.close(fd);
```

## 12.6 Módulo subprocess

O módulo subprocess fornece uma interface consistente para criar e manipular processos filhos criados a partir de seu script. Este módulo oferece uma interface de alto nível que são mais fáceis de lidar do que os.system(), os.spawn() e os.popen()

Neste módulo vamos encontrar as principais funções de interface, são estas:

- subprocess.run()
- subprocess.Popen()
- subprocess.call()
- subprocess.check\_call()
- subprocess.check\_output()

### 12.6.1 subprocess.run()

Antes do Python 3.5 dentre as opções do módulo a função popen era a mais utilizada, na verdade, a popen sempre foi utilizada para fins gerais, para executar um binário, um script, simples ou complexo, tanto para abordagem de muito ou pouco controle sobre o filho.

Mas no python3.5 em diante a interface run passou a ser a interface recomendada pela equipe Python o que revela que um grande trabalho foi realizada nesta versão sobre esta interface, e suplantou ou espera-se, a interface popen que segue em alto uso pelos adeptos do python (basta ver nos fóruns).

A sintaxe da interface run é:

```
1. subprocess.run(args, *, stdin=None, input=None, stdout=None, stderr=None,
capture_output=False, shell=False, cwd=None, timeout=None, check=False,
encoding=None, errors=None, text=None, env=None, universal_newlines=None,
**other_popen_kwargs)
```

**args** é o comando que quer executar como processo filho, veja exemplos:

Para executar uma listagem em um diretório use como args `["ls", "/home"]`

Para executar um script python sem permissão de execução no script `["/usr/bin/python3", "/tmp/meuscript.py"]`

Para executar um script python que tem permissão de execução: `["/tmp/meuscript.py"]`

Se `capture_output` é verdade, os descritores `stdout` e `stderr` do processo filho serão capturados, segue opções:

**subprocess.DEVNULL**: Valor especial que pode ser usado como argumento `stdin`, `stdout` ou `stderr` e indica que o arquivo especial `os.devnull` será usado para despejo dos descritores;

**subprocess.PIPE**: Valor especial que pode ser usado como argumento `stdin`, `stdout` ou `stderr` e indica que um PIPE para o fluxo padrão dos descritores devem ser aberto entre ambos os processos e pode se salvar o output do processo filho em uma variável;

**subprocess.STDOUT**: Valor especial que pode ser usado como o argumento `stderr` e indica que o erro padrão deve ir para o mesmo descritor `OUTPUT`.

Se `shell` é `True`, o comando especificado será executado através do shell, o argumento **universal\_newlines** é equivalente a `text` e é fornecido para compatibilidade com versões anteriores, pois por padrão, os objetos de arquivo são abertos no modo binário.

No próximo exemplo será executado um comando GNU/Linux clássico com uso do `run`, o output do comando será abstraindo para dentro do descritor do python e o output do processo pai será entrelaçado com o output do processo filho.



```
GNU nano 4.8
run_cmd.py
#!/usr/bin/python3
#/tmp/run_cmd.py
import subprocess

process = subprocess.run(['ls', '/', '-lha'], universal_newlines=True)
```

A saída deste processo mas a saída do processo filho será:

```
well@wpo:~/tmp$./run_cmd.py
total 2,1G
drwxr-xr-x 21 root root 4,0K jun 15 2020 .
drwxr-xr-x 21 root root 4,0K jun 15 2020 ..
lrwxrwxrwx 1 root root 7 jun 6 2020 bin -> usr/bin
drwxr-xr-x 4 root root 4,0K dez 17 11:23 boot
drwxrwxr-x 2 root root 4,0K jun 6 2020 cdrom
drwxr-xr-x 2 root root 4,0K jun 15 2020 .config
drwxr-xr-x 22 root root 5,4K dez 17 11:57 dev
drwxr-xr-x 162 root root 12K dez 17 11:23 etc
```

Uma opção muito comum é a captura do output de um processo filho e armazenamento em uma variável, conforme exemplo abaixo.

```
GNU nano 4.8 run_cmd.py
#!/usr/bin/python3
#/tmp/run_cmd.py
import subprocess;

process = subprocess.run(['ls', '/', '-lha'], stdout=subprocess.PIPE, universal_newlines=True);

aqui tem o output capturado
output = process.stdout;

print("fim");
```

A variável `output` passa a ter toda a saída do processo filho, e é natural que por não a imprimir o output do processo pai se torna distinto do output do processo filho, mas veja que na interface `run` tive que adicionar um `stdout=subprocess.PIPE`.

```
well@wpo:~/tmp$./run_cmd.py
:fim
well@wpo:~/tmp$
```

Para manipular o output do processo filho é simples, basta saber interagir com texto pois afinal a comunicação entre processos é feita por texto em formato de bytes, mas o python trabalha muito bem com textos em bytes, no exemplo abaixo estou demonstrando um laço de repetição percorrendo linha por linha do output do comando filho.

```
GNU nano 4.8 run_cmd.py
#!/usr/bin/python3
#/tmp/run_cmd.py
import subprocess;

process = subprocess.run(['ls', '/', '-lha'], stdout=subprocess.PIPE, universal_newlines=True);

aqui tem o output capturado
output = process.stdout;

for linha in output.splitlines():
 print("Linha: ", linha);

print("fim");
```

## 12.6.2 subprocess.Popen()

O mais genérico porém o mais complexo, genérico pois pode-se realizar qualquer intração com o processo filho e naturalmente se adapta a qualquer situação por parte da necessidade, mas esta liberdade também traz grandes consequências, a tal complexidade.

“Com grandes poderes vêm grandes responsabilidades”  
(frase de subprocess para popen)

Não descarta-se nada do que foi dito até o momento sobre subprocess nem das opções definidas em run(), esta interface possui a seguinte sintaxe.

1. `subprocess.Popen(args, bufsize=-1, executable=None, stdin=None, stdout=None, stderr=None, preexec_fn=None, close_fds=True, shell=False, cwd=None, env=None, universal_newlines=None, startupinfo=None, creationflags=0, restore_signals=True, start_new_session=False, pass_fds=(), *, group=None, extra_groups=None, user=None, umask=-1, encoding=None, errors=None, text=None, pipesize=-1)`

No exemplo abaixo o Popen() será utilizado para demonstrar o mesmo exemplo anterior realizado no run(), a alteração se dá principalmente no retorno output.

```
GNU nano 4.8 /tmp/popen_cmd.py
#!/usr/bin/python3
#/tmp/run_cmd.py
import subprocess;

process = subprocess.Popen(['ls', '/', '-lha'], stdout=subprocess.PIPE, universal_newlines=True);

aqui tem o output capturado
output = process.stdout;

for linha in output:
 print("Linha: ", linha.strip());
print("fim");
```

No próximo exemplo será realizado uma interação por descritor IN e OUT, então do processo pai será enviado um JSON, o filho deverá processar o JSON e retornar uma saída. Começamos pelo processo filho conforme código abaixo.

```
GNU nano 4.8 /tmp/filho.py
#!/usr/bin/python3
#/tmp/filho.py
import json, sys;

data = sys.stdin.readlines();
input_process = json.loads(data[0].rstrip());

print("Dentro do filho, o processo pai enviou: ", input_process['mensagem']);
```

Pelo módulo sys será capturado o stdin no processo filho, como é um texto então será extraído as linhas, após isso é feita a conversão para JSON e um print básico para mostrar que o processo filho recebeu o input. Agora vamos codificar o processo pai.

```
GNU nano 4.8 /tmp/pai.py
#!/usr/bin/python3
#/tmp/pai.py
import subprocess, json;

filho = subprocess.Popen(["python3", "/tmp/filho.py"], stdout=subprocess.PIPE, stderr=subprocess.STDOUT,
 stdin=subprocess.PIPE);

input = json.dumps({"mensagem": "ola"});
output_err = filho.communicate(input=input.encode('utf-8'));
print(output_err[0].decode());
```

O destaque no Popen() é para o stdout e stdin como PIPE permitindo assim a comunicação entre o processo filho e o pai e o stderr apontando para o STDOUT. Já o input tem que ser em formato texto, então o JSON é transformado em string e enviado pelo communicate(), este communicate() é um método muito usado em Popen().

O output e o err serão armazenados em output\_err ao término da execução do processo filho, a posição 0 (zero) será o output e a posição 1 (um) será o err. Veja que o processo pai é executado e ele imprime o output do processo filho.

```
well@wpo:~/tmp$./tmp/pai.py
Dentro do filho, o processo pai enviou: ola
well@wpo:~/tmp$
```

No próximo exemplo será capturada a saída de um processo filho em tempo de execução, e o processo pai também deverá aguardar a finalização do processo filho.

## 12.5 Scripts profissionais em Python

Um grande problema para administradores de sistemas operacionais é lidar rotineiramente com scripts que não possui nenhum padrão ou que não são profissionais, pode-se citar:

- Falta de padronização de output;
- Output verboso com alertas excessivos;
- Output com dados de depuração;
- Input por command line fora de padrão;
- Input por command line pouco explicativo;
- Manipulação errônea de arquivos do sistema;
- Falta de dependência;
- Scripts que saem de controle;

Nos próximos tópicos o autor irá definir itens interessantes que se deve ter em um script para sanar a dificuldade dos usuários de scripts Python.

### 12.5.1 Trabalhando com parâmetros

O módulo argparse facilita a manipulação de argumentos passados para o script por meio do Command Line, e também provê uma técnica profissional de interação com o usuário que em grande parte são usuários que buscam a manutenção do sistema operacional. O argparse módulo também gera automaticamente mensagens de ajuda e uso e emite erros quando os usuários dão ao programa argumentos inválidos.

Para iniciar o desenvolvedor deve importar o módulo argparse com import, não é preciso instalar pois este módulo está embutido no Python. Após isso deve-se criar um objeto ArgumentParser, conforme exemplo abaixo.

```
GNU nano 4.8 /tmp/exemplo.py
#!/usr/bin/python3
/tmp/exemplo.py
#
Um exemplo de uso do canal aiedonline

import argparse

parser = argparse.ArgumentParser(description='Um exemplo de uso do canal aiedonline');

print(type(parser));
```

O print foi utilizado apenas para exibir o tipo do objeto retornado por argparse.ArgumentParser(), demonstrado abaixo.

```
well@wpo:/tmp$./exemplo.py
<class 'argparse.ArgumentParser'>
well@wpo:/tmp$
```

Dos parâmetros opcionais de argparse.ArgumentParser os principais são:

**prog** - O nome do programa (padrão: os.path.basename(sys.argv[0]))  
**description** - Texto a ser exibido antes da ajuda do argumento (padrão: nenhum);  
**prefix\_chars** - O conjunto de caracteres que prefixam argumentos (padrão: '-')  
**argument\_default** - O valor padrão global para argumentos (padrão: None)  
**exit\_on\_error** - Determina se ArgumentParser sai ou não com informações de erro quando ocorre um erro. (padrão: True)

Mas o objeto ArgumentParser não irá definir automaticamente os parâmetros, o desenvolvedor do script deve realizar a definição dos parâmetros (obrigatórios ou não) por meio do uso do método add\_argument(), conforme exemplo abaixo.

```
GNU nano 4.8 /tmp/exemplo.py
#!/usr/bin/python3
/tmp/exemplo.py
#
Um exemplo de uso do canal aiedonline

import argparse

parser = argparse.ArgumentParser(description='Um exemplo de uso do canal aiedonline');

parser.add_argument('-a', '--arquivo');
args = parser.parse_args();

print("O arquivo é:", args.arquivo);
```

Um argumento foi adicionado ao parser, o argumento arquivo que não é obrigatório, logo em seguida é usado o método parse\_args() para realizar o parser em sys.argv. Para utilizar um parâmetro é só informar o nome conforme demonstrado no print.

```
well@wpo:/tmp$./exemplo.py -a /etc/passwd
O arquivo é: /etc/passwd
well@wpo:/tmp$
```

Os principais parâmetros para o método add\_argument() são:

**name** - Um nome ou uma lista de strings de opções;

**nargs** - O número de argumentos de linha de comando que devem ser consumidos.  
**default** - O valor produzido se o argumento estiver ausente da linha de comando.  
**type** - O tipo para o qual o argumento de linha de comando deve ser convertido.  
**required** - Se a opção de linha de comando pode ou não ser omitida (somente opcionais).  
**help** - Uma breve descrição do que o argumento faz.  
**dest** - O nome do atributo a ser adicionado ao objeto retornado por `parse_args()`.

Quando se utiliza um sinal asterisco em **nargs** (\*) vários valores são adicionados ao argumento, na forma de um array, vamos ao exemplo. No script abaixo vários path de arquivos podem ser informados por command line.

```
GNU nano 4.8 /tmp/exemplo.py
#!/usr/bin/python3
/tmp/exemplo.py
#
Um exemplo de uso do canal aiedonline

import argparse

parser = argparse.ArgumentParser(description='Um exemplo de uso do canal aiedonline');
parser.add_argument('-a', '--arquivos', nargs='*');
args = parser.parse_args();

print("O arquivo é:", args.arquivos);
```

veja que ao informar 3 arquivos são abstraídos para 3 itens em um array dentro de args.arquivos.

```
well@wpo:/tmp$./exemplo.py -a /etc/passwd /etc/shadow /etc/sysctl.conf
0 arquivo é: ['/etc/passwd', '/etc/shadow', '/etc/sysctl.conf']
well@wpo:/tmp$
```

Quando se utiliza uma interrogação em **nargs** (?), um argumento será consumido do input proveniente do Command Line, se não foi passado o argumento então uma variável é criada com o valor de DEFAULT, se o argumento foi informado porém não foi informado um valor, será usado o valor de CONST, já se foi passado o argumento e o valor então utiliza-se o valor do input do command line.

```
GNU nano 4.8 /tmp/exemplo.py
#!/usr/bin/python3
/tmp/exemplo.py
#
Um exemplo de uso do canal aiedonline

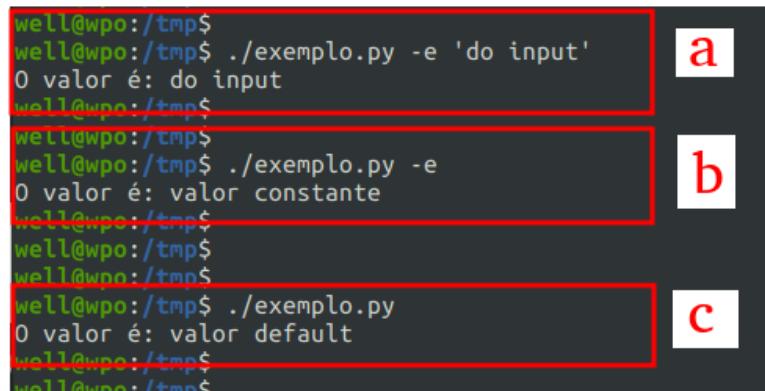
import argparse

parser = argparse.ArgumentParser(description='Um exemplo de uso do canal aiedonline');

parser.add_argument('-e', '--exemplo', nargs='?', const='valor constante', default='valor default');
args = parser.parse_args();

print("O valor é:", args.exemplo);
```

Repare que const é utilizado quando o argumento -e é utilizado porém nenhum valor é passado (em **b**), agora, caso não seja passado o argumento -e é utilizado o valor definido em default (em **c**), esta é a diferença entre o valor default e o valor em const.



well@wpo:/tmp\$ ./exemplo.py -e 'do input'  
O valor é: do input  
well@wpo:/tmp\$  
well@wpo:/tmp\$ ./exemplo.py -e  
O valor é: valor constante  
well@wpo:/tmp\$  
well@wpo:/tmp\$  
well@wpo:/tmp\$ ./exemplo.py  
O valor é: valor default  
well@wpo:/tmp\$  
well@wpo:/tmp\$

Uma opção muito utilizada é o uso do parâmetro required que força a necessidade do argumento, mas este tipo de opção só se faz eficiente quando um help for descrito para o argumento, conforme exemplo abaixo.

```
GNU nano 4.8 /tmp/exemplo.py
#!/usr/bin/python3
/tmp/exemplo.py
#
Um exemplo de uso do canal aiedonline

import argparse;

parser = argparse.ArgumentParser(description='Um exemplo de uso do canal aiedonline');

parser.add_argument('-a', '--arquivo', required=True, help="Informe o caminho do arquivo");
args = parser.parse_args();

print("O arquivo é:", args.arquivo);
```

Caso não seja informado o argumento, veja figura abaixo, é elevado um exception que finaliza a execução do script.

```
well@wpo:/tmp$./exemplo.py
usage: exemplo.py [-h] -a ARQUIVO
exemplo.py: error: the following arguments are required: -a/--arquivo
well@wpo:/tmp$
```

O help se torna útil quando o usuário informa como argumento `-h`, então uma descrição completa do script e argumentos.

```
well@wpo:/tmp$./exemplo.py -h
usage: exemplo.py [-h] -a ARQUIVO

Um exemplo de uso do canal aiedonline

optional arguments:
 -h, --help show this help message and exit
 -a ARQUIVO, --arquivo ARQUIVO
 Informe o caminho do arquivo
well@wpo:/tmp$
```

Scripts recebem argumentos como texto, afinal tudo que sai do teclado são caracteres descritos como texto, caso queira que um argumento receba um cast durante o parse deve-se informar com o parâmetro `type`, veja que o argumento `-n` será convertido em `int` de inteiro.

```
GNU nano 4.8 /tmp/exemplo.py
#!/usr/bin/python3
/tmp/exemplo.py
#
Um exemplo de uso do canal aiedonline

import argparse;

parser = argparse.ArgumentParser(description='Um exemplo de uso do canal aiedonline');

parser.add_argument('-n', '--numero', type=int, required=True, help="Informe um número Inteiro");
args = parser.parse_args();

print("O tipo é:", type(args.numero));
```

Ao executar o script, no `print` é exibido o tipo de variável criada pelo parse do `argparse`.

```
well@wpo:/tmp$./exemplo.py -n 5
0 tipo é: <class 'int'>
well@wpo:/tmp$
```

O método `parse_args` então é utilizado após a definição dos argumentos, por padrão `parse_args()` obtém dados do `sys.argv` e processa de acordo com os argumentos definidos anteriormente. O que poucos sabem é que é possível invocar o `parse_args` informando os valores. São possíveis parâmetros que se pode utilizar neste método:

**args** - Lista de strings para analisar. O padrão é obtido de `sys.argv`.

**namespace** - Um objeto para receber os atributos. O padrão é um novo objeto `Namespace` vazio.

## 12.5.2 Output profissional

O objetivo do output é retratar o ocorrido na execução do script para o usuário, e é lógico que o usuário precisa entender a saída de uma rotina. Muitas saídas de programas, output são complexos e trazem tantos caracteres que nem informação se torna na mente do usuário.

Um output profissional possui:

- Uma definição do procedimento;
- Detalhes do procedimento que será executado;
- Saída sumarizada de preferência só com o que o usuário precisa;
- Parametrização pode ser utilizada para adicionar dados;
- Um resumo;

Vamos analisar 2 outputs interessantes, o primeiro é o output de uma ferramenta chamada nmap, que em:

**a:** Uma definição do procedimento;

**b:** Execução propriamente dita;

**c:** Um resumo da execução;

```
well@wpo:~$ sudo nmap -sV 127.0.0.1
Starting Nmap 7.80 (https://nmap.org) at 2022-02-01 17:05 -03
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000010s latency).
Not shown: 993 closed ports
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
80/tcp open http Apache httpd 2.4.41 ((Ubuntu))
111/tcp open rpcbind 2-4 (RPC #100000)
139/tcp open netbios-ssn Samba smbd 4.6.2
445/tcp open netbios-ssn Samba smbd 4.6.2
631/tcp open ipp CUPS 2.3
9050/tcp open tor-socks Tor SOCKS proxy
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
a
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.48 seconds
b
c
well@wpo:~$
```

O output embora simples é perfeito, na próxima imagem temos a mesma sequência de elementos no output do PING.

```
well@wpo:~$ ping google.com
PING google.com(2800:3f0:4001:82c::200e) 56 data bytes
a
64 bytes from 2800:3f0:4001:82c::200e (2800:3f0:4001:82c::200e): icmp_seq=1 ttl=114 time=9.84 ms
b
64 bytes from 2800:3f0:4001:82c::200e (2800:3f0:4001:82c::200e): icmp_seq=2 ttl=114 time=11.9 ms
^C
c
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 9.836/10.862/11.888/1.026 ms
well@wpo:~$
```

Um outro recurso interessante é o uso de cores, no python pode-se adicionar alguns caracteres para que o output seja colorido, vamos a tabela.

| COR            | Texto    |
|----------------|----------|
| Vermelho       | \033[91m |
| Verde          | \033[92m |
| Amarelo        | \033[93m |
| Roxo Claro     | \033[94m |
| Roxo           | \033[95m |
| Ciano          | \033[96m |
| Cinza          | \033[97m |
| Preto          | \033[98m |
| RESET          | \033[00m |
| <b>NEGRITO</b> | \033[1m  |

Vamos ao exemplo, crie um novo script /tmp/exemplo\_cor.py e edite o código abaixo, repare que no print é utilizado os caracteres especiais da tabela acima para, o format está sendo utilizado para concatenação.

```
GNU nano 4.8
#!/usr/bin/python3
#/tmp/exemplo_core.py
Exibindo uma forma de imprimir no output com cor

print("\033[91m {}\033[00m".format("Um texto vermelho"));
print("\033[92m {}\033[00m".format("Um texto verde"));
print("\033[93m {}\033[00m".format("Um texto amarelo"));
```

Ao executar o script o output assume as cores utilizadas no print, repare que sempre é utilizado \033[00 para resetar o estilo de cor.

```
well@wpo:/tmp$./exemplo_cor.py
Um texto vermelho
Um texto verde
Um texto amarelo
well@wpo:/tmp$
```

Com estes caracteres especiais pode-se indexar com níveis elementos no output. Você pode expressar os caracteres control-g, backspace, tab, newline, vertical tab, formfeed, space, return, del e escape como, conforme listagem abaixo.

```
\a control-g
\b retrocesso
\t tab
\n nova linha
\v tabulação vertical
\f caractere formfeed
```

```
\r retorno de carro
\` caractere de escape
\`s caractere de espaço
\\ caractere de barra invertida
\`d excluir caractere
```

Algo que é importante e poucos desenvolvem em seus produtos são os Banners, tais banners causam impacto e demonstram o capricho dos desenvolvedores com o seu produto, embora não seja recomendado para scripts rotineiros e simples, para scripts que se tornam produtos ou que tem grande adesão de um grupo Banner é fundamental. Na figura abaixo um banner está sendo exibido, é o banner do script Metasploitable muito usado por hackers.



O desenvolvedor pode escrever seu próprio banner usando print e os caracteres especiais ou instalar bibliotecas que fazem tal operação, vamos ao exemplo, mas será preciso instalar o pyfiglet.

```
1. python3 -m pip install pyfiglet
```

O script é simples, define-se uma fonte e renderiza um texto conforme script abaixo (vantagem de se usar uma biblioteca pronta), além do banner está sendo utilizado o artifício das cores, a cor escolhida será verde.

```
GNU nano 4.8
#!/usr/bin/python3
#/tmp/exemplo_banner.py
Criando um banner para um programa

from pyfiglet import Figlet;
f = Figlet(font='slant');

print('\033[92m'); # COR VERDE
print(f.renderText('Aied Online')); # BANNER
print("\033[00m"); # ESTILO COR PADRÃO DO TERMINAL
```

O resultado é TOP, e Aied Online aparece como um banner de terminal conforme imagem abaixo, é impactante e naturalmente ajuda a difundir e estabilizar um script no mercado de scripts.



Em muitas áreas do conhecimento se utiliza um tipo de sintaxe em scripts, isso é possível ver quando entramos no mundo dos hackers, na figura abaixo temos um script desenvolvido para hackers analisaram tráfego de rede desenvolvido no livro de [Hacker entre a luz e as trevas](#) do mesmo autor deste livro.

```
well@wpo:/tmp$./script.py
[+] 2005-01-14 01:50:16.484263
 Quadro Ethernet: 00:0e:35:78:0c:02 ff:ff:ff:ff:ff:ff 2054

[+] 2005-01-14 01:50:16.501471
 Quadro Ethernet: 00:0e:35:78:0c:02 00:90:d0:eb:46:e7 2048
/home/well/.local/lib/python3.8/site-packages/dpkt/ip.py:123: UserWarning: IP.off is deprecated
 deprecation_warning("IP.off is deprecated")
 Pacote IP: 192.168.1.3 -> 192.168.1.1 (len=70 ttl=128 DF=0 MF=0 offset=0)

[+] 2005-01-14 01:50:16.501497
 Quadro Ethernet: 00:90:d0:eb:46:e7 00:0e:35:78:0c:02 2054

[+] 2005-01-14 01:50:16.504144
 Quadro Ethernet: 00:90:d0:eb:46:e7 00:0e:35:78:0c:02 2048
 Pacote IP: 192.168.1.1 -> 192.168.1.3 (len=98 ttl=64 DF=0 MF=0 offset=0)

[+] 2005-01-14 01:50:16.580303
 Quadro Ethernet: 00:0e:35:78:0c:02 ff:ff:ff:ff:ff:ff 9298

[+] 2005-01-14 01:50:17.600303
 Quadro Ethernet: 00:90:d0:eb:46:e7 00:10:c6:30:6b:b3 9298

[+] 2005-01-14 01:50:19.113417
 Quadro Ethernet: 00:0e:35:78:0c:02 00:90:d0:eb:46:e7 2048
 Pacote IP: 192.168.1.3 -> 192.168.1.1 (len=61 ttl=128 DF=0 MF=0 offset=0)
```

É comum uso de colchetes com caracteres nesta área, repare o uso da indentação por meio de \t e as cores reforçando que tipo de PDU foi possível capturar na rede. Não existe uma norma para o símbolo que se utiliza dentro de colchetes, mas pode-se utilizar:

- \* Quando seu script cria algo novo;
- + Quando foi possível detectar algo importante;
- Quando algo é removido ou perdido;
- ! Atenção para uma falha;
- > Dá a idéia de nível;

Embora pareça trivial o print possui uma série de parâmetros que permite reduzir código, imagine ter que imprimir uma série de elementos com vírgula (,) como separador, sem uso de for com print é possível. Vamos a assinatura do print na definição abaixo.

```
1. print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

Onde:

- objects** são os objetos que quer imprimir no arquivo (veja o parâmetro file);
- sep** é o separador entre os objects;
- end** é o caracter usado no final do print, por padrão em python é uma quebra de linha;

**file** é a saída, ou seja, o Stream File de saída que por padrão é o registrador do próprio programa;  
**flush** é usado para avançar ponteiros em registradores liberando espaços;

Pode-se utilizar caracteres especiais, conforme já visto aqui em um texto de saída, conforme figura abaixo.

```
>>> print("Aied é um canal\nmuito bom");
Aied é um canal
muito bom
>>>
```

Por padrão o print do Python após escrever a cadeia de caracteres no Stream adiciona o caractere padrão definido em **end**, que é um `\n`. Caso queira escrever e manter o cursor na mesma linha, defina um valor para **end**.

```
1. print("Siga sempre o canal:", end=" ");
2. print("Aiedonline");
```

No exemplo acima a saída será: **Siga sempre o canal: Aiedonline**

## 12.6 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para resolver, isto já é levado em consideração.

### 12.6.1 Prática py0001 checkpoint01: Preparando o ambiente

Nesta prática o aluno deverá configurar o ambiente python com um módulo necessário.

1. Instale o pip3 para o python3;
2. Após instalar o **pip3** instale o módulo [network](#);

Agora execute o comando aied para validar a atividade:

```
1. sudo aied validar py0001 checkpoint01
```

Veja abaixo a execução da validação da atividade.

```
usuario@debian:~$ sudo aied validar py0001 checkpoint01
Ação que será executada: validar

Prática: Prática da aula de Python - instalando o ambiente
Será enviado o OUTPUT do comando: dpkg -l | grep python3-pip
Será enviado o OUTPUT do comando: su usuario -c "python3 -c 'import network as mod
'''"

Deseja continuar? (s para SIM) s
si - Comando: dpkg -l | grep python3-pip
1.1 Validar por regex /ii s+python3-pip/
2 - Comando: su usuario -c "python3 -c 'import network as modulo; print(modulo)''"
2.1 Validar por text network.py

Total de acertos: 2 do total de 2 validações equivalente a 100%.
Identificação: ad2c5925c2
AIED v(8)
```

## 12.6.2 Prática py0001 checkpoint02: Imprimindo o usuário corrente do Environment

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

1. Crie com o nano um arquivo /home/userlinux/pythonscript1.py;
2. No script aponte o Shebang para o python3;
3. Import o módulo os;
4. Imprima o usuário corrente;
5. Adicione o poder de execução para o script /home/userlinux/pythonscript1.py;

Agora execute o comando aied para validar a atividade:

```
1. sudo aied validar py0001 checkpoint02
```

Veja abaixo a execução da validação da atividade.

```
usuario@debian:~$ sudo aied validar py0001 checkpoint02
Ação que será executada: validar

Prática: Prática do capítulo de Python
Será enviado o OUTPUT do comando: /home/usuario/pythonscript1.py
Será enviado o OUTPUT do comando: cat /home/usuario/pythonscript1.py
Será enviado o OUTPUT do comando: ls /home/usuario/pythonscript1.py -l
Será enviado o OUTPUT do comando: su usuario -c "python3 /home/usuario/pythonscript1.py"

Deseja continuar? (s para SIM) s
s1 - Comando: /home/usuario/pythonscript1.py
1.1 Validar por regex / w/
2 - Comando: cat /home/usuario/pythonscript1.py
2.1 Validar por text #!/usr/bin/python3
2.2 Validar por regex /import s+os/
3 - Comando: ls /home/usuario/pythonscript1.py -l
3.1 Validar por regex /-rwxr-xr-x(.*)?pythonscript1.py/
4 - Comando: su usuario -c "python3 /home/usuario/pythonscript1.py"
4.1 Validar por text usuario

Total de acertos: 5 do total de 5 validações equivalente a 100%.
Identificação: ad2c5925c2
AIED V(8)
```

## 12.6.3 Prática py0001 checkpoint03: Criando uma hash em MD5 do primeiro argumento

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

1. Crie com o nano um arquivo /home/userlinux/pythonscript2.py;
2. No script aponte o Shebang para o python3;
3. Import o módulo hashlib e sys;
4. Receba o primeiro argumento, e gere um MD5 deste argumento, guarde em uma variável;
5. Imprima esta variável com o MD5;
6. Adicione o poder de execução para o script /home/userlinux/pythonscript2.py;

Agora execute o comando aied para validar a atividade:

```
2. sudo aied validar py0001 checkpoint03
```

Veja abaixo a execução da validação da atividade.

```

usuario@debian:~$ sudo aied validar py0001 checkpoint03
Ação que será executada: validar

Prática: Prática do capítulo de Python
Será enviado o OUTPUT do comando: /home/usuario/pythonscript2.py
Será enviado o OUTPUT do comando: cat /home/usuario/pythonscript2.py
Será enviado o OUTPUT do comando: ls /home/usuario/pythonscript2.py -l
Será enviado o OUTPUT do comando: /home/usuario/pythonscript2.py aied

Deseja continuar? (s para SIM) s
s1 - Comando: /home/usuario/pythonscript2.py
1.1 Validar por regex / w+/
2 - Comando: cat /home/usuario/pythonscript2.py
2.1 Validar por text #!/usr/bin/python3
2.2 Validar por regex /import s+(sys|hashlib)/
3 - Comando: ls /home/usuario/pythonscript2.py -l
3.1 Validar por regex /-rwxr-xr-x(.*)?pythonscript2.py/
4 - Comando: /home/usuario/pythonscript2.py aied
4.1 Validar por text a31b95a837056076d5a9a4f7669799fd

Total de acertos: 5 do total de 5 validações equivalente a 100%.
Identificação: ad2c5925c2
AIED v(8)

```

#### 12.6.4 Prática py0001 checkpoint04: Listando usuários com shell /bin/bash

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

1. Crie com o nano um arquivo /home/userlinux/pythonscript3.py;
2. No script aponte o Shebang para o python3;
3. Abra o arquivo **/etc/passwd** para leitura, carregue as linhas em um array;
4. Faça um laço de repetição, e imprima o usuário do Linux que possui com default shell o **/bin/bash**;
5. Adicione o poder de execução para o script /home/userlinux/pythonscript3.py;

Agora execute o comando aied para validar a atividade:

1. sudo aied validar py0001 checkpoint04

Veja abaixo a execução da validação da atividade.

```

usuario@debian:~$ sudo aied validar py0001 checkpoint04
[sudo] password for usuario:
Ação que será executada: validar

Prática: Prática do capítulo de Python
Será enviado o OUTPUT do comando: /home/usuario/pythonscript3.py
Será enviado o OUTPUT do comando: cat /home/usuario/pythonscript3.py
Será enviado o OUTPUT do comando: ls /home/usuario/pythonscript3.py -l
Será enviado o OUTPUT do comando: /home/usuario/pythonscript3.py aied

Deseja continuar? (s para SIM) s
s1 - Comando: /home/usuario/pythonscript3.py
1.1 Validar por regex / w+/
2 - Comando: cat /home/usuario/pythonscript3.py
2.1 Validar por text #!/usr/bin/python3
2.2 Validar por regex /import s+os/
3 - Comando: ls /home/usuario/pythonscript3.py -l
3.1 Validar por regex /-rwxr-xr-x(.*)?pythonscript3.py/
4 - Comando: /home/usuario/pythonscript3.py aied
4.1 Validar por text usuario
4.2 Validar por text root

Total de acertos: 6 do total de 6 validações equivalente a 100%.
Identificação: ad2c5925c2
AIED v(8)

```

#### 12.6.5 Prática py0001 checkpoint05: Persistir um arquivo

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

1. Crie com o nano um arquivo /home/userlinux/pythonscript4.py;
2. No script aponte o Shebang para o python3;
3. Receba por parâmetro na posição 1 um valor que será passado;

4. Salve o valor passado por argumento na posição 1 em um arquivo **/tmp/argumento.txt**
  5. Adicione o poder de execução para o script `/home/userlinux/pythonscript4.py`;
- Agora execute o comando aied para validar a atividade:

```
1. sudo aied validar py0001 checkpoint05
```

Veja abaixo a execução da validação da atividade.

```
usuario@debian:~$ sudo aied validar py0001 checkpoint05
Ação que será executada: validar

Prática: Prática do capítulo de Python
Será enviado o OUTPUT do comando: /home/usuario/pythonscript4.py
Será enviado o OUTPUT do comando: cat /home/usuario/pythonscript4.py
Será enviado o OUTPUT do comando: ls /home/usuario/pythonscript4.py -l
Será enviado o OUTPUT do comando: /home/usuario/pythonscript4.py aied
Será enviado o OUTPUT do comando: cat /tmp/argumento.txt

Deseja continuar? (s para SIM) s
s1 - Comando: /home/usuario/pythonscript4.py
1.1 Validar por regex / w/
2 - Comando: cat /home/usuario/pythonscript4.py
2.1 Validar por text #!/usr/bin/python3
2.2 Validar por regex /import s+(sys|os)/
3 - Comando: ls /home/usuario/pythonscript4.py -l
3.1 Validar por regex /-rwxr-xr-x(.*)?pythonscript4.py/
4 - Comando: /home/usuario/pythonscript4.py aied
5 - Comando: cat /tmp/argumento.txt
5.1 Validar por text aied

Total de acertos: 5 do total de 5 validações equivalente a 100%.
Identificação: ad2c5925c2
AIED v(8)
```

# 13 Adicionando um serviço na inicialização do GNU/Linux (atuando...)

Conforme descrito no capítulo de Processos Linux, um processo é o resultado do carregamento de um arquivo regular com Bit mático ativo, ou seja, um arquivo que se pode executar. Quando carregado para a memória uma complexa estrutura é montada para armazenar o código bem como variáveis e execuções de métodos. Os principais eventos que levam à criação de processos:

- Início do sistema;
- Execução de chamada ao sistema de criação de processos;
- Solicitação do usuário para criar um novo processo;
- Início de um job em lote;

Neste capítulo será demonstrado todo o processo de carregamento, desde a entrada da energia nos equipamentos até o término do carregamento dos aplicativos de apoio do Sistema Operacional.

## 13.2 Passos executados pela BIOS ou UEFI

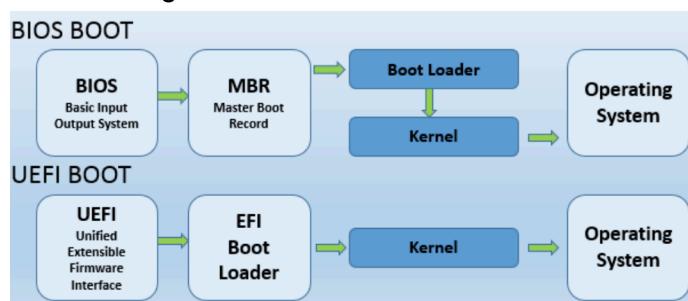
Antes da inicialização de um Sistema Operacional ocorre no computador uma série de eventos sequenciais que levam o carregamento correto do Sistema bem como aplicativos, tudo começa ainda na placa mãe, pois é dentro do chipset da máquina que encontramos alguns elementos fundamentais.

O primeiro elemento é BIOS (Basic Input/Output System), trata-se de sistema pré-gravado no chipset da placa mãe em equipamentos antigos, em computadores novos temos o advento da UEFI. A função da BIOS ou do UEFI é iniciar o processo de inicialização, chamamos esta etapa de boot, no qual alguns **subprocesso** devem acontecer. Alguns hardwares mantêm o legado, afinal chegamos neste patamar tecnológico porque nunca ignoramos o legado.

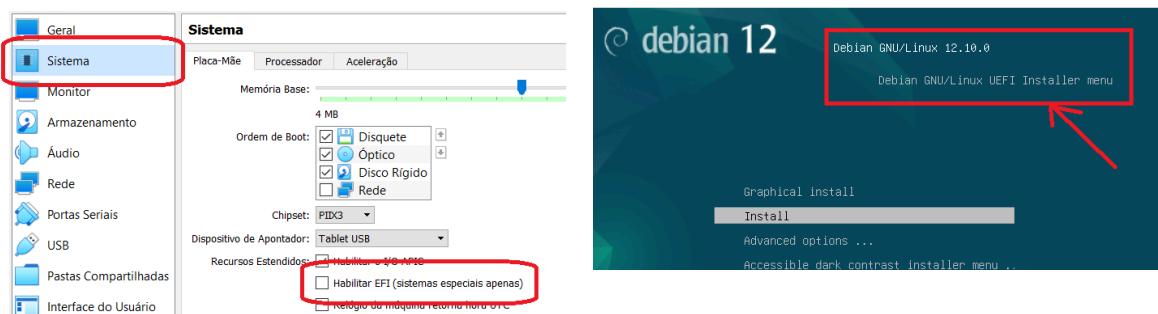


Partindo do pressuposto que seu computador utiliza a BIOS, este é responsável pelo carregamento da memória RAM e outros dispositivos ES com vídeo, teclado, USB, etc. As seguinte etapas são seguidas no carregamento orquestrado pela BIOS:

1. **A memória CMOS é acessada**, por esta memória a BIOS obtém dados para reconhecer os dispositivos de ES principais e iniciar o carregamento e posteriormente testes;
2. **O Power-on Self Test (POST) entra em ação**, é um conjunto de teste que a BIOS realiza para saber se tudo está se inicializando da maneira correta, esta é a fase dos famosos beeps;
3. **Procura de fonte de inicialização**, algum dispositivo que possui informações para arranque do Sistema Operacional, ser um disco rígido (padrão), CD-ROM, pendrive, disquete, entre outros.
4. **Supondo que utilize a BIOS e MBR**, a BIOS lê o setor zero (que contém apenas 512 bytes, denominado Master Boot Record) do sistema secundário. Essa área contém um código que alavanca a inicialização do sistema operacional, no caso do GNU/Linux vamos carregar o bootloader.



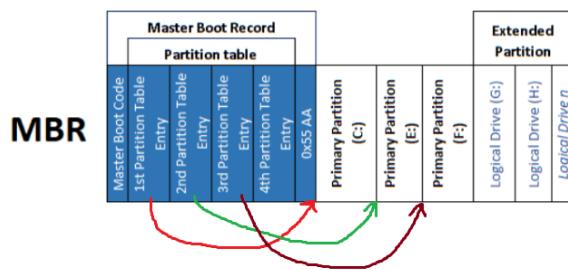
Se o Bootloader for carregado com sucesso, ele irá carregar um arranque do Sistema Operacional, no caso do Debian GNU/Linux é usado o Bootloader GRUB, mas existem outros. No GRUB encontraremos opções de carregamento do SO, mas nem todo Bootloader permite seleção. No caso do UEFI ele irá carregar um programa complexo chamado EFI que fará vários trabalhos, incluindo a inicialização (o serviço do Bootloader), então carregará ele o arranque do SO. Para ativar esta opção no VirtualBox na aba de Sistema habilitar EFI conforme figura abaixo.



Verá que na instalação (ver figura acima) aparecerá a informação que o Debian será instalado com UEFI.

### 13.3 MBR e GPT

Se chegou neste ponto de seu estudo então sabe o que é uma estrutura de dados e as diferenças entre elas, sabe inclusive como as manipular. Pense que o antigo padrão **Master Boot Record (MBR)** como um VETOR de 4 posições fixas, e cada posição neste vetor aponta para uma posição no disco, que estará devidamente particionada e formatada.



Então o número máximo de partições é um valor fixo e imutável. A organização da tabela de partições no MBR limita o espaço máximo de armazenamento endereçável de um disco a 2 TiB ( $232 \times 512$  bytes).

Abordagens para elevar levemente esse limite presumindo que a aritmética de 33 bits ou setores de 4096 bytes não seja oficialmente suportada, pois quebra a compatibilidade com carregadores de inicialização existentes e com a maioria dos sistemas operacionais e ferramentas de sistema compatíveis com MBR e podem causar sérios danos aos dados quando usados fora de ambientes de sistemas controlados. Outro grande problema é que está obrigatoriamente no início, ou seja, é possível que haja uma falha neste ponto do disco e requerer atuação de especialistas para isolar essa área afetada.

```

1 well@usb:~$ sudo dd if=/dev/sda bs=512 count=1 | hexdump -C
2
3
4

```

1. Endereço do disco;

2. Dados convertidos de hexadecimal para algo mais humano;

3. Número de registros encontrados;

4. Uma entrada que informa que será carregado o bootloader GRUB.

```

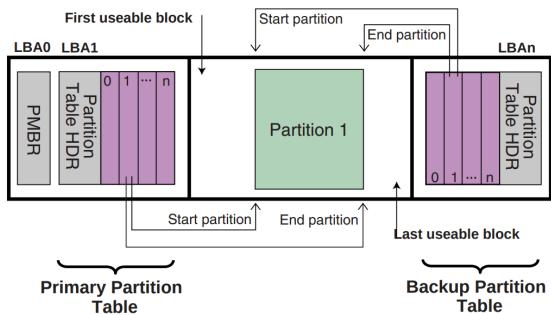
well@usb:~$ sudo dd if=/dev/sda bs=512 count=1 | hexdump -C
00000000 eb 63 90 10 8e d0 bc 00 b0 b8 00 00 8e d8 8e c0 |.c.....
00000010 fb be 07 7c bf 00 06 b9 00 02 f3 a4 ea 21 06 00 |....|.....!..
00000020 00 be 07 38 04 75 0b 83 c6 10 81 fe 07 75 |....8.u.....u|
00000030 f3 eb 16 b4 02 b0 01 bb 00 7c b2 80 8a 74 01 8b |.....|....t..|
00000040 4c 02 cd 13 ea 00 7c 00 00 eb fe 00 00 00 00 00 |L....|....|
00000050 00 00 00 00 00 00 00 00 00 00 00 00 80 01 00 00 |.....|....|
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|....|
00000070 74 02 02 80 ea 79 7c 00 00 31 c0 8e d8 8e d0 bc |t....y|..1....|
1+0 records in
1+0 records out
512 bytes copied, 0.000245449 s, 2.1 MB/s
00000080 00 20 fb a0 64 7c 3c ff 74 02 88 c2 52 be 80 7d |...d|<.t..R..|
00000090 e8 17 01 be 05 7c b4 41 bb aa 55 cd 13 5a 52 72 |.....|A..U..ZRx|
000000a0 3d 81 fb 55 aa 75 37 83 e1 01 74 32 31 c0 89 44 |=..U.u7...t21..D|
000000b0 04 40 88 44 ff 89 44 02 c7 04 10 00 66 8b 1e 5c |.|@.D..D....f..`|
000000c0 7c 66 89 5c 08 66 8b 1e 60 7c 66 89 5c 0c c7 44 ||f.\.f..|f.\..D|
000000d0 06 00 70 b4 42 cd 13 72 05 bb 00 70 eb 76 b4 08 |..p.B..r..p.v..|
000000e0 cd 13 73 0d 5a 84 d2 0f 83 d8 00 be 8b 7d e9 82 |..s.Z.....}..|
000000f0 00 66 0f b6 c6 88 64 ff 40 66 89 44 04 0f b6 d1 |.f....d.@f.D....|
00000100 c1 e2 02 88 e8 88 f4 40 89 44 08 0f b6 c2 c0 e8 |.....|..D....|
00000110 02 66 89 04 66 a1 60 7c 66 09 c0 75 4e 66 a1 5c |.f..f.|f..UNf.\|
00000120 7c 66 31 d2 66 f7 34 88 d1 31 d2 66 f7 74 04 3b ||f1.f.4..1.f.t.;|
00000130 44 08 7d 37 fe c1 88 c5 30 c0 c1 e8 02 08 c1 88 |D.}7....0....|
00000140 d0 5a 88 c6 bb 00 70 8e c3 31 db b8 01 02 cd 13 |.Z....p..1....|
00000150 72 1e 8c c3 60 1e b9 00 01 8e db 31 f6 bf 00 80 |x....`.....1....|
00000160 8e c6 fc f3 a5 1f 61 ff 26 5a 7c be 86 7d eb 03 |.....a.&Z|...|
00000170 be 95 7d e8 34 00 be 9a 7d e8 2e 00 cd 18 eb fe |...).4....|
00000180 47 52 55 42 20 00 47 65 6f 6d 00 48 61 72 64 20 |GRUB .Geom.Hard|
00000190 44 69 73 6b 00 52 65 61 64 00 20 45 72 72 6f 72 |Disk.Read. Error|
000001a0 0d 0a 00 bb 01 00 b4 0e cd 10 ac 3c 00 75 f4 c3 |.....<.u..|

```

Onde:

1. Endereço do disco;
2. Dados convertidos de hexadecimal para algo mais humano;
3. Número de registros encontrados;
4. Uma entrada que informa que será carregado o bootloader GRUB.

No caso do **GUID Partition Table (GPT)** temos uma lista encadeada em uma área muito ampla, permitindo uma quantidade absurda de tabelas de partição, o GPT tipicamente utiliza 16.384 bytes para armazenar as tabelas, com isso é possível se ter 128 entradas (a letra n na imagem abaixo) nesta lista encadeada, cada um contendo 128 bytes para fazer os apontamentos.



Mas por questões de compatibilidade com aplicações legadas é reservada uma área protegida chamada PMBR. Também é armazenada uma cópia da tabela principal para que caso tenha problemas, seja possível restaurar.

Com 128 entradas é possível ter em um único meio de armazenamento 8ZB, ou seja, 8.000.000.000 TB. Está bem claro que essa é a evolução.

## 13.4 Bootloaders

O bootloader é o primeiro programa software executado quando um computador é inicializado, visto que o processo anterior foi a BIOS/UEFI. É responsável por carregar e transferir o controle para um kernel do sistema operacional, já o kernel, por sua vez, inicializa o resto do sistema operacional.

Existem diversos bootloaders, mas os mais utilizados são:

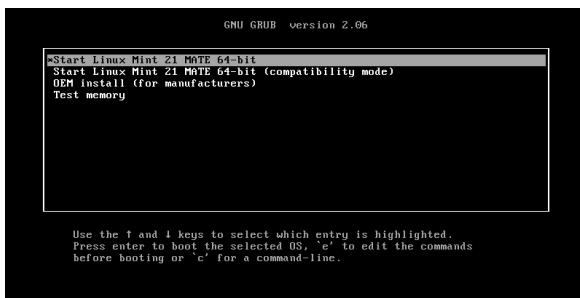
- GNU/LINUX: Lilo, GNU GRUB, etc;
- Mac OS: Chameleon ou rEFIt;
- Windows: NTLDLR;

| Arquitetura        | Bootloader padrão | Bootloaders alternativos                     |
|--------------------|-------------------|----------------------------------------------|
| i386, amd64 (BIOS) | GRUB 2            | LILO, Syslinux, GRUB 1                       |
| i386, amd64 (UEFI) | GRUB-EFI 2        | LILO, Syslinux, GRUB-efi-v2, Kernel EFI stub |

Neste material o foco será o GNU GRUB 2 utilizado em várias distribuições GNU/Linux, incluindo o Debian no momento que escrevo este livro.

### 13.4.1 GNU GRand Unified Bootloader (GRUB)

GNU GRUB é um gerenciador de inicialização muito poderoso, que pode carregar uma ampla variedade de sistemas operacionais, um dos recursos importantes do GRUB é a flexibilidade, GRUB entende sistemas de arquivos e formatos executáveis de kernel, então você pode carregar um sistema operacional arbitrário da maneira que quiser, sem registrar a posição física de seu kernel no disco, assim, você pode carregar o kernel apenas especificando seu nome de arquivo e a unidade e partição onde o kernel reside.



O GRUB 2 lê sua configuração do arquivo **/boot/grub2/grub.cfg** em máquinas tradicionais baseadas em BIOS e do arquivo **/boot/efi/EFI/redhat/grub.cfg** em máquinas UEFI.

Este arquivo contém informações de menu, conforme imagem ao lado.

O arquivo de configuração **grub.cfg**, é gerado durante a instalação ou invocando o utilitário **/usr/sbin/grub2-mkconfig** e é atualizado automaticamente sempre que um novo kernel é instalado (versão). Quando regenerado manualmente usando **grub2-mkconfig**, o arquivo é gerado de acordo com os arquivos de modelo localizados em **/etc/grub.d/**, e as configurações personalizadas no arquivo **/etc/default/grub**.

```
GNU nano 7.2 /boot/grub/grub.cfg
#
DO NOT EDIT THIS FILE
#
It is automatically generated by grub-mkconfig using templates
from /etc/grub.d and settings from /etc/default/grub
#
BEGIN /etc/grub.d/00_header
if [-s $prefix/grubenv]; then
 set have_grubenv=true
 load_env
fi
if ["${next_entry}"] ; then
 set default="${next_entry}"
 set next_entry=
 save_env next_entry

```

Repare que a advertência deixa claro que não se deve editar manualmente este arquivo.

Toda configuração que o usuário pode fazer, ou seja, parametrização é realizado em outro arquivo, este arquivo fica em **/etc/default/grub**, conforme figura abaixo.

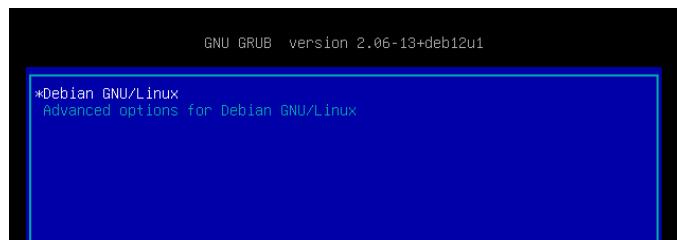
```
GNU nano 7.2 /etc/default/grub
#
If you change this file, run 'update-grub' afterwards to update
/boot/grub/grub.cfg.
For full documentation of the options in this file, see:
info -f grub -n 'Simple configuration'
#
GRUB_DEFAULT=0
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
GRUB_CMDLINE_LINUX=""
#
If your computer has multiple operating systems installed, then you
probably want to run os-prober. However, if your computer is a host
for virtual machines, you probably don't want to run os-prober.
In that case, you can either disable os-prober in /etc/default/grub
or you can run os-prober manually after booting into the host OS.
```

Aqui fazemos todas as customizações.

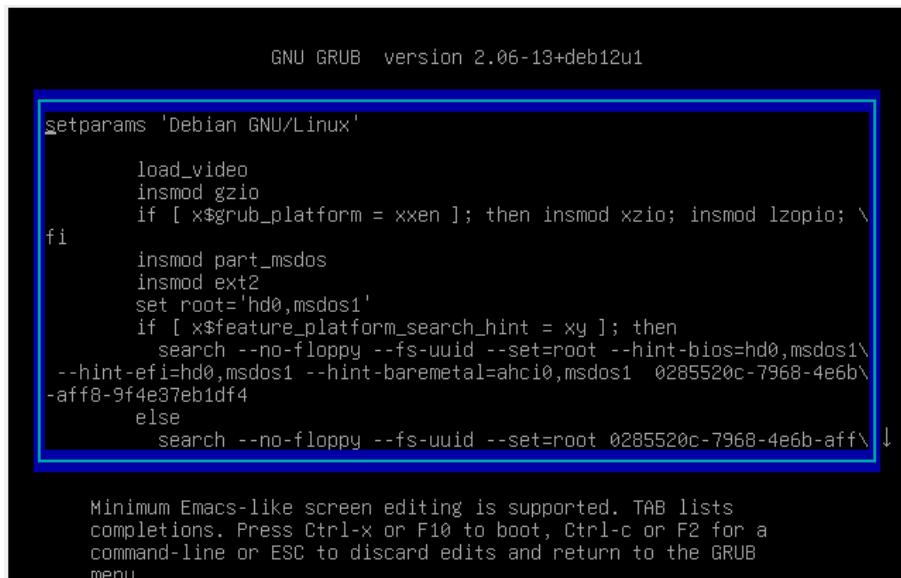
Para atualizar os arquivos de configuração do GRUB utilize o comando **update-grub**, conforme figura abaixo. Este comando também irá atualizar as opções de escolha na tela inicial do seu computador.

```
userlinux@debian:/boot/grub$ sudo update-grub
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.1.0-32-amd64
Found initrd image: /boot/initrd.img-6.1.0-32-amd64
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
done
userlinux@debian:/boot/grub$
```

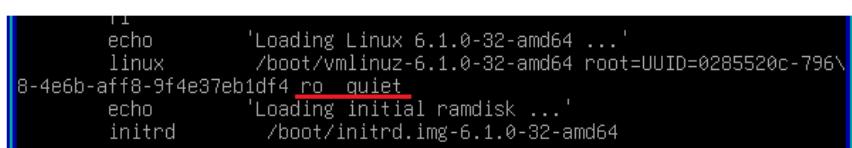
Quando o computador inicializa o GRUB é invocado e é exibido uma lista de opções, no universo Linux não existe a idéia de modo de manutenção, na verdade quando temos algum problema carregamos o Linux com menos pacotes e programas. No exemplo abaixo a primeira opção é a padrão, e também é a opção que carrega o Linux normalmente.



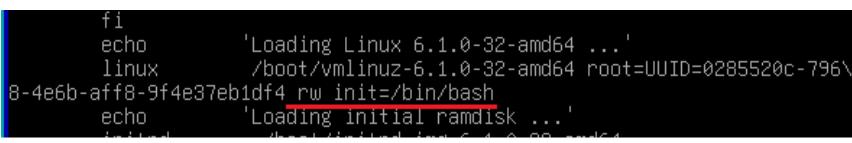
Caso precise fazer uma alteração profunda no seu Linux, como por exemplo corrigir um erro em **/etc/fstab** (é comum iniciantes errarem neste arquivo). Para isso, na opção padrão pressione a tecla **e** do seu teclado, verá que entrará em modo de edição. Aqui você pode mudar a parametrização de inicialização.



Uma das linhas, a linha `linux` (conforme figura abaixo) deve estar sempre como **ro** e **quiet**, vamos alterar este campo, movimente o cursor até este ponto, veja que é como usar o `nano`.



Troque **ro** para **rw** e coloque que a inicialização será feita usando o **/bin/bash** conforme figura abaixo. Agora pressione **F10**.



Veja que o Linux será carregado, nada normal e mesmo que seu Linux seja gráfico o carregamento será terminal, sem cor e com fonte padrão Linux, isso é bom. Sem nada supérfluo é garantido que o Linux iniciará.

```
/dev/sda1: clean, 38061/1774192 files, 550435/7089664 blocks
done.
[3.250722] EXT4-fs (sda1): mounted filesystem with ordered data mode. Quo
mode: none.
done.
Begin: Running /scripts/local-bottom ... done.
Begin: Running /scripts/init-bottom ... done.
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
root@none:/# _
```

Altere o que quiser e poderá reiniciar o computador.

### 13.4.2 Segurança do GRUB

Se realmente procura por segurança, recomendo que utilize um disco LVM criptografado, mas terá que iniciar sempre o sistema operacional manualmente. Para somente aumentar a segurança do GRUB edite **/boot/grub/grub.cfg** adicione as duas linhas a seguir no topo. Isso impede que os usuários editem os itens de inicialização. Já o parâmetro **timeout 3** especifica um atraso de 3 segundos antes GRUB iniciar a configuração padrão.

- 1. **timeout 3**
- 2. **password PASSWORD**

Para reforçar ainda mais a integridade da senha, você pode armazená-la em um formato criptografado. O utilitário **grub-md5-crypt**<sup>78</sup> gera uma senha com hash compatível com o algoritmo de senha criptografada do GRUB (MD5). Para especificar que uma senha no formato MD5 será usada, use a seguinte directiva (**\$1\$**):

- 1. **timeout 3**
- 2. **password --md5 **\$1\$bw0ez\$tljnxxKLfMzmnDVaQWgjP0****

<sup>78</sup> Descrição de criptografia na página oficial:

<https://manpages.debian.org/testing/grub-legacy/grub-md5-crypt.8.en.html>

# 14 Agendamento de tarefas com cron, crontab e at no GNU/Linux (finalizado)

Utiliza-se agendamento de procedimentos no Linux por vários motivos, os principais são:

1. Reduzir o tempo de contato dos administradores em ambientes de produção;
2. Execução de procedimentos fora do horário de alto consumo de serviços;
3. O ser humano vai esquecer, vai por mim!!!

O GNU/Linux assim como o Unix possui o mais conhecido agendador de execuções, o **cron**, e com este pode-se agendar qualquer rotina recorrente, outro comando interessante é o comando **at**, que permite agendamento de uma execução singular.

## 14.2 Daemon cron e comando crontab

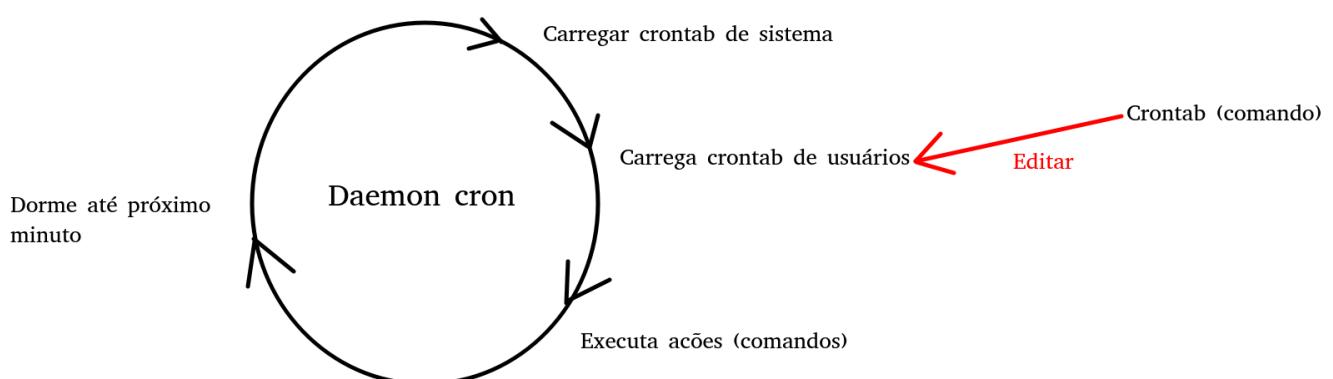
Um dos recursos mais conhecidos no GNU/Linux e Unix é o cron, desenvolvido no final da década de 70 é um poderoso agendador e naturalmente uma solução estável e precisa para agendamentos recorrentes.

### 14.2.1 Daemon cron

Quando o Sistema Operacional inicia um dos primeiros processos que são iniciados é o processo cron, cron é um daemon que é iniciado com o Sistema Operacional e morre com o mesmo.

```
systemd
├── atd
├── cron
├── dbus-daemon
├── dhclient
├── exim4
├── login—bash—pstree
├── rsyslogd—3*[{rsyslogd}]
├── systemd—(sd-pam)
├── systemd-journal
├── systemd-logind
└── systemd-timesync—{systemd-timesync}
└── systemd-udevd
```

Em um laço infinito que ocorre sempre no segundo 0 de cada minuto, o cron carrega arquivos de configuração e executa as tarefas que são chamadas de cronjob.



Existem dois tipos de arquivos de configuração que contém os cronjob, estes são:

**Tarefas de sistema:** Neste arquivo os jobs são dispostos em linhas e encontramos aqui as ações sobre o sistema como um todo;

**Tarefas de usuário:** Cada usuário possui um arquivo que possui as cronjobs específicas para estes usuários;

Estes arquivos podem ser editados tanto com um editor de texto, tal como vimos ou com um comando chamado crontab.

```
usuario@debian:~$ sudo systemctl status cron.service
[sudo] password for usuario:
● cron.service - Regular background program processing daemon
 Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
 Active: active (running) since Tue 2021-07-27 10:01:21 -03; 17s ago
 Docs: man:cron(8)
 Main PID: 327 (cron)
 Tasks: 1 (limit: 2355)
 Memory: 360.0K
 CGroup: /system.slice/cron.service
 └─327 /usr/sbin/cron -f

Jul 27 10:01:21 debian systemd[1]: Started Regular background program processing daemon.
Jul 27 10:01:21 debian cron[327]: (CRON) INFO (pidfile fd = 3)
Jul 27 10:01:21 debian cron[327]: (CRON) INFO (Running @reboot jobs)
usuario@debian:~$ _
```

#### 14.2.1 Comando crontab

O comando **crontab** deve criar, substituir ou editar os **arquivos crontab** de usuários, uma crontab de usuário é um arquivo com uma lista de comandos e os horários em que eles serão executados.

Um novo cronjob, ou seja, uma nova entrada do crontab pode ser inserida no arquivo como uma linha, o arquivo crontab pode ser livremente manipulado por um editor de texto devida a sua sintaxe simples, mas recomenda-se utilizar o comando crontab (comando) para editar o arquivo também chamado crontab.

1. crontab -l

Parâmetros para execução do comando crontab:

- e Edite uma cópia do arquivo crontab do usuário que está invocando o comando, caso não exista o comando cria um novo arquivo vazio para edição. Quando a edição for concluída, o arquivo crontab CÓPIA é então movido para o diretório de crontabs de usuários.
- l Lista a entrada do crontab do usuário que está invocando (l de laranja).
- r Remove o arquivo crontab do usuário que está invocando.

No exemplo a seguir será descrito os passos para que o usuário possa fazer backups automatizadas de seus arquivos, lógico, algo bem simples para exemplo. Primeiro deve-se entender o problema, o problema é “fazer backups periódicas” do diretório do usuário.

Após compreender o problema constrói-se uma solução, que vem como forma de script, no exemplo abaixo vemos um script que faz backup do diretório do usuário, o script deve estar em `~/`, neste exemplo será chamado de **script.sh**.

```

1.#!/bin/bash
2.#nome: script.sh
3.#diretório: ~/
4.#descricao: Faz a backup do diretório do usuário, é apenas um exemplo de aula
5.
6.data=`/bin/date +%d-%m-%Y`;
7.
8.# atenção, vou guardar em /tmp para
9.tar -cvzf /tmp/backup-${LOGNAME}-$data.tar.gz /home/${LOGNAME}/;
10.

```

Como o script é do usuário, não existe um local padrão, neste exemplo estou armazenando apenas no diretório do usuário. Após criar o script será dada a permissão de execução usando o comando:

```
1. chmod +x ~/script.sh
```

Teste o script!!!! Agora edite a crontab (arquivo) do usuário utilizando o comando crontab, conforme comando abaixo.

```
1. crontab -e
```

O comando cron vai criar um arquivo com o LOGIN do usuário no diretório **/var/spool/cron/crontabs/** caso o usuário não possua uma crontab (arquivo), após isso faz uma cópia do arquivo para uma área temporária. Isso é feito pois não se faz alterações diretamente em um arquivo que está constantemente em uso, aprenda ai programadores :).

Após fazer esta cópia então o comando crontab invoca um editor de texto apontando para este arquivo temporário, então o usuário edita este arquivo. Se é a primeira vez que o usuário está utilizando o comando crontab, então o comando questiona sobre qual o editor ele quer utilizar, conforme figura abaixo, pressione **Enter** para continuar.

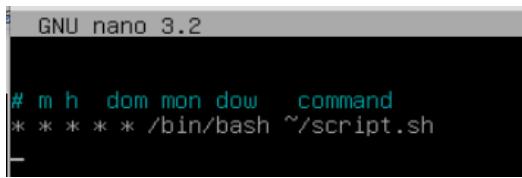
```

usuario@debian:~$ crontab -e
no crontab for usuario - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano <---- easiest
 2. /usr/bin/vim.tiny

Choose 1-2 [1]: _
```

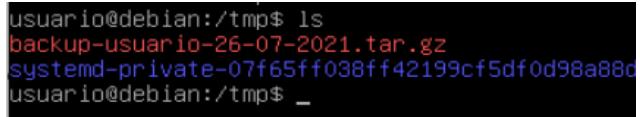
Neste arquivo temporário, para este exemplo digite a linha abaixo, esta linha será explicada ainda neste capítulo, salve o arquivo no seu editor.



```
m h dom mon dow command
* * * * * /bin/bash ~/script.sh
```

Quando o arquivo é alterado então o comando crontab copia de volta o arquivo temporário para o diretório /var/spool/cron/crontabs/ com o nome do usuário, neste caso, conforme figura acima o usuário do sistema se chama **usuario**.

Monitore o diretório /tmp e constate o arquivo de backup, conforme figura abaixo.



```
usuario@debian:/tmp$ ls
backup-usuario-26-07-2021.tar.gz
systemd-private-07f65ff038ff42199cf5df0d98a88d
usuario@debian:/tmp$ _
```

#### 14.2.2 Formato de entradas cronjob na crontab de usuário

Quando foi configurado a crontab do usuário foi adicionado uma linha com o seguinte dado.

|                                                         |
|---------------------------------------------------------|
| <b>1. #min hor dia mês sem comando</b>                  |
| <b>2. * * * * * /bin/bash /home/userlinux/script.sh</b> |

A primeira coluna é referente ao minuto que quer que execute, as entradas possíveis são:

\*, todos os minutos, de 0 até 59;

**Um número**, será executado somente quando o minuto chegar exatamente naquele número;

**Uma sequência de números separados por vírgula**, será executado a cada momento que coincidir o minuto do seu datetime com um destes elementos;

A segunda coluna é referente as horas, as entradas possíveis são:

\*, todas as horas, de 0 até 23;

**Um número**, será executado somente quando a hora chegar exatamente naquele número;

**Uma sequência de números separados por vírgula**, será executado a cada momento que coincidir a hora do seu datetime com um destes elementos;

A terceira coluna é referente ao dia do mês, as entradas possíveis são:

\*, todos os dias, de 1 até 31;

**Um número**, será executado somente quando o dia chegar exatamente naquele número;

**Uma sequência de números separados por vírgula**, será executado a cada momento que coincidir o dia do seu datetime com um destes elementos;

A quarta coluna é referente ao mês do ano, as entradas possíveis são:

\*, todos os meses, de 1 até 12;

**Um número**, será executado somente quando o mês chegar exatamente naquele número;

**Uma sequência de números separados por vírgula**, será executado a cada momento que coincidir mês do seu datetime com um destes elementos;

A quinta coluna é o dia da semana, as entradas possíveis são:

\*, todos os dias da semana, onde 0 é Domingo, 1 Segunda ... mas saiba que 7 também é Domingo.

**Um número**, será executado somente quando o dia da semana chegar exatamente naquele número;

**Uma sequência de números separados por vírgula**, será executado a cada momento que coincidir o dia da semana do seu datetime com um destes elementos;

A sexta coluna é o comando, o comando que será executado caso os valores das colunas anteriores encontrarem correspondência com o date da máquina.

### 14.2.3 A crontab do sistema

A crontab (arquivo) de sistema é destinada aos jobs que afetaram todo o sistema, tal como uma backup geral, atualização, validação, extração de dados para monitoramento, etc..

Não está ligado por padrão a um usuário, mas dá permissão para associar um cronjob específico a um usuário específico, isso é feito pois é saudável para o sistema que se tenha vários usuários para controle dos serviços e processos (conforme vimos no capítulo de usuários Linux).

Então uma crontab (arquivo) de sistema possui uma coluna a mais se comparado com a crontab (arquivo) de usuário, conforme pode-se ver na figura abaixo.

```
usuario@debian:/tmp$ cat /etc/crontab
/etc/crontab: system-wide crontab
Unlike any other crontab you don't have to run the `crontab'
command to install the new version when you edit this file
and files in /etc/cron.d. These files also have username fields,
that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

Example of job definition:
----- minute (0 - 59)
| ----- hour (0 - 23)
| | ----- day of month (1 - 31)
| | | ----- month (1 - 12) OR jan,feb,mar,apr ...
| | | | ----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,
| | | |
* * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || (cd / && run-parts --report /
47 6 * * 7 root test -x /usr/sbin/anacron || (cd / && run-parts --report /
52 6 1 * * root test -x /usr/sbin/anacron || (cd / && run-parts --report /
)
```

Para editar este arquivo, deve-se ter privilégios administrativos, ou ser o root ou estar no arquivo sudoers com permissão para tal, com um editor de texto pode-se editar o arquivo /etc/crontab

#### 14.2.4 Execuções de comandos

Quando um comando é executado o Cron cria um environment, uma caixa de areia para o comando que deverá ser um processo, neste environment que é montado a partir do ponto de instalação do Cron instancia as seguintes variáveis deste environment:

**HOME**: Um nome de caminho do diretório pessoal do usuário.

**LOGNAME**: O nome de login do usuário.

**PATH**: Uma string que representa os caminhos dos utilitários padrões, tal como /bin, /sbin, etc..

**SHELL**: Um nome de caminho do interpretador de comandos;

#### 14.2.5 Permissão de acesso ao crontab

O controle de acesso ao comando crontab está relacionado à existência ou não de dois arquivos e algumas entradas destes arquivos, os arquivos são **/etc/cron.allow** e **/etc/cron.deny**.

Esses arquivos de controle de acesso funcionam juntos da seguinte maneira:

- Se cron.allow existir, apenas os usuários listados neste arquivo podem criar, editar, exibir ou remover arquivos crontab.
- Se cron.allow não existir, todos os usuários podem enviar arquivos crontab, exceto os usuários listados em cron.deny.
- Se cron.allow nem cron.deny existir, os privilégios de super usuário são necessários para executar o comando crontab.

### 14.3 Comando at

O comando at permite agendar comandos para serem executados em um determinado momento. Os trabalhos criados com **at** são executados apenas uma vez. Dependendo da distribuição, **at** pode ou não estar presente em seu sistema GNU/Linux. Se **at** não estiver instalado, você pode instalá-lo facilmente usando o gerenciador de pacotes da sua distribuição. No Debian é muito simples, conforme listagem abaixo.

```
1. sudo apt update -y
2. sudo apt install at -y
```

Agora após instalado vamos demonstrar o uso deste comando.

#### 14.3.1 Criando um agendamento com at

A sintaxe simplificada do comando at é a seguinte:

at [OPÇÕES] TEMPO

O comando at usa a data e a hora (runtime) quando você deseja executar o trabalho como um parâmetro de linha de comando, após pressionar ENTER o comando at entra em um procedimento interativo com o usuário, no exemplo abaixo será criado um agendamento para as 12:00.

```
usuario@debian:~$ at 12:00
warning: commands will be executed using /bin/sh
at>
```

Repare o cursor passa para **at>** ao invés usuário e máquina, que é o padrão Debian, agora pode digitar o comando que quer executar neste horário, conforme exemplo abaixo no qual é criado um arquivo em /tmp.

```
usuario@debian:~$ at 12:00
warning: commands will be executed using /bin/sh
at> touch /tmp/porat.txt
at> _
```

Para sair deste modo interativo basta pressionar **Ctrl+d**, então o comando exibe que o job será criado com um número, este número é “uma espécie de ID” e a data e horário do agendamento.

```
usuario@debian:~$ at 12:00
warning: commands will be executed using /bin/sh
at> touch /tmp/porat.txt
at> <EOT>
Job 2 at Tue Jul 27 12:00:00 2021
usuario@debian:~$ _
```

Outra forma de criar o agendamento é executado diretamente por linha de comando, este procedimento é útil para agendamentos feitos por scripts sem interação, veja o exemplo.

```
usuario@debian:~$ echo "rm /tmp/porat.txt" | at 13:00
```

Ao pressionar ENTER o comando at faz o agendamento e já retorna o output semelhante ao modelo interativo, conforme visto abaixo.

```
usuario@debian:~$ echo "rm /tmp/porat.txt" | at 13:00
warning: commands will be executed using /bin/sh
job 3 at Tue Jul 27 13:00:00 2021
usuario@debian:~$ _
```

Para agendar a execução de um script é fácil, tendo o script já devidamente desenvolvido e estando este script com permissão de execução o usuário simplesmente utiliza o parâmetro **-f** e o caminho do script.

```
usuario@debian:~$ at 14:00 -f /etc/cron.d/script.sh
warning: commands will be executed using /bin/sh
job 4 at Tue Jul 27 14:00:00 2021
usuario@debian:~$
```

### 14.3.2 Especificando o tempo de execução

O comando at aceita uma ampla gama de especificações de tempo, você pode especificar a hora, a data e o incremento a partir da hora atual:

**Hora:** para especificar uma hora, use o formulário HH:MM ou HHMM;

**Data:** o comando permite agendar a execução do trabalho em uma determinada data, a data pode ser especificada usando o nome do mês seguido pelo dia e um ano opcional. Você pode usar nomes, como today, tomorrow, ou dia da semana, mas caso queira a data podem também ser indicados usando os formatos MMDD[CC]YY, MM/DD/[CC]YY, DD.MM.[CC]YY ou [CC]YY-MM-DD.

**Incremento:** o comando at também aceita incrementos no **now + count time-unit**, onde count um número e time-unit pode ser um dos seguintes textos: minutes, hours, days, ou weeks.

Exemplos do uso:

Agende um trabalho para o próxima Segunda-feira em um horário dez minutos depois do horário atual:

```
1. at monday +10 minutes
```

Agende um trabalho para ser executado às 13h em dois dias a partir de agora:

```
1. at 13:00 + 2 days
```

Agende um trabalho para ser executado às 13:30 de **24 de dezembro de 2021**:

```
1. at 13:30 122421
```

### 14.3.3 Listagem de tarefas pendentes

Para listar as tarefas pendentes do usuário, execute o comando at ou at -l. A saída listará todos os trabalhos, um por linha, e cada linha inclui o número do trabalho, data, hora e usuário. Quando o comando é chamado por um usuário administrativo, ele listará os trabalhos pendentes de todos os usuários.

```
usuario@debian:~$ at -l
3 Tue Jul 27 13:00:00 2021 a usuario
2 Tue Jul 27 12:00:00 2021 a usuario
4 Tue Jul 27 14:00:00 2021 a usuario
usuario@debian:~$ _
```

#### 14.3.4 Removendo trabalhos pendentes

Para remover um trabalho pendente, invoque o comando atrm ou at -r seguido pelo número do trabalho. Por exemplo, para remover o trabalho com o número 3, você executaria:

```
1. at -r 3
```

#### 14.3.5 Restringindo usuários

Os arquivos **/etc/at.deny** e **/etc/at.allow** permitem que você controle quais usuários podem criar trabalhos com o comando at. Os arquivos consistem em uma lista de nomes de usuário, um nome de usuário por linha.

Por padrão, apenas o /etc/at.deny arquivo existe e está vazio, o que significa que todos os usuários podem usar o comando at. Se você deseja negar permissão a um usuário específico, adicione o nome de usuário a este arquivo.

Se o arquivo /etc/at.allow existir, apenas os usuários listados neste arquivo podem usar o comando at. Se nenhum dos arquivos existir, apenas os usuários com privilégios administrativos podem usar o comando at.

### 14.4 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para resolver, isto já é levado em consideração.

#### 14.4.1 Prática 4d4f6ae 01: Instalação do comando apt e um agendamento

Nesta prática o aluno deverá instalar o comando at e criar um agendamento.

1. Utilize o apt e instale o comando at;
2. utilize o comando at para executar o comando **touch /tmp/teste.txt** exatamente as **12:00**;

```
1. sudo aied validar 4d4f6ae checkpoint01
```

#### 14.4.2 Prática d7a527b 01: Um script de backup

Nesta prática o aluno deverá criar um script de backup conforme exemplificado neste capítulo e agendar na crontab do usuário usuario.

1. Crie um script chamado **script.sh** e posicione ele em **/home/userlinux/**

2. Neste script edite o código de acordo com o tópico “[Comando crontab](#)”
3. Agende com o comando **crontab** a execução do script para ocorrer todos os dias exatamente às 12:00

```
1. sudo aied validar d7a527b checkpoint01
```

1 - Fazer um script que acha todos os arquivos alterado (na última hora) do tipo (txt, sh, odt) no diretório do usuário ~/ , e então salvar em um relatório /home/userlinux/relatorio.rel, cada linha deve ter o caminho (path) do arquivo bem como horário. Faça esse arquivo relatorio.rel ser um arquivo incremental, apendendo o novo relatório no relatório anterior. Use o agendamento recorrente de hora em hora para isso.

2 - Todo minuto 0 da hora 10 e da hora 15, faça um update no sistema.

3 - A cada minuto, usar agendamento. Faça um relatório de portas abertas no servidor, vai precisar fazer um script com netstat.

# 15 Roteador, Firewall e Gateway

Se o aluno chegou até este ponto, acredito que já tenha estudado a teoria de Redes de Computadores<sup>79</sup>, e o autor recomendado é **Andrew Stuart Tanenbaum**<sup>80</sup>. Conceitos abordados neste ponto do livro levam isso em consideração.

As redes de computadores são muitas vezes complexas, em parte por ser heterogênea quanto aos meios e também pela variedade de protocolos. A Organização Internacional de Normalização<sup>81</sup> (ISO) atuou por anos para reduzir tal complexidade, focando seus esforços nos serviços, nas funções e nos protocolos.

É com base neste esforço que regras e padrões foram definidas. Para agravar a complexidade, nem todos querem atingir os mesmos objetivos, há pessoas que acordam todos os dias pensando em desmontar e agredir os esforços positivos, o que eleva a complexidade com a exigência de segurança. Um elemento importante neste cenário é o Roteador, que naturalmente é peça fundamental no funcionamento e na segurança. Este capítulo é fundamental para quem busca ferramentas para segurança em redes de computadores.

## 15.1 MAC ADDRESS, IP e Port Number

Em algumas camadas do modelo OSI encontramos alguns identificadores importantes que são utilizados para que um PDU chegue até seu destino e usamos estes para criar regras, estes endereços são:

- **Enlace**: MAC Address;
- **Rede**: IP Address;
- **Transporte**: Port Number;

Um endereço de Medium Access Control (MAC) é um identificador único atribuído a uma Network Interface Controller (NIC). Para comunicações dentro de um enlace de rede, é usado como endereço de rede para a maioria das tecnologias de rede IEEE 802, incluindo Ethernet, Wi-Fi e Bluetooth. No modelo Open Systems Interconnection (OSI), os endereços MAC são usados na subcamada de protocolo do Medium Access Control da camada de enlace de dados. Como normalmente representado, os endereços MAC são reconhecíveis como seis grupos de dois dígitos hexadecimais, separados por hífens, dois pontos ou nenhum separador.

Este endereço está fixado na interface, ou seja, no dispositivo de I/O e por isso não pode ser alterado por definitivo, quando se necessita alterar tal endereço esta alteração é feita

---

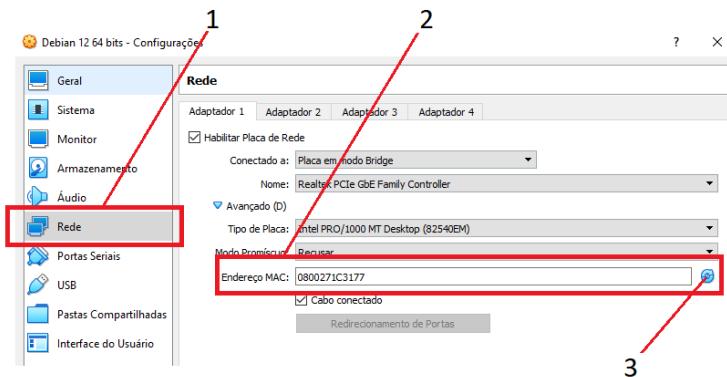
<sup>79</sup> Existe um curso no Youtube que aborda este livro, disponível no link:

[https://www.youtube.com/playlist?list=PL1d\\_HUKMz4Rm3Tqv8sMXSaR3J4rYdzj0e](https://www.youtube.com/playlist?list=PL1d_HUKMz4Rm3Tqv8sMXSaR3J4rYdzj0e)

<sup>80</sup> Sua biografia pode ser obtida neste link [https://pt.wikipedia.org/wiki/Andrew\\_Stuart\\_Tanenbaum](https://pt.wikipedia.org/wiki/Andrew_Stuart_Tanenbaum)

<sup>81</sup> Este livro se baseia no modelo Open Systems Interconnection (OSI) da Organização Internacional de Normalização (ISO);

em memória pois os dispositivos de I/O são mapeados na memória, e naturalmente, a memória é volátil. No VirtualBox, este endereço é editável mas só pode ser editável quando a máquina virtual estiver desligada.



Onde:

1. Configuração de Rede da máquina virtual;
2. O endereço MAC gerado pelo VirtualBox;
3. Botão que randomicamente muda o MAC Address.

Há comandos no Linux que editam o endereço, mas como a Interface de rede é um dispositivo de I/O, o Linux mapeia seus atributos em memória e a memória é volátil.

```
usuario@debian:~$ ip link
1: lo: <LOOPBACK,up,LOWER_UP> mtu 65536 qdisc noqueue state
1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
 link/ether 08:00:27:1c:31:77 brd ff:ff:ff:ff:ff:ff
 usuario@debian:~$
```

```
usuario@debian:~$ ip link set enp0s3 address 64:1C:67:CE:61:CE
usuario@debian:~$
```

Para obter o endereço MAC utilize o comando ip com parâmetro link.

Com o próprio comando ip é possível alterar o endereço, no comando acima, estou alterando de **08:00:27:1C:31:77** para **64:1C:67:CE:61:CE**.

Segundo o padrão IEEE o endereço MAC é dividido em 2 partes, cada parte contendo 24 bits. A primeira parte é chamada de Organizationaly Unique Identifier (OUI), e conforme o próprio nome já diz, é um identificador único por organização desenvolvedora do chipset das interfaces de rede, tal como Realtek, INTEL, Broadcom, etc.. Já a segunda parte é um auto incremento da organização, assim cada OUI de cada fabricante pode produzir 2 elevado 24 interfaces, mas a empresa pode acabar reiniciando a contagem ou até mesmo pedindo novos OUI. É comum ter um OUI para cada produto ou grupo de produtos de uma organização, é desta forma por exemplo, que os hackers sabem que tipo de equipamento estão atacando. Lembre-se que é por endereço MAC que um frame se movimenta na rede local.

O endereço IP versão 4 possui 32 bits<sup>82</sup>, e que estes 32 bits são divididos em 4 octetos, por isso variam de 0 até 255 cada um dos quatro octetos. Teoricamente temos 0.0.0.0 até 255.255.255.255, mas na prática, os endereços foram divididos em classes<sup>83</sup>, algumas destas classes estão em uso, são as classes:

- **Classe A:** 0.0.0.0 até 127.255.255.255
  - **Classe B:** 128.0.0.0 até 191.255.255.255
  - **Classe C:** 192.0.0.0 até 223.255.255.255
  - **Classe D:** 224.0.0.0 até 239.255.255.255 para Endereço multicast
  - **Classe E:** 240.0.0.0 até 247.255.255.255 para uso futuro

Seque reserva de endereços na descrição em bits (Bit-wise representation).

The second type of address, class B, has a 14-bit network number and a 16-bit local address. The two highest-order bits are set to 1-0. This allows 16,384 class B networks.

### Class B Address

The third type of address, class C, has a 21-bit network number and a 8-bit local address. The three highest-order bits are set to 1-1-0. This allows 2,097,152 class C networks.

| 1                                                | 2                                                | 3                                                |
|--------------------------------------------------|--------------------------------------------------|--------------------------------------------------|
| 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1      | 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1      | 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1      |
| +-+-+---+-+-+---+-+-+---+-+-+---+-+-+---+-+-+--- | +-+-+---+-+-+---+-+-+---+-+-+---+-+-+---+-+-+--- | +-+-+---+-+-+---+-+-+---+-+-+---+-+-+---+-+-+--- |
| -+  1 0                                          | NETWORK                                          | Local Address                                    |

### Class C Address

Vamos aos cálculos. segura os bits aí.

<sup>82</sup> Referencia oficial <https://www.rfc-editor.org/rfc/rfc791#section-1.2>

<sup>83</sup> Referência oficial: <https://datatracker.ietf.org/doc/html/rfc870>

Diferente do endereço MAC Address, o endereço IP precisa ser adicionado na interface de rede que vem a sem configuração lógica. Faz parte do projeto de rede definir qual faixa de IP utilizado nos dispositivos e como levar tal configuração para as interfaces, seja manualmente ou por DHCP.

Mas em uma rede privada não é pode utilizar qualquer faixa de IP, por padrão a IANA definiu que em uma rede privada deve-se utilizar uma faixa de endereço conforme listagem abaixo:

- **Classe A:** 10.0.0.0 até 10.255.255.255 com máscara **padrão** 255.0.0.0;
- **Classe B:** 172.16.0.0 até 172.31.255.255 com máscara **padrão** 255.255.0.0;
- **Classe C:** 192.168.0.0 até 192.168.255.255 com máscara **padrão** 255.255.255.0;

Em uma rede cada interface possui um endereço único, mas todas as interfaces possuem a mesma máscara de rede, desta forma a abrangência. A abrangência de uma rede vai depender da sua máscara de rede, então vamos imaginar uma rede Classe C. Um endereço Classe C é reservado a uma rede consideravelmente pequena, podendo ter até 256 endereços dos quais apenas 254 endereços realmente são válidos para os equipamentos. Isso ocorre pois a máscara desta classe possui no mínimo três octetos completamente preenchidos com 1.

Uma máscara da classe C<sup>84</sup> pode variar de 255.255.255.0 até 255.255.255.252<sup>85</sup>, e a alteração da máscara então também altera o escopo da rede/subrede. Por padrão a rede Classe C tem como máscara padrão 255.255.255.0, onde o último octeto conforme já dito possui 8 bits como zero, se calcular 2 elevado a 8 terá 256, ou seja, 256 endereços como já dito.

Independente de ser uma rede Classe A, B ou C, toda rede/sub-rede precisa de um endereço para ela, por exemplo, a rede em que estou neste momento é a rede 192.168.0.0 e o meu computador está usando o endereço 192.168.0.5. Geralmente os equipamentos começam a usar os endereços após o endereço da rede. Saiba também que toda rede/sub-rede precisa de um canal para troca de dados entre os equipamentos, ou seja, dados que não são de uso humano, por convenção este endereço chamado de broadcast é o último endereço da rede/sub-rede.

Então na rede Classe C 192.168.0.0 (endereço da rede) no qual estou no computador 192.168.0.5 (endereço que minha interface de rede usa) também temos o endereço 192.168.0.255 que está em uso do Broadcast. Uma coisa deve ser comum para todos os equipamentos dentro desta rede/sub-rede, que é a máscara. Agora imagine o seguinte problema, temos na rede pessoas que são confiáveis e pessoas que não são confiáveis, podem eles estarem na mesma rede/sub-rede?

Tanto o endereço IP quanto o MAC Address são endereços de uma interface, ou seja, usados para localizar o computador, mas em um computador existem vários programas sendo executados e muitos destes são serviços, mesmo que você esteja em um

<sup>84</sup> Especificação oficial: <https://www.rfc-editor.org/rfc/rfc1878>

<sup>85</sup> Com 2 bits para os Hosts, 2 elevado a 2 é 4 ou seja, 4 endereços disponíveis. Mas como deve-se deduzir o endereço da subrede e o endereço do broadcast, sobra apenas 2 endereços para os hosts

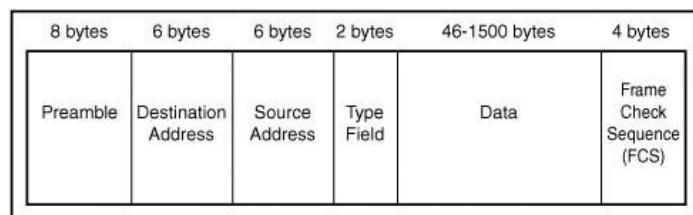
computador de uso doméstico, internamente ele está cheio de serviços que atendem vários processos, locais, ou até remotos.

Então para se localizar um processo a que se destina um T-PDU utiliza-se portas, tais portas são endereços numéricos de 16 bits que podem variar de 0 até 65.535. Muitas destas portas já estão em uso pois são usadas em serviços internos do próprio Sistema Operacional. A lista completa de serviços esperados em relação a portas usadas estão no arquivo **/etc/services**.

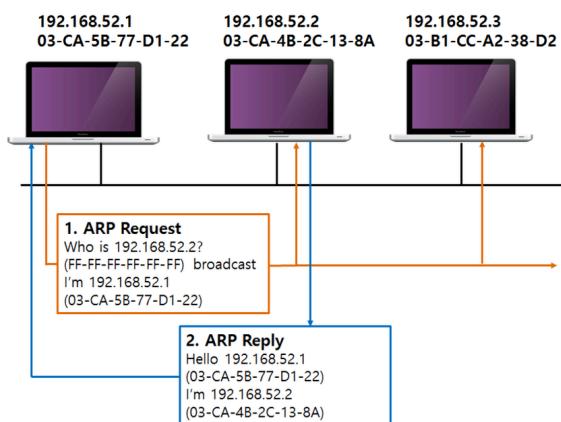
## 15.2 Protocolo Ethernet, IP, TCP e UDP

Quando afirmamos que um protocolo está em trânsito por várias redes, sobre a camada de enlace é complicado afirmar que o protocolo é o Ethernet em todas as redes que este passará, pois é uma camada extremamente dependente do meio físico utilizado na topologia da rede, mas segundo a literatura e a prática é que Ethernet é um protocolo dominante quando falamos em Rede Privada, ou LAN. Ethernet é um protocolo presente em redes cabeadas com UTP por exemplo, e esse meio é barato e eficiente.

Este protocolo possui 8 bytes de enquadramento (Leia o livro de redes do Tanenbaum) seguido de dois endereços de 6 bytes, com 2 bytes para definição de tipo ou tamanho dos dados, seguido dos dados que pode variar de 46 bytes até 1500 bytes, finalizando com 4 bytes de CheckSum (método de validação de sucesso/falha), totalizando no máximo 1526 bytes.



O que é muito comum é o uso dos dois endereços de 6 bytes, utiliza-se tanto para criar negação de DHCP quanto para associar um computador a um nome. Internamente o computador utiliza para comunicação, pois toda comunicação entre dois computadores dentro da mesma Rede Privada ocorre por endereço MAC.

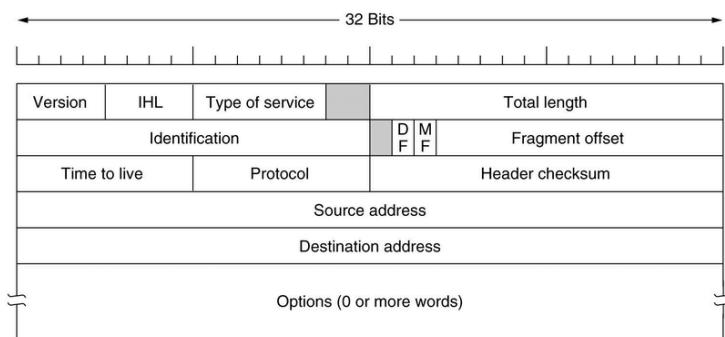


Vamos exemplificar, se um computador 192.168.52.1 (origem) quer se comunicar com o computador 192.168.52.2 (destino) ele calcula se o destino está dentro de sua rede local, para isso ele utiliza a máscara de rede, neste caso 255.255.255.0.

Estão na mesma rede, confirmado, então o computador de origem solicita para a rede o MAC ADDRESS do computador destino por protocolo ARP.

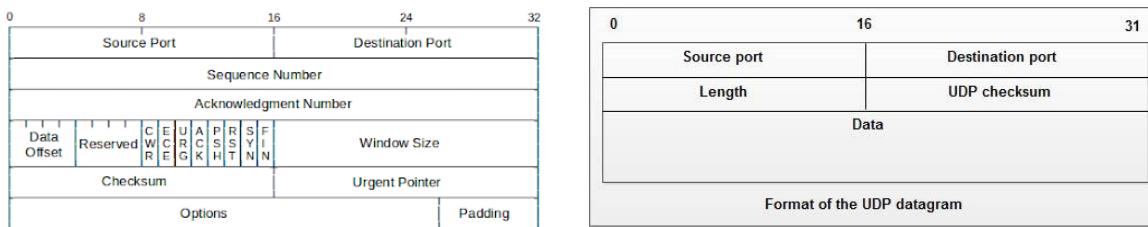
Isso confunde muita gente, pois por mais que use o endereço IP para que dois computadores na mesma rede se comuniquem, internamente há uma tradução de endereços para se obter o MAC Address da máquina que possui o endereço IP de destino. Se o computador de destino estiver fora da rede privada, então é utilizado com endereço MAC Address o endereço de um computador chamado GATEWAY que está na rede, se este não existir, então ocorre um erro.

Um pouco mais acima, na pilha de protocolos OSI temos o protocolo IP que é pertencente a camada de Rede, este protocolo é fundamental para um PDU ir de uma rede para outra rede, ou seja, para comunicações entre computadores em redes diferentes. Um PDU de camada Rede pode assumir dois comportamentos na rede, ou ele opera com um circuito virtual ou como datagrama. Então um PDU de Rede pode ser um Pacote ou um Datagrama, a particularidade entre ambos é que o datagrama possui dois endereços de 32 bits usados por todo o caminho para um PDU sair da origem (primeiro endereço) até chegar no destino (segundo endereço). Já o pacote, este possui um ENDEREÇO DE CIRCUITO após fechar um handshake entre todos os computadores envolvidos e os PDUs usam este endereço de circuito que é virtual para passar todos os pedaços da mensagem. Conforme imagem abaixo.



Muitas regras são criadas sobre esses endereços de 32 bits, pois os equipamentos Firewall estão sempre entre duas redes onde uma delas sempre é mais perigosa, por exemplo, entre sua rede privada e a Internet.

O PDU da camada de transporte, ou T-PDU (na literatura CISCO) pode assumir dois nomes, são: UDP e TCP. Tanto o protocolo TCP quanto UDP são protocolos de ponta a ponta, que devido a sua posição acima da camada de Rede, conseguem então proporcionar uma comunicação direta entre cliente e destinatário. A diferença é que na comunicação TCP tenta-se dar qualidade de conexão, com artifícios de sincronismo, handshake e contadores, fora um extenso algoritmo intrincado de regras para conseguir tal façanha. Já o protocolo UDP nem tenta. Ambos possuem o endereço de porta de 16 bits.



Exemplo de header TCP.

Exemplo de header UDP.

Os protocolos de camada de aplicação se apoiam nestes dois, então há um grupo de protocolos de camada de aplicação que usam TCP e outro grupo de protocolos de camada de aplicação que usam o UDP. Para citar o FTP usa o protocolo TCP, e tem todo o trâmite burocrático de uma comunicação controlada, já o protocolo TFTP utiliza o UDP, qualquer arquivo obtido por TFTP não tem nenhuma certeza de integridade.

Muitas regras são criadas para portas de serviços bem como para protocolos de camada de transporte (UDP e TCP), isso é normal pois queremos receber requisições nos serviços, só queremos que seja controlado e seguro. Quanto aos protocolos de camada de Aplicação, tal como dito FTP e TFTP, estes não podem ser associados a portas, por exemplo, associar a porta número 21 ao FTP, afinal não há obrigatoriedade em se usar as mesmas portas para os mesmos serviços conforme /etc/services. Por isso temos complicações em criar regras, sempre recomendo trabalhar por protocolo por ser mais humano, mas entendo a simplicidade de se operar por porta numérica e certeza de processo aguardando nestas portas numéricas.

Quando configuramos as regras de permissão/negação utilizamos todas as particularidades destes protocolos e destes endereços para criar as regras, de atributos até endereços, de origens ou destinos, tudo levamos em consideração.

## 15.3 Router, Gateway e Firewall

Qualquer distribuição Linux pode servir as três configurações básicas descritas no título. Lembre-se que todo Router é um computador e pode ser configurado com objetivos estratégicos diferentes, e atuar como Gateway, Firewall ou até mesmo Web Application Firewall (WAF).

### 15.1.2 Segmentação da LAN

Segmenta-se a rede quando é fundamental<sup>86</sup>:

- Separar pessoas em uma LAN;
- Isolar aplicações perigosas;
- Organizar aplicações em forma de serviços;
- Reduzir o domínio de Broadcast;

<sup>86</sup> Mais detalhes em <https://ccna.network/segmentacao-de-rede/>

O cálculo é simples, tomo como exemplo a rede 192.168.1.0/24 e a necessidade é de se dividir esta rede em dois segmentos. Para-se compreender este código deve estar claro que cada número destes é um octeto binário. Onde:

**192** é 11000000;

**168** é 10101000;

**1** é 00000001;

**0** é 00000000.

Da mesma forma, na máscara 24 que representa 255.255.255.0 é:

**255** é 11111111;

**0** é 00000000.

Por incrível que pareça, trabalhamos olhando para a máscara e não para o IP neste momento, veja que o último octeto binário é 00000000, logo se precisamos dividir a rede em 2 é necessário trabalhar com o bit mais significativo do octeto que é zero.

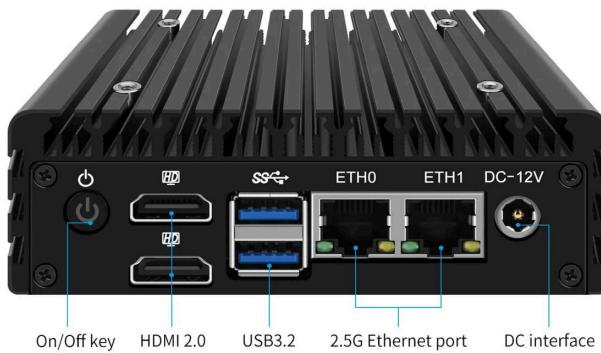
Transformando 00000000 em 10000000 estamos dividindo a rede em duas, já se transformamos em 11000000 estamos dividindo a rede em 4, se usamos 11100000 estamos dividindo a rede em 8, veja, sempre 2 elevado ao número de UNs usado na máscara. Se precisamos dividir a rede em duas, então é 10000000, ou seja, 2 elevado a 1. Ao transformar este último octeto em decimal vamos obter o seguinte número: 128, então a máscara para todas as máquinas dentro dos dois segmentos será 255.255.255.128.

Mas quantas endereços se pode ter em cada segmento? Simples, ainda trabalhando com o último octeto, então deve-se inverter, veja que 10000000 para rede será 01111111 para endereços, e ao converter este último para Decimal se tem 127, cada segmento pode ter 127 endereços então. **Atenção**, lembre-se que cada sub-rede temos que reservar dois endereços, o primeiro para identificar o segmento de rede e o último para Broadcast.

O primeiro segmento desta rede começa então em 192.168.1.0 com máscara 255.255.255.128 e termina em 192.168.1.127, repare que foi adicionado no último octeto 127. Neste segmento não se deve usar 192.168.1.0 e nem 192.168.0.127. No segundo segmento desta rede o primeiro endereço é 192.168.1.128 e o último é o 192.168.1.255, todos os endereços devem ser utilizados com a máscara 255.255.255.128. Mas neste segmento o endereço 192.168.1.128 é reservado bem como 192.168.1.255.

### 15.3.1 Router

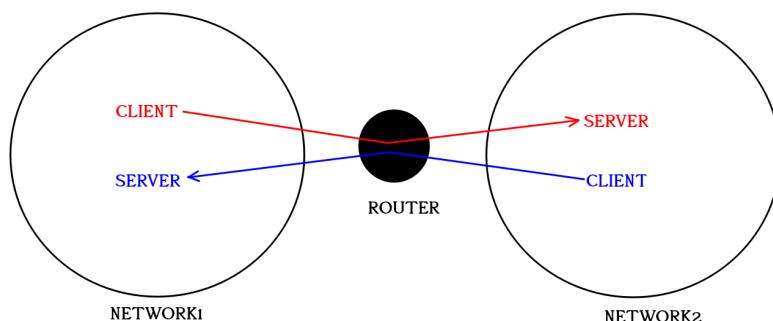
O básico é um Router, este equipamento interliga duas redes/sub-redes e seu foco está no roteamento, dentro de uma empresa ele é muito útil na segmentação da rede. Então precisa de ter no mínimo 2 interfaces de rede, na figura abaixo temos uma imagem de um mini-pc com duas interfaces de rede Ethernet.



Lembre-se que equipamentos em redes/sub-redes diferentes precisam de um elemento intermediário para a comunicação. Todo router tem uma tabela interna de roteamento, e existem dos tipos de redes para o router:

- Rede conectada diretamente;
- Rede remota (distante);

A diferença em termos de tratativa administrativa é que as redes conectadas diretamente automaticamente são reconhecidas pelo Router e este então redireciona naturalmente, já as redes distantes, estas precisam da intervenção administrativa criando regras de roteamento.



Imagine que uma empresa possui dois tipos de usuários, usuários administrativos e funcionários, ou que esta rede seja muito grande, então é natural que vamos precisar segmentar a rede. Segundo a CISCO uma rede é segmentada quando:

- É grande demais, e a comunicação de broadcast compete com os usuários;
- Quando diferente tipo de usuários devem coexistir na no mesmo ambiente;
- Quando aplicativos são inseguros;
- Quando queremos isolar serviços e aplicativos.

O uso do Router em uma rede empresarial é feito na interligação entre redes, no qual aplicações clientes e aplicações servidoras podem estar em qualquer uma das redes que o Router interliga.

### 15.3.2 Firewall

Sendo simplista, um firewall atua na filtragem de pacotes criando um perímetro de segurança, e consiste em uma lista de regras de aceitação e negação. Essas regras definem explicitamente quais pacotes serão ou não permitidos através da interface de rede. As regras de firewall usam os campos de cabeçalho do pacote descritos no livro de Redes

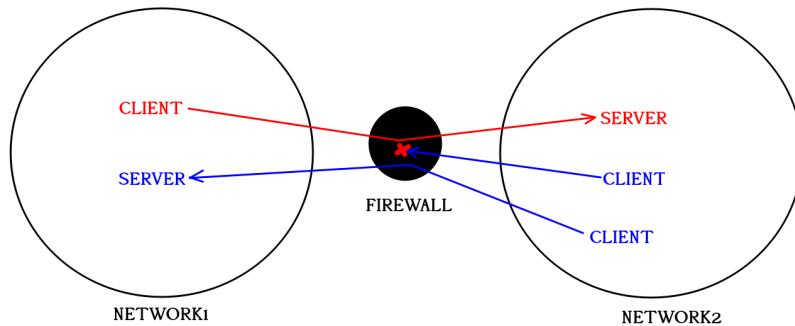
de Computadores do autor Andrew Stuart Tanenbaum. Usando um modelo de referência TCP/IP, um firewall de filtragem de pacotes funciona nas camadas de Rede e Transporte.

Decidir se encaminhará um pacote para seu destino, descartar o pacote silenciosamente ou bloquear o pacote e talvez retornar uma mensagem de erro à máquina remetente. Essas regras podem ser baseadas em uma ampla gama de fatores, incluindo a origem ou o destino Endereços IP, as portas de origem e (mais comumente) de destino e partes de pacotes individuais, como os sinalizadores de cabeçalho TCP, os tipos de protocolo, o endereço MAC e muito mais.

A ideia geral é que você precisa controlar com muito cuidado o que se passa entre a Internet e a máquina que você conectou diretamente à Internet. Na interface externa para a Internet, você filtra individualmente o que entra de fora e o que sai da máquina da maneira mais exata e explícita possível.

Nem todos os protocolos de comunicação de aplicativos se prestam à filtragem de pacotes. Esse tipo de filtragem é de nível muito baixo para permitir autenticação e controle de acesso refinados. Esses serviços de segurança devem ser fornecidos em níveis mais elevados. O IP não tem a capacidade de verificar se o remetente é quem afirma ser. A única informação de identificação disponível neste nível é o endereço de origem no cabeçalho do pacote IP.

Um nível acima, nem a camada de Rede nem a camada de Transporte podem verificar se os dados do aplicativo estão corretos. No entanto, o nível do pacote permite um controle maior e mais simples sobre o acesso direto à porta, o conteúdo dos pacotes e os protocolos de comunicação corretos, o que pode ser feito de maneira fácil ou conveniente em níveis mais elevados.



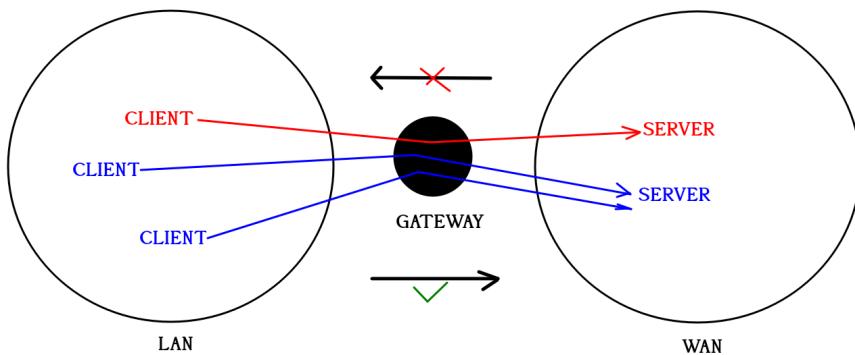
Sem filtragem em nível de pacote, as medidas de filtragem de nível superior e de segurança de proxy são prejudicadas ou potencialmente ineficazes. Até certo ponto, pelo menos, elas devem confiar na correção do protocolo de comunicação subjacente. Cada camada na pilha de protocolos de segurança adiciona outra peça que outras camadas não podem fornecer facilmente.

Vamos imaginar a prática comum das empresas, que é crescer seu parque de computadores sem nenhum planejamento e de forma caótica, a primeira etapa sempre é aplicação de segmentação e organização por meio de Router simples. Então as empresas começam a prestar a atenção e percebem que inseguranças existem na rede, e então

começam a criar regras nos Routers, estes passam então a ter a função de interligação (Router) e filtragem (Firewall).

### 15.3.3 Gateway

Imagine a seguinte necessidade, de uma rede menos abrangente e mais controlada (como uma LAN), e a partir desta rede fosse possível iniciar conexões tendo como destino uma rede mais abrangente (como uma WAN), toda conexão então poderia ser feita, ser enviada e obtida a resposta. Mas nenhuma conexão pode ser ORIGINADA dentro da rede mais abrangente (como a WAN) tendo como destino um host dentro da rede menos abrangente (como uma LAN).



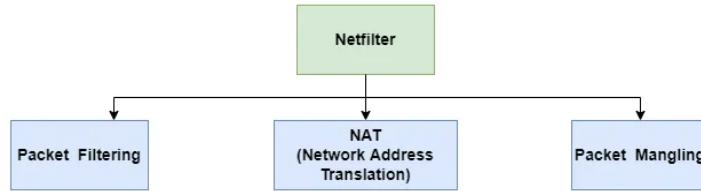
Veja que a grosso modo poderia dizer que é um FIREWALL, mas há particularidades, no FIREWALL todas as conexões são estudadas e todos os serviços também, então a negação ou permissão é de ambas para ambas, aqui não.

Gateway é um importante equipamento usado para dar segurança a toda uma rede menos abrangente. Também são equipamentos simples com no máximo três regras, no caso do Linux. Toda empresa possui um bem como toda casa também, a diferença é que a caixinha de plástico oferecida pela operadora para as redes caseiras não possuem todo o potencial dos gateways empresariais e naturalmente, do Linux.

Neste sentido foi concebido um Linux chamado DD-WRT para routers caseiros, no qual é possível acessar serviços complexos só ofertados por um verdadeiro Sistema Operacional.

## 15.4 Projetos Netfilter

A iniciativa [netfilter.org](http://netfilter.org) desenvolve software para o kernel Linux, que é lançado sob os termos da Licença Pública Geral GNU versão 2 (GPL-2.0) e licenças compatíveis. Este projeto também fornece bibliotecas e comandos lançados sob a GPL-2.0. O projeto netfilter é um projeto FOSS colaborativo conduzido pela comunidade que fornece software de filtragem, modificação e redirecionamento de pacotes. O projeto netfilter é comumente associado ao iptables e seu sucessor, nftables.



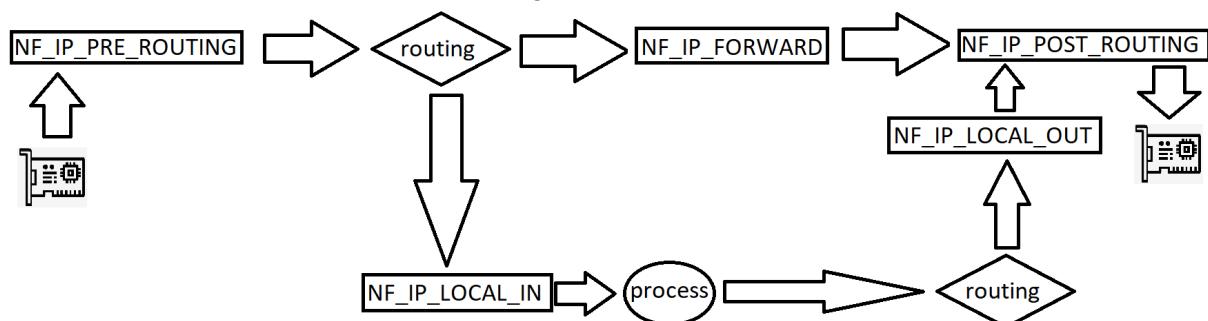
O projeto netfilter permite filtragem de pacotes, tradução de endereços e portas de rede, registro de pacotes, enfileiramento de pacotes no "userspace" e outras manipulações de pacotes. Os **hooks netfilter** são uma estrutura dentro do **kernel Linux** que permite que os módulos do kernel registrem funções de retorno de chamada em diferentes locais da pilha de rede. A função de retorno de chamada registrada é então chamada de volta para cada pacote que atravessa o respectivo hook na pilha.

Existem cinco netfilter hooks nos quais os programas podem registrar eventos. À medida que os pacotes avançavam pela pilha, eles acionaram os módulos do kernel que foram registrados com esses hooks, disparando eventos. Os hooks que um pacote irá disparar dependem se o pacote está entrando ou saindo, o destino do pacote e se o pacote foi descartado ou rejeitado em um ponto anterior.

Os hooks a seguir representam esses pontos bem definidos na pilha de rede para **IPv4**:

- **NF\_IP\_PRE\_ROUTING**: Este hook será acionado por qualquer tráfego de entrada logo após entrar na pilha da rede. Esse hook é processado antes de qualquer decisão de roteamento ser tomada em relação ao local para onde enviar o pacote;
- **NF\_IP\_LOCAL\_IN**: Este hook é acionado após um pacote receberd ter sido roteado se o pacote for destinado ao sistema local;
- **NF\_IP\_FORWARD**: Este hook é acionado após um pacote receberd ter sido roteado se o pacote for encaminhado para outro host;
- **NF\_IP\_LOCAL\_OUT**: esse hook é acionado por qualquer tráfego de saída criado localmente assim que atinge a pilha da rede;
- **NF\_IP\_POST\_ROUTING**: Este hook é acionado por qualquer tráfego de saída ou encaminhado após o roteamento ter ocorrido e pouco antes de ser enviado pela rede;

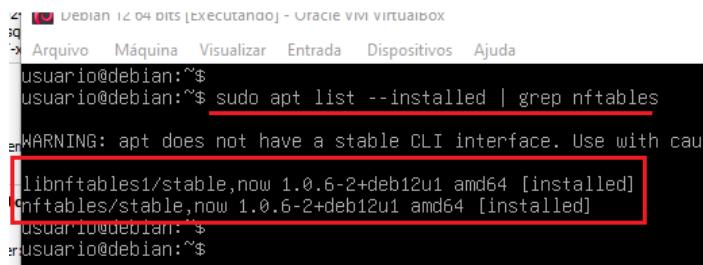
A sequência de hooks está definida na figura abaixo.



Sendo assim, para uma mensagem entrar no host e ser levada até um processo local (tal como sshd), deve entrar pela interface de rede passar pelo hook **NF\_IP\_PRE\_ROUTING** não ser roteada e ir para hook local **NF\_IP\_LOCAL\_IN** e então chegar até o processo.

Os módulos do kernel que precisam ser registrados nesses hooks também devem fornecer um número de prioridade para ajudar a determinar a ordem em que serão chamados quando o hook for acionado. Isso fornece os meios para que vários módulos (ou múltiplas instâncias do mesmo módulo) sejam conectados a cada um dos hooks com ordem determinística. Cada módulo será chamado por sua vez e retornará uma decisão ao netfilter framework após o processamento que indica o que deve ser feito com o pacote.

O iptables é um software de firewall genérico que permite definir conjuntos de regras. Cada regra dentro de uma tabela consiste em vários classificadores (correspondências de iptables) e uma ação conectada (alvo de iptables), já o nftables é o sucessor do iptables, permite uma classificação de pacotes muito mais flexível, escalável e de maior desempenho. Neste livro tanto Iptables quanto Nftables serão devidamente descritos e exemplificados para que o profissional aprenda a lidar com qualquer distribuição Linux de qualquer versão. Para saber se tem o nftables instalado (por exemplo), utilize o comando **apt** com a opção **list** e **--installed**, se tiver os pacotes **nftables** e **libnftables** estarão instalados e prontos para o uso em seu GNU/Linux.



```
Debian 12/04 DTS [executando] - Oracle VIVI VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
usuario@debian:~$ sudo apt list --installed | grep nftables
WARNING: apt does not have a stable CLI interface. Use with caution.
libnftables1/stable,now 1.0.6-2+deb12u1 amd64 [installed]
nftables/stable,now 1.0.6-2+deb12u1 amd64 [installed]
usuario@debian:~$ usuario@debian:~$
```

Se não tem nftables, procure por iptables.

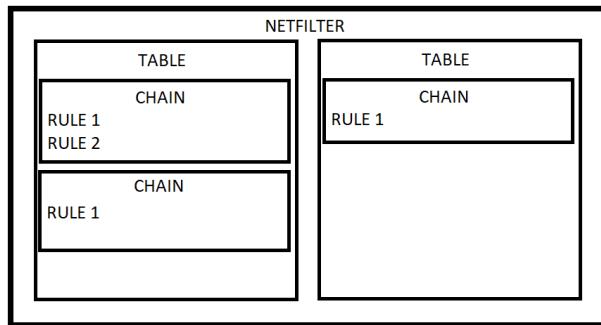
#### 15.4.1 Pacote Iptables

O subsistema de processamento de pacotes de rede em algumas versões do kernel Linux é chamado Iptables. Como o Netfilter e o Iptables estão fortemente acoplados, vou usar “Iptables” para se referir a um ou ambos ao longo deste tópico.

A arquitetura Iptables agrupa o processamento de pacotes de rede com baseado em regras e estas estão organizadas em tabelas, as tabelas por sua vez possuem um objetivo específico, que pode ser:

- packet filtering;
- network address translation;
- packet mangling.

Nas tabelas temos agrupamentos de regras que chamamos de chains, um chain é executado com base em um determinado tipo de evento, por exemplo, entrada, saída, redirecionamento, etc.. O esquema abaixo exemplifica a organização dos elementos, table, chain e rule.



Regras consistem em matches que por exemplo podem ser utilizados para determinar quais pacotes a regra se aplicará e uma target que determinam o que será feito com os pacotes correspondentes a regra. Estas regras Iptables operam no modelo OSI na **camada 3** (Network), na **camada 2** (Enlace), mas é possível utilizar outras tecnologias como ebtables e também é possível criar regras para portas ou protocolos. Um exemplo de regra iptables:

```
1. iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j DNAT --to-destination 192.168.1.3:8080
```

Onde,

- t nat Opera na tabela nat;
- A PREROUTING Anexa a chain PREROUTING;
- i eth1 Pacotes que estão chegando pela interface 1;
- p tcp Usa o modelo TCP/IP em especial o protocolo tcp;
- dport 80 Porta na interface eth1;
- j DNAT Vá para;
- to-destination 192.168.1.3:8080 Destino na rede interna;

Conforme já dito existem tabelas, e nestas tabelas temos as chains definidas, cada tabela possui um objetivo específico sobre o fluxo de dados que passa nela interface de rede, são elas:

- **FILTER**: Usado para definir políticas para o tipo de tráfego permitido na I/O de tráfego;
- **NAT**: Usado para redirecionar conexões para *network address translation*, normalmente com base nos endereços de origem ou destino;
- **MANGLE**: Usado para alteração de pacotes especializados, como remoção de opções do protocolo IP;

O netfilters e em especial o Iptables define pontos que são chave no processamento das regras, estes são 5 pontos (em inglês chamado de hook point):

- PREROUTING (hook **NF\_IP\_PRE\_ROUTING**);
- INPUT (hook **NF\_IP\_LOCAL\_IN**);
- FORWARD (hook **NF\_IP\_FORWARD**);
- POSTROUTING (hook **NF\_IP\_POST\_ROUTING**);
- OUTPUT (hook **NF\_IP\_LOCAL\_OUT**);

Chains (lista de regras) são anexadas a estes pontos e com isso é possível anexar sequências de regras nestes pontos e cada regra representa uma oportunidade de afetar ou monitorar fluxo de pacotes.

Uma regra iptables consiste em um ou mais critérios de correspondência que determinam quais pacotes de rede ela afeta e uma especificação de destino que determina como os pacotes de rede serão afetados. O sistema mantém contadores de pacotes e bytes para cada regra. Cada vez que um pacote atinge uma regra e corresponde aos critérios da regra, o contador de pacotes é incrementado e o contador de bytes é aumentado pelo tamanho do pacote correspondente.

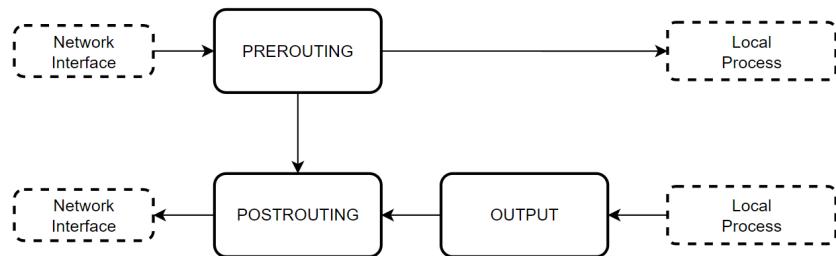
#### 15.4.1.1 A tabela NAT e seu uso

NAT é a tabela que possui regras para modificação dos endereços e ou portas dos pacotes de rede à medida que eles passam pelo netfilter. O computador que está executando as regras de NAT sobre os pacotes pode ser a fonte ou destino dos próprios pacotes, ou pode ser um computador na rota entre a origem e o destino.

Principais usos aplicados no tráfego de rede:

- Redirecionar requisições da rede mais abrangente para a menos abrangentes;
- Mascarar requisições internas para o da rede menos abrangente para a rede mais abrangente;
- Alterar porta , principalmente para uso em proxy transparente;
- Balanceamento de carga de uma rede menos abrangente para duas redes mais abrangentes;

A tabela NAT pode ser usada para realizar uma variedade de funções úteis com base nas manipulações de endereços e portas. Essas funções podem ser agrupadas com base em endereços de origem e destino.



Nesta tabela temos as seguintes chains envolvidas:

- **PREROUTING**: Adicionamos aqui as regras que manipulam os envelopes antes do processo de redirecionamento ou filtro;
- **POSTROUTING**: Adicionamos aqui as regras que serão executadas na saída do envelope;
- **OUTPUT**: Regras para os envelopes originados na própria máquina que possui esta regra, criada por processos locais;

A lógica NAT iptables permite que os módulos (plug-in) ajudem a lidar com pacotes para protocolos que incorporam endereços nos dados que estão sendo trocados (portas por

exemplo). Sem o módulo auxiliar, os pacotes seriam modificados para ir para hosts diferentes apenas.

Um exemplo de uso é definido de **Source NAT (SNAT)**, é usado para compartilhar uma única conexão de Internet entre computadores em uma rede, o computador conectado à Internet atua como um **gateway** e usa SNAT para reescrever pacotes para conexões entre a Internet e a rede interna. O endereço de origem dos pacotes de saída é substituído pelo endereço IP estático da conexão de Internet do gateway, e quando computadores externos responderem, eles definirão o endereço de destino para o endereço IP da conexão de Internet do gateway, e o gateway irá interceptar esses pacotes, alterar seus endereços de destino para o computador interno correto e encaminhá-los para a rede interna.

Visto que SNAT envolve a modificação dos endereços de origem e/ou portas dos pacotes antes de saírem do kernel, ele é executado por meio da cadeia POSTROUTING da tabela nat. Existem duas maneiras de realizar SNAT com iptables:

- Gateway com IP estático;
- Gateway com IP dinâmico;

A extensão de destino SNAT é destinada a situações em que o computador gateway possui um endereço IP estático e a extensão de destino MASQUERADE é destinada a situações em que o computador gateway possui um endereço IP dinâmico. A extensão de destino MASQUERADE fornece lógica adicional que trata da possibilidade de a interface de rede ficar offline e voltar a funcionar com um endereço diferente. Sobrecarga adicional está envolvida nesta lógica, portanto, se você tiver um endereço IP estático, deverá usar a extensão de destino SNAT.

Você pode configurar SNAT na interface **enp0s8** colocando uma regra simples na cadeia POSTROUTING da tabela nat:

```
1. iptables -t nat -A POSTROUTING -o enp0s8 -j SNAT
```

O comando correspondente para mascaramento é:

```
1. iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE
```

Outro uso é o **Destination NAT (DNAT)**, este expõe serviços específicos em uma rede interna ao mundo externo sem conectar os computadores internos diretamente à Internet, desde que não haja mais de um serviço a ser exposto em uma determinada porta, apenas uma conexão de Internet é necessária. O computador gateway redireciona as conexões às portas especificadas para os computadores e portas internas designadas e organiza o retorno do tráfego para o endereço original fora da rede.

Uma vez que o DNAT envolve a modificação dos endereços de destino e/ou portas dos pacotes antes de serem roteados para processos locais ou encaminhados para outros computadores, ele é executado por meio da cadeia PREROUTING da tabela nat.

Por exemplo, para encaminhar conexões de entrada que chegam na porta 80 de um gateway (HTTP) para um servidor da web interno em execução na porta 8080 de 192.168.200.2, você pode usar uma regra como esta:

```
1. iptables -t nat -A PREROUTING -i enp0s8 -p tcp --dport 80 -j DNAT --to-destination 192.168.200.2:8080
```

Outro uso comum para tabela nat é a implantação de proxy transparente é uma forma de interceptar conexões de saída específicas e direcioná-las a um computador que as atenderá no lugar do computador de destino original, esta técnica ser descrita, exemplificada e cobrada no capítulo [Proxy Squid-cache](#).

Essa técnica permite que você configure proxies para serviços sem ter que configurar cada computador na rede interna. Uma vez que todo o tráfego para o mundo externo passa pelo gateway, todas as conexões com o mundo externo na porta fornecida serão proxy transparentemente.

Se você tiver um proxy HTTP (como o Squid) configurado para ser executado como um proxy transparente em seu computador com firewall e escutar na porta 8888, poderá adicionar uma regra para redirecionar o tráfego HTTP de saída para o proxy HTTP:

```
1. iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

É mais complicado fazer proxy de forma transparente para um serviço executado em um host diferente, você pode encontrar detalhes sobre como fazer isso funcionar para o Squid.

Você pode distribuir a carga por vários hosts participantes usando a enésima extensão de correspondência e a extensão de destino DNAT. O balanceamento de carga é um refinamento da distribuição de carga que envolve o uso de estatísticas de carga para os hosts de destino para aconselhar a escolha do destino para os pacotes, a fim de manter os hosts participantes próximos e igualmente carregados.

#### 15.4.1.2 A tabela Filter e seu uso

A função da tabela filter é restringir/permitir tanto o fluxo de pacotes para dentro de uma máquina, para fora de uma máquina e também pela máquina, regras devem ser aplicadas não só para a entrada e a saída, mas também pode ser aplicada na passagem dos pacotes.

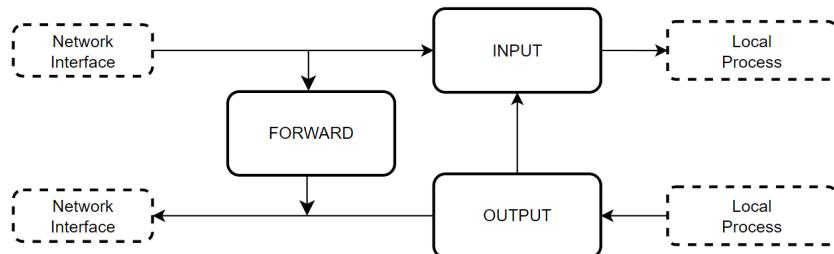
A table filter é tão requisitada que por padrão caso não seja informado no comando iptables qual tabela foi escolhida o comando automaticamente envia a regra para a table filter, visto que a maioria das regras se aplicam às permissões.

Principais usos aplicados no tráfego de rede:

- Bloquear ou permitir acesso a uma porta na máquina local;
- Bloquear ou permitir a passagem de requisições pelo redirecionamento forward;
- Bloquear ou permitir acesso a uma rede específica, tanto da rede mais abrangente para a menos quanto da rede menos abrangente para a mais abrangente;

- Bloquear ou permitir um protocolo específico;

Toda a teoria de firewall pode ser aplicada utilizando este recurso de iptables, um tópico para este assunto será reservado neste capítulo.



A table filter possui as seguintes chains:

- INPUT: realiza o filtro sobre pacotes **destinados** a processos locais da máquina;
- OUTPUT: realiza filtro sobre pacotes **originados** na máquina;
- FORWARD: realiza filtro sobre pacotes que **passam** pela máquina;

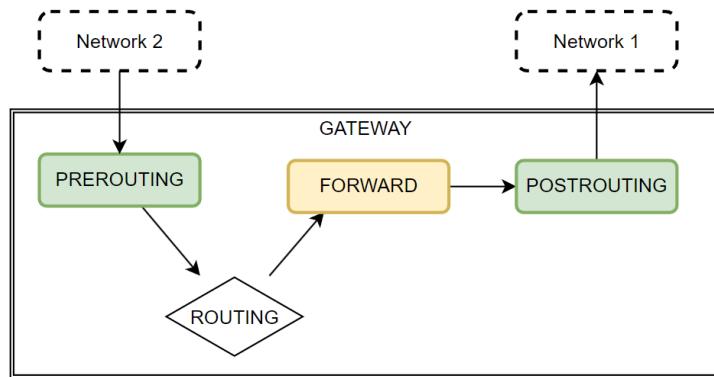
Um firewall é um computador gateway que restringe o fluxo do tráfego de rede entre as redes que ele conecta, também é comum o redirecionamento de determinados pacotes para outra máquina para uma devida tratativa ou armazenamento de LOG.

Existem inúmeras ferramentas que atuam neste sentido, mas uma simples iptables com tcpdump resolve o problema, as principais tables e chains envolvidas são:

- table filter, seguintes chains:
  - INPUT;
  - OUTPUT;
  - FORWARD;
- table nat, seguintes chains:
  - PREROUTING;
  - POSTROUTING;

Por padrão se uma máquina gateway possui a configuração “Masquerade” na interface externa (mais abrangente) é natural que da rede mais abrangente (externa) as requisições são negadas para a rede menos abrangente (interna) caso a conexão tenha origem na rede externa.

No esquema abaixo a NETWORK 1 é mais abrangente, para que sua requisição avance para a NETWORK 1 é necessário aplicar uma regra em PREROUTING alterando o IP de destino para o IP utilizado na máquina interna na NETWORK 2.



Onde:

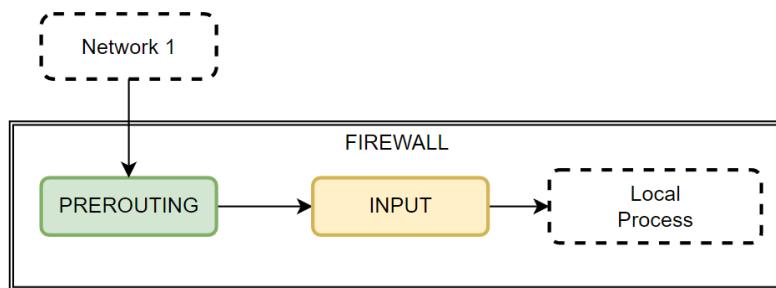
- PREROUTING da table NAT;
  - FORWARD da table FILTER;
  - POSTROUTING da table NAT.

Imagine que a NETWORK 2 (rede interna) está disponibilizando um serviço na porta 80, tal como Apache já discutido neste curso, para que a máquina Gateway aceite a requisição e envie a requisição para a máquina Interna, requisiçõe estas iniciadas na NETWORK 1 (externa), é preciso uma regra Destination NAT conforme exemplo abaixo.

Neste exemplo a rede externa está acessível pelo Gateway pela Interface de rede enp0s3 e a rede interna está acessível pela enp0s8, tendo o apache na NETWORK 2 com IP definido 192.168.200.2 e a porta 80, pode-se definir a seguinte regra.

```
1. iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT --to-destination 192.168.200.2:80
2. iptables -t filter -A FORWARD -p tcp -d 192.168.200.2 --dport 80 -j ACCEPT
```

Uma prática comum é a liberação de uma porta de uma máquina para uma determinada rede, embora o autor não recomenda uma solicitação comum é a abertura da porta 22 para o OpenSSH-server de uma determinada máquina, por ser uma requisição para a máquina local então a tabela trabalhada será a filter.



Onde:

- PREROUTING da table NAT;
  - INPUT da table FILTER;

Utilizando o ambiente anterior, vamos liberar o ssh da máquina Gateway para a Rede I mais abrangente, para isso no arquivo /usr/local/sbin/gateway.sh adicione a seguinte em destaque na figura abaixo, e logo em seguida reinicie a máquina virtual.

```
GNU nano 3.2 /usr/local/sbin/gateway.sh

#!/bin/bash
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT --to-destination 192.168.200.2:80

iptables -A INPUT -i enp0s3 -p tcp --dport 22 -j ACCEPT
```

Para verificar é muito simples, basta executar o comando iptables conforme listagem abaixo.

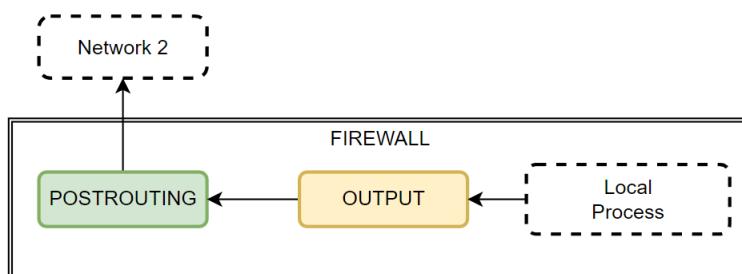
1. sudo iptables -L

Veja na figura abaixo que a table filter foi listada e existe a regra para a porta 22 (ssh).

```
[sudo] password for usuario:
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED
```

Mesmo que a requisição seja gerada por um processo interno, está sujeita a aplicação das regras, mesmo que o normal seja permitir tudo que saia por default. No esquema abaixo temos a sequência básica de CHAINs para saída de uma comunicação originada no processo P1.



Onde:

- POSTROUTING da table NAT;
- OUTPUT da table FILTER;

Na figura do tópico anterior é possível ver que na CHAIN INPUT existe uma regra para a porta 22 mas não existe uma saída, também é importante uma regra de OUTPUT para conexões iniciadas dentro ou fora.

```
GNU nano 3.2 /usr/local/sbin/gateway.sh

#!/bin/bash
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT --to-destination 192.
iptables -A INPUT -i enp0s3 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o enp0s3 -p tcp --sport 22 -j ACCEPT
```

Na figura acima em destaque encontra-se a regra que permite que conexões com porta de origem é 22, quando aplicado veja na figura abaixo que a CHAIN OUTPUT possui uma regra de saída 22.

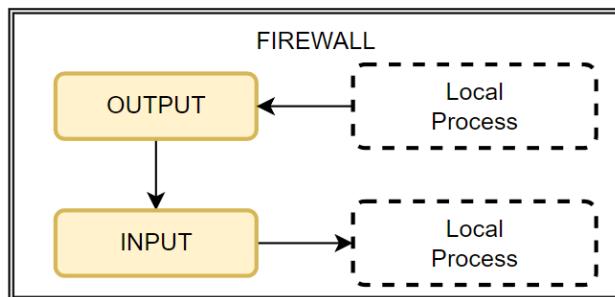
```
usuario@debian:~$ sudo iptables -L
[sudo] password for usuario:
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere state RELATED,ES
ACCEPT tcp -- anywhere anywhere state RELATED,ES
tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere state RELATED,ES

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
ACCEPT tcp -- anywhere anywhere tcp spt:ssh
```

Há 3 formas de se organizar um Sistema Operacional, o mais clássico é um monolítico sem hierarquia de funcionalidades, a segunda forma é a organização das funcionalidades em camadas, em ambas a comunicação entre processo ocorre por arquivos, environment e pipes. A TERCEIRA organização é baseada em serviços, a comunicação entre os processos ocorre por um modelo Cliente-Servidor interno.

O trunfo para permitir essa comunicação é liberar todas as portas na interface de loopback, isso deverá criar 2 regras uma em INPUT e OUTPUT.



Onde:

- INPUT da table FILTER;
- OUTPUT da table FILTER;

Veja como aplicar as regras por comando iptables.

```
GNU nano 3.2 /usr/local/sbin/gateway.sh
#!/bin/bash
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT --to-destination 192
iptables -A INPUT -i enp0s3 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o enp0s3 -p tcp --sport 22 -j ACCEPT

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

#### 15.4.1.3 Política padrão

Caso nenhuma regra se aplique ao PDU que está trafegando então as regras padrão serão aplicadas, que por padrão em um Debian GNU/Linux é ACCEPT. Isso é muito permissivo e é com fundamento um ponto a se criticar nesta distribuição.

Se todas as regras estão corretas, se o SSH está liberado ou tem-se acesso ao dispositivo, então altere as 3 regras padrões para DROP.

```
GNU nano 3.2 /usr/local/sbin/gateway.sh
#!/bin/bash
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT --to-destination 192
iptables -A INPUT -i enp0s3 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o enp0s3 -p tcp --sport 22 -j ACCEPT

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

#### 15.4.1.4 Limpando as regras em vigor

Caso queira organizar todas as regras já discutidas em um script, é comum a necessidade de executar o script mais de uma vez enquanto na fase de testes, recomenda-se então que se limpe as regras no início do script, conforme figura abaixo, use -F e -X para eliminar as regras e purgar da memória.

```
GNU nano 3.2 /usr/local/sbin/gateway.sh
#!/bin/bash

iptables -t filter -F
iptables -t filter -X
iptables -t nat -F
iptables -t nat -X

iptables -A TINPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

#### 15.4.1.5 Regras baseadas em flags TCP

O protocolo TCP possui uma série de flags que são utilizados para manter a orientação da conexão (se não conhece este termo, procure ler o livro do Tanenbaum). O protocolo já inicia executando três passos, tais passos formam o Three-way Handshake. Durante o processo de transferência de dados tais campos também são utilizados para manter o status da conexão e naturalmente comandar operações no outro lado.

São Flags do protocolo TCP:

**SYN**: Quando uma conexão começa ela realiza três passos, tais passos são chamados de Three-way Handshake. O flag Synchronization é usado no primeiro passo. Somente o primeiro pacote do remetente e também do destinatário deve ter esse flag definido. Isto é usado para sincronizar o número de sequência, ou seja, para informar à outra extremidade qual número de sequência eles devem aceitar;

**ACK**: Já o Acknowledgement, é usado para reconhecer pacotes que foram recebidos com sucesso pelo host. O sinalizador será definido se o campo do número de confirmação contiver um número de confirmação válido;

**FIN**: Finish é utilizado para solicitar o encerramento da conexão por parte do cliente, ou seja, quando não há mais dados do remetente, ele solicita o encerramento da conexão;

**RST**: Reset é usado para encerrar a conexão se o remetente se algo está errado com a conexão TCP ou que a conexão não deveria existir. Ele pode ser enviado do lado do receptor;

**URG**: Urgent é utilizado para indicar que os dados contidos no pacote devem ser priorizados e tratados com urgência pelo receptor. Este flag é usado em combinação com o campo Urgent Pointer;

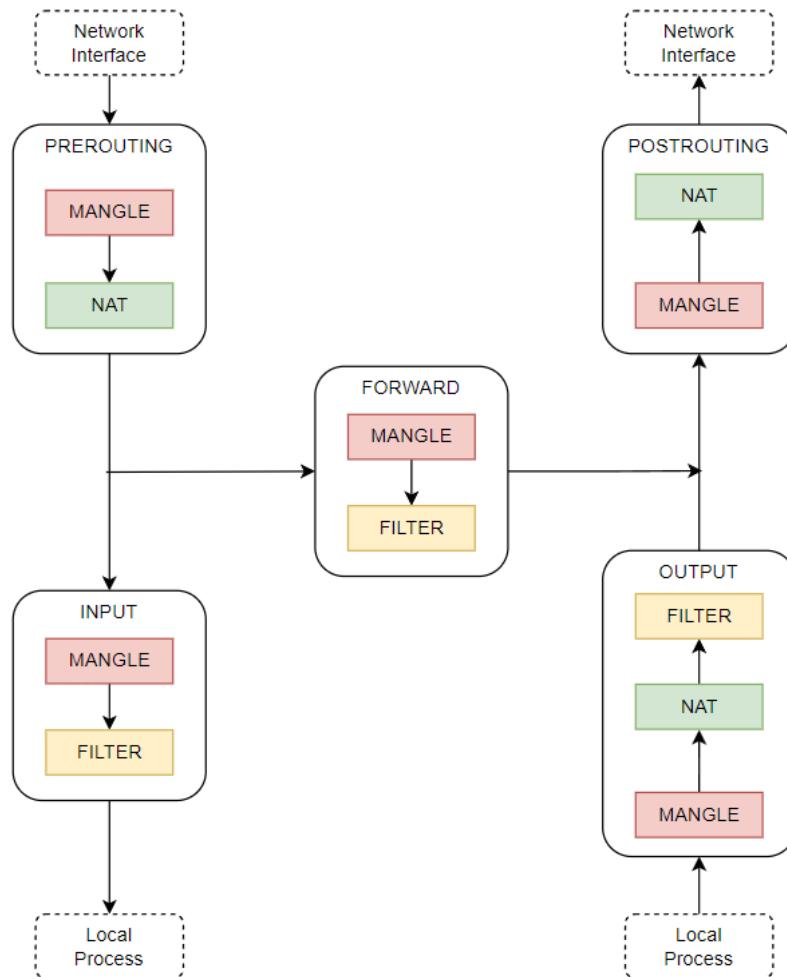
**PSH**: Push é usado para solicitar a entrega imediata de dados ao host receptor, sem esperar que dados adicionais sejam armazenados em buffer no lado do remetente, usado com frequência para streaming de áudio ou vídeo em tempo real;

Deliberadamente hackers utilizam tais parâmetros para mapear serviços em um servidor, mapeando portas, tenho um bom link para entender este processo que foge do escopo deste texto, para isso acesse [este link](#). Veja exemplos de regras que derrubam (DROP) pacotes TCP por flags.

```
1. iptables -A INPUT -p tcp -tcp-flags FN,URG,PSH FIN,URG,PSH -j DROP
2. iptables -A INPUT -p tcp -tcp-flags ALL ALL -j DROP
3. iptables -A INPUT -p tcp -tcp-flags ALL NONE -j DROP
4. iptables -A INPUT -p tcp -tcp-flags SYN,RST SYN, RST -j DROP
```

#### 15.4.1.6 Flow de tabelas e chains Iptables

Além da sequência de eventos hook também temos uma segunda dimensão que vai ditar a precedência (PRIORIDADE) das chains, cada chain possui uma prioridade. Um exemplo é o hook **NF\_IP\_LOCAL\_OUT** que dispara as chains OUTPUT, mas várias tables possuem a chain OUTPUT, então qual a precedência? O grupo netfilter criou todas as tables e chains, só temos o trabalho de adicionar as regras, então é natural que essa decisão de precedência já foi tomada, que no caso do hook **NF\_IP\_LOCAL\_OUT** é a chain OUTPUT da table **MANGLE**, seguido da chain OUTPUT da table **NAT** e por último a chain OUTPUT da table **FILTER**.



Na figura acima temos as chains definidas pelos hooks e naturalmente a prioridade de cada tabela.

#### 15.4.2 Pacote Nftables

O nftables é um subsistema do kernel Linux que fornece filtragem e classificação de pacotes de rede/datagramas/quadros. Ele está disponível desde o kernel Linux 3.13 lançado em 19 de janeiro de 2014. nftables substitui as partes legadas do iptables do Netfilter. Entre as vantagens do nftables sobre o iptables está menos duplicação de código e extensão mais fácil para novos protocolos, também o nftables é configurado por meio do comando `nft`, enquanto as ferramentas legadas são configuradas por meio dos utilitários `iptables`, `ip6tables`, `arptables` e `ebtables` frameworks.

O nftables utiliza os blocos de construção da infraestrutura do Netfilter, como os ganchos existentes na pilha de rede, sistema de rastreamento de conexão, componente de enfileiramento de espaço de usuário e subsistema de registro.

O pacote nftables vem para substituir iptables e naturalmente não coexistindo no mesmo kernel, isso irá causar um afastamento do público do iptables ao passar dos anos. Nftables também incorpora **ip6tables** para tratar IPv6, **arptables** para filtragem de ARP na camada de enlace e **ebtables** que é utilizado para filtragem nas pontes Ethernet.

É natural que com tantos recursos a mais se comparado com iptables, o nftables possui uma sintaxe de comandos diferentes, e também temos que entender que ao passar dos anos o Linux passou a ser normatizado. Para começar o comando iptables deve ser trocado para o comando nft. Ao contrário do iptables, o nftables não inclui nenhuma tabela integrada, e cabe ao administrador determinar quais tabelas são necessárias e adicioná-las seguidas pelas regras de processamento. A sintaxe do comando é:

1. **nft** <comando> <subcomando> <chain> <rule>

Onde os comandos possíveis são:

- **add**: Adiciona no final;
- **list**: Lista os elementos;
- **insert**: Insere em uma posição;0
- **delete**: Deleta uma posição;
- **flush**: Limpa os elementos da listagem.

Já os subcomandos podem ser:

- **table**: Operações sobre as tabelas;
- **chain**: Operações sobre as chains;
- **rule**: Operações sobre as regras.

O pacote nftables inclui alguns recursos semelhantes a linguagens de programação, como a capacidade de definir variáveis e incluir arquivos externos, o nftables também pode ser usado para filtrar e processar vários tipos de endereços. Tais endereços são:

- **ip**: IPv4 addresses;
- **ip6**: IPv6 addresses;
- **inet**: Tanto IPv4 e IPv6 addresses;
- **arp**: Address Resolution Protocol (ARP) addresses;
- **bridge**: Processamento para pacotes em ponte.

Dentro das regras você especifica os critérios de correspondência para uma determinada regra e um veredicto ou decisão sobre o que deve acontecer com um pacote que corresponda a essa regra, as regras nele criadas usam várias instruções e expressões para criar a definição. As instruções **nftables** são semelhantes às instruções do iptables e geralmente afetam como o pacote será processado, seja interrompendo o processamento, enviando o processamento para outra cadeia ou simplesmente registrando o pacote.

Declarações e veredictos incluem o seguinte:

- **accept**: aceita o pacote e interrompe o processamento;
- **continue**: Continue processando o pacote;
- **drop**: interrompe o processamento e descarta silenciosamente o pacote;
- **goto**: envia o processamento para a cadeia especificada, mas não retorna para a cadeia de chamada;
- **jump**: envia o processamento para a cadeia especificada e retorna à cadeia de chamada quando terminar ou quando uma instrução de retorno for executada;
- **limit**: processa o pacote de acordo com a regra se o limite de pacotes recebidos correspondentes for atingido;

- **log**: registra o pacote e continua o processamento;
- **queue**: interrompe o processamento e envia o pacote para o processo do espaço do usuário;
- **reject**: interrompe o processamento e rejeita o pacote;
- **return**: envia o processamento de volta para a cadeia de chamada.

Expressões de carga útil são aquelas coletadas de informações de pacotes. Por exemplo, existem certas expressões de cabeçalho, como sport e dport que se aplicam a pacotes TCP e UDP e não fazem sentido nas camadas IPv4 e IPv6, uma vez que essas camadas não usam portas. As metas expressões podem ser usadas para regras que se aplicam amplamente ou estão vinculadas a pacotes comuns ou propriedades de interface.

São exemplos de expressões usadas:

- **iif**: A interface de rede que recebe a requisição (índice);
- **iifname**: A interface de rede que recebe a requisição (nome);
- **iiftype**: Tipo de interface que recebe o pacote;
- **length**: Comprimento do pacote em bytes;
- **mark**: The packet mark
- **oif**: A interface de rede que irá gerar o pacote (índice);
- **oifname**: A interface de rede que irá gerar o pacote (nome);
- **priority**: Prioridade do pacote;
- **protocol**: O protocolo EtherType;

#### 15.4.2.1 Criando tabelas e chains

Para iniciar deve-se criar as tabelas, e como já dito uma diferença entre iptables e nftables é que nftables não possui nenhuma table ou chain pré-definida. E antes que venham os curiosos me perguntar, pela falta de criatividade e devido a tal fama do iptables, criamos tabelas com os mesmos nomes.

Então para exemplificar, será criado uma table filter e dentro da filter uma chain INPUT.

```
1. sudo nft add table filter
2. sudo nft add chain ip filter input '{ type filter hook input priority 0; }'
```

Para criar um firewall, precisará abrir ou fechar portas, o procedimento é simples, conforme listagem.

```
1. nft add table inet nat
2. nft add table inet filter
3.
4. nft add chain inet nat prerouting '{ type nat hook prerouting priority -100 ; }'
5. nft add chain inet filter forward '{ type filter hook forward priority 0; }'
6.
7. nft add rule inet nat prerouting iifname enp0s3 tcp dport 80 dnat ip to
192.168.200.5:8080
8. nft add rule inet filter forward iifname enp0s3 oifname enp0s8 ip daddr
192.168.200.5 tcp dport 8080 accept
```

Para fazer o mascaramento SNAT, siga a listagem abaixo.

```

1. nft add table inet nat
2.
3. nft add chain inet nat postrouting '{ type nat hook postrouting priority 100 ; }'
4. nft add rule inet nat postrouting oif enp0s3 masquerade

```

## 15.5 Segmentando uma rede com Router

Existe uma necessidade comum nas grandes organizações que é a segmentação de uma rede, esta necessidade ocorre por pelo menos um dos motivos abaixo:

- Necessidade de segregar grupos usuários da rede;
- Necessidade de isolar aplicações;
- Necessidade de reduzir o broadcast dos sistemas;

No exemplo a seguir, será criado um Router Linux com regras básicas, segmentando uma rede em dois segmentos.

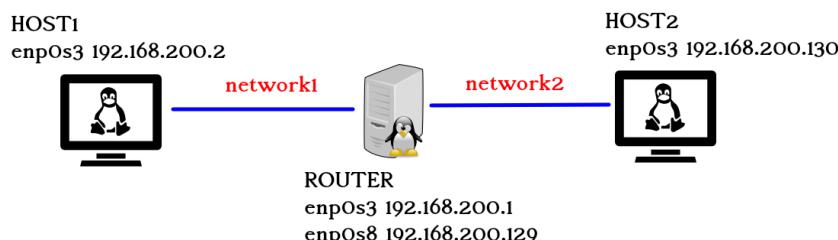
A rede segmentada é a rede 192.168.200.0/24, como será preciso apenas dois segmentos a máscara deverá ser modificada em todas as interfaces de 255.255.255.0 para 255.255.255.128 totalizando 25 bits.

Isso irá segregar duas subredes, a primeira iniciando em 192.168.200.0/25 indo até 192.168.200.127/25, já a segunda parte, indo de 192.168.200.128/25 até 192.168.200.255/25.

Mas lembre-se que em ambos os seguimentos, tanto o primeiro e o último endereço são reservados, sendo assim, endereços úteis para os equipamentos são:

- **Network 1:** 192.168.200.1/25 até 192.168.200.126/25;
- **Network 2:** 192.168.200.129/25 até 192.168.200.254/25;

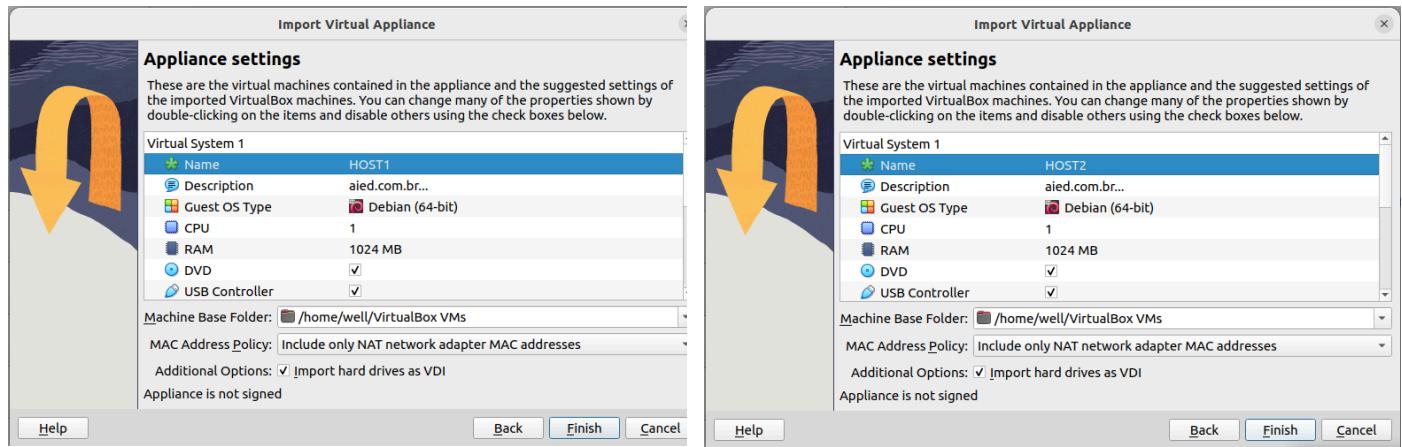
Segue abaixo uma ilustração de como será a topologia bem como a distribuição dos endereços.



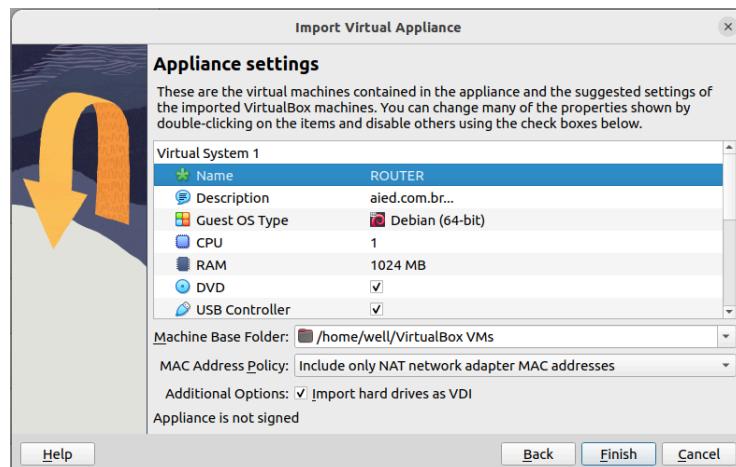
Neste exemplo serão necessárias três máquinas virtuais, são estas:

- Host1: máquina cliente ligada na rede network1;
- Host2: máquina cliente ligada na rede network2;
- Router: servidor que permite a conexão de ambas as redes, e por isso possui duas interfaces de rede;

Comece importando o arquivo .ova inicial (criado no Capítulo 1) duas vezes, altere o nome das importações para HOST1 e HOST2, conforme figuras abaixo.

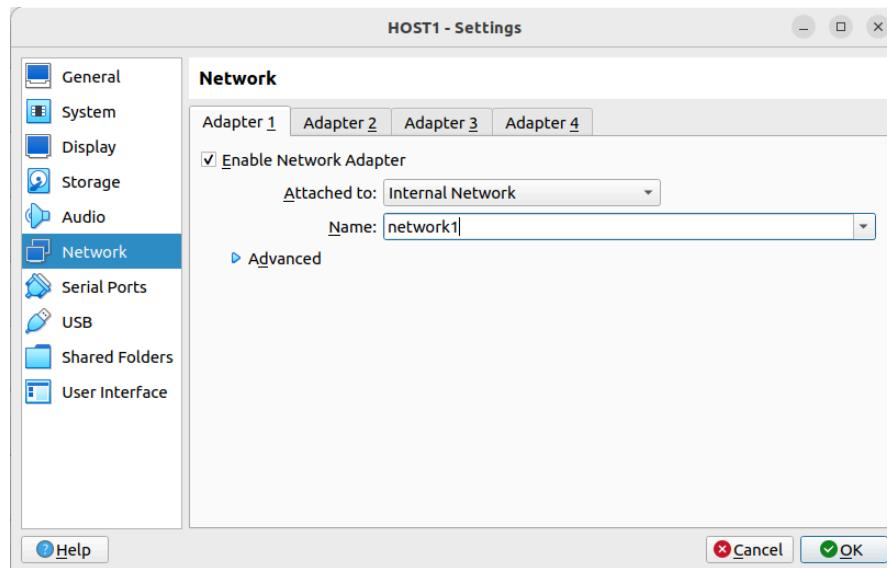


O próximo passo é importar mais uma vez o arquivo .ova, mas desta vez dê o nome de ROUTER, conforme figura abaixo.

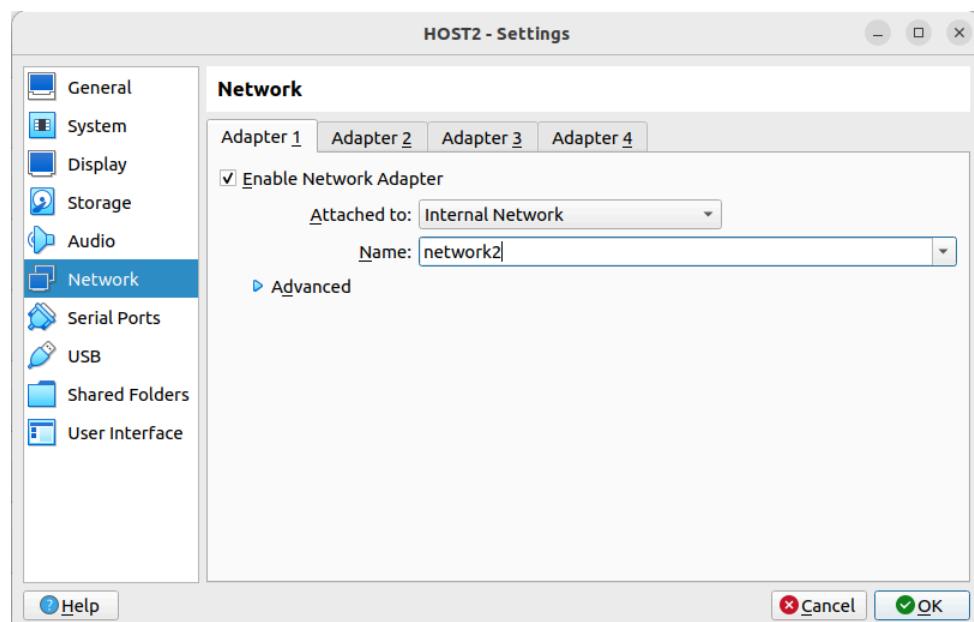


Com as três máquinas criadas, o próximo passo é alterar a configuração física dos adaptadores de rede, criando assim as redes: network1 e network2. No HOST1 acesse a

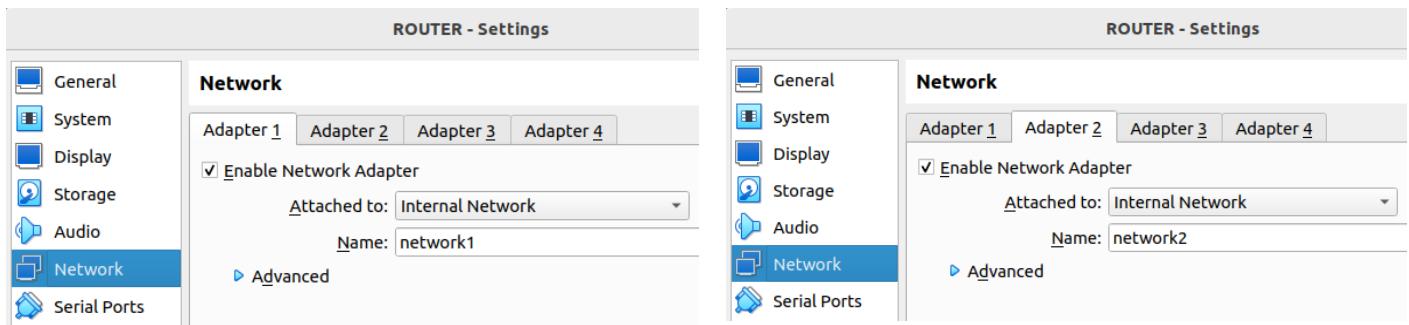
configuração e na aba Network habilite somente o Adapter 1, para este adapter selecione Internal Network e digite o nome **network1**, sim, tem que ser digitado corretamente.



Já para o HOST2 o Adapter 1 deve ter como rede a **network2**, conforme figura abaixo.



Já o ROUTER precisa de dois Adapters, onde o Adapter 1 estará ligado na rede **network1** e o Adapter 2 ligado na rede **network2**.



### 15.5.1 Configuração lógica

Até o momento foi realizado o projeto físico, seguimos agora para o projeto lógico que é a configuração dos endereços e serviços, para começar inicie a máquina virtual HOST1 e com o nano abra para edição o arquivo `/etc/network/interfaces`, configure da seguinte forma.

```
GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.2
 netmask 255.255.255.128
 broadcast 192.168.200.127
 gateway 192.168.200.1
 dns-nameservers 8.8.8.8
```

Tanto o HOST1 quanto o HOST2 possuem apenas um Adapter, ou seja, o primeiro Adapter, que quando o Linux é iniciado assume o nome de `enp0s3`, por isso a configuração deve ser realizada nesta interface. O endereço dado é um pertencente a `network1` que inicia em `192.168.200.1/25` até `192.168.200.126/25`, será escolhido o endereço `192.168.200.2/25` para manter o padrão que é: o primeiro endereço útil para hosts é dado ao router.

Da mesma forma, para o HOST2 será escolhido o endereço `192.168.200.130/25` pois é o segundo endereço válido na `network2`.

```
GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.130
 network 192.168.200.128
 netmask 255.255.255.128
 broadcast 192.168.200.255
 gateway 192.168.200.129
 dns-nameservers 8.8.8.8
```

Já a máquina virtual ROUTER possui dois adaptadores de rede, onde o Adapter 1 será a interface enp0s3 e o Adapter 2 a interface enp0s8. Como é o router entre as duas redes, terá como endereço o primeiro endereço útil de cada uma das redes.

Os endereços são 192.168.200.1/25 para a enp0s3 e 192.168.200.129/25 para a interface enp0s8, repare na configuração de HOST1 e HOST2 que ambos os endereços são usados como gateway.

ROUTER [Running] - Oracle VM VirtualBox

|      |         |      |       |         |      |
|------|---------|------|-------|---------|------|
| File | Machine | View | Input | Devices | Help |
|------|---------|------|-------|---------|------|

```
GNU nano 7.2 /etc/network/interfaces
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

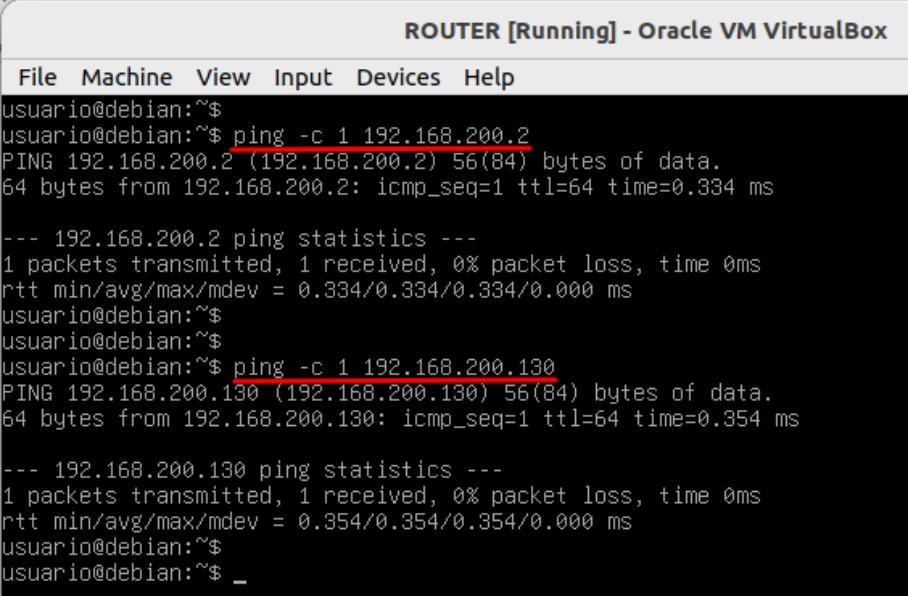
source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.1
 network 192.168.200.0
 netmask 255.255.255.128
 broadcast 192.168.200.127

auto enp0s8
iface enp0s8 inet static
 address 192.168.200.129
 network 192.168.200.128
 netmask 255.255.255.128
 broadcast 192.168.200.255
```

Reiniciando todas as três máquinas virtuais, verá que é possível da ROUTER enviar um protocolo ICMP para ambas as máquinas HOSTs, conforme figura abaixo. Esse teste é importante pois avalia que tanto a parte física quanto a parte lógica de endereçamento está funcionando.



```

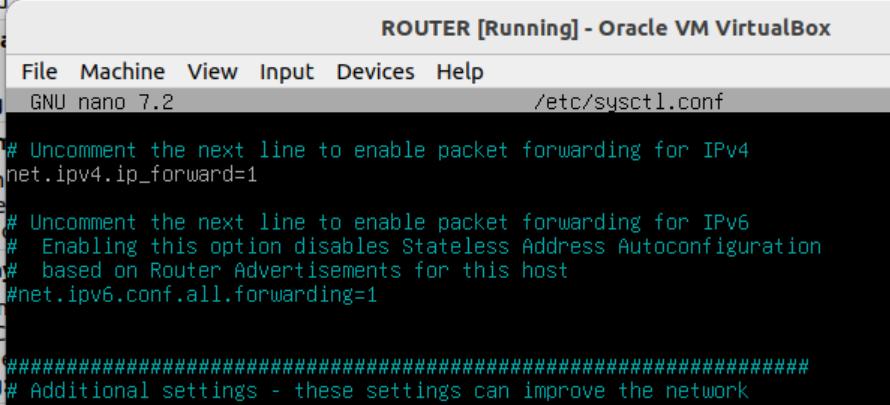
ROUTER [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~$
usuario@debian:~$ ping -c 1 192.168.200.2
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.
64 bytes from 192.168.200.2: icmp_seq=1 ttl=64 time=0.334 ms

--- 192.168.200.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.334/0.334/0.334/0.000 ms
usuario@debian:~$
usuario@debian:~$
usuario@debian:~$ ping -c 1 192.168.200.130
PING 192.168.200.130 (192.168.200.130) 56(84) bytes of data.
64 bytes from 192.168.200.130: icmp_seq=1 ttl=64 time=0.354 ms

--- 192.168.200.130 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.354/0.354/0.354/0.000 ms
usuario@debian:~$
usuario@debian:~$

```

No GNU/Linux para que uma mensagem passe de uma interface de rede para outra é necessário ativar a opção `net.ipv4.ip_forward` e atribuir o valor 1, e caso queira IPv6 a opção `net.ipv6.ip_foward` também com o valor 1. Para esta prática será utilizado somente IPv4 e por isso abra o arquivo `/etc/sysctl.conf` para edição, e habilite o atributo `net.ipv4.ip_forward=1` conforme figura abaixo.



```

ROUTER [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/sysctl.conf

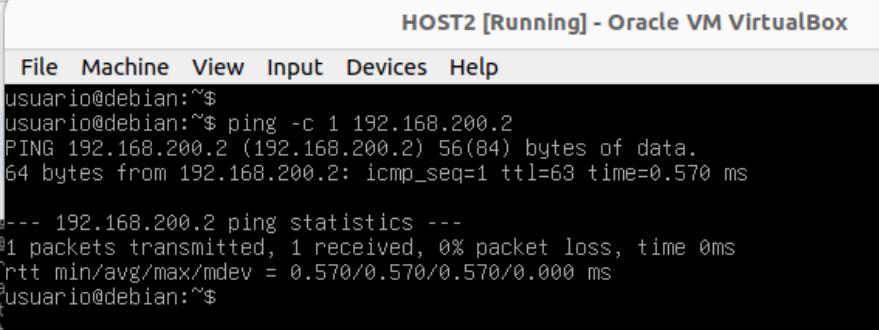
Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

Uncomment the next line to enable packet forwarding for IPv6
Enabling this option disables Stateless Address Autoconfiguration
based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

#####
Additional settings - these settings can improve the network

```

Tal configuração só terá efeito caso **reinicie o router**. Depois de configurar tudo, e tendo reiniciado todas as máquinas, será a hora de realizar os testes, de qualquer HOST se for enviado um ICMP para o outro HOST e se tiver sucesso, toda a configuração foi realizada com sucesso. Na figura abaixo, está sendo enviado um ICMP com ping da máquina HOST2 com o IP 192.168.200.130/25 para o HOST1 com IP 192.168.200.2/25.



```

HOST2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~$ ping -c 1 192.168.200.2
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.
64 bytes from 192.168.200.2: icmp_seq=1 ttl=63 time=0.570 ms

--- 192.168.200.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.570/0.570/0.570/0.000 ms
usuario@debian:~$
```

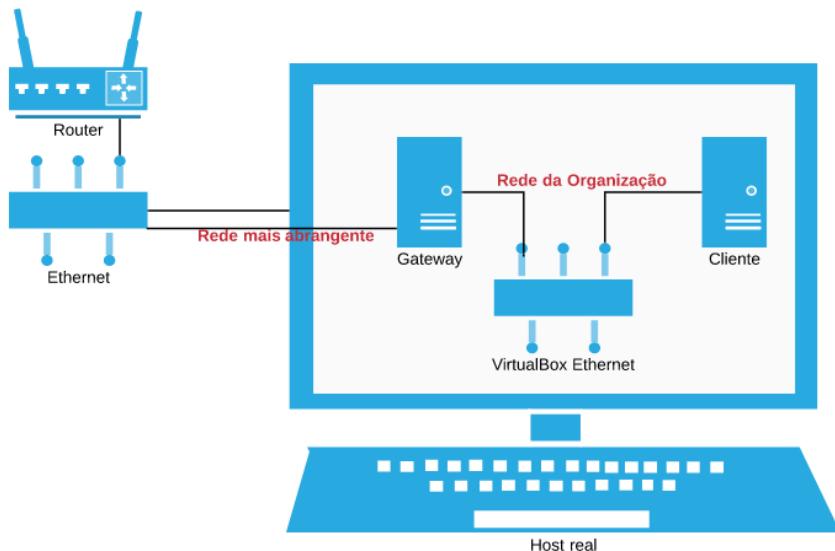
## 15.6 Serviço de Internet com Gateway

Um roteador pode ser utilizado como um elemento que segmenta uma rede de uma organização que detém o controle de todos os segmentos **ou** pode ser utilizado como uma ponte entre uma rede menos abrangente e uma rede mais abrangente, a rede menos abrangente pode ser uma LAN ou um segmento, e a rede mais abrangente pode ser um serviço WAN ou uma LAN respetivamente.

Nesta prática, o aluno vai configurar um Gateway (chamado servidor) que deverá segregar uma segunda máquina virtual (chamada cliente) em uma rede interna, realizando a configuração básica no qual, um cliente em uma rede menos abrangente terá acesso a rede mais abrangente, e poderá utilizar os serviços da rede mais abrangente, tal como Internet.

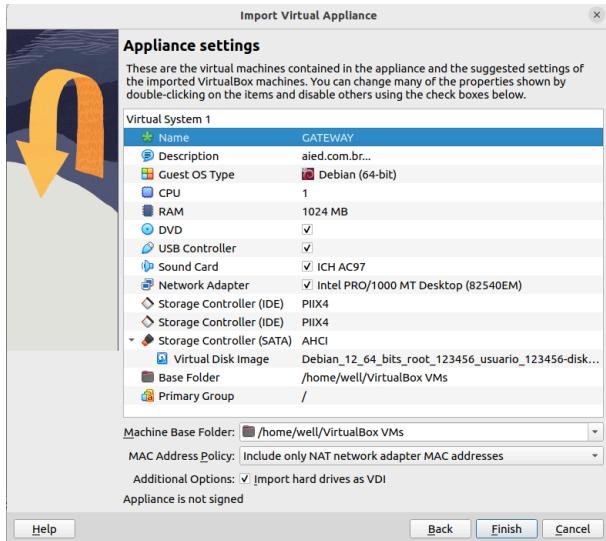
### 15.6.1 Configurando o ambiente virtual e a rede virtual no VirtualBox

Nesta prática vamos instalar um Debian 12 GNU/Linux como Router com a função de “ponte” onde a rede da casa do leitor será classificado como a rede mais abrangente e a rede interna do VirtualBox será a rede menos abrangente. Na figura abaixo este esquema está descrito em esquema.



A rede mais abrangente deve estar configurada no VirtualBox em um Adaptador Bridge e a rede da Organização (menos abrangente) em um Adaptador em rede Interna.

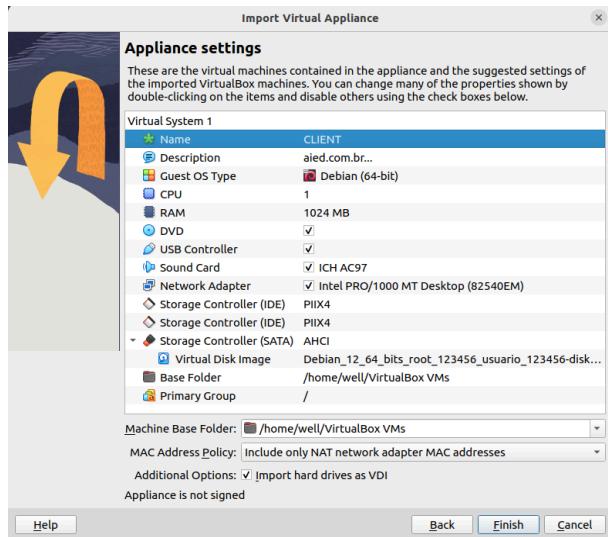
Após realizar a instalação de seu próprio Debian, conforme **Capítulo 1**, você deve importar o arquivo **.ova** duas vezes conforme próximos passos, para importar basta executar um duplo clique sobre o arquivo **.ova**. Na aba de assistente, altere o nome da primeira máquina para **Gateway** e mantenha 1024 MB de RAM.



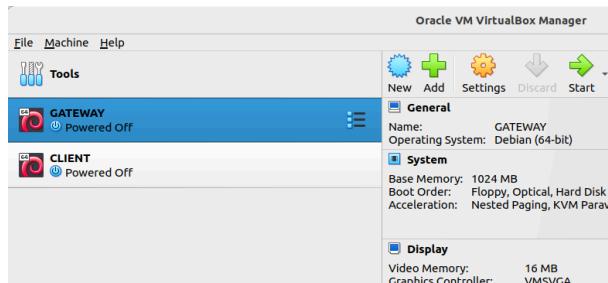
O processo de importação pode demorar, aguarde.



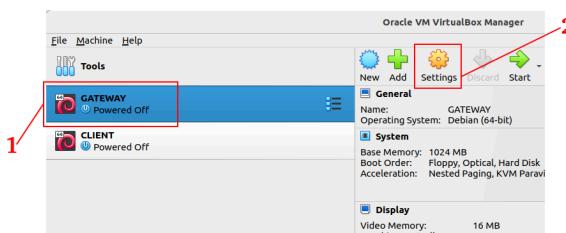
Ao término da importação é possível ver a Virtual Machine **Gateway** na lista de máquinas. Agora vamos criar uma máquina cliente que servirá de Workstation dentro da nossa LAN, para isso vamos importar novamente o mesmo arquivo **.ova** mas agora vamos alterar as informações para poder distinguir esta máquina das demais. Vamos chamar esta Virtual Machine de **Cliente**.



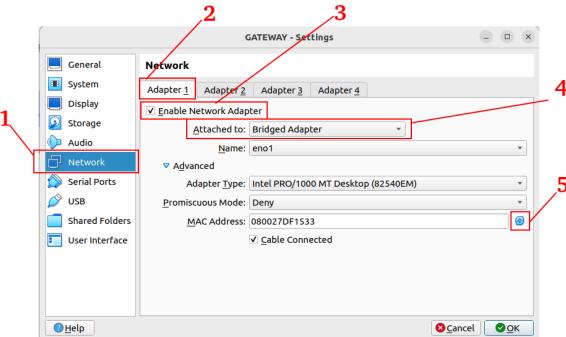
Ao término da implantação, a máquina **Cliente** será listada na lista de máquinas virtuais.



Agora precisamos configurar as ligações das interfaces de rede nas Virtual Machine conectando estas nas redes corretas, lembre-se que a LAN real (a mais abrangente) será a nossa WAN (para fins de prática) e a LAN será a rede interna do VirtualBox.

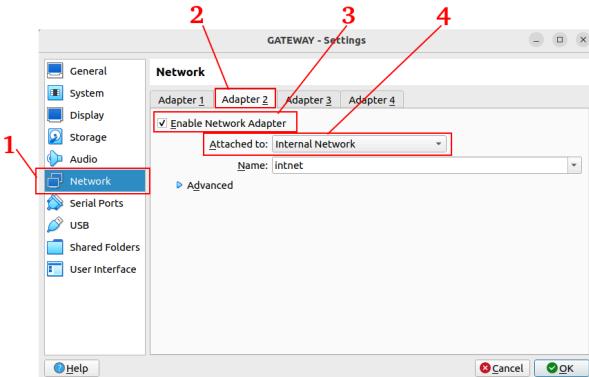


Na Virtual Machine Servidor vamos configurar o primeiro adaptador de rede para que este se conecte na LAN real que fará o papel de WAN nesta nossa prática, o ideal é que utilize a configuração em Modo Bridge no Adaptador 1 da Virtual Machine, conforme figura abaixo.



**Observação:** Sempre que entrar nesta interface, clique várias vezes no botão de mudar o MAC Address que está destacado com o quadro verde, isso é fundamental para que o MAC Address das VMs seja sempre diferente.

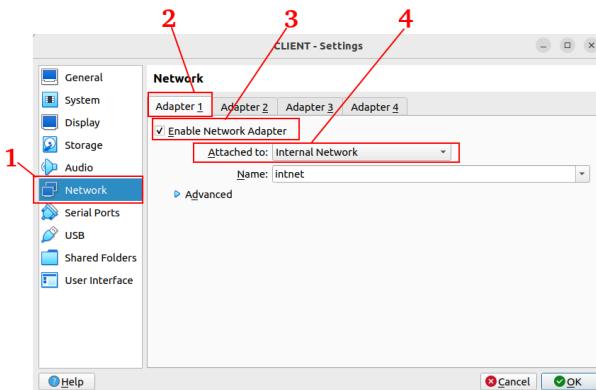
Já no segundo Adaptador de rede, a configuração do adaptador deve ser apontada para a Rede Interna do VirtualBox, é nesta rede virtual que vamos conectar a Virtual Machine **Cliente**.



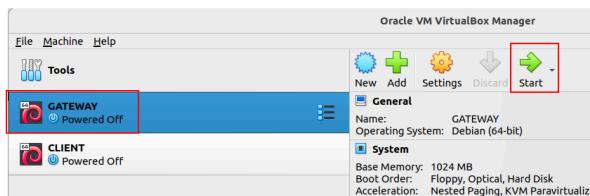
Agora deve-se configurar o adaptador de rede da Virtual Machine **Cliente**, para isso acesse a configuração do **Cliente** conforme figura abaixo.



No Adaptador 1 conecte este na Rede Interna do VirtualBox, ou seja, a mesma rede que foi conectado o Adaptador 2 da Virtual Machine **Gateway**.



Agora podemos iniciar as duas máquinas, para isso selecione a máquina e clique em Start conforme figura abaixo, faça isso para as duas.



### 15.6.2 Configurando as Interfaces de rede

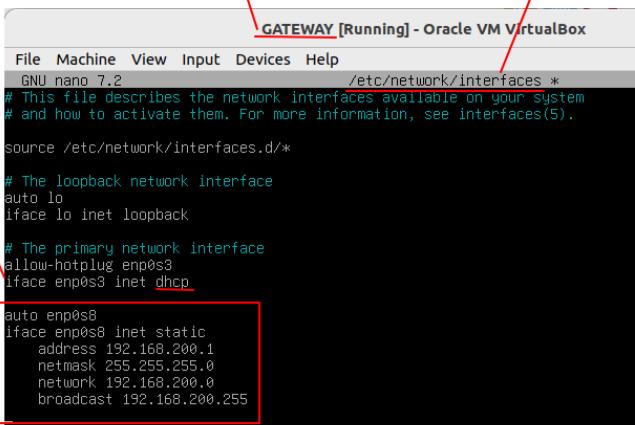
Agora que a rede está conectada e ambas as máquinas estão ligadas, o projeto físico de sua rede está finalizado, agora é preciso fazer a configuração lógica da rede, ou seja, configurar as máquinas com os dados da rede.

Comece configurando o Gateway, para isso na Virtual Machine **Gateway** faça o login.



Após o login, é fundamental que confirme o nome das interfaces de rede e naturalmente veja a rede que já está configurada automaticamente por DHCP.

O Adaptador 1 da Virtual Machine **Gateway** é a **enp0s3** e o adaptador 2 é o **enp0s8**, isso pode mudar de distribuição para distribuição. Comece alterando o arquivo **/etc/network/interfaces**, mantenha o **enp0s3** como está e altere apenas adicionando a configuração do **enp0s8** conforme figura abaixo.



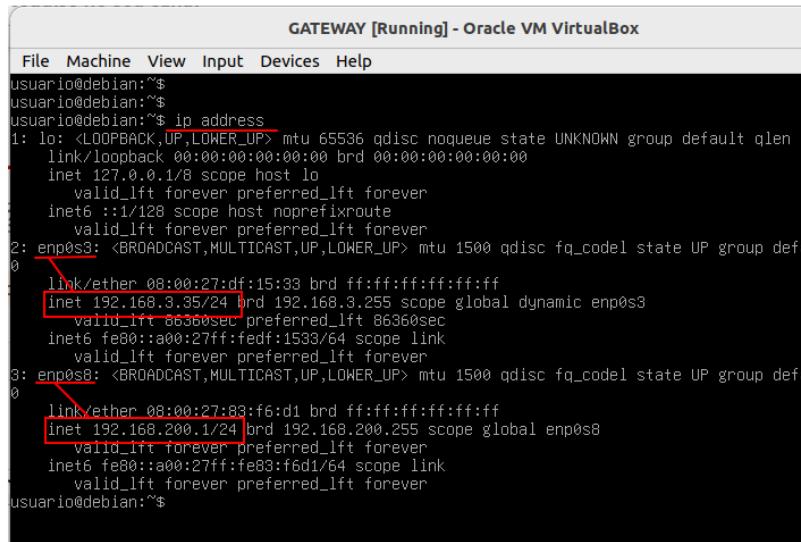
```

1
GATEWAY [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).
source /etc/network/interfaces.d/*
The loopback network interface
auto lo
iface lo inet loopback
The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp
auto enp0s8
iface enp0s8 inet static
 address 192.168.200.1
 netmask 255.255.255.0
 network 192.168.200.0
 broadcast 192.168.200.255

```

Esta máquina vai pegar além de suas próprias requisições as requisições que chegam pela interface **enp0s8** e enviar para interface **enp0s3**. Reinicie a máquina com o comando de reboot.

Agora após reiniciar, digite o comando `ip address` e confirme que a configuração de **enp0s3** e **enp0s8** foram obtidas, saiba que o ip obtido na interface **enp0s3** é dinâmico e depende muito da casa do leitor<sup>87</sup>, mas a interface enp0s8 deve estar igual à imagem abaixo.



```

GATEWAY [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
 link/ether 08:00:27:df:15:33 brd ff:ff:ff:ff:ff:ff
 inet 192.168.3.35/24 brd 192.168.3.255 scope global dynamic enp0s3
 valid_lft 86360sec preferred_lft 86360sec
 inet6 fe80::a00:27ff:fedf:1533/64 scope link
 valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
 link/ether 08:00:27:83:f6:d1 brd ff:ff:ff:ff:ff:ff
 inet 192.168.200.1/24 brd 192.168.200.255 scope global enp0s8
 valid_lft forever preferred_lft forever
 inet6 fe80::a00:27ff:fe83:f6d1/64 scope link
 valid_lft forever preferred_lft forever
usuario@debian:~$

```

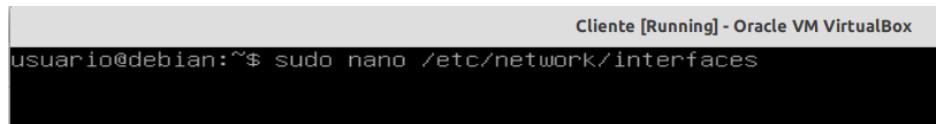
O próximo passo é garantir que o Gateway possui informações de acesso a um DNS Server, para isso edite o arquivo **/etc/resolv.conf** adicionando a última linha da imagem, só adicione esta última linha.

<sup>87</sup> Leia o capítulo 1 deste livro com atenção, encontrará respostas sobre Adaptadores de rede;



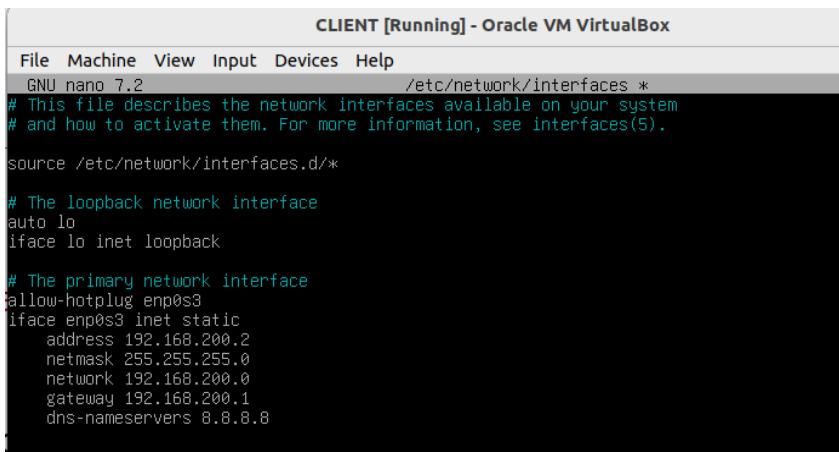
```
File Machine View Input Devices Help
GNU nano 7.2 /etc/resolv.conf *
nameserver 8.8.8.8
```

A configuração das interfaces de rede da máquina Gateway está finalizada, agora vamos configurar a interface de rede do Cliente. No terminal, configure a interface de rede utilizando o nano, para isso edite o arquivo **/etc/network/interfaces**



```
usuario@debian:~$ sudo nano /etc/network/interfaces
```

Será ligado a única interface na rede Interna do VirtualBox com IP estático 192.168.200.2, veja a figura abaixo.



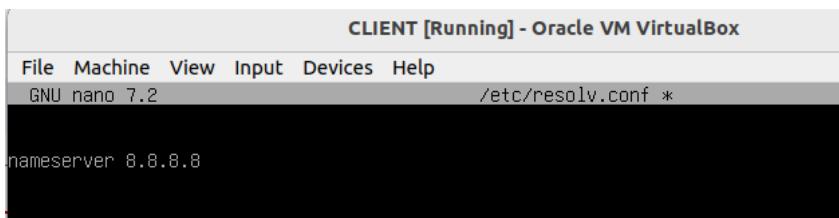
```
File Machine View Input Devices Help
GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

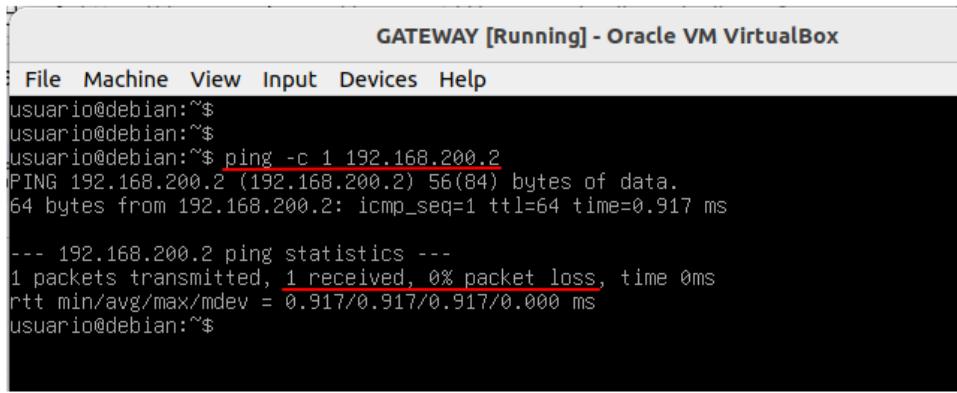
The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.2
 netmask 255.255.255.0
 network 192.168.200.0
 gateway 192.168.200.1
 dns-nameservers 8.8.8.8
```

O próximo passo é garantir que o Gateway possui informações de acesso a um DNS Server, para isso edite o arquivo **/etc/resolv.conf** adicionando a última linha da imagem, só adicione esta última linha.



```
File Machine View Input Devices Help
GNU nano 7.2 /etc/resolv.conf *
nameserver 8.8.8.8
```

Reinicie a máquina com **reboot** ou **systemctl reboot**. Após reiniciar então volte para Virtual Machine Gateway, com ambas ligadas vamos executar o comando ping e vamos validar a conectividade entre os elementos configurados.



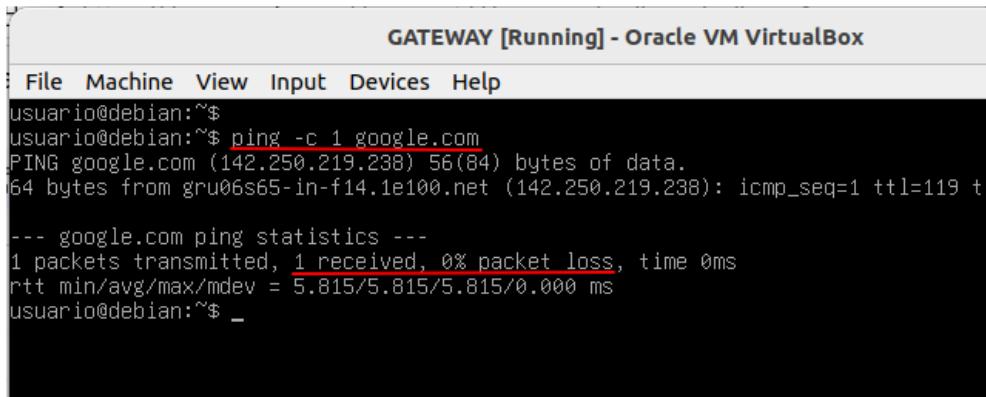
```

GATEWAY [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help
usuario@debian:~$ ping -c 1 192.168.200.2
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.
64 bytes from 192.168.200.2: icmp_seq=1 ttl=64 time=0.917 ms

--- 192.168.200.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.917/0.917/0.917/0.000 ms
usuario@debian:~$
```

Sempre teste a conectividade entre as máquinas virtuais para depois testar acesso à Internet.



```

GATEWAY [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help
usuario@debian:~$ ping -c 1 google.com
PING google.com (142.250.219.238) 56(84) bytes of data.
64 bytes from gru06s65-in-f14.1e100.net (142.250.219.238): icmp_seq=1 ttl=119 ti
--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 5.815/5.815/5.815/0.000 ms
usuario@debian:~$
```

O primeiro ping valida a conectividade da Virtual Machine Gateway com a Cliente, já o segundo ping valida a conectividade do Gateway com a Internet.

### 15.6.3 Configurando o servidor Gateway

Abra a Virtual Machine Gateway, em **/etc/sysctl.conf** devemos habilitar o parâmetro forward.



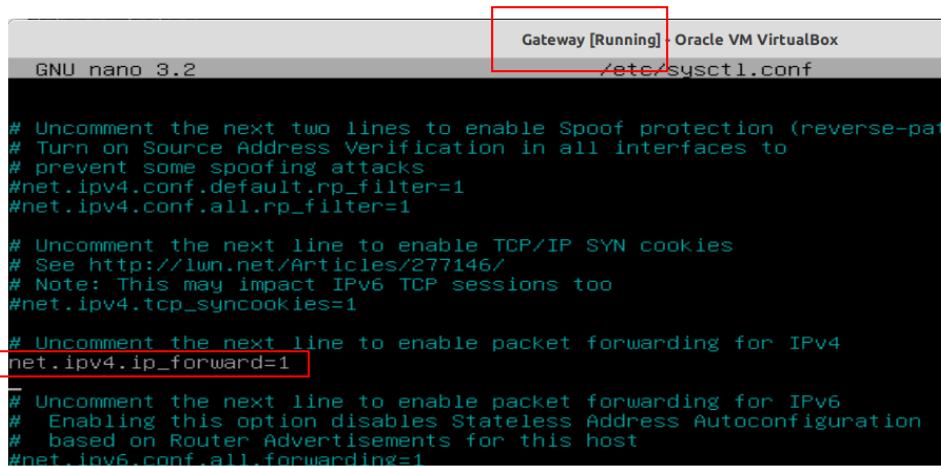
```

Gateway [Running] - Oracle VM VirtualBox

usuario@debian:~$ sudo nano /etc/sysctl.conf
```

Procure a linha que possui **#net.ipv4.ip\_forward=1** e retire o comentário e ficará assim:  
**net.ipv4.ip\_forward=1**

Veja na imagem abaixo:



```

Uncomment the next two lines to enable Spoof protection (reverse-path
Turn on Source Address Verification in all interfaces to
prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

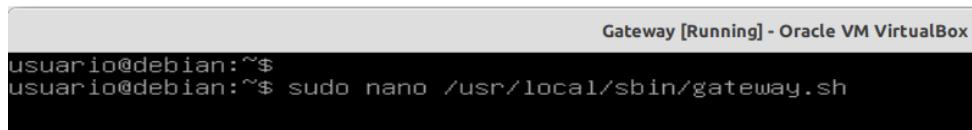
Uncomment the next line to enable TCP/IP SYN cookies
See http://lwn.net/Articles/277146/
Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

Uncomment the next line to enable packet forwarding for IPv4
#net.ipv4.ip_forward=1

Uncomment the next line to enable packet forwarding for IPv6
Enabling this option disables Stateless Address Autoconfiguration
based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

```

Reinic peace a máquina com **reboot** ou **systemctl reboot**. O próximo passo é criar um arquivo script com três regras de Iptables, e naturalmente agendar a execução deste script para a inicialização do sistema.



```

usuario@debian:~$ sudo nano /usr/local/sbin/gateway.sh

```

Algumas distribuições Linux possuem no kernel o Iptables instalado, já outras possuem o nftables. Ambas as bibliotecas executam a mesma ação, que é redirecionar, filtrar e modificar pacotes de rede antes do processamento. O Debian 12 possui o pacote nftables, já os anteriores o Iptables.

#### 15.6.3.1 Utilizando Iptables

O Iptables é um programa escrito em C, utilizado como ferramenta que configura regras para o protocolo de IPv4, pode ser utilizado para filtragem, modificação e redirecionamento de protocolos. Utilizando os módulos e framework do kernel Linux na versão 2.3.15 ou posterior. As configurações de firewall feitas ficarão guardadas no kernel, logo serão perdidas quando o sistema for reiniciado.

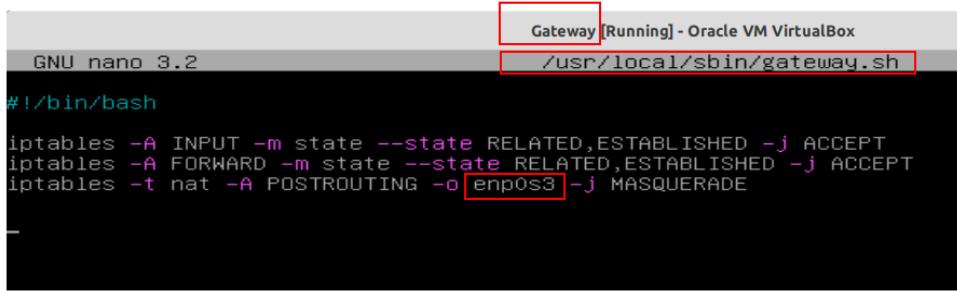
Se seu Debian GNU/Linux utiliza Iptables, digite as seguintes regras no arquivo **/usr/local/sbin/gateway.sh**:

```

1.#!/bin/bash
2.iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
3.iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
4.iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

```

Conforme imagem abaixo, salve.



```
GNU nano 3.2
#!/bin/bash

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

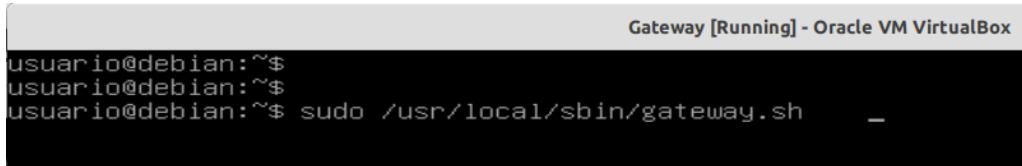
-
```

Para garantir que seja executado, permita que o script recém criado tenha tal permissão com o comando chmod.



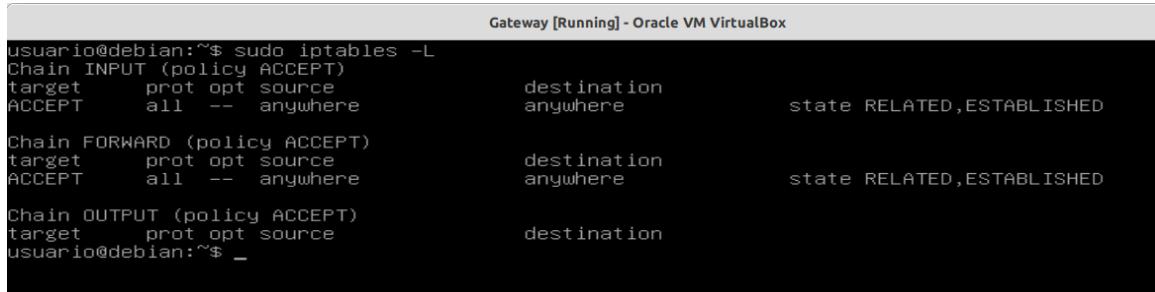
```
usuario@debian:~$ sudo chmod +x /usr/local/sbin/gateway.sh
```

Para testar o script é fácil, apenas informe o caminho do script que o sistema operacional irá executar este arquivo com /bin/bash.



```
usuario@debian:~$ sudo /usr/local/sbin/gateway.sh
```

O IPTABLES roda apenas se o usuário estiver usando sudo, mesmo na conta privilegiada ROOT o comando iptables não permite sua execução. Se nenhum erro foi lançado (ver figura acima) então liste as regras do servidor para averiguar se foram criadas.



```
usuario@debian:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
usuario@debian:~$
```

### 15.6.3.2 Utilizando nftables

O nftables é um subsistema do kernel Linux que fornece filtragem e classificação de pacotes de rede, datagramas e quadros. Ele está disponível desde o kernel Linux 3.13 lançado em 19 de janeiro de 2014.

O nftables substitui as partes legadas do iptables do projeto Netfilter. Entre as vantagens do nftables sobre o iptables está menos duplicação de código e extensão mais fácil para novos protocolos. nftables é configurado por meio do utilitário de pelo usuário chamado nft,

enquanto as ferramentas legadas são configuradas por meio dos utilitários iptables, ip6tables, arptables e ebtables frameworks.

O nftables utiliza os blocos de construção da infraestrutura do Netfilter, como os ganchos existentes na pilha de rede, sistema de rastreamento de conexão, componente de enfileiramento de espaço de usuário e subsistema de registro.

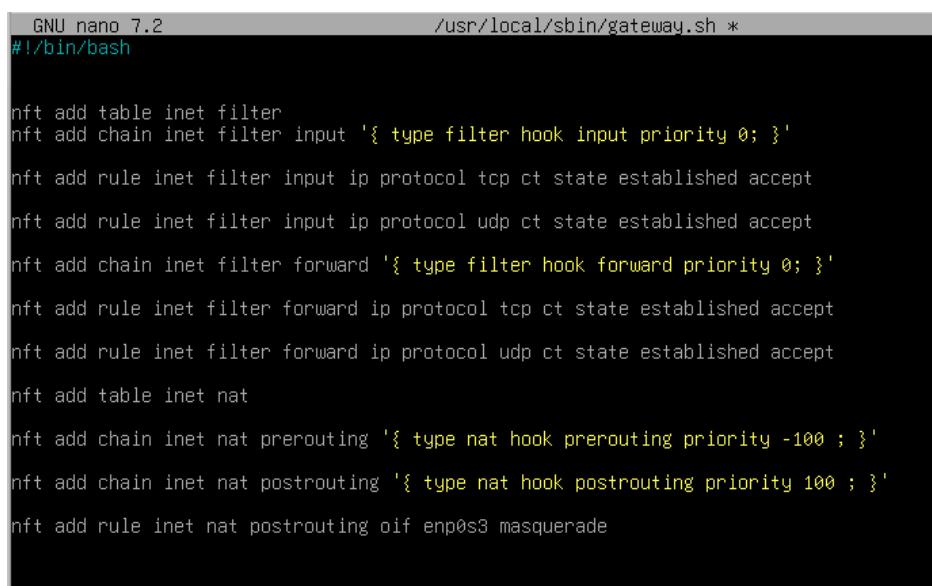
Se seu Debian GNU/Linux utiliza Nftables, digite as seguintes regras no arquivo **/usr/local/sbin/gateway.sh**:

```

1.#!/bin/bash
2.#
3.
4.nft flush ruleset
5.
6.nft add table inet filter
7.nft add chain inet filter input '{ type filter hook input priority 0; }'
8.
9.nft add rule inet filter input ip protocol tcp ct state established accept
10.nft add rule inet filter input ip protocol udp ct state established accept
11.
12.nft add chain inet filter forward '{ type filter hook forward priority 0; }'
13.nft add rule inet filter forward ip protocol tcp ct state established accept
14.nft add rule inet filter forward ip protocol udp ct state established accept
15.
16.nft add table inet nat
17.nft add chain inet nat prerouting '{ type nat hook prerouting priority -100 ; }'
18.nft add chain inet nat postrouting '{ type nat hook postrouting priority 100 ; }'
19.nft add rule inet nat postrouting oif enp0s3 masquerade
20.

```

Veja o exemplo após digitação.



```

GNU nano 7.2 /usr/local/sbin/gateway.sh *
#!/bin/bash

nft flush ruleset
nft add table inet filter
nft add chain inet filter input '{ type filter hook input priority 0; }'
nft add rule inet filter input ip protocol tcp ct state established accept
nft add rule inet filter input ip protocol udp ct state established accept
nft add chain inet filter forward '{ type filter hook forward priority 0; }'
nft add rule inet filter forward ip protocol tcp ct state established accept
nft add rule inet filter forward ip protocol udp ct state established accept
nft add table inet nat
nft add chain inet nat prerouting '{ type nat hook prerouting priority -100 ; }'
nft add chain inet nat postrouting '{ type nat hook postrouting priority 100 ; }'
nft add rule inet nat postrouting oif enp0s3 masquerade

```

Para verificar se está correto, digite o comando abaixo, você verá no output a lista completa de regras.

- 
1. sudo chmod +x /usr/local/sbin/gateway.sh
  2. /usr/local/sbin/gateway.sh
  3. sudo nft list ruleset
- 

Veja o output:

```
usuario@debian:~$
usuario@debian:~$ sudo nft list ruleset
[sudo] password for usuario:
table inet filter {
 chain input {
 type filter hook input priority filter; policy accept;
 ip protocol tcp ct state established accept
 ip protocol udp ct state established accept
 }

 chain forward {
 type filter hook forward priority filter; policy accept;
 ip protocol tcp ct state established accept
 ip protocol udp ct state established accept
 }
}
table inet nat {
 chain prerouting {
 type nat hook prerouting priority dstnat; policy accept;
 }

 chain postrouting {
 type nat hook postrouting priority srcnat; policy accept;
 oif "enp0s3" masquerade
 }
}
usuario@debian:~$
```

Para garantir que seja executado, permita que o script recém criado tenha tal permissão com o comando chmod.

Gateway [Running] - Oracle VM VirtualBox

```
usuario@debian:~$
usuario@debian:~$
usuario@debian:~$
usuario@debian:~$ sudo chmod +x /usr/local/sbin/gateway.sh
```

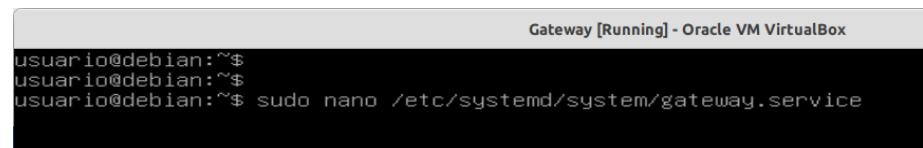
Para testar o script é fácil, apenas informe o caminho do script que o sistema operacional irá executar este arquivo com **/bin/bash**.

Gateway [Running] - Oracle VM VirtualBox

```
usuario@debian:~$
usuario@debian:~$
usuario@debian:~$ sudo /usr/local/sbin/gateway.sh _
```

### 15.6.3.3 Executando script na inicialização do GNU/Linux

Este script funciona, mas não deve ser executado manualmente, será criado um arquivo que será usado pelo Systemd para inicialização automática, para isso crie o arquivo **/etc/systemd/system/gateway.service** com o nano conforme imagem abaixo.



```
Gateway [Running] - Oracle VM VirtualBox
usuario@debian:~$ sudo nano /etc/systemd/system/gateway.service
```

O conteúdo deste arquivo será utilizado para iniciar o script, então digite conforme figura abaixo.



```
GNU nano 3.2
/etc/systemd/system/gateway.service

[Unit]
Description=Gateway da aula AIED.com.br
After=network.target

[Service]
ExecStart=/usr/local/sbin/gateway.sh

[Install]
WantedBy=multi-user.target
```

Arquivo criado, sem erro de digitação, então ative o serviço com **enable**, para isso execute o comando **systemctl** conforme imagem abaixo.



```
Gateway [Running] - Oracle VM VirtualBox
usuario@debian:~$ sudo systemctl enable gateway.service
Created symlink /etc/systemd/system/multi-user.target.wants/gateway.service
usuario@debian:~$ _
```

Com o enable agendamos a execução do script para inicialização, então reinicie o servidor Gateway com **reboot**. Após o carregamento então utilizando o comando **iptables/nftables** liste as regras criadas.

#### 15.6.4 Realizando os testes com ICMP

Vamos fazer testes básicos de conectividade, com Ping teste a conectividade agora com a Virtual Machine Cliente conforme figura abaixo.

Cliente [Running] - Oracle VM VirtualBox

```

usuario@debian:~$ ping -c 1 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data.
64 bytes from 192.168.200.1: icmp_seq=1 ttl=64 time=0.625 ms

--- 192.168.200.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.625/0.625/0.625/0.000 ms
usuario@debian:~$
usuario@debian:~$ ping -c 1 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=12.7 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 12.678/12.678/12.678/0.000 ms
usuario@debian:~$
usuario@debian:~$
```

Os pings acima testam a conectividade e o Gateway. Agora realize o teste do serviço DNS realizando um ping no domínio google.com conforme imagem abaixo.

Cliente [Running] - Oracle VM VirtualBox

```

usuario@debian:~$ ping -c 1 google.com
PING google.com (216.58.202.14) 56(84) bytes of data.
64 bytes from gru06s26-in-f14.1e100.net (216.58.202.14): icmp_seq=1 ttl=115 time=12.9 ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 12.941/12.941/12.941/0.000 ms
usuario@debian:~$
usuario@debian:~$
usuario@debian:~$
```

## 15.7 Regras de filtro e redirecionamento em um Firewall

Para este exemplo, será necessário realizar a importação de quatro máquinas virtuais, uma ficará no meio de duas redes virtuais e irá realizar o trabalho de um Firewall. No Host1 já temos o OpenSSH-Server instalado conforme capítulo anterior, ou seja, já está instalado.

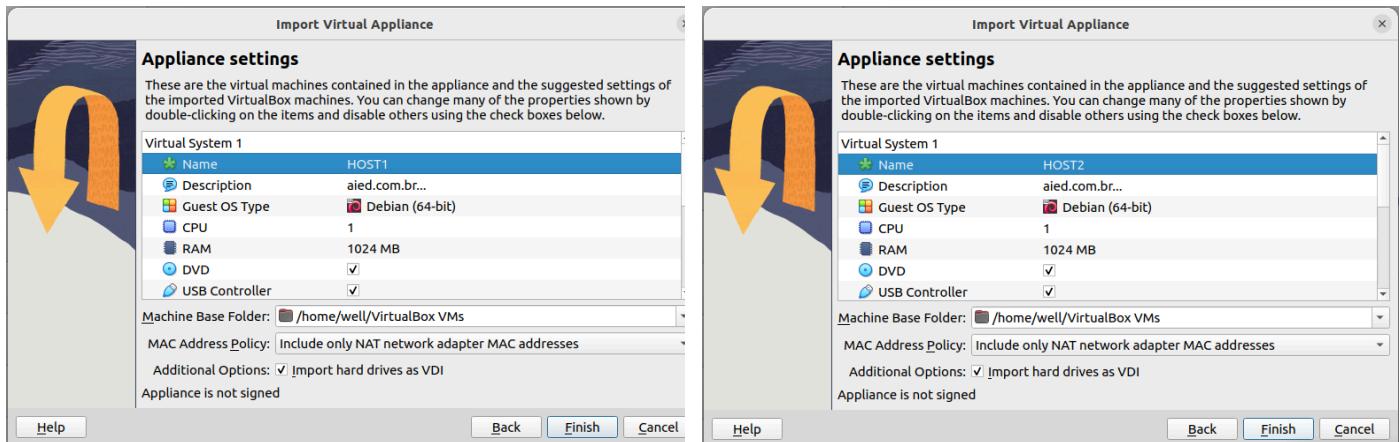
Segue abaixo uma ilustração de como será a topologia bem como a distribuição dos endereços.

<imagem da rede>

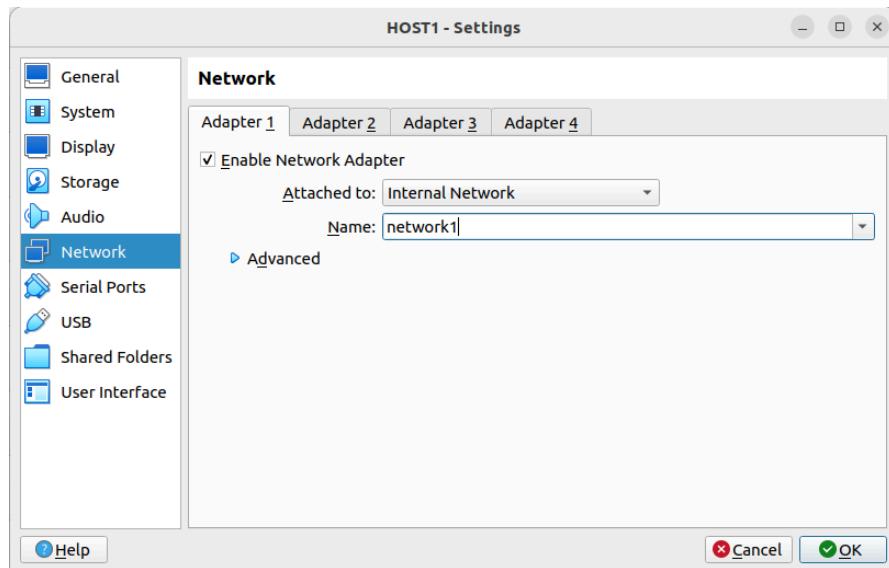
Neste exemplo serão necessárias três máquinas virtuais, são estas:

- Host1: máquina cliente ligada na rede network1;
- Host2: máquina cliente ligada na rede network2;
- Host3: máquina cliente ligada na rede network2;
- Firewall: servidor que permite a conexão de ambas as redes, e por isso possui duas interfaces de rede;

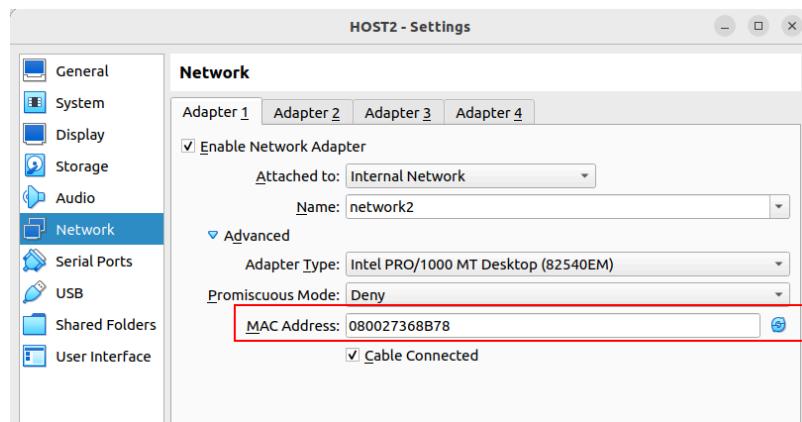
Comece importando o arquivo .ova inicial (criado no Capítulo 1) três vezes, altere o nome das importações para HOST1, HOST2 e HOST3, conforme figuras abaixo.



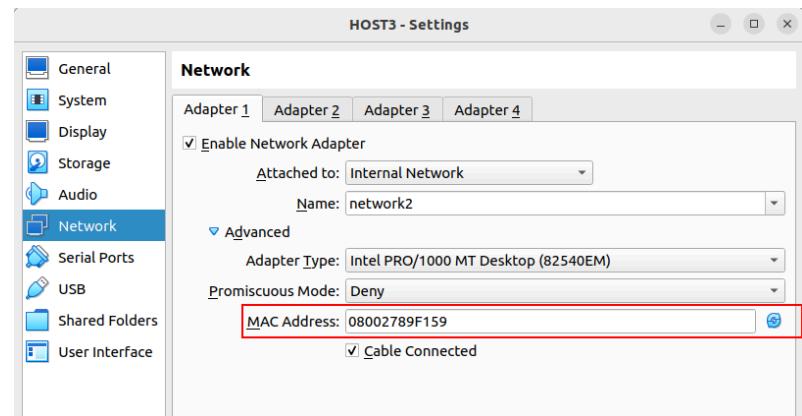
O próximo passo é importar mais uma vez o arquivo .ova, mas desta vez dê o nome de FIREWALL. Com as três máquinas criadas, o próximo passo é alterar a configuração física dos adaptadores de rede, criando assim as redes: **network1** e **network2**. No HOST1 acesse a configuração e na aba Network habilite somente o Adapter 1, para este adapter selecione Internal Network e digite o nome **network1**, sim, tem que ser digitado corretamente.



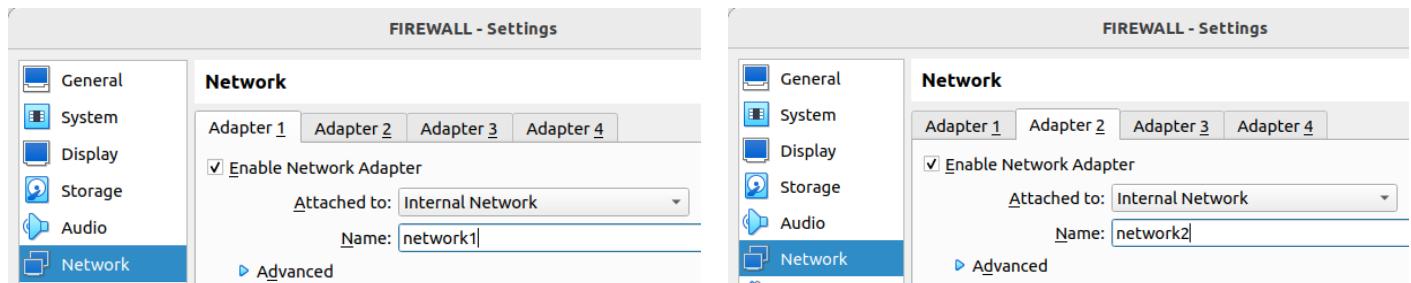
Já para o HOST2 o Adapter 1 deve ter como rede a **network2**, conforme figura abaixo.



E para o HOST3 vamos conectar na rede **network2**, sim, tanto HOST2 quanto HOST3 estarão na mesma rede. A idéia é que somente o HOST2 irá poder acessar o serviço SSH do HOST1 que está na outra rede, então o FIREWALL deverá bloquear as requisições de qualquer máquina que não seja o HOST2 para o serviço SSH porta 22. Você tem que garantir que o endereço MAC Address não seja o mesmo se comparado o HOST2 com o HOST3.



Já o FIREWALL precisa de dois Adapters, onde o Adapter 1 estará ligado na rede **network1** e o Adapter 2 ligado na rede **network2**.



### 15.7.1 Configuração lógica

Até o momento foi realizado o projeto físico, seguimos agora para o projeto lógico que é a configuração dos endereços e serviços, para começar inicie a máquina virtual HOST1 e

com o nano abra para edição o arquivo **/etc/network/interfaces**, configure da seguinte forma.

```
GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.2
 network 192.168.200.0
 netmask 255.255.255.128
 broadcast 192.168.200.127
 gateway 192.168.200.1
 dns-nameservers 8.8.8.8
```

Tanto o HOST1, HOST2 e HOST3 possuem apenas um Adapter, ou seja, o primeiro Adapter, que quando o Linux é iniciado assume o nome de enp0s3, por isso a configuração deve ser realizada nesta interface. O endereço dado é um pertencente a network1 que inicia em 192.168.200.1/25 até 192.168.200.126/25, será escolhido o endereço 192.168.200.2/25 para manter o padrão que é: o primeiro endereço útil para hosts é dado ao router. Da mesma forma, para o HOST2 será escolhido o endereço 192.168.200.130/25 pois é o segundo endereço válido na **network2**.

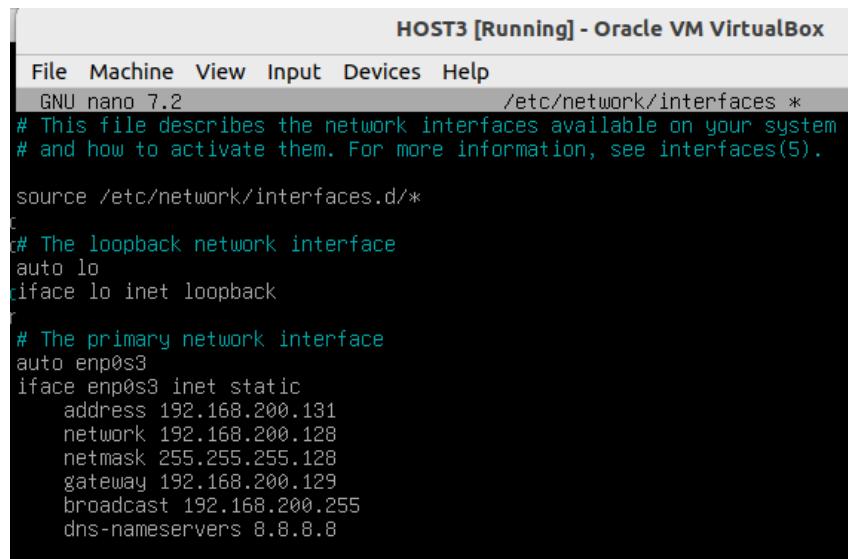
```
GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.130
 network 192.168.200.128
 netmask 255.255.255.128
 broadcast 192.168.200.255
 gateway 192.168.200.129
 dns-nameservers 8.8.8.8
```

Já o HOST3 terá o endereço 192.168.200.131/25, pois assim como o HOST2 ele também está na **network2**.



```

HOST3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

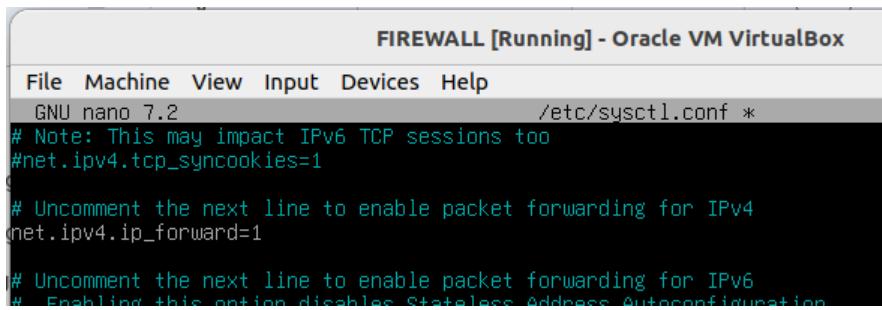
source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
auto enp0s3
iface enp0s3 inet static
 address 192.168.200.131
 network 192.168.200.128
 netmask 255.255.255.128
 gateway 192.168.200.129
 broadcast 192.168.200.255
 dns-nameservers 8.8.8.8

```

No GNU/Linux FIREWALL para que uma mensagem passe de uma interface de rede para outra é necessário ativar a opção `net.ipv4.ip_forward` e atribuir o valor 1, e caso queira IPv6 a opção `net.ipv6.ip_forward` também com o valor 1. Para esta prática será utilizado somente IPv4 e por isso abra o arquivo `/etc/sysctl.conf` para edição, e habilite o atributo `net.ipv4.ip_forward=1` conforme figura abaixo.



```

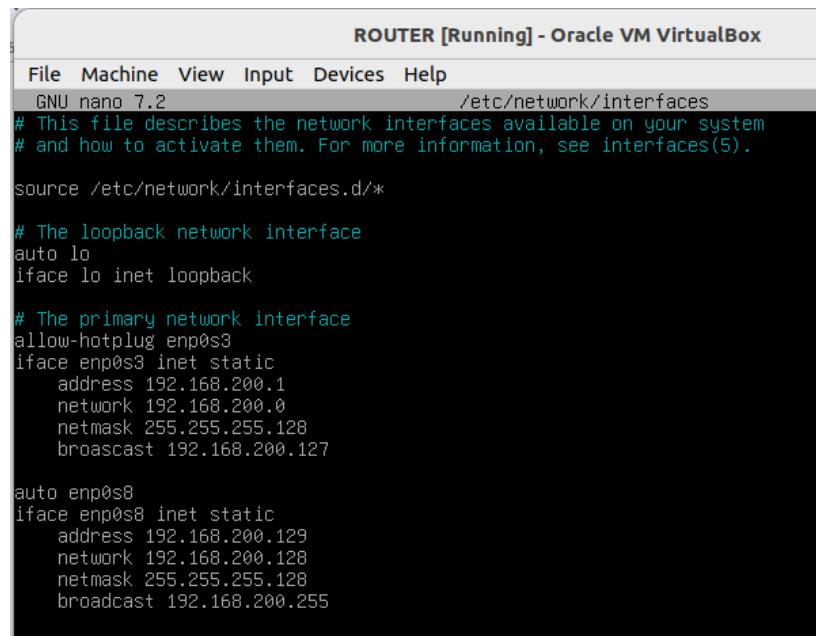
FIREWALL [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/sysctl.conf *
Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

Uncomment the next line to enable packet forwarding for IPv6
Enabling this option disables Stateless Address Auto-configuration

```

Já a máquina virtual FIREWALL possui dois adaptadores de rede, onde o Adapter 1 será a interface `enp0s3` e o Adapter 2 a interface `enp0s8`. Como é o router entre as duas redes, terá como endereço o primeiro endereço útil de cada uma das redes. Os endereços são 192.168.200.1/25 para a `enp0s3` e 192.168.200.129/25 para a interface `enp0s8`, repare na configuração de HOST1 e HOST2 que ambos os endereços são usados como gateway padrão.



```

ROUTER [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/network/interfaces
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

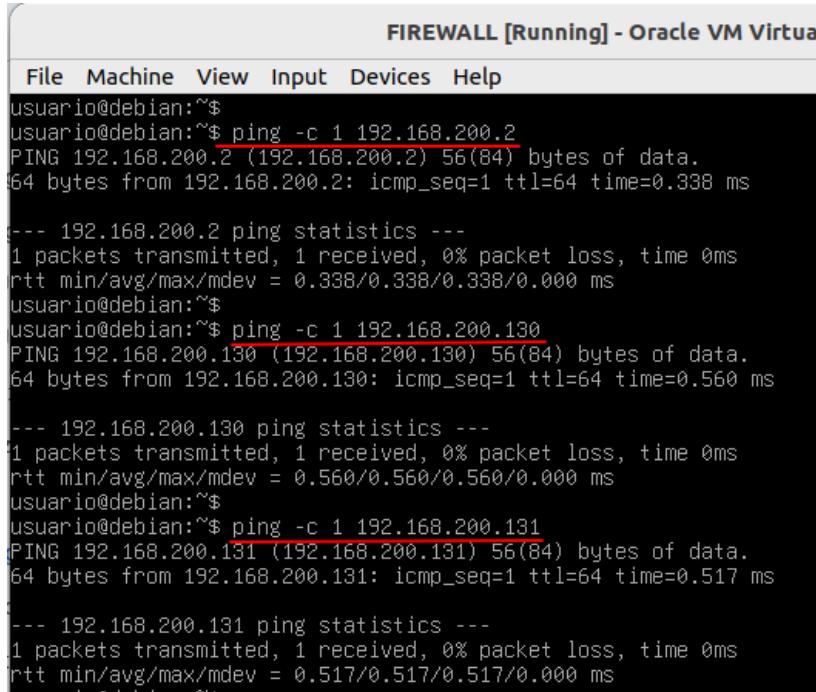
The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.1
 network 192.168.200.0
 netmask 255.255.255.128
 broadcast 192.168.200.127

auto enp0s8
iface enp0s8 inet static
 address 192.168.200.129
 network 192.168.200.128
 netmask 255.255.255.128
 broadcast 192.168.200.255

```

Reiniciando todas as três máquinas virtuais, verá que é possível da ROUTER enviar um protocolo ICMP para ambas as máquinas HOSTs, conforme figura abaixo. Esse teste é importante pois avalia que tanto a parte física quanto a parte lógica de endereçamento estão funcionando.



```

FIREWALL [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~$ usuario@debian:~$ ping -c 1 192.168.200.2
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.
64 bytes from 192.168.200.2: icmp_seq=1 ttl=64 time=0.338 ms

--- 192.168.200.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.338/0.338/0.338/0.000 ms
usuario@debian:~$ usuario@debian:~$ ping -c 1 192.168.200.130
PING 192.168.200.130 (192.168.200.130) 56(84) bytes of data.
64 bytes from 192.168.200.130: icmp_seq=1 ttl=64 time=0.560 ms

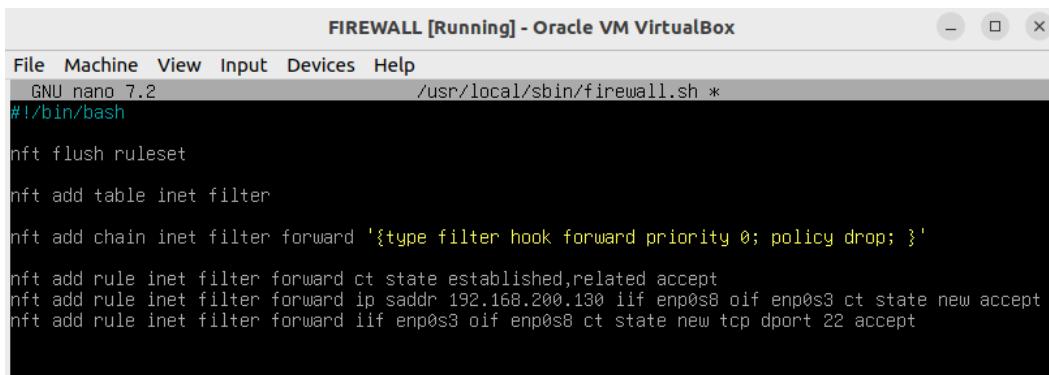
--- 192.168.200.130 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.560/0.560/0.560/0.000 ms
usuario@debian:~$ usuario@debian:~$ ping -c 1 192.168.200.131
PING 192.168.200.131 (192.168.200.131) 56(84) bytes of data.
64 bytes from 192.168.200.131: icmp_seq=1 ttl=64 time=0.517 ms

--- 192.168.200.131 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.517/0.517/0.517/0.000 ms

```

### 15.7.2 Criando as regras

Como no nftables as tables e as chains não estão criadas, vamos iniciar criando tais elementos. Para esta prática vamos apenas atuar na permissão de passagem, ou seja, na chain **FORWARD**. Como somos pouco criativos, vamos usar **FILTER** para filtragem e **FORWARD** para passagem.



```

FIREWALL [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /usr/local/sbin/firewall.sh *
#!/bin/bash

nft flush ruleset
nft add table inet filter
nft add chain inet filter forward '{type filter hook forward priority 0; policy drop; }'
nft add rule inet filter forward ct state established,related accept
nft add rule inet filter forward ip saddr 192.168.200.130 iif enp0s8 oif enp0s3 ct state new accept
nft add rule inet filter forward iif enp0s3 oif enp0s8 ct state new tcp dport 22 accept

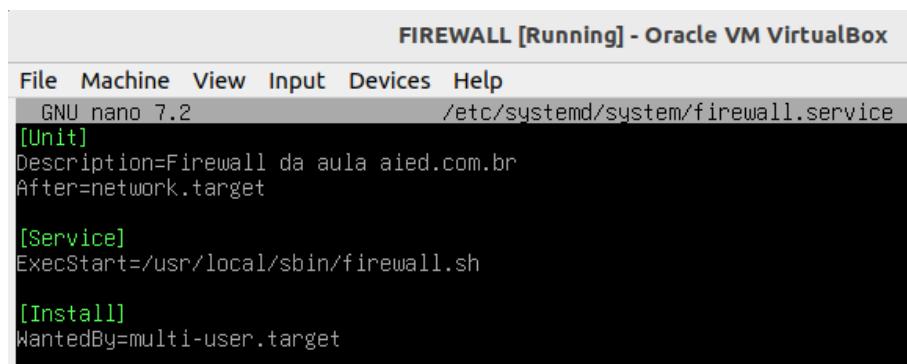
```

A ideia é só permitir abertura de novas conexões quando são iniciadas na máquina 192.168.200.130/25, repare que a política padrão é DROP. Salve o arquivo script e também não se esqueça de dar poder de execução para este script.

```
1. sudo chmod +x /usr/local/sbin/firewall.sh
```

### 15.7.3 Script de inicialização

Assim como as iptables a nftables só existem em memória, existem componentes que persistem, mas não recomendo pois o próprio GNU/Linux possui artifícios para isso. Comece então criando um arquivo de serviço chamado **/etc/systemd/system/firewall.service**.



```

FIREWALL [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/systemd/system/firewall.service *
[Unit]
Description=Firewall da aula aied.com.br
After=network.target

[Service]
ExecStart=/usr/local/sbin/firewall.sh

[Install]
WantedBy=multi-user.target

```

Tenha sempre uma descrição digna, nada mais digno do que exaltar **aied**. Somente execute este script de inicialização após configuração da placa de rede e rode quando o Linux terminal for iniciado.

Para recarregar os arquivos de inicialização use o parâmetro `daemon-reload` no comando `systemctl`, após isso então ative o serviço na inicialização com `enable`.

**FIREWALL [Running] - Oracle VM VirtualBox**

```

File Machine View Input Devices Help
usuario@debian:~$ sudo systemctl daemon-reload
usuario@debian:~$ sudo systemctl enable firewall.service
Created symlink /etc/systemd/system/multi-user.target.wants/firewall.service → /etc/systemd/system/firewall.service.
usuario@debian:~$ sudo systemctl start firewall.service
usuario@debian:~$ sudo systemctl status firewall.service
● firewall.service - Firewall da aula aied.com.br
 Loaded: loaded (/etc/systemd/system/firewall.service; enabled; preset: enabled)
 Active: inactive (dead) since Thu 2024-03-28 18:12:21 -03; 24s ago
 Duration: 18ms
 Process: 959 ExecStart=/usr/local/sbin/firewall.sh (code=exited, status=0/SUCCESS)
 Main PID: 959 (code=exited, status=0/SUCCESS)
 CPU: 13ms

Mar 28 18:12:21 debian systemd[1]: Started firewall.service - Firewall da aula aied.com.br.
Mar 28 18:12:21 debian systemd[1]: firewall.service: Deactivated successfully.
usuario@debian:~$
```

#### 15.7.4 Testando a conexão

Para testar é simples, basta iniciar uma conexão a partir das máquinas 192.168.200.130/25 e 192.168.200.131/25. Veja na imagem abaixo que a conexão iniciada da máquina 192.168.200.130/25 obteve sucesso.

**HOST2 [Running] - Oracle VM VirtualBox**

```

File Machine View Input Devices Help
usuario@debian:~$ usuario@debian:~$ usuario@debian:~$ ssh usuario@192.168.200.2
usuario@192.168.200.2's password:
```

Mas a partir da máquina 192.168.200.131/25 não foi possível.

**HOST3 [Running] - Oracle VM VirtualBox**

```

File Machine View Input Devices Help
usuario@debian:~$ usuario@debian:~$ usuario@debian:~$ ssh usuario@192.168.200.2
```

## 15.8 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

### 15.8.1 Prática acb1b800 01: Configurando interfaces de rede do Gateway

Nesta prática o aluno deve ter a expertise de editar a Interface de rede do Gateway utilizando o arquivo `/etc/network/interface`.

1. No servidor Gateway, informe que a interface de rede na qual está ligada o Adaptador em modo bridge obtenha IP por DHCP;
2. No servidor Gateway, informe que a interface de rede na qual está ligada o Adaptador em modo rede Interna que o IP será estático e que terá como IP 192.168.200.1 com máscara 255.255.255.0 (ver aquivo abaixo);
3. Neste mesmo adaptador informe o Broadcast e o endereço da rede;
4. Reinicie a máquina GATEWAY;
5. Rode o comando aied baixo no GATEWAY;

O arquivo será semelhante ao arquivo abaixo, lembrando que pode-se ter alterações nos nomes das interfaces de rede.

```
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*
The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp

auto enp0s8
iface enp0s8 inet static
 address 192.168.200.1
 netmask 255.255.255.0
 network 192.168.200.0
 broadcast 192.168.200.255
```

A primeira validação deve ser feita na Virtual Machine Gateway, entre na Virtual Machine e digite o seguinte comando:

1. `sudo aied validate acb1b800 checkpoint01`

```
usuario@debian:~$ sudo aied validate acb1b800 checkpoint01
Action that will be performed: validate

Practice: Linux Gateway class practice - Configuring Gateway network interface
The OUTPUT of the command will be sent: cat /etc/network/interfaces
The OUTPUT of the command will be sent: ip address
The OUTPUT of the command will be sent: ping -c 1 8.8.8.8

Do you wish to continue? (y for YES) y

Total hits: 11 out of a total of 11. This is equivalent to 100% (hit).
Identification: 25e64062-c
AIED v(10)
```

### 15.8.2 Prática acb1b801 02: Ativando o serviço de **Gateway**

Nesta prática o aluno deve ter a expertise de editar o arquivo `/etc/sysctl.conf` bem como criar as regras NFTABLES para compartilhar o serviço de rede.

1. No arquivo `/etc/sysctl.conf` ative a opção de `net.ipv4.ip_forward=1` removendo o comentário da linha;
2. Crie um script em `/usr/local/sbin/` chamado `gateway.sh` e adicione neste arquivo as regras nftables, conforme tópico [Utilizando nftables](#);
3. Cadastre o script `gateway.sh` no serviço Systemd do GNU/Linux, não se esqueça de dar permissão de execução para o script `gateway.sh`<sup>88</sup> e ativar o **enable** do serviço;
4. Reinicie o servidor **Gateway**;
5. Execute o comando aied abaixo;

A segunda validação deve ser feita na Virtual Machine **Gateway**, entre na Virtual Machine e digite o seguinte comando:

```
1. sudo aied validar acb1b801 checkpoint02
```

```
usuario@debian:/tmp$ sudo aied validar acb1b801 checkpoint02
Action that will be performed: validar

Practice: Validation of Gateway practice - Configuring the Customer's network interface and Resolv.conf
The OUTPUT of the command will be sent: cat /etc/sysctl.conf
The OUTPUT of the command will be sent: cat /usr/local/sbin/gateway.sh
The OUTPUT of the command will be sent: cat /etc/systemd/system/gateway.service
The OUTPUT of the command will be sent: ls -l /usr/local/sbin/
The OUTPUT of the command will be sent: nft list ruleset
The OUTPUT of the command will be sent: systemctl status gateway.service --no-pager

Do you wish to continue? (y for YES) y
Total hits: 33 out of a total of 33. This is equivalent to 100% (hit).
Identification: 25e64062-c
AIED v(10)

usuario@debian:/tmp$ _
```

### 15.8.3 Prática acb1b800 03: Configurando a interface de rede do **Cliente**

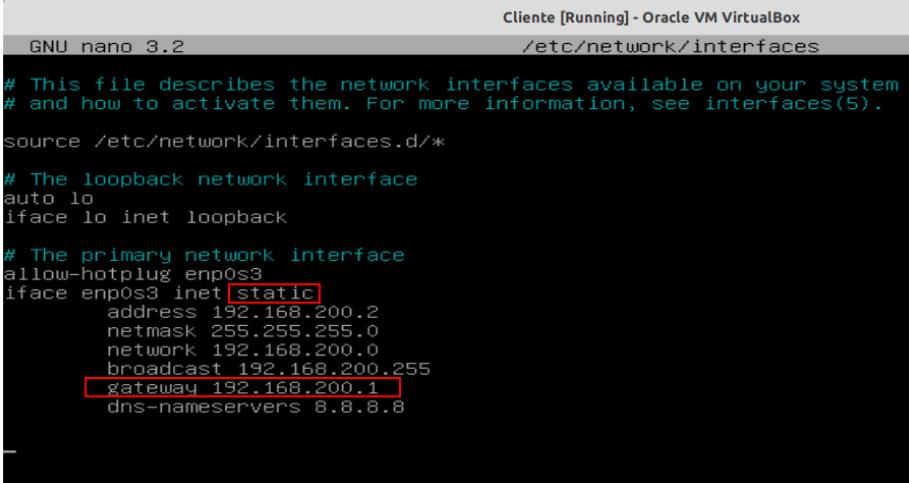
Nesta prática o aluno deve ter a expertise de editar a Interface de rede do Cliente utilizando o arquivo `/etc/network/interface`.

1. Na máquina **Cliente**, informe que a interface de rede na qual está ligada o Adaptador em modo rede Interna que o IP será estático e que terá como IP `192.168.200.2` com máscara `255.255.255.0` (ver figura abaixo);
2. Altere o conteúdo do arquivo `/etc/resolv.conf` onde a propriedade `nameserver` deve ter o valor `8.8.8.8` (somente);
3. Neste mesmo adaptador informe o Broadcast e o endereço da rede.
4. Reinicie o cliente;
5. Rode o comando aied abaixo;

O arquivo será semelhante ao arquivo abaixo, lembrando que pode-se ter alterações nos nomes das interfaces de rede.

---

<sup>88</sup> Caso tenha dúvidas de como fazer esta operação no Systemctl, veja vídeo:  
<https://youtu.be/7d2q8RF2xY4>



```

GNU nano 3.2 Cliente [Running] - Oracle VM VirtualBox
 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

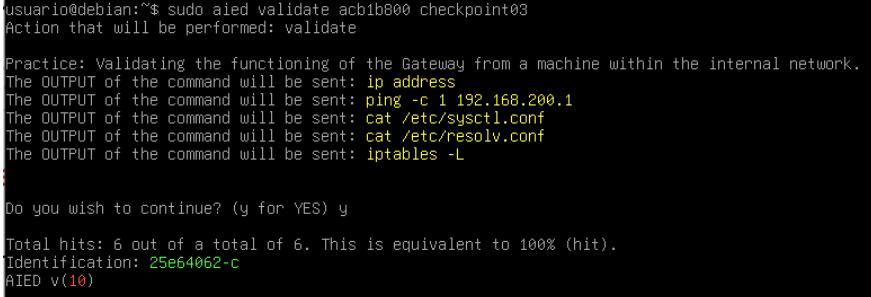
The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.2
 netmask 255.255.255.0
 network 192.168.200.0
 broadcast 192.168.200.255
 gateway 192.168.200.1
 dns-nameservers 8.8.8.8

```

A primeira validação deve ser feita na Virtual Machine Cliente, entre na Virtual Machine e digite o seguinte comando:

1. sudo aied validate acb1b800 checkpoint03

A ferramenta deve informar que enviará dados para o servidor aied.com.br, se confirmar ela deverá validar a prática.



```

usuario@debian:~$ sudo aied validate acb1b800 checkpoint03
Action that will be performed: validate

Practice: Validating the functioning of the Gateway from a machine within the internal network.
The OUTPUT of the command will be sent: ip address
The OUTPUT of the command will be sent: ping -c 1 192.168.200.1
The OUTPUT of the command will be sent: cat /etc/sysctl.conf
The OUTPUT of the command will be sent: cat /etc/resolv.conf
The OUTPUT of the command will be sent: iptables -L
:
Do you wish to continue? (y for YES) y

Total hits: 6 out of a total of 6. This is equivalent to 100% (hit).
Identification: 25e64062-c
AIED v(10)

```

## 16 Openssh-server

Secure Socket Shell (SSH) é um protocolo de comunicação de Camada de Aplicação do modelo TCP/IP, permite que usuários acessem terminais remotos GNU/Linux mesmo por redes adversas e inseguras, como a WAN utilizando funcionalidades de criptografia descrita no Modelo OSI na Camada de Apresentação, lembre-se que no modelo TCP/IP a camada de Aplicação corresponde às camadas de Apresentação, Sessão e Aplicação do modelo OSI.

Nesta prática, vamos aprender a utilizar não só o pipe bidirecional de comunicação entre os terminais para execução de comandos, mas também vamos aprender a transferir arquivos de forma segura.

Desenvolvido em 1995 por Tatu Ylonen, um especialista em Cyber Security e ao longo dos anos se mostrou muito seguro, e logo foi substituindo o uso do Telnet, o SSH ganhou o apego popular de quem atua em manutenção de servidores GNU/Linux.

Telnet é um modo de comunicação que realiza a mesma função do SSH, porém o Telnet transmite protocolos pela rede com conteúdo não criptografado, e por isso é o protocolo mais procurado por hackers quando estes estão realizando um sniffer na rede. Inclusive é muito comum o envio de credenciais no Telnet. O Telnet ainda é utilizado para se dar manutenção em Routers, Switches e Access Points.

Na figura abaixo temos a captura de uma comunicação com uso de Telnet, e em um dado momento um equipamento solicita o password para o outro equipamento, este segundo possui alguém digitando a senha, e a senha vai trafegar na rede caracter por caracter.

|             |               |               |        |                                                       |
|-------------|---------------|---------------|--------|-------------------------------------------------------|
| 22 0.608842 | 192.168.1.140 | 192.168.1.194 | TCP    | 66 56760 → 23 [ACK] Seq=116 Ack=67 Win=5888 Len=0 TS\ |
| 23 0.776660 | 192.168.1.140 | 192.168.1.194 | TELNET | 67 Telnet Data ...                                    |
| 24 0.777664 | 192.168.1.194 | 192.168.1.140 | TELNET | 67 Telnet Data ...                                    |
| 25 0.777673 | 192.168.1.140 | 192.168.1.194 | TCP    | 66 56760 → 23 [ACK] Seq=117 Ack=68 Win=5888 Len=0 TS\ |
| 26 0.874608 | 192.168.1.140 | 192.168.1.194 | TELNET | 67 Telnet Data ...                                    |
| 27 0.875528 | 192.168.1.194 | 192.168.1.140 | TELNET | 67 Telnet Data ...                                    |
| 28 0.875555 | 192.168.1.140 | 192.168.1.194 | TCP    | 66 56760 → 23 [ACK] Seq=118 Ack=69 Win=5888 Len=0 TS\ |
| 29 1.607927 | 192.168.1.140 | 192.168.1.194 | TELNET | 68 Telnet Data ...                                    |
| 30 1.608922 | 192.168.1.194 | 192.168.1.140 | TELNET | 68 Telnet Data ...                                    |
| 31 1.608938 | 192.168.1.140 | 192.168.1.194 | TCP    | 66 56760 → 23 [ACK] Seq=120 Ack=71 Win=5888 Len=0 TS\ |
| 32 1.634015 | 192.168.1.194 | 192.168.1.140 | TELNET | 76 Telnet Data ...                                    |
| 33 1.634025 | 192.168.1.140 | 192.168.1.194 | TCP    | 66 56760 → 23 [ACK] Seq=120 Ack=81 Win=5888 Len=0 TS\ |
| 34 1.947921 | 192.168.1.140 | 192.168.1.194 | TELNET | 67 Telnet Data ...                                    |
| 35 1.986862 | 192.168.1.194 | 192.168.1.140 | TCP    | 66 23 → 56760 [ACK] Seq=81 Ack=121 Win=5792 Len=0 TS\ |

Frame 32: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)  
 ▷ Ethernet II, Src: OpenGear\_00:55:a5 (00:13:06:05:55:a5), Dst: ASUSTekC\_b3:01:84 (00:1d:60:b3:01:84)  
 ▷ Internet Protocol Version 4, Src: 192.168.1.194, Dst: 192.168.1.140  
 ▷ Transmission Control Protocol, Src Port: 23, Dst Port: 56760, Seq: 71, Ack: 120, Len: 10  
 ▷ Telnet  
 Data: Password: ←

Tudo passa em texto plano, esta captura é real e foi retirada de uma prática com equipamentos CISCO, [esta captura pode ser obtida neste link](#).

### 16.2 Introdução OpenSSH e Protocolo SSH

OpenSSH é a implementação mais amplamente implantada do protocolo SSH. Começou como uma ramificação de uma versão licenciada gratuitamente do software SSH original, mas foi fortemente reescrita, expandida e atualizada. OpenSSH é desenvolvido como parte do Projeto OpenBSD, uma comunidade conhecida por escrever software seguro.

OpenSSH é a implementação SSH padrão no mundo Linux e BSD e também é usado em produtos de grandes empresas como HP, Cisco, Oracle, Novell, Juniper, IBM e assim por diante.

O OpenSSH criptografa todo o tráfego, de credencial aos dados, tudo, basicamente uma criptografia transforma um texto (plaintext) em um conjunto de caracteres ilegíveis para qualquer cultura, é indecifrável, chamamos este texto de texto criptografado (ciphertext). Eu tenho um livro especial e um capítulo especial para você compreender a fundo estes processos, [acesse este link](#).

A criptografia simétrica é rápida, mas não oferece nenhuma maneira para os hosts trocarem chaves com segurança. A criptografia assimétrica permite que os hosts troquem chaves públicas, mas é lenta e dispendiosa em termos computacionais.

O problema do SSH é: COMO TRANSPORTAR UMA CHAVE POR UM CANAL INSEGURO?

Existem muitas abordagens para isso, uma possibilidade é o uso de uma técnica chamada Diffie-Hellman, onde o cliente e o servidor negociam algoritmos mutuamente aceitáveis em cada conexão. Na figura abaixo dois computadores definiram que a criptografia será por AES com 128 bits e que a troca de informações privadas será realizada por Diffie-Hellman.

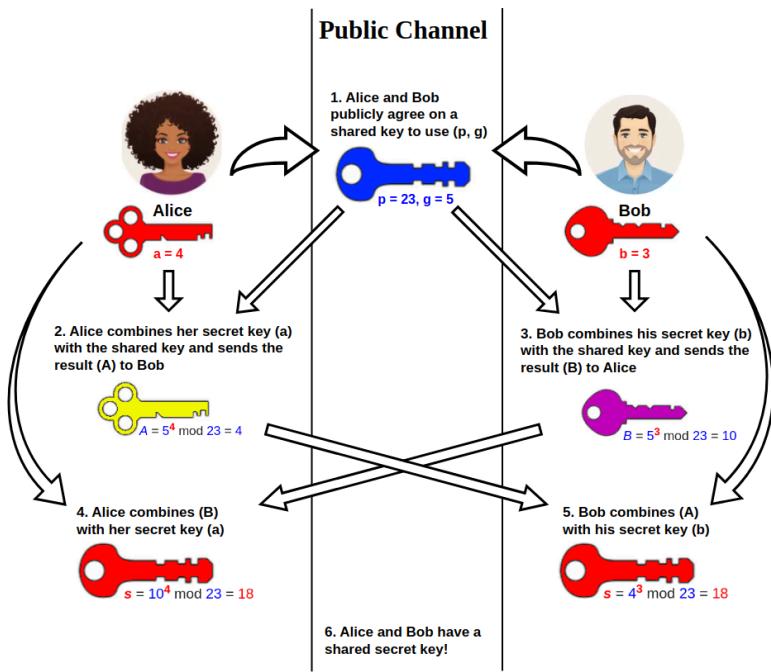
```

7 0.080015 10.0.0.2 10.0.0.1 SSHv2 334 Server: Key Exchange Init
8 0.080017 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=85 Ack=20 Win=4109 Len=64 [TCP]
9 0.088006 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=149 Ack=20 Win=4109 Len=64 [TCP]
10 0.096005 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=213 Ack=20 Win=4109 Len=64 [TCP]
11 0.104006 10.0.0.1 10.0.0.2 SSHv2 78 Client: Key Exchange Init
12 0.112029 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=301 Ack=300 Win=3829 Len=64 [TCP]
13 0.120006 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=365 Ack=300 Win=3829 Len=64 [TCP]
14 0.128007 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=433 Ack=300 Win=3829 Len=64 [TCP]

encryption_algorithms_server_to_client length: 41
encryption_algorithms_server_to_client string: aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc
mac_algorithms_client_to_server length: 43
mac_algorithms_client_to_server string: hmac-sha1,hmac-sha1-96,hmac-md5,hmac-md5-96
mac_algorithms_server_to_client length: 43
mac_algorithms_server_to_client string: hmac-sha1,hmac-sha1-96,hmac-md5,hmac-md5-96
compression_algorithms_client_to_server length: 4
compression_algorithms_client_to_server string: none
compression_algorithms_server_to_client length: 4
compression_algorithms_server_to_client string: none
languages_client_to_server length: 0
languages_client_to_server string:
languages_server_to_client length: 0
languages_server_to_client string:
First KEY Packet Follows: 0
Reserved: 00000000
[hashhServerAlgorithms: diffie-hellman-group1-sha1;aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc;hmac-sha1,hmac-sha1-96,hmac-md5,hmac-md5-96]
[hashhServer: 3cc67862bceac0f334c62ad1b76895b4]
Padding String: 00000000
[Direction: server-to-client]

```

Diffie-Hellman é uma forma de trocar dados em ambiente público para que ambos geram chaves privadas, já o algoritmo AES é um algoritmo de chave simétrica. Na imagem abaixo temos o melhor exemplo e explicação para Diffie-Hellman.



Essa troca de informações entre ambas as máquinas com parâmetros, pode ser vista na captura de tráfego de redes, conforme figura abaixo.

```

0 0.072002 10.0.0.1 10.0.0.2 TCP 110 59139 -> 22 [ACK] Seq=211 Ack=20 Win=4109 Len=04 [T]
1 0.080015 10.0.0.2 10.0.0.1 TCP 334 Server: Key Exchange Init
2 0.080017 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=85 Ack=20 Win=64 Len=64 [T]
3 0.088006 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=149 Ack=20 Win=4109 Len=64 [T]
4 0.096005 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=213 Ack=20 Win=4109 Len=64 [T]
5 0.104006 10.0.0.1 10.0.0.2 SSHv2 78 Client: Key Exchange Init
6 0.112029 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=301 Ack=300 Win=3829 Len=64 [T]
7 0.120006 10.0.0.1 10.0.0.2 TCP 118 59139 -> 22 [ACK] Seq=365 Ack=300 Win=3829 Len=64 [T]
8 0.128007 10.0.0.1 10.0.0.2 SSHv2 70 Client: Diffie-Hellman Key Exchange Init
9 0.160020 10.0.0.2 10.0.0.1 SSHv2 374 Server: Diffie-Hellman Key Exchange Reply
10 0.168014 10.0.0.2 10.0.0.1 SSHv2 70 Server: New Keys
11 0.184011 10.0.0.1 10.0.0.2 SSHv2 70 Client: New Keys
12 0.192010 10.0.0.1 10.0.0.2 SSHv2 106 Client: Encrypted packet (len=52)
13 0.200010 10.0.0.1 10.0.0.2 SSHv2 106 Client: Encrypted packet (len=52)

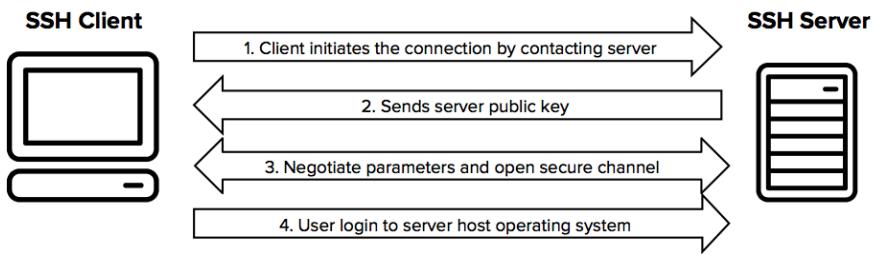
▼ Key Exchange (method:diffie-hellman-group1-sha1)
 Message Code: Key Exchange Init (20)
 ▼ Algorithms
 Cookie: f3cad290ecf47c47554c88cf3a722bb2
 kex_algorithms length: 26
 kex_algorithms string: diffie-hellman-group1-sha1
 server_host_key_algorithms length: 7
 server_host_key_algorithms string: ssh-rsa
 encryption_algorithms_client_to_server length: 41
 encryption_algorithms_client_to_server string: aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc
 encryption_algorithms_server_to_client length: 41
 encryption_algorithms_server_to_client string: aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc
 mac_algorithms_client_to_server length: 43

```

Outro método de autenticação é por uso de chaves assimétricas, ou seja chave pública e privada são utilizadas para transportar dados em canais inseguros, às vezes, é uma técnica muito usada por administradores de sistema para logon. A ideia é ter um par de chaves criptográficas (chave pública e chave privada), as chaves usadas para autenticação são chamadas de SSH Keys.

Assim que uma conexão for estabelecida entre o cliente SSH e o servidor, os dados transmitidos são criptografados de acordo com os parâmetros negociados na configuração. Durante a negociação, o cliente e o servidor concordam com o algoritmo de criptografia simétrica a ser usado e geram a chave de criptografia que será usada. O tráfego entre as partes comunicantes é protegido com algoritmos de criptografia forte padrão da indústria (como AES (Advanced Encryption Standard)), e o protocolo SSH também inclui um

mecanismo que garante a integridade dos dados transmitidos usando algoritmos de hash padrão (como SHA-2 (Algoritmo de hash padrão)).



O passo 3 negocia um pipe de comunicação na camada de sessão do modelo OSI que garante que a comunicação seja redundante a falha e naturalmente eficiente, pois, caso haja problemas no pipe de comunicação é natural que todo processo que não foi iniciado com nohup seja finalizado automaticamente após a falha do pipe de conexão, o mesmo é válido para caso o cliente feche o ssh.

O comando nohup é fundamental então para ambientes cuja falha é comum, principalmente para rotinas que demandam tempo de processamento, como backups e transferência de arquivos.

Um grande problema de se trabalhar com chaves assimétricas é o enorme peso computacional e naturalmente o uso excessivo de recursos e ainda a massa de dados que pode ser criptografada por bloco não passa de 128 caracteres. Uma prática muito comum que inclusive é utilizada no openssh é utilizar a criptografia assimétrica apenas para trocar a chave simétrica, que está representada na figura acima no passo 3.

Toda a comunicação é criptografada, conforme captura de pacotes de rede, conforme figura abaixo.

|             |          |          |       |                                                      |
|-------------|----------|----------|-------|------------------------------------------------------|
| 41 4.032228 | 10.0.0.1 | 10.0.0.2 | TCP   | 60 59139 → 22 [ACK] Seq=973 Ack=1056 Win=3073 Len=0  |
| 42 4.776279 | 10.0.0.1 | 10.0.0.2 | SSHv2 | 106 Client: Encrypted packet (len=52)                |
| 43 4.784321 | 10.0.0.2 | 10.0.0.1 | SSHv2 | 106 Server: Encrypted packet (len=52)                |
| 44 4.792300 | 10.0.0.2 | 10.0.0.1 | SSHv2 | 106 Server: Encrypted packet (len=52)                |
| 45 4.800301 | 10.0.0.2 | 10.0.0.1 | SSHv2 | 106 Server: Encrypted packet (len=52)                |
| 46 4.808515 | 10.0.0.2 | 10.0.0.1 | SSHv2 | 106 Server: Encrypted packet (len=52)                |
| 47 4.816332 | 10.0.0.2 | 10.0.0.1 | SSHv2 | 106 Server: Encrypted packet (len=52)                |
| 48 4.824434 | 10.0.0.2 | 10.0.0.1 | SSHv2 | 614 Server: Encrypted packet (len=560)               |
| 49 4.832295 | 10.0.0.1 | 10.0.0.2 | TCP   | 60 59139 → 22 [ACK] Seq=1025 Ack=1876 Win=4128 Len=0 |
| 50 4.832321 | 10.0.0.2 | 10.0.0.1 | SSHv2 | 554 Server: Encrypted packet (len=500)               |
| 51 4.840289 | 10.0.0.1 | 10.0.0.2 | SSHv2 | 106 Client: Encrypted packet (len=52)                |

Frame 46: 106 bytes on wire (848 bits), 106 bytes captured (848 bits)  
Ethernet II, Src: c2:02:69:49:00:00 (c2:02:69:49:00:00), Dst: c2:01:69:49:00:00 (c2:01:69:49:00:00)  
Internet Protocol Version 4, Src: 10.0.0.2, Dst: 10.0.0.1  
Transmission Control Protocol, Src Port: 22, Dst Port: 59139, Seq: 1212, Ack: 1025, Len: 52  
SSH Protocol  
- SSH Version 2 (encryption: aes128-cbc mac: hmac-sha1 compression:none)  
Packet Length (encrypted): dea73268  
Encrypted Packet: 80f82c3acdbb6d08a909a2c3dfa8a0906dd3180bca3342dfda2c8e9af  
MAC: fd4198e935d00e458705b3bba298506a00f8a390  
[Direction: server-to-client]

Se houver uma falha de conexão, o openssh deverá renegociar uma nova chave simétrica pelo canal seguro usando a criptografia assimétrica. O protocolo SSH é alvo de constantes ataques, basta abrir uma porta 22 na Internet para constatar isso, então o OpenSSH possui um grande número de algoritmos de criptografia que podem ser utilizados

Todos os arquivos de configuração OpenSSH de todo o sistema residem em /etc/ssh por padrão. Depois de encontrar o diretório de configuração, você encontrará um conjunto bastante padrão de arquivos.

As configurações padrão para o cliente ssh aparecem em ssh\_config. Os arquivos que começam com ssh\_host e terminam com \_key são as chaves privadas do servidor. O meio de cada nome de arquivo fornece o algoritmo de criptografia — por exemplo, ssh\_host\_ecdsa\_key contém a chave do host que usa o algoritmo ECDSA.

```
usuario@debian:/etc/ssh$ ls -l
total 584
-rw-r--r-- 1 root root 565189 Jan 31 2020 moduli
-rw-r--r-- 1 root root 1580 Jan 31 2020 ssh_config
-rw-r--r-- 1 root root 3250 Jan 31 2020 sshd_config
-rw----- 1 root root 505 Feb 15 15:23 ssh_host_ecdsa_key
-rw-r--r-- 1 root root 173 Feb 15 15:23 ssh_host_ecdsa_key.pub
-rw----- 1 root root 399 Feb 15 15:23 ssh_host_ed25519_key
-rw-r--r-- 1 root root 93 Feb 15 15:23 ssh_host_ed25519_key.pub
-rw----- 1 root root 1811 Feb 15 15:23 ssh_host_rsa_key
-rw-r--r-- 1 root root 393 Feb 15 15:23 ssh_host_rsa_key.pub
usuario@debian:/etc/ssh$ _
```

Cada chave privada tem um arquivo correspondente com o mesmo nome, mas um .pub adicionado no final. Esta é a chave pública para esse arquivo. O servidor oferecerá o conteúdo desses arquivos a qualquer cliente. Finalmente, sshd\_config contém a configuração do servidor.

O arquivo de configuração permite que se tenha uma configuração específica para uma máquina ou um array de máquinas de uma determinada rede, basicamente existe uma regra para filtrar a máquina e em seguida um conjunto de parâmetros para a prestação do serviço do OpenSSH para aquela máquina.

1. Host <WILDCARD>
2.     parametro 1
3.     parametro 2
4.     parametro 3

Veja exemplo<sup>89</sup>:

---

<sup>89</sup> A definição estes parâmetros está descrito no site oficial do projeto SSH, acessível pelo link: <https://www.ssh.com/academy/ssh/config>

```
GNU nano 3.2 ssh_config

list of available options, their meanings and defaults, please see
ssh_config(5) man page.

Host *
ForwardAgent no
ForwardX11 no
ForwardX11Trusted yes
PasswordAuthentication yes
HostbasedAuthentication no
GSSAPIAuthentication no
GSSAPIDelegateCredentials no
GSSAPIKeyExchange no
GSSAPITrustDNS no
BatchMode no
CheckHostIP yes
AddressFamily any
ConnectTimeout 0
StrictHostKeyChecking ask
IdentityFile ~/.ssh/id_rsa
```

Quando um parâmetro está precedido por # significa que está desabilitado porém caso queira habilitar é só remover o caractere #.

```
GNU nano 3.2 /etc/ssh/ssh_config

ConnectTimeout 0
StrictHostKeyChecking ask
IdentityFile ~/.ssh/id_rsa
IdentityFile ~/.ssh/id_dsa
IdentityFile ~/.ssh/id_ecdsa
IdentityFile ~/.ssh/id_ed25519
Port 22
Protocol 2
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
MACs hmac-md5,hmac-sha1,umac-64@openssh.com
EscapeChar ~
Tunnel no
TunnelDevice any:any
PermitLocalCommand no
VisualHostKey no
ProxyCommand ssh -q -W %h:%p gateway.example.com
RekeyLimit 1G 1h
SendEnv LANG LC_*
HashKnownHosts yes
GSSAPIAuthentication yes
```

No trecho acima, podemos observar que o parâmetro IdentityFile define os certificados assimétricos, já port define a porta e Ciphers define os algoritmos de criptografia. Como todos estes parâmetros estão comentados, o OpenSSH deverá utilizar sua configuração default.

Geralmente se utiliza a porta 22, algumas raras vezes esta porta é alterada para 2222, mas saiba, não adianta esconder portas, pois após uma varredura de 5 minutos um hacker localiza todas as portas abertas, e essa ação é considerada rotineira para um hacker. Já o arquivo sshd\_config reúne os parâmetros do serviço OpenSSH Server, que está sendo executado no servidor. Pode variar de portas a parâmetros de criptografia.

O serviço OpenSSH (service sshd) é altamente configurável e permite restringir quem pode se conectar ao servidor, quais ações esses usuários podem realizar e quais ações ele permite. Todo sistema operacional moderno semelhante ao Unix vem com o sshd instalado como parte do sistema operacional básico.

```
GNU nano 3.2 /etc/systemd/system/sshd.service

[Unit]
Description=OpenBSD Secure Shell server
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

O arquivo acima é o arquivo de inicialização do serviço OpenSSH, que é invocado pelo Systemd na inicialização do GNU/Linux após a rede estar ativa. Se tudo estiver OK então o script **/usr/sbin/sshd** será executado.

Quando um cliente se conecta pela primeira vez em um servidor (no serviço **openssh-server**) por meio de fingerprint é impresso um certificado dentro do diretório **~/.ssh/known\_hosts** do usuário.

```
well@wpo:~$ ssh usuario@192.168.0.100
The authenticity of host '192.168.0.100 (192.168.0.100)' can't be established.
ECDSA key fingerprint is SHA256:GNyCCeMkH6ut4v+mSMn0Wws0y6vu02uv9fG+xnBsp0w.
Are you sure you want to continue connecting (yes/no/[fingerprint])? █
```

No comando acima,

**usuario** é o nome do usuário na máquina que se pretende conectar;  
**192.168.0.100** é o IP da máquina que se pretende conectar.

Um detalhe importante que deve ser levado em consideração é que o IP do servidor OpenSSH fica registrado no cliente através do certificado e caso em sua rede os IPs sejam distribuídos por DHCP, pode ocorrer que nas próximas conexões a chave.

Caso ocorra, deve-se eliminar a chave da máquina específica com o comando:

```
1. ssh-keygen -R 192.168.0.6
```

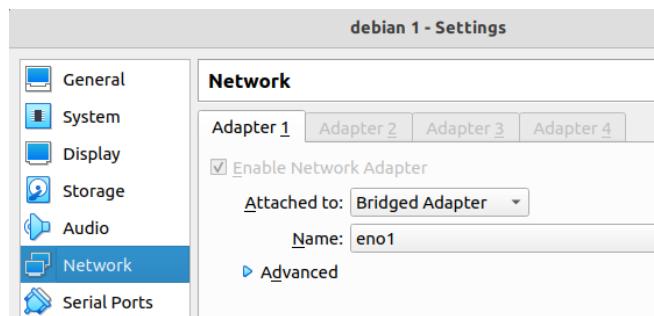
Onde,

-R é utilizado para remover;

**192.168.0.6** é o IP de uma máquina específica, você deve usar o IP da máquina que teve problema;

## 16.2 Instalando e configurando o openssh-server

Antes de iniciar a instalação, crie uma máquina virtual com Debian 12 no modo terminal, veja o [Capítulo 1](#). Configure essa máquina virtual com o Adaptador de Rede 1 apontado para o seu Router da sua LAN, utilizando a opção **Bridge Adapter** e escolhendo como Name a interface de rede da sua máquina física que está realmente conectado na LAN.



No exemplo acima, **eno1** é a interface de rede da máquina física que está ligada na LAN. Dentro da máquina virtual garanta que a interface de rede esteja com a configuração DHCP.

```
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
auto enp0s3
iface enp0s3 inet dhcp
```

Tendo o Router físico na rede com serviço DHCP está Virtual Machine irá obter um IP por intermédio do DHCP, reinicie esta máquina com reboot caso tenha alterado o arquivo. Obtenha o IP da Virtual Machine utilizando o comando **ip address** conforme imagem abaixo.

```
root@debian:~# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo
 link/ether 08:00:27:cf:2f:d4 brd ff:ff:ff:ff:ff:ff
 inet 192.168.0.18/24 brd 192.168.0.255 scope global dynamic
 valid_lft 35901sec preferred_lft 35901sec
```

## 16.3 Instalando o serviço SSH

A instalação do openssh-server não requer outras ferramentas, mas para esta prática vamos utilizar um comando **netstat** para validar a prática, então digite o comando em uma máquina virtual dentro do VirtualBox.

```
1. sudo apt update -y
2. sudo apt install net-tools -y
```

Onde,

**install** é o parâmetro para o apt fazer o download e instalação de um novo recurso; **net-tools** é o pacote básico de ferramentas para atuar na interface de rede;

```
root@debian:/home/usuario# apt install net-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and
 libgnutls-dane0 libunbound8
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
```

O próximo passo é instalar o openssh-server, conforme comando abaixo.

```
1. sudo apt install openssh-server -y
```

Onde,

**openssh-server** é o conjunto de recursos do programa ser será executado como serviço SSH;

No próximo passo, é fundamental ativar o serviço openssh-server para ser executado na inicialização do GNU/Linux, para isso, digite o comando:

```
1. sudo systemctl enable ssh.service
```

Onde,

**enable** é o parâmetro utilizado para ativar o serviço ssh na inicialização do GNU/Linux;

**ssh.service** é o parâmetro que informa que será aplicado sobre o arquivo **ssh.service** em **/etc/systemd/system/** ou em versões mais antigas o arquivo **ssh** em **/etc/inti.d/**.

Agora é importante avaliar se o serviço SSH está em execução, para isso digite o comando

```
1. sudo systemctl status ssh.service
```

Onde,

**status** é o parâmetro que informa ao **systemctl** que deve ser listado;

```
root@debian:~# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
 Loaded: loaded (/lib/systemd/system/ssh.service; enabled; v
 Active: active (running) since Wed 2020-11-18 19:12:02 -03;
 Docs: man:sshd(8)
 man:sshd_config(5)
 Process: 387 ExecStartPre=/usr/sbin/sshd -t (code=exited, st
 Main PID: 399 (sshd)
 Tasks: 1 (limit: 2354)
 Memory: 2.9M
 CGroup: /system.slice/ssh.service
 └─399 /usr/sbin/sshd -D

Nov 18 19:12:02 debian systemd[1]: Starting OpenBSD Secure She
Nov 18 19:12:02 debian sshd[399]: Server listening on 0.0.0.0
```

Caso o OpenSSH Server não esteja “**active**”, então digite o comando abaixo.

```
1. sudo systemctl start ssh.service
```

Onde,

**start** é o parâmetro inicia o serviço ssh;

## 16.4 Acessando o terminal por uso de cliente SSH

Sabendo o IP e o usuário da Virtual Machine que possui o **openssh-server** instalado, em um computador com GNU/Linux digite o comando para se conectar.

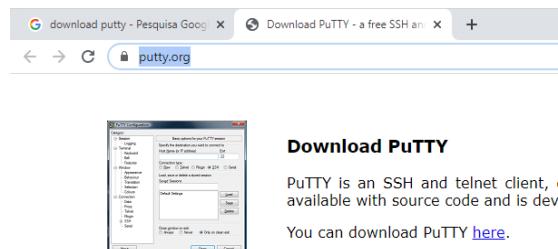
```
1. ssh usuario@192.168.0.18
```

Onde,

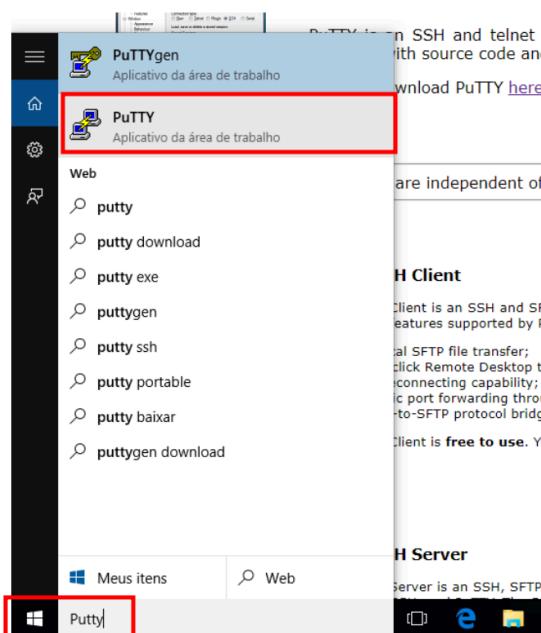
**usuário** é o usuário do GNU/Linux que possui o openssh-server instalado;  
**192.168.0.18** é o Endereço IP do GNU/Linux que possui o openssh-server instalado;

```
well@wpo:~$ ssh usuario@192.168.0.18
The authenticity of host '192.168.0.18 (192.168.0.18)' can't be established.
ECDSA key fingerprint is SHA256:GNyCCeMkH6ut4v+mSMn0Hws0y6vu02uv9fG+xnBsp0w.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.18' (ECDSA) to the list of known hosts.
usuario@192.168.0.18's password: ■
```

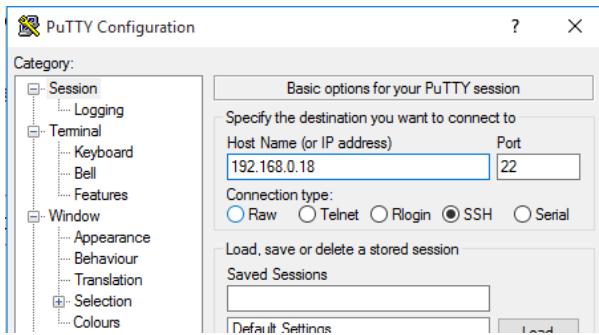
Digite a senha e entre no terminal da máquina remota, pronto, estando em um GNU/Linux acessando um GNU/Linux tudo é fácil, vamos agora utilizar o Windows. Para a plataforma Windows é preciso instalar o PUTTY, um pequeno programa distribuído gratuitamente e é o clássico quando se fala de SSH em Windows.



Instale o Putty, é um padrão Next, Next e Finish. Após a instalação ele está acessível pelo menu do Windows, conforme figura abaixo.



O layout do Putty é bem simples, conforme figura abaixo.



Informe o IP da máquina que possui o openssh-server instalado e clique em “Open”, o Putty deverá questionar sobre Certificado, informe “Sim” e continue.



Informe o usuário e a senha, e pronto, está acessando a Virtual Machine por SSH.

## 16.4 Transferindo arquivos

Secure Copy Protocol (SCP) é um protocolo de transmissão de arquivos entre computadores com segurança independente da rede, pois se apoia no protocolo Secure Shell (SSH). Utilizando o SSH para transferência de dados ele usa os mesmos mecanismos de autenticação, garantindo assim a autenticidade e confidencialidade dos dados em trânsito.

O comando SCP no GNU/Linux é um programa cliente para SFTP este comando foi projetado para ser semelhante ao comando ftp e naturalmente faz parte do pacote OpenSSH.

Em uma GNU/Linux como cliente pode-se enviar arquivos para um servidor com serviço openssh-server conforme comando abaixo:

```
1. scp /tmp/meuarquivo.txt usuario@192.168.0.18:/home/userlinux/
```

Onde,

**/tmp/meuarquivo.txt** é um arquivo na máquina local que será enviado para a máquina com openssh-server;

**usuario@192.168.0.18** é o usuário na máquina remota bem como o IP da máquina remota que possui o openssh-server instalado;

**:/home/userlinux/** é o diretório que receberá o arquivo que será enviado.

Caso queira enviar um diretório e incluindo o diretório basta apenas apontar para o diretório (sem a barra no final) e utilizar o parâmetro recursivo.

```
1. scp -rp /tmp/diretorio usuario@192.168.0.18:/home/userlinux/
```

Onde,

**-rp** é uma sequencia de parâmetros que informa que será recursivo e caso algum diretório esteja vazio é para ser criado lá na máquina de destino;

**/tmp/diretorio** é um diretório com vários arquivos e naturalmente outros diretórios;

Caso queira enviar o conteúdo de um diretório altere o comando conforme exemplo abaixo.

```
1. scp -rp /tmp/diretorio/* usuario@192.168.0.18:/home/userlinux/diretorio/
```

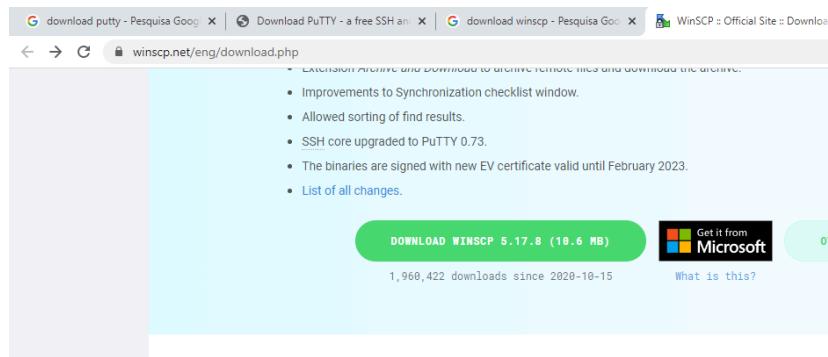
Onde,

**/\*** indica que é o conteúdo do diretório.

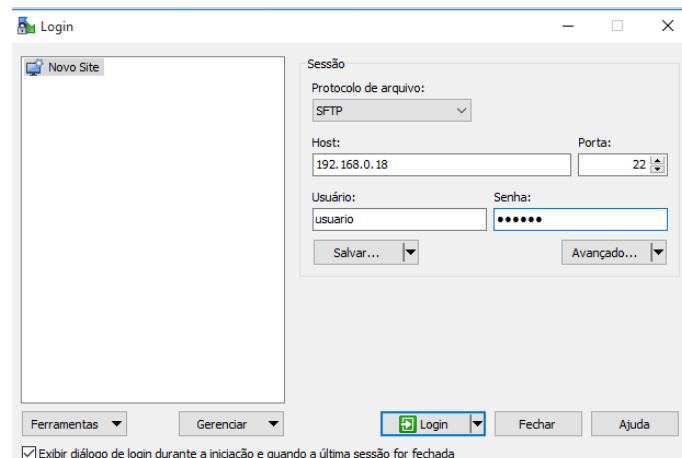
Para trazer arquivos do computador remoto para o computador local, basta inverter os parâmetros dos exemplos anteriores, veja o exemplo.

```
1. scp -rp usuario@192.168.0.18:/home/userlinux/diretorio/* /tmp/diretorio/
```

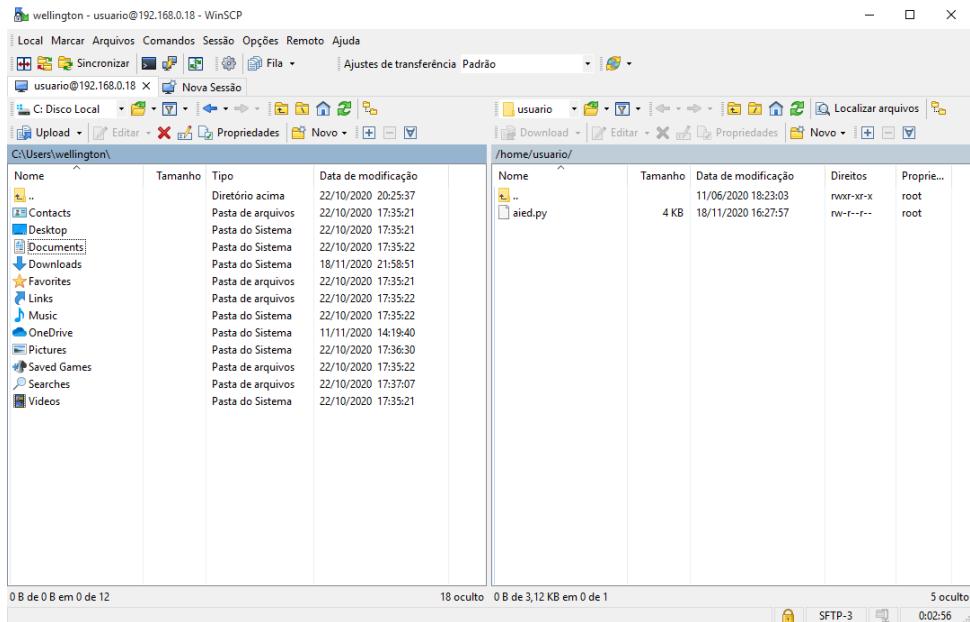
Na plataforma Windows você deve baixar e instalar o WinSCP, que é um aplicativo gráfico, conforme figura abaixo.



Ao rodar o aplicativo, cadastre o servidor que possui o openssh-server instalado, conforme figura abaixo.



Quando clicar em Login pela primeira vez, aceite o certificado que será salvo no Windows, processo obrigatório para o protocolo SSH. Na tela abaixo você pode ver que do lado esquerdo é possível acessar arquivos da máquina Windows, já na aba da direita, é possível acessar o sistema de arquivos do GNU/Linux que possui o openssh-server instalado.



Como prática, copie arquivos do Windows para o diretório /tmp do GNU/Linux.

## 16.5 Biblioteca libssh

O projeto libssh é para programadores que precisam de uma implementação de SSH em funcionamento por meio de uma biblioteca. O controle completo do cliente é feito pelo programador. Com o libssh, você pode executar programas remotamente, transferir arquivos, usar um túnel seguro e transparente para seus programas remotos (automatização). Com sua implementação de FTP Seguro, você pode transferir arquivos para servidores remotos facilmente, sem programas de terceiros que não sejam libcrypto (de openssl), libgcrypt ou mbedTLS.

O site oficial da biblioteca pode ser acessado pela url <https://www.libssh.org/>

Quem utiliza:

- KDE** Utiliza para transferência de arquivos por sftp;
- GitHub** Implementaram seu servidor git ssh com libssh
- QEMU** driver de bloco ssh usa libssh
- GNU** Utiliza para transmissão de mídia;

Para desenvolver, deve-se instalar a biblioteca libssh para desenvolvedores, conforme comando abaixo.

```
1. sudo apt install libssh-dev -y
```

Para codificação, será necessário importar a biblioteca libssh.h e naturalmente a biblioteca iostream para uso do cout. Crie um novo arquivo com o nome chamado sshclient.cpp e codifique o código abaixo. No código repare que somente será realizada a autenticação, para isso confirme o IP da máquina virtual que possui o Openssh instalado.

```

1. #include <libssh/libssh.h>
2. #include <iostream>
3.
4. int main()
5. {
6. ssh_session my_ssh_session;
7. int rc;
8. std::string password = "123456";
9. std::string usuario = "usuario";
10.
11. // Abrir uma sessão
12. my_ssh_session = ssh_new();
13. if (my_ssh_session == NULL)
14. exit(-1);
15. ssh_options_set(my_ssh_session, SSH_OPTIONS_HOST, "192.168.3.26");
16.
17. // Conectar com o servidor
18. rc = ssh_connect(my_ssh_session);
19. if (rc != SSH_OK)
20. {
21. fprintf(stderr, "Erro ao se conectar com o servidor: %s\n",
22. ssh_get_error(my_ssh_session));
23. ssh_free(my_ssh_session);
24. exit(-1);
25. }
26. // Autenticar
27. rc = ssh_userauth_password(my_ssh_session, usuario.c_str(),
28. password.c_str());
29. if (rc != SSH_AUTH_SUCCESS)
30. {
31. fprintf(stderr, "Erro ao autenticar: %s\n",
32. ssh_get_error(my_ssh_session));
33. ssh_disconnect(my_ssh_session);
34. ssh_free(my_ssh_session);
35. exit(-1);
36. }
37. // Executar alguma ação
38. std::cout << "Conectado, aí de é top." << std::endl;
39.
40. // Desconectar do servidor
41. ssh_disconnect(my_ssh_session);
42. ssh_free(my_ssh_session);
43. }
```

Para compilar é fácil, com g++ informe o caminho do arquivo .cpp e informe que o compilador deve utilizar a biblioteca libssh com o parâmetro -lssh.

```
1. g++ -o sshclient sshclient.cpp -lssh
```

Na linha 37 do código anterior, é o trecho que enviamos comandos ou arquivos para o servidor, se tudo der certo a mensagem será exibida na interface. No código abaixo temos um exemplo de execução de um processo no servidor, foi criado uma função.

```
1. int show_remote_processes(ssh_session session)
2. {
3. ssh_channel channel;
4. int rc;
5. char buffer[256];
6. int nbytes;
7.
8. channel = ssh_channel_new(session);
9. if (channel == NULL)
10. return SSH_ERROR;
11. // No ssh você tem a conexão, e abre sessões com o servidor para
12. execução.
12. rc = ssh_channel_open_session(channel);
13. if (rc != SSH_OK)
14. {
15. ssh_channel_free(channel);
16. return rc;
17. }
18.
19. rc = ssh_channel_request_exec(channel, "ps aux");
20. if (rc != SSH_OK)
21. {
22. ssh_channel_close(channel);
23. ssh_channel_free(channel);
24. return rc;
25. }
26.
27. nbytes = ssh_channel_read(channel, buffer, sizeof(buffer), 0);
28. while (nbytes > 0)
29. {
30. if (write(1, buffer, nbytes) != (unsigned int) nbytes)
31. {
32. ssh_channel_close(channel);
33. ssh_channel_free(channel);
34. return SSH_ERROR;
35. }
36. nbytes = ssh_channel_read(channel, buffer, sizeof(buffer), 0);
37. }
38.
39. if (nbytes < 0)
40. {
41. ssh_channel_close(channel);
42. ssh_channel_free(channel);
43. return SSH_ERROR;
44. }
45.
46. ssh_channel_send_eof(channel);
47. ssh_channel_close(channel);
48. ssh_channel_free(channel);
49.
50. return SSH_OK;
51. }
```

O programador pode desenvolver uma função que recebe como parâmetro o comando e os argumentos.

## 16.5 Práticas do capítulo

Nesta sessão o aluno deve realizar a prática, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para resolver, isto já é levado em consideração.

### 16.5.1 Prática 57671beb 01: Configurando o serviço Openssh-server

Nesta prática o aluno deve ter a expertise de instalar pacotes com apt e configurar a interface de rede.

1. Configure o Adaptador de Rede para modo Bridge, e configure a interface de rede para auto DHCP (geralmente é padrão);
2. execute o comandos de instalação:
  - a. sudo apt update
  - b. sudo apt install net-tools -y
  - c. sudo apt install openssh-server -y;
3. anote o IP da Virtual Machine;
4. No seu computador, se for Windows instale o putty;
5. Pelo seu computador, acesse o putty e informe o IP da Virtual Machine;
6. De dentro da Virtual Machine pelo putty, execute o validador aied.

Para validar a prática, execute o comando no servidor utilizando uma sessão aberta por SSH cliente:

```
1. sudo aied validar 57671beb checkpoint01
```

Então todos os Check-points devem estar em verde conforme imagem, caso algum item esteja em vermelho a ferramenta vai descrever a ação que o aluno deve tomar para solucionar o item.

```
Práticas aied.com.br
Pratica de instalação do OpenSSH
Atencao, serao enviado os seguintes dados para o servidor de validacao:
 Output do comando: ping -c 1 google.com
 Output do comando: openssl version -a
 Output do comando: systemctl status ssh --no-pager
 Output do comando: netstat -V
 Output do comando: netstat -tanp | grep ESTABLISHED | grep ssh | awk '{print $5}' | sed 's/:.*'
///

'Deseja continuar (y|n)?:y
++++++ Inicio da validacao ++++++
1 - command -> ping -c 1 google.com
1.1 regex /0%\s+packet\s+loss/ Status: True
2 - command -> openssl version -a
2.1 regex /OpenSSL\s+\d+/ Status: True
3 - command -> systemctl status ssh --no-pager
3.1 text active (running) Status: True
3.2 text enabled; Status: True
4 - command -> netstat -V
4.1 regex /net-tools(.*)/ Status: True
5 - command -> netstat -tanp | grep ESTABLISHED | grep ssh | awk '{print $5}' | sed 's/:.*//'
5.1 regex /\d+\.\d+\.\d+/ Status: True
```

Onde,

**1 validar** se a máquina possui acesso a rede mundial para se fazer o download dos programas;

**2 validar** a versão do SSH, naturalmente válida se está instalado;

**3 validar** se o serviço está ativo e ativo na inicialização do Sistema Operacional;

**4 validar** a instalação no netstat, para teste do item 5;

**5 validar** que o aluno está usando o SSH cliente para se conectar no servidor.

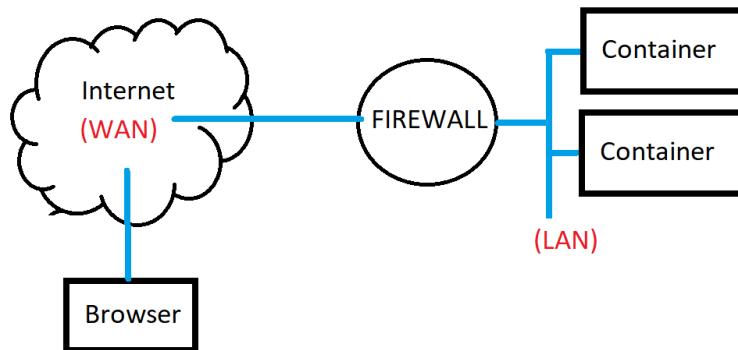
adicionar questões de segurança

evoluir e mostrar como fazer tunnel ssh

## 17 Serviço WEB com Tomcat e Java

O Apache Tomcat é um servidor container especialista em aplicações WEB com Java sendo a solução mais utilizada para servlet e JavaServer Pages. O Tomcat é uma implementação de código aberto das tecnologias Java Servlet e JavaServer Pages, lançada pela [Apache Software Foundation](#).

Em uma empresa o container web não está exposto na Internet, sempre temos servidores atrás de Routers e Firewalls. A arquitetura básica é:



O Browser sendo utilizado por uma pessoa acessa a rede mundial e pelo serviço de Internet e realiza requisições para o IP público do FIREWALL, o IP público está na ligação entre o FIREWALL e a WAN. O FIREWALL então analisa a requisição e então se é permitido repassar a requisição para o Container adequado.

Como o material é para estudo, então o aluno deve admitir que a WAN (da imagem) é a rede local de sua casa ou de sua escola, e a LAN (da imagem) é a rede interna do VirtualBox. Na prática vamos utilizar apenas 1 Container e 1 Firewall.

Comece importando a máquina virtual chamada GATEWAY e vamos a chamar de FIREWALL, a configuração física desta máquina está perfeita para esta prática, vamos apenas editar o arquivo de regras Nftables.

Importe a segunda máquina virtual com apenas um adaptador de rede, chame esta máquina de TOMCAT e adicione nela 2048 MB de RAM. O único adaptador desta máquina virtual deve estar ligada na rede interna do VirtualBox.

### 17.2 Criando a regra nftables no FIREWALL

Para realizar esta prática o firewall será um computador virtual, comece a prática importando a máquina Gateway com o nome de Firewall.

Será necessário adicionar mais duas regras no arquivo /usr/local/sbin/gateway.sh, essas duas regras redirecionaram as requisições da rede externa mais abrangente para a rede interna.

```

1. nft add rule inet nat prerouting iifname enp0s3 tcp dport 80 dnat ip to
 192.168.200.5:8080
2. nft add rule inet filter forward iifname enp0s3 oifname enp0s8 ip daddr
 192.168.200.5 tcp dport 8080 accept

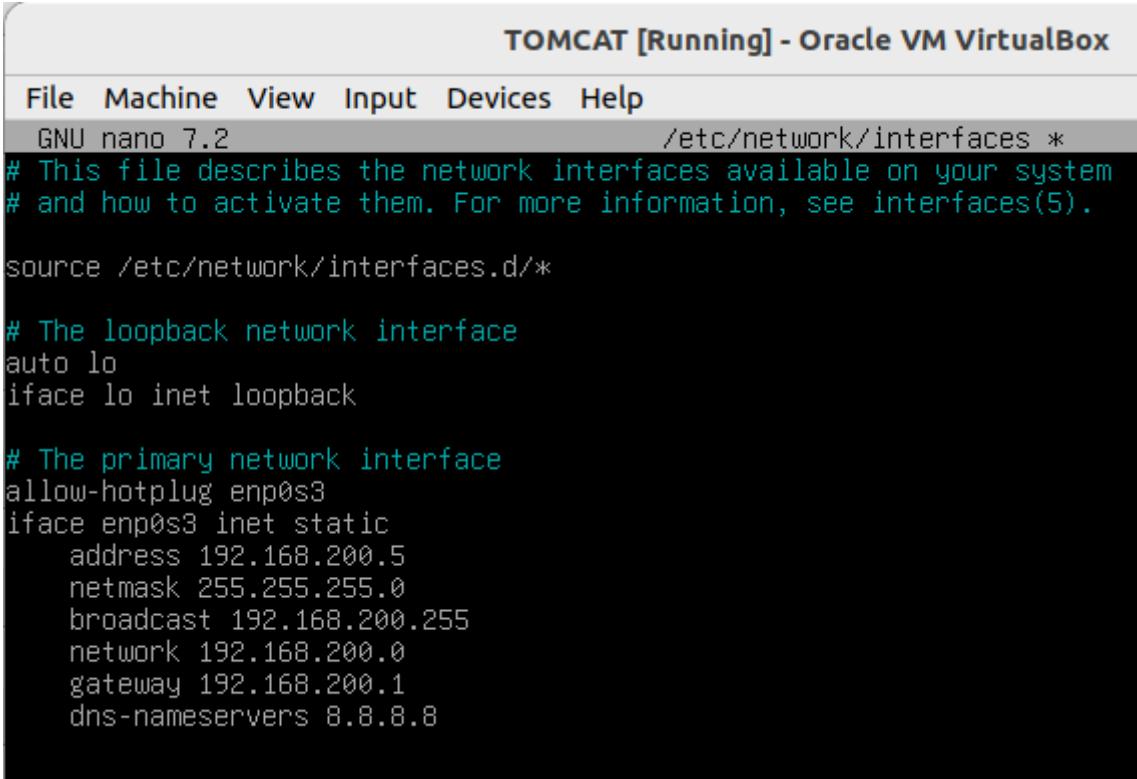
```

A primeira regra adiciona na chain Prerouting um redirecionamento de protocolos tcp destinados para a porta 80 na interface externa para a IP 192.168.200.5 porta 8080. Já a segunda regra permite o redirecionamento na tabela filter.

### 17.3 Configuração de rede da máquina TOMCAT

Importe uma máquina virtual existente, uma que não tenha nenhuma prática já executada nela, importe com o nome Tomcat. Essa máquina virtual deve ter apenas 1 único adaptador ligado a rede interna. Inicie a máquina virtual e atribua o endereço conforme imagem abaixo.

**TOMCAT [Running] - Oracle VM VirtualBox**



```

GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.5
 netmask 255.255.255.0
 broadcast 192.168.200.255
 network 192.168.200.0
 gateway 192.168.200.1
 dns-nameservers 8.8.8.8

```

Reinic peace a máquina. Quando logar novamente, execute envio de ICMP para o endereço google.com.br, espera-se que o gateway com nome firewall redirecione as requisições por SNAT.

### 17.2 Instalando o Java no Linux Terminal

O Tomcat exige que o Java seja instalado no servidor para que qualquer código de aplicativo da Web Java possa ser executado. Podemos satisfazer esse requisito instalando o OpenJDK com o apt-get.

Primeiro, atualize seu índice de pacote apt, execute um Update sempre que for instalar um ambiente.

```
1. sudo apt update -y
```

Veja os arquivos sendo baixados dos repositórios. Em seguida, instale o pacote Java Development Kit com o apt-get:

```
1. sudo apt install default-jdk -y
```

Veja a saída do output da instalação:

```
Get:18 http://ftp.br.debian.org/debian stretch/main i386 libxrender1 i386 1:0.9.10-1 [34.1 kB]
Get:19 http://ftp.br.debian.org/debian stretch/main i386 libcairo2 i386 1.14.8-1 [827 kB]
Get:20 http://ftp.br.debian.org/debian stretch/main i386 libcroco3 i386 0.6.11-3 [149 kB]
Get:21 http://ftp.br.debian.org/debian stretch/main i386 libthai-data all 0.1.26-1 [166 kB]
Get:22 http://ftp.br.debian.org/debian stretch/main i386 libdatrie1 i386 0.2.10-4+b1 [38.0 kB]
Get:23 http://ftp.br.debian.org/debian stretch/main i386 libthai0 i386 0.1.26-1 [53.4 kB]
Get:24 http://ftp.br.debian.org/debian stretch/main i386 libpango-1.0-0 i386 1.40.5-1 [326 kB]
Get:25 http://ftp.br.debian.org/debian stretch/main i386 libgraphite2-3 i386 1.3.10-1 [86.9 kB]
Get:26 http://ftp.br.debian.org/debian stretch/main i386 libharfbuzz0b i386 1.4.2-1 [680 kB]
Get:27 http://ftp.br.debian.org/debian stretch/main i386 libpangoft2-1.0-0 i386 1.40.5-1 [208 kB]
Get:28 http://ftp.br.debian.org/debian stretch/main i386 libpangocairo-1.0-0 i386 1.40.5-1 [195 kB]
Get:29 http://ftp.br.debian.org/debian stretch/main i386 librsvg2-2 i386 2.40.16-1+b1 [293 kB]
11% [29 librsvg2-2 174 kB/293 kB 59%] 240 kB/s 7min 23s
```

## 17.3 Criando um ambiente seguro para seu container

Agora que o Java está instalado, podemos criar um Tomcat usuário, que será usado para executar o serviço Tomcat, isso é fundamental para se ter um ambiente seguro para o seu container.

Por questões de segurança, o Tomcat deve ser executado como um usuário sem privilégios (ou seja, não como root). Criaremos um novo usuário e grupo que executará o serviço Tomcat. Primeiro, crie um novo grupo chamado tomcat conforme comando abaixo:

```
1. sudo groupadd tomcat
```

Em seguida, crie um novo usuário chamado tomcat. Tornaremos esse usuário um membro do grupo tomcat, com um diretório inicial de /opt/tomcat (onde instalaremos o Tomcat) e com um shell de /bin/false (para que ninguém possa fazer login na conta):

```
1. sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

## 17.4 Instalar o Tomcat

A melhor maneira de instalar o Tomcat é baixar a versão binária mais recente e configurá-la manualmente. Encontre a versão mais recente do [Tomcat na página de downloads do Tomcat](http://tomcat.apache.org/download-10.cgi). No momento da redação deste texto, a versão mais recente (estável) é 10.0.5, mas você deve usar uma versão estável posterior, se estiver disponível, para isso navegue na página aberta (link acima). Evite versões alfa ou beta.

Save the date!

### Apache Tomcat

- Home
- Taglibs
- Maven Plugin

### Download

Which version?

- Tomcat 10
- Tomcat 9
- Tomcat 8
- Tomcat 7
- Tomcat Migration Tool for Jakarta EE
- Tomcat Connectors
- Tomcat Native
- Taglibs
- Archives

### Documentation

- Tomcat 10.0
- Tomcat 9.0
- Tomcat 8.5
- Tomcat 7.0
- Tomcat Connectors
- Tomcat Native
- Wiki
- Migration Guide
- Presentations
- Specifications

### Problems?

- Security Reports
- Find help
- FAQ
- Mailing Lists
- Bug Database
- IRC

Unsure which version you need? Specification versions implemented, minimum Java version required and

Users of Tomcat 10 onwards should be aware that, as a result of the migration tool, the version of Tomcat 10 and earlier to Tomcat 10 and later. A [migration tool](#) is under development.

### Quick Navigation

[KEYS](#) | [10.0.5](#) | [Browse](#) | [Archives](#)

### Release Integrity

You **must verify** the integrity of the downloaded files. We provide OpenPGP signatures for every release file. You can verify the checksum for your download, and make sure it is the same as ours.

### Mirrors

You are currently using <https://downloads.apache.org/>. If you encounter problems with this mirror, please try another.

Other mirrors: <https://downloads.apache.org/>

### 10.0.5

Please see the [README](#) file for packaging information. It explains what every distribution contains.

### Binary Distributions

- Core:
  - [zip \(pgp, sha512\)](#)
  - [tar.gz \(pgp, sha512\)](#)
  - [32-bit Windows zip \(pgp, sha512\)](#)
  - [64-bit Windows zip \(pgp, sha512\)](#)
  - [32-bit/64-bit Windows Service Installer \(pgp, sha512\)](#)
- Full documentation:
  - [tar.gz \(pgp, sha512\)](#)
- Deployer:
  - [zip \(pgp, sha512\)](#)
  - [tar.gz \(pgp, sha512\)](#)
- Embedded:

**Versoes do TOMCAT**

**Faca download do .tar.gz**

Copie o endereço do arquivo tar.gz, com o botão direito do mouse.

10.0.10

Please see the [README](#) file for packaging information. It explains what every

**Binary Distributions**

- Core:
  - [zip \(pgp, sha512\)](#)
  - [tar.gz](#) Open link in new tab
  - [32-b](#) Open link in new window
  - [64-b](#) Open link in incognito window
  - [32-b](#) Send link to Motorola Phone
- Full document:
  - [tar.gz](#) Save link as...
  - [Copy link address](#)
- Deployer:
  - [zip \(pgp, sha512\)](#) Inspect
  - [tar.gz \(pgp, sha512\)](#)
- Embedded:
  - [tar.gz \(pgp, sha512\)](#)
  - [zip \(pgp, sha512\)](#)

Em seguida, mude o cursor do sistema de arquivos usando o comando cd para o **/tmp**. Este é um bom diretório para baixar itens, como o Tomcat. Use curl para baixar o link que você copiou do site da Tomcat, se não tiver instalado é fácil, baixa executar o comando

1. sudo apt install curl -y

```
usuario@debian:/tmp$
usuario@debian:/tmp$ sudo apt install curl
```

Com o curl instalado, então informe a url da versão do Tomcat escolhida, conforme exemplo abaixo.

1. cd /tmp/
2. curl -O  
<https://downloads.apache.org/tomcat/tomcat-10/v10.0.5/bin/apache-tomcat-10.0.5.tar.gz>

Vamos instalar o Tomcat no diretório **/opt/tomcat**. Crie o diretório e extraia o arquivo compactado com estes comandos:

1. sudo mkdir /opt/tomcat

No comando abaixo, digite apache e já pressione o TAB

1. sudo tar xzvf apache-tomcat-10.0.5.tar.gz -C /opt/tomcat --strip-components=1

O CURL realiza download de arquivos, caso digite errado, ou seja, uma url que não existe então o CURL irá realizar o download da página 404. É natural que uma página HTML não pode ser descompactada pois não é tar.gz, então na etapa de descompactação irá falhar. Observe o tamanho do arquivo.

```
usuario@debian:/tmp$ curl -O http://ftp.unicamp.br/pub/apache/tomcat/tomcat-8/v8.5.46/bin/apache-tomcat-8.5.46.tar.gz
 % Total % Received % Xferd Average Speed Time Time Time Current
 0 0 0 0 6564k 0 0:00:01 0:00:01 --:--:-- 6561k
usuario@debian:/tmp$ ls -l
total 11364
-rw-r--r-- 1 usuario usuario 11623939 Oct 8 14:34 apache-tomcat-8.5.46.tar.gz
drwxr-xr-x 2 root root 4096 Oct 8 14:18 hsperfdata_root
drwx----- 3 root root 4096 Oct 8 14:05 systemd-private-7f615277d5ef47cab8469131d9e23e51-systemd-timesyncd.service-HCb201
usuario@debian:/tmp$ sudo tar xzvf apache-tomcat-8.5.46.tar.gz -C /opt/tomcat --strip-components=1
```

Em seguida, podemos configurar as permissões de usuário adequadas para nossa instalação.

## 17.5 Configurando as permissões para o serviço

O usuário **tomcat** que configuramos precisa ter acesso à instalação do Tomcat. Vamos configurar isso agora. Mude para o diretório em que descompactamos a instalação do Tomcat, o /opt/tomcat.

```
usuario@debian:/tmp$ cd /opt/tomcat/
usuario@debian:/opt/tomcat$ ls
bin CONTRIBUTING.md logs RELEASE-NOTES webapps
BUILDING.txt lib NOTICE RUNNING.txt work
conf LICENSE README.md temp
usuario@debian:/opt/tomcat$
```

Conceda ao tomcat grupo a propriedade de todo o diretório de instalação:

1. sudo chgrp -R tomcat /opt/tomcat

```
usuario@debian:/opt/tomcat$ sudo chgrp -R tomcat /opt/tomcat
usuario@debian:/opt/tomcat$
```

Em seguida, forneça ao tomcat grupo acesso de leitura ao conf diretório e a todo o seu conteúdo e execute o acesso ao próprio diretório:

1. sudo chmod -R g+r /opt/tomcat/conf
2. sudo chmod g+x /opt/tomcat/conf

Faça o tomcat usuário proprietário dos webapps, work, temp, e logs diretórios:

1. sudo chown -R tomcat /opt/tomcat/webapps/ /opt/tomcat/work/ /opt/tomcat/temp/ /opt/tomcat/logs/

Agora que as permissões apropriadas foram configuradas, podemos criar um arquivo de serviço systemd para gerenciar o processo do Tomcat.

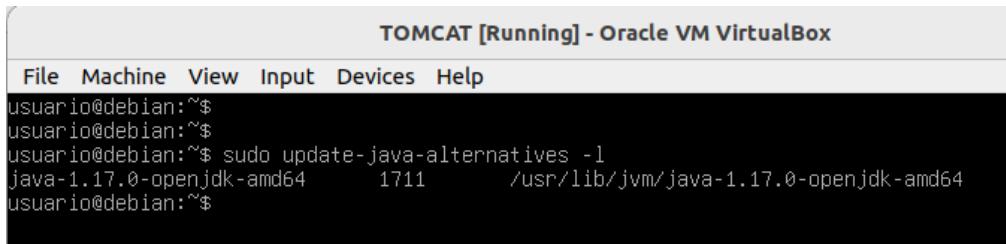
```
usuario@debian:/opt/tomcat$ sudo chmod -R g+r conf
usuario@debian:/opt/tomcat$ sudo chmod g+x conf
usuario@debian:/opt/tomcat$ sudo chown -R tomcat webapps/ work/ temp/ logs/
usuario@debian:/opt/tomcat$
```

## 17.6 Adicionando o Tomcat na inicialização do Linux

Queremos poder executar o Tomcat como um serviço, portanto, configuramos o arquivo de serviço systemd. O Tomcat precisa saber onde o Java está instalado. Esse caminho é conhecido como "JAVA\_HOME". A maneira mais fácil de procurar esse local é executando este comando:

```
1. sudo update-java-alternatives -l
```

A saída deste comando será similar ao exemplo abaixo:



```
TOMCAT [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~$
usuario@debian:~$
usuario@debian:~$ sudo update-java-alternatives -l
java-1.17.0-openjdk-amd64 1711 /usr/lib/jvm/java-1.17.0-openjdk-amd64
usuario@debian:~$
```

A JAVA\_HOME variável correta pode ser construída pegando a saída da última coluna (destacada em vermelho). Dado o exemplo acima, o correto JAVA\_HOME para este servidor seria: /usr/lib/jvm/java-1.8.0-openjdk-amd64

Com essa informação, podemos criar o arquivo de serviço systemd. Abra um arquivo chamado tomcat.service no /etc/systemd/system diretório digitando:

```
1. sudo nano /etc/systemd/system/tomcat.service
```

```
usuario@debian:/opt/tomcat$
usuario@debian:/opt/tomcat$ cd /etc/systemd/system/
usuario@debian:/etc/systemd/system$
usuario@debian:/etc/systemd/system$ sudo nano tomcat.service_
```

Cole o seguinte conteúdo no seu arquivo de serviço. Modifique o valor de, JAVA\_HOME se necessário, para corresponder ao valor encontrado em seu sistema. Você também pode modificar as configurações de alocação de memória especificadas em CATALINA\_OPTS:

---

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target
```

```

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.17.0-openjdk-amd64
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true
-Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

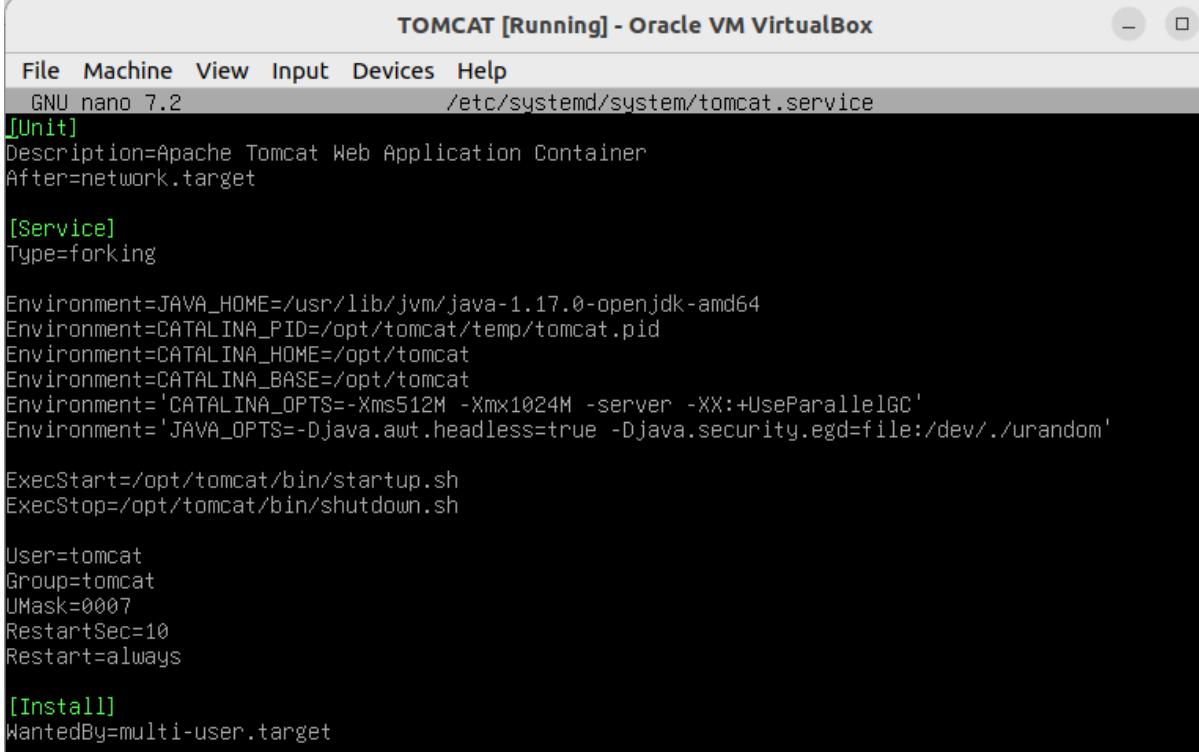
User=tomcat
Group=tomcat
UMask=0007
RestartSec=60
Restart=always

[Install]
WantedBy=multi-user.target

```

---

Veja como fica:



```

TOMCAT [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/systemd/system/tomcat.service
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.17.0-openjdk-amd64
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target

```

Quando terminar, salve e feche o arquivo. Em seguida, recarregue o daemon `systemd` para que ele conheça nosso arquivo de serviço e inicie o serviço:

```
1. sudo systemctl daemon-reload
```

```
2. sudo systemctl enable tomcat
3. sudo systemctl start tomcat
4. sudo systemctl status tomcat
```

Veja a saída do status:

```
usuario@debian:~$ sudo systemctl daemon-reload
usuario@debian:~$ sudo systemctl start tomcat
usuario@debian:~$ sudo systemctl status tomcat
● tomcat.service - Apache Tomcat Web Application Container
 Loaded: loaded (/etc/systemd/system/tomcat.service; disabled; vendor preset:
 Active: active (running) since Tue 2019-10-08 15:05:17 -03; 9s ago
 Process: 661 ExecStart=/opt/tomcat/bin/startup.sh (code=exited, status=0/SUCCE
 Main PID: 668 (java)
 Tasks: 44 (limit: 4915)
 CGroup: /system.slice/tomcat.service
 └─668 /usr/lib/jvm/java-1.8.0-openjdk-i386/jre/bin/java -Djava.util.l

Oct 08 15:05:17 debian systemd[1]: Starting Apache Tomcat Web Application Contai
Oct 08 15:05:17 debian systemd[1]: Started Apache Tomcat Web Application Contain
lines 1-11/11 (END)
```

## 17.7 Configurar a interface de gerenciamento da Web do Tomcat

Para usar o aplicativo Web do gerente que acompanha o Tomcat, precisamos adicionar um login ao nosso servidor Tomcat. Faremos isso editando o `tomcat-users.xml` arquivo:

```
1. sudo nano /opt/tomcat/conf/tomcat-users.xml
```

Você deseja adicionar um usuário que possa acessar os manager-gui admin-gui(aplicativos da web que acompanham o Tomcat). Você pode fazer isso definindo um usuário, semelhante ao exemplo abaixo, entre as `tomcat-userstags`.

TOMCAT [Running] - Oracle VM VirtualBox

```
File Machine View Input Devices Help
GNU nano 7.2 /opt/tomcat/conf/tomcat-users.xml *
```

The users below are wrapped in a comment and are therefore ignored. If you wish to configure one or more of these users for use with the manager web application, do not forget to remove the <!... ...> that surrounds them. You will also need to set the passwords to something appropriate.

```
-->
<!--
<user username="admin" password="" roles="manager-gui"/>
<user username="robot" password="" roles="manager-script"/>
-->
<!--
```

The sample user and role entries below are intended for use with the examples web application. They are wrapped in a comment and thus are ignored when reading this file. If you wish to configure these users for use with the examples web application, do not forget to remove the <!... ...> that surrounds

```
<tomcat-users
 <user username="admin" password="123456" roles="manager-gui,admin-gui"/>
</tomcat-users>
```

Veja como fica:

TOMCAT [Running] - Oracle VM VirtualBox

```
File Machine View Input Devices Help
GNU nano 7.2 /opt/tomcat/conf/tomcat-users.xml *
```

The users below are wrapped in a comment and are therefore ignored. If you wish to configure one or more of these users for use with the manager web application, do not forget to remove the <!... ...> that surrounds them. You will also need to set the passwords to something appropriate.

```
-->
<user username="admin" password="123456" roles="manager-gui,admin-gui"/>
<!--
<user username="robot" password="" roles="manager-script"/>
-->
<!--
```

The sample user and role entries below are intended for use with the examples web application. They are wrapped in a comment and thus are ignored when reading this file. If you wish to configure these users for use with the

Salve e feche o arquivo quando terminar. Por padrão, as versões mais recentes do Tomcat restringem o acesso aos aplicativos Manager e Host Manager a conexões provenientes do próprio servidor. Como estamos instalando em uma máquina remota, você provavelmente desejará remover ou alterar essa restrição. Para alterar as restrições de endereço IP, abra os context.xml arquivos apropriados. Para o aplicativo Manager, digite:

1. sudo nano /opt/tomcat/webapps/manager/META-INF/context.xml

```
usuario@debian:/tmp$ sudo nano /opt/tomcat/webapps/manager/META-INF/context.xml
```

Para o aplicativo Host Manager, digite:

```
File: /opt/tomcat/webapps/manager/META-INF/context.xml

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<Context antiResourceLocking="false" privileged="true" >
 <Valve className="org.apache.catalina.valves.RemoteAddrValve"
 allow="^\d+\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:1" />
 <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer$"
</Context>
```

1. sudo nano /opt/tomcat/webapps/**host-manager**/META-INF/context.xml

```
usuario@debian:/tmp$ sudo nano /opt/tomcat/webapps/host-manager/META-INF/context
.xml
```

No interior, comente a restrição de endereço IP para permitir conexões de qualquer lugar. Como alternativa, se você deseja permitir acesso apenas a conexões provenientes de seu próprio endereço IP, você pode adicionar seu endereço IP público à lista:

1. <Context antiResourceLocking="false" privileged="true" >
2. <!--<Valve className="org.apache.catalina.valves.RemoteAddrValve"
3. &nbsp;&nbsp;&nbsp;allow="^\d+\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:1" />-->
4. </Context>

```
File: /opt/tomcat/webapps/host-manager/META-INF/context.xml

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<Context antiResourceLocking="false" privileged="true" >
 <Valve className="org.apache.catalina.valves.RemoteAddrValve"
 allow="^\d+\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:1" />
 <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer$"
</Context>
```

Salve e feche os arquivos quando terminar.

1. sudo systemctl restart tomcat.service
2. sudo systemctl status tomcat.service

## 17.8 Acessando a interface da Web

Vamos agora testar, para isso navegue até /tmp conforme imagem abaixo. E em seguida execute o comando wget.

```
usuario@debian:~$ cd /tmp
usuario@debian:/tmp$ wget http://localhost:8080/
--2019-10-08 15:15:07-- http://localhost:8080/
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)|::1|:8080... connected.
HTTP request sent, awaiting response... 200
Length: unspecified [text/html]
Saving to: 'index.html'

index.html [<=>] 10.93K --.-KB/s in 0.02s

2019-10-08 15:15:11 (629 KB/s) - 'index.html' saved [11195]
```

Veja que o arquivo index.html foi baixado, agora pode ver o arquivo.

```
usuario@debian:/tmp$ cat index.html

<!DOCTYPE html>
<html lang="en">
 <head>
 <meta charset="UTF-8" />
 <title>Apache Tomcat/8.5.46</title>
 <link href="favicon.ico" rel="icon" type="image/x-icon" />
 <link href="favicon.ico" rel="shortcut icon" type="image/x-icon" />
 <link href="tomcat.css" rel="stylesheet" type="text/css" />
 </head>

 <body>
 <div id="wrapper">
 <div id="navigation" class="curved container">
 Home

```

Agora que criamos um usuário, podemos acessar a interface de gerenciamento da web novamente em um navegador da web. Mais uma vez, você pode acessar a interface correta digitando o nome de domínio ou endereço IP do FIREWALL no seu navegador: <http://server domain or IP>

## 18 Serviço WEB com Apache e PHP (finalizado)

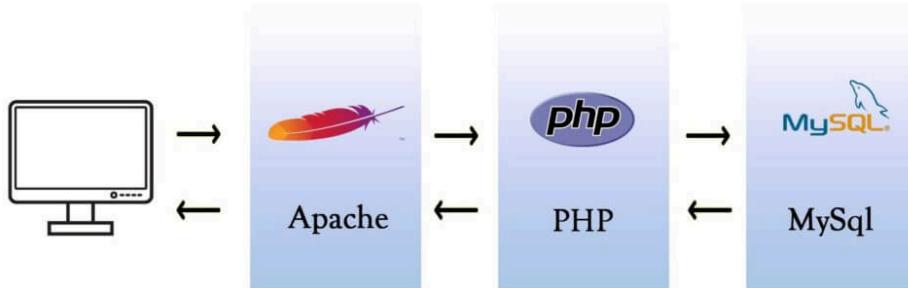
Na área de TI a hospedagem de aplicações PHP é um processo cotidiano, pois esta tecnologia associada ao Apache é barata, eficiente e possui uma ampla rede de profissionais que atuam com esta tecnologia.

O Apache HTTP Server Project é um esforço de desenvolvimento de software colaborativo que visa criar uma implementação de código-fonte robusta, de nível comercial, cheia de recursos e disponível gratuitamente de um servidor HTTP (Web), pode ser executado em UNIX, GNU/Linux e Microsoft Windows.

O projeto é administrado em conjunto por um grupo de voluntários localizados ao redor do mundo, usando a Internet e a Web para se comunicar, planejar e desenvolver o servidor e sua documentação relacionada e este projeto faz parte da Apache Software Foundation.

Além disso, centenas de usuários contribuíram com ideias, código e documentação para o projeto. Este arquivo tem a intenção de descrever brevemente a história do servidor Apache HTTP e reconhecer os muitos contribuidores.

O PHP (um acrônimo recursivo para PHP: Hypertext Preprocessor) é uma linguagem de script open source de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML. PHP é uma linguagem executada no lado do servidor exigindo processamento é extremamente simples e fácil de se aprender, extremamente poderoso e proporciona um ambiente poderoso com muitos recursos.



Nesta prática o aluno deve instalar o Apache2 e PHP (última versão disponível no PPA) em um servidor GNU/Linux Debian 12. Após instalação e configuração o aluno deve na máquina Host (real) abrir um Browser de sua preferência e digitar o endereço do site como sendo: [http://IP\\_DA\\_VM\\_NA\\_REDE](http://IP_DA_VM_NA_REDE)

Para conseguir realizar esta placa o Adaptador de rede no VirtualBox deve estar em modo “Bridge” conforme Capítulo 1 explica como configurar.

### 18.2 Instalando e configurando o Apache2

O processo de instalação é simples, primeiro o aluno deve fazer um update com apt.

```
1. sudo apt update -y
```

Para instalar o Apache2 basta usar o comando apt conforme código abaixo.

```
1. sudo apt install apache2 -y
```

Onde,

- apache2** é o recurso que se pretende instalar;
- y** informa ao apt que não será interativo e que a instalação deve ser feita;

Para testar é muito simples, dirija-se para o diretório /tmp e utilize o wget conforme código abaixo.

```
1. cd /tmp
2. wget http://127.0.0.1
```

Onde na linha 2,

<http://127.0.0.1> é o endereço da interface loopback que toda máquina possui.

No output deste comando deve ser exibido o código **200 OK** conforme figura abaixo.

```
usuario@debian:/tmp$ wget http://127.0.0.1
--2020-11-23 10:33:04-- http://127.0.0.1/
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html'

index.html 100%[=====] 10.45K
2020-11-23 10:33:04 (92.2 MB/s) - 'index.html' saved [10701/10701]
usuario@debian:/tmp$
```

Confirme se o serviço está **enabled** conforme figura abaixo.

```
usuario@debian:/tmp$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
 Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor
 Active: active (running) since Mon 2020-11-23 10:32:41 -03; 1min 2
 Docs: https://httpd.apache.org/docs/2.4/
 Main PID: 960 (apache2)
 Tasks: 55 (limit: 2354)
 Memory: 11.7M
 CGroup: /system.slice/apache2.service
 └─960 /usr/sbin/apache2 -k start
 ├─962 /usr/sbin/apache2 -k start
 ├─963 /usr/sbin/apache2 -k start
```

Este **enabled** ativo garante que o Apache sempre será carregado na inicialização do sistema operacional. Caso esteja com o valor **disabled**, então execute o comando abaixo.

```
1. sudo systemctl enable apache2.service
```

Agora instale o último PHP, para isso utilize o apt conforme comando abaixo.

```
1. sudo apt install php -y
```

Este comando irá instalar as seguintes bibliotecas no sistema:

- php-common
- php(última versão)
- php(última versão)-cli
- php(última versão)-common
- php(última versão)-fpm
- php(última versão)-json
- php(última versão)-opcache
- php(última versão)-readline

Confirme a versão instalada com o comando

```
1. php -v
```

O output do comando será:

```
usuario@debian:/tmp$ php -v
PHP 7.3.19-1~deb10u1 (cli) (built: Jul 5 2020 06:46:45) (NTS)
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.19, Copyright (c) 1998-2018 Zend Technologies
 with Zend OPcache v7.3.19-1~deb10u1, Copyright (c) 1999-2018
usuario@debian:/tmp$ _
```

Mas como o Apache foi instalado antes do PHP (corretamente) é preciso recarregar o serviço Apache para que ele faça a leitura do módulo PHP, para isso então reinicie o serviço com o comando abaixo:

```
1. sudo systemctl restart apache2.service
```

### 18.3. Instalando certificado digital

O primeiro trabalho é instalar o openssl, que é a implementação dos protocolos de segurança para HTTPS, nesta prática vamos utilizar um certificado auto assinado, ou seja, conhecido apenas pelas duas partes.

```
1. sudo apt update -y
2. sudo apt install openssl -y
```

O próximo passo é habilitar os módulos mod\_ssl e mod\_rewrite. Para isso, utilizamos o script a2enmod, que nos permite habilitar e desabilitar módulos na configuração do apache.

Use os comandos como mostrado abaixo:

```
1. sudo a2enmod ssl
2. sudo a2enmod rewrite
```

Vamos navegar até o final do arquivo apache2.conf com nano, veja comando abaixo.

```
1. sudo nano /etc/apache2/apache2.conf
```

Adicione no final:

```
1. <Directory /var/www/html>
2. AllowOverride ALL
3. </Directory>
```

Agora precisamos gerar um certificado auto assinado, mas existem várias maneiras de obter um certificado SSL gratuito. Ferramentas como certbot e SSL generators são ótimas opções.

No entanto, neste guia, criaremos um certificado autoassinado usando o utilitário OpenSSL. Vamos organizar nossos certificados em um diretório, para isso crie o seguinte diretório e navegue até ele.

```
1. sudo mkdir /etc/apache2/certs
2. cd /etc/apache2/certs
```

Execute o utilitário OpenSSL para gerar seu certificado auto assinado conforme fornecido no comando abaixo:

```
1. sudo openssl req -new -newkey rsa:4096 -x509 -sha256 -days 365 -nodes -out
apache.cert -keyout apache.key
```

Você pode fornecer qualquer informação neste processo, exceto Nome Comum. Certifique-se de fornecer um endereço IP ou nome de host.

Assim que o processo for concluído com sucesso, você deverá ter o apache.crt e o apache.key no diretório /etc/apache2/certs. Adicione o certificado editando o arquivo de configuração do site padrão do Apache, para isso edite o arquivo:

```
1. sudo nano /etc/apache2/sites-enabled/000-default.conf
```

Adicione um bloco de host virtual na porta 443, conforme mostrado:

**ATENÇÃO: SE JÁ EXISTIR O VIRTUALHOST PARA PORTA 80, COMENTE O CONTEÚDO E ESCREVA O CONTEÚDO ABAIXO.**

```
1. <VirtualHost *:80>
2. RewriteEngine On
3. RewriteCond %{HTTPS} off
4. RewriteRule (.*) https:// %{HTTP_HOST} %{REQUEST_URI}
5. </VirtualHost>
6.
7. <VirtualHost *:443>
8. ServerAdmin webmaster@localhost
9. DocumentRoot /var/www/html
10. ErrorLog ${APACHE_LOG_DIR}/error.log
11. CustomLog ${APACHE_LOG_DIR}/access.log combined
```

```

12. SSLEngine on
13. SSLCertificateFile /etc/apache2/certs/apache.cert
14. SSLCertificateKeyFile /etc/apache2/certs/apache.key
15.</VirtualHost>

```

Na maioria dos casos, você desejará redirecionar os usuários de nenhum endpoint SSL para SSL. Você faz isso adicionando uma regra de regravação nos hosts virtuais da porta 80.

**VOU REMOVER ABAIXO, POIS VOU TIRAR DO .HTACCESS**

**Crie um novo arquivo na raiz do WebSite chamado .htaccess, edite de acordo com as regras de redirecionamento abaixo.**

```

1. RewriteEngine on
2. RewriteCond %{HTTPS} !=on
3. RewriteRule ^/?(.*) https:// %{SERVER_NAME}/$1 [R=301,L]

```

Reinic peace o serviço do apache com o comando systemctl.

```
1. sudo systemctl restart apache2.service
```

## 18.4 Correções de erros

## 18.5 Testando o PHP Instalado

Para testar é simples, momentaneamente vamos disponibilizar no site uma página de teste que será apagada após os testes.

Para isso com o nano crie/edito o seguinte arquivo:

```
1. sudo nano /var/www/html/info.php
```

E edite o seguinte código PHP:

```

1. <?php
2.
3. phpinfo();
4.
5. ?>

```

Agora você deve localizar o IP que está em uso pela máquina virtual, se ela está com obtenção de IP por DHCP e está em modo Bridge, ela obteve um IP na mesma rede que sua máquina real Host. Utilize o comando ip conforme exemplo abaixo.

```
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
 link/ether 08:00:27:51:1d:06 brd ff:ff:ff:ff:ff:ff
 inet 192.168.0.6/24 brd 192.168.0.255 scope global
 valid_lft 35826sec preferred_lft 35826sec
 inet6 2804:14c:bf41:9956:a00:27ff:fe51:1d06/64 scope global
 valid_lft 86395sec preferred_lft 71995sec
 inet6 fe80::a00:27ff:fe51:1d06/64 scope link
 valid_lft forever preferred_lft forever
usuario@debian:~$ _
```

Na imagem acima, a Virtual Machine está utilizando o IP 192.168.0.6, na imagem abaixo é possível ver que a maquina real está utilizando o IP 192.168.0.9, mas lembre-se, estes valores mudam de rede para rede.

```
well@wpo:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
 link/ether 18:c0:4d:f0:21:db brd ff:ff:ff:ff:ff:ff
 inet 192.168.0.9/24 brd 192.168.0.255 scope global
 valid_lft 28157sec preferred_lft 28157sec
 inet6 2804:14c:bf41:9956::1/128 scope global dynamic
 valid_lft 2836sec preferred_lft 1036sec
 inet6 2804:14c:bf41:9956:608c:4c49:bdf:51ea/64
 valid_lft 86378sec preferred_lft 71978sec
 inet6 2804:14c:bf41:9956:c98b:2249:a4f4:70e2/64
 valid_lft 86378sec preferred_lft 71978sec
 inet6 fe80::70a8:805a:a641:521d/64 scope link
 valid_lft forever preferred_lft forever
```

Para testar a conectividade entre os elementos, utilize o comando ping, uma máquina deve pingar na outra. Após ter certeza que ambas as máquinas se comunicam, abra um Browser de sua preferência na máquina real, e digite:

[http://IP\\_DA\\_MAQUINA\\_VIRTUAL/info.php](http://IP_DA_MAQUINA_VIRTUAL/info.php)

Veja que o PHP deve exibir as informações conforme imagem abaixo.



System	Linux debian 4.19.0-12-686 #1 SMP Debian 4.19.152-1 (2)
Build Date	Jul 5 2020 06:46:45
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.3/apache2
Loaded Configuration File	/etc/php/7.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.3/apache2/conf.d
Additional .ini files parsed	/etc/php/7.3/apache2/conf.d/10-opcache.ini, /etc/php/7.3/apache2/conf.d/20-calendar.ini, /etc/php/7.3/apache2/conf.d/20-exif.ini, /etc/php/7.3/apache2/conf.d/20-gettext.ini, /etc/php/7.3/apache2/conf.d/20-json.ini, /etc/php/7.3/apache2/conf.d/20-readline.ini, /etc/php/7.3/apache2/conf.d/20-sockets.ini, /etc/php/7.3/apache2/conf.d/20-sysvsem.ini, /etc/php/7.3/apache2/conf.d/20-sysvshm.ini

## 18.6 Prática do capítulo

Nesta sessão o aluno deve realizar a prática na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

### 18.6.1 Prática bn13la21 01: Configurando o serviço WEB com Apache e PHP

Nesta prática o aluno deve ter a expertise instalar e configurar o Apache e o PHP, para isso ele deve realizar toda a configuração conforme o capítulo.

A validação deve ser feita na Virtual Machine com Apache, entre na Virtual Machine e digite o seguinte comando:

**Atenção:** o código é b de bola, n de navio, 1 de um, 3 de três, l de laranja, a de abóbora, 2 de dois e 1 de um, 😊 aied é engraçado!!!

```
1. sudo aied validar bn13la21 checkpoint01
```

## 19 Serviço DHCP

O DHCP (Dynamic Host Configuration Protocol) é um protocolo de gerenciamento de rede usado para automatizar o processo de configuração de dispositivos em redes TCP/IP, permitindo que eles usem serviços de rede como DNS, NTP e qualquer protocolo de comunicação baseado em UDP ou TCP.

Um servidor DHCP atribui dinamicamente um endereço IP e outros parâmetros de configuração de rede a cada dispositivo em uma rede para que eles possam se comunicar com outras redes IP. O DHCP é um aprimoramento de um protocolo mais antigo chamado BOOTP. Mas nem todos os dispositivos operam com o protocolo DHCP, routers e servidores possuem o endereço IP fixo em suas configurações locais, geralmente reservamos os primeiros endereços válidos para as NICs para estes dispositivos.

Outros dispositivos por questão de dinâmica de trabalho utilizam o DHCP porém seu endereço sempre é fixado no DHCP utilizando o MAC Address de suas interfaces NIC, são estes impressoras, telefones IP, switch (3 camadas) e access point.

São dispositivos que por padrão configuramos para obter IP pelo serviço DHCP:

- Dispositivos móveis, como Smartphone, PDA, IoT;
- Workstation e Laptop;

São dispositivos que NÃO utilizam o serviço **DHCP dinâmico** da rede:

- Impressoras;
- Routers;
- Access Point;
- Servidores;

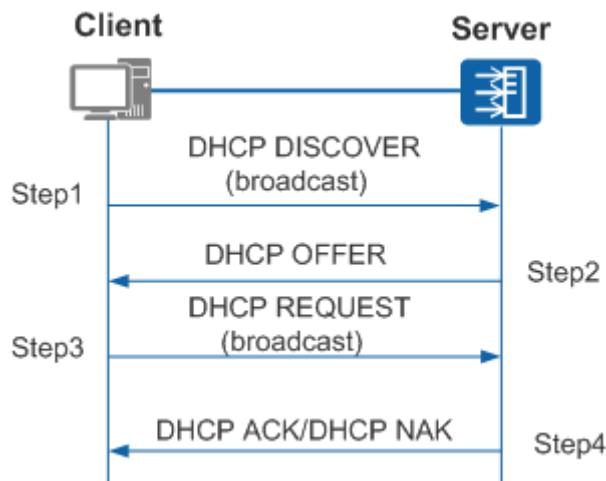
**OBS: Você pode configurar estes equipamentos por DHCP dando para eles IPs definidos e baseados no MAC Address (é interessante um LDAP).**

O fluxo básico é que um servidor DHCP entrega dados de configuração, com base na política descrita pelo administrador, a um cliente solicitante. Os parâmetros de rede comuns (às vezes chamados de "**option** DHCP") solicitados incluem:

- máscara de sub-rede;
- endereço da rede bem como broadcast;
- roteador gateway;
- servidor de nomes de domínio;
- nome de hosts;
- nome de domínio.

Como o cliente solicitante não tem endereço IP ao ingressar na rede, ele transmite a solicitação. O protocolo é, portanto, usado em um estágio muito inicial da comunicação de rede de um computador.

A troca de mensagem entre Servidor e Cliente<sup>90</sup> ocorre assim:



1. Durante o processo de inicialização, um computador cliente configurado como cliente DHCP envia um pacote de transmissão chamado DHCPDISCOVER pelo Broadcast com endereço **FF:FF:FF:FF:FF:FF**. Este pacote Discover contém o nome do computador (dele cliente) e o endereço MAC (Media Access Control) do cliente, para que os servidores DHCP possam responder. Basicamente, o pacote Discover diz: "**Estou procurando um servidor DHCP que possa conceder um endereço IP :)**".
2. Os servidores DHCP na rede respondem à transmissão com um DHCPOFFER. Em essência, o DHCPOFFER diz: "**Sou um servidor DHCP e tenho uma concessão para você ;)**". Se vários servidores DHCP responderem à solicitação, o cliente aceitará a primeira oferta que receber.
3. O cliente responde através de uma mensagem no canal de Broadcast chamada DHCPREQUEST. Essa mensagem basicamente diz: "**Aceito sua oferta de locação e gostaria de um endereço IP**". Se outros servidores DHCP fizerem ofertas, eles também verão que suas ofertas de concessão não foram aceitas pela mensagem no Broadcast e, por isso, cancelam suas ofertas. **Eles não devem gostar de ser desprezados por um computador cliente :(**
4. O servidor DHCP cuja oferta foi aceita responde com uma mensagem DHCPACK, que reconhece a aceitação da concessão e contém a concessão do endereço IP do cliente, bem como outras informações de endereçamento IP que você configura o servidor para fornecer. O cliente agora é um cliente TCP/IP e pode participar da rede.

Na figura abaixo podemos ver o LOG de um servidor DHCP em um Linux, repare que a cada 4 minutos (média) ocorre o processo de manutenção de concessão de endereço IP.

<sup>90</sup>

[https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr\\_dhcp/configuration/15-sy/dhcp-15-sy-book/config-dhcp-client.pdf](https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_dhcp/configuration/15-sy/dhcp-15-sy-book/config-dhcp-client.pdf)

```
Jun 16 11:49:01 debian dhcpcd[466]: DHCPACK on 192.168.200.10 to 08:00:27:cf:2f:d4 (debian) via
Jun 16 11:53:04 debian dhcpcd[466]: DHCPREQUEST for 192.168.200.10 from 08:00:27:cf:2f:d4 (debi
Jun 16 11:53:04 debian dhcpcd[466]: DHCPACK on 192.168.200.10 to 08:00:27:cf:2f:d4 (debian) via
Jun 16 11:57:01 debian dhcpcd[466]: DHCPREQUEST for 192.168.200.10 from 08:00:27:cf:2f:d4 (debi
Jun 16 11:57:01 debian dhcpcd[466]: DHCPACK on 192.168.200.10 to 08:00:27:cf:2f:d4 (debian) via
Jun 16 12:01:20 debian dhcpcd[466]: DHCPREQUEST for 192.168.200.10 from 08:00:27:cf:2f:d4 (debi
Jun 16 12:01:20 debian dhcpcd[466]: DHCPACK on 192.168.200.10 to 08:00:27:cf:2f:d4 (debian) via
Jun 16 12:05:23 debian dhcpcd[466]: DHCPREQUEST for 192.168.200.10 from 08:00:27:cf:2f:d4 (debi
Jun 16 12:05:24 debian dhcpcd[466]: DHCPACK on 192.168.200.10 to 08:00:27:cf:2f:d4 (debian) via
Jun 16 12:08:09 debian dhcpcd[466]: DHCPRELEASE of 192.168.200.10 from 08:00:27:cf:2f:d4 (debi
```

O serviço DHCP traz três valores para a organização:

- 1) As tarefas de operação são reduzidas: o administrador da rede não precisa mais configurar manualmente cada cliente antes de poder usar a rede
- 2) O plano de endereçamento IP é otimizado: os endereços que não estão mais sendo usados são liberados e disponibilizado para novos clientes que se conectam
- 3) A mobilidade do usuário é facilmente gerenciada: o administrador não precisa configurar manualmente um cliente quando seu ponto de acesso à rede é alterado.

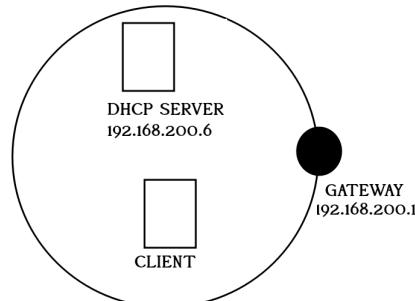
As informações de endereço IP atribuídas pelo DHCP são válidas apenas por um período limitado de tempo e são conhecidas como concessão de DHCP. O período de validade é chamado de tempo de concessão do DHCP. Quando a concessão expira, o cliente não pode mais usar o endereço IP e precisa interromper toda a comunicação com a rede, a menos que ele solicite estender a “locação” da concessão por meio do ciclo de renovação da concessão DHCP.

Para evitar que o servidor DHCP não esteja disponível no final do período de concessão, os clientes geralmente começam a renovar sua concessão no meio do período. Esse processo de renovação garante uma alocação robusta de endereço IP para os dispositivos.

É fundamental que o aluno faça a prática completa do capítulo [Gateway Debian GNU/Linux](#) para então iniciar esta prática.

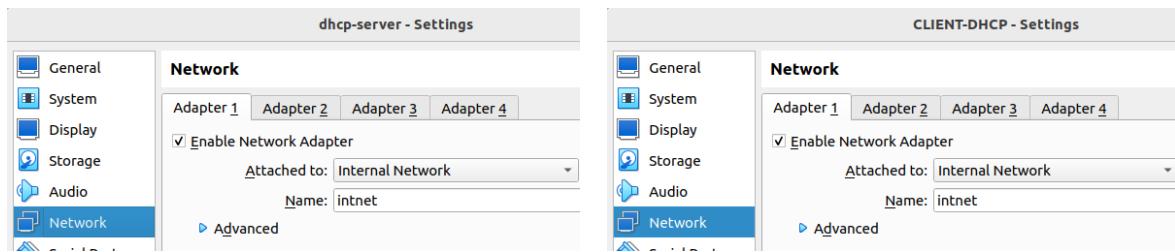
## 19.1 No VirtualBox

Para prática no VirtualBox é preciso três máquinas virtuais com Debian 12 Terminal (sem interface gráfica) configurado segundo o capítulo [Gateway Debian GNU/Linux](#). O serviço DHCP deverá ficar dentro da rede interna da organização.



Na aula de Gateway o endereço da interface de rede da máquina cliente foi definida de forma estática, nesta prática o endereço da interface de rede da máquina cliente será definida pelo DHCP instalado no Gateway e apontado para a rede interna da organização.

Importe duas novas máquinas virtuais, e conecte estas duas máquinas virtuais na rede interna, para isso em configuração selecione o Adapter 1 e escolha Internal Network.



Lembre-se que o Gateway possui duas interfaces, uma das interfaces está ligada também nesta rede interna. Não vou descrever o Gateway pois a prática já foi realizada e o aluno só precisa iniciar o Gateway para esta prática.

## 19.2 Instalando e Configurando o DHCP Server

Por padrão a equipe de infra está localizada na rede interna da organização, se algum elemento está na parte externa então utiliza-se uma VPN (tópico que será demonstrado posteriormente) para ingressar de forma segura na rede interna da organização para fazer as devidas manutenções nos serviços, e DHCP é um destes serviços.

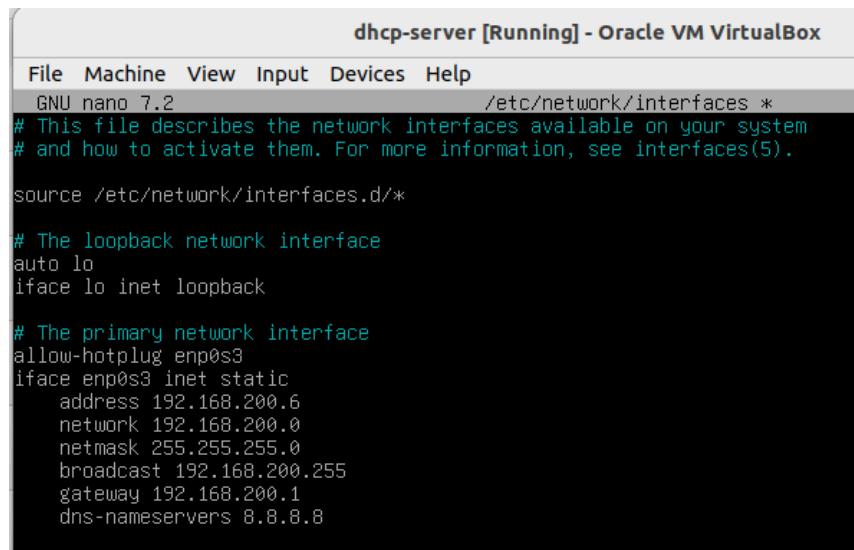
Então a interface de rede do Gateway configurada na rede interna da organização deverá ter um IP fixado na configuração do próprio Gateway em `/etc/network/interfaces`. Já a interface do Gateway apontada para a rede externa mais abrangente é definida pelo operador Internet.

### 19.2.1 Configurando o IP do servidor DHCP

O endereço IP da interface de rede interna do Gateway será 192.168.200.1 e a interface de rede do DHCP Server será 192.168.200.6, para realizar esta configuração por arquivos de configuração digite no terminal o comando abaixo:

```
1. sudo nano /etc/network/interfaces
```

No caso do GNU/Linux o nome do adaptador de rede é "enp0s3" conforme figura abaixo, vamos utilizar como endereço para a interface enp0s3 o IP 192.168.200.6 conforme configuração. Para saber qual o nome das interfaces de rede no Debian utilize o comando `ip` conforme descrito no capítulo [Configurando Interface de rede no Linux](#).



```

GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.6
 network 192.168.200.0
 netmask 255.255.255.0
 broadcast 192.168.200.255
 gateway 192.168.200.1
 dns-nameservers 8.8.8.8

```

Reinicie o serviço de rede, para isso pode usar uma das 3 opções abaixo:

- 1) por comando: `sudo systemctl restart networking.service`
- 2) executando o script: `sudo /etc/init.d/networking restart`
- 3) reiniciando o servidor (usar este método somente se tem certeza que nenhum serviço está sendo prestado pela máquina)

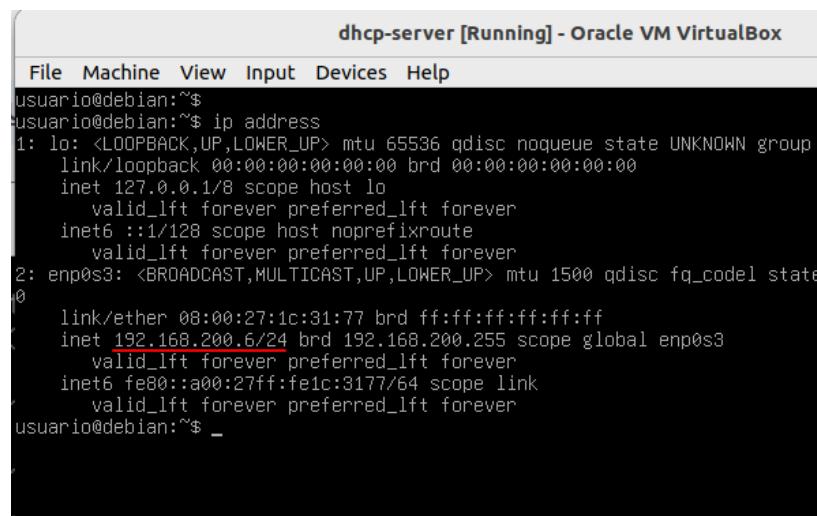


Aula sobre Init e Systemd no Linux, [acesse este link](#).

Teste a configuração utilizando o comando `ip` conforme exemplo abaixo:

1. `ip address`

Veja o Output:



```

usuario@debian:~$ ip address
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
 link/ether 00:00:27:1c:31:77 brd ff:ff:ff:ff:ff:ff
 inet 192.168.200.6/24 brd 192.168.200.255 scope global enp0s3
 valid_lft forever preferred_lft forever
 inet6 fe80::a00:27ff:fe1c:3177/64 scope link
 valid_lft forever preferred_lft forever
usuario@debian:~$

```

### 19.2.2 Instalando o servidor DHCP

Antes de prosseguir com a instalação de um servidor DHCP, primeiro atualize os pacotes executando o update e só depois execute a instalação, caso tenha dúvida de como se faz o uso do apt, acesse o capítulo [Instalação de programas e pacotes no Linux](#):

1. sudo apt update -y
2. sudo apt install isc-dhcp-server -y

```
usuario@debian:~$ sudo apt install isc-dhcp-server -y
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:
 #1) Respect the privacy of others.
 #2) Think before you type.
 #3) With great power comes great responsibility.

[sudo] password for usuario:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
 libirs-export161 libisccfg-export163 policycoreutils selinux-utils
Suggested packages:
 policykit-1 isc-dhcp-server-ldap
The following NEW packages will be installed:
 isc-dhcp-server libirs-export161 libisccfg-export163 policycoreutils selinux-utils
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,638 kB of archives.
After this operation, 6,668 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main i386 isc-dhcp-server i386 4.4.1-2 [569 kB]
Get:2 http://deb.debian.org/debian buster/main i386 selinux-utils i386 2.8-1+b1 [102 kB]
Get:3 http://deb.debian.org/debian buster/main i386 policycoreutils i386 2.8-1 [466 kB]
Get:4 http://security.debian.org/debian-security buster/updates/main i386 libisccfg-export163 i386 1
:9.11.5.P4+dfsg-5.1+deb10u1 [264 kB]
Get:5 http://security.debian.org/debian-security buster/updates/main i386 libirs-export161 i386 1:9.
11.5.P4+dfsg-5.1+deb10u1 [237 kB]
Fetched 1,638 kB in 0s (3,976 kB/s)
Preconfiguring packages ...
Selecting previously unselected package libisccfg-export163.
(Reading database ... 27643 files and directories currently installed.)
Preparing to unpack .../libisccfg-export163_1%3a9.11.5.P4+dfsg-5.1+deb10u1_i386.deb ...
Unpacking libisccfg-export163 (1:9.11.5.P4+dfsg-5.1+deb10u1) ...
Selecting previously unselected package libirs-export161.
Preparing to unpack .../libirs-export161_1%3a9.11.5.P4+dfsg-5.1+deb10u1_i386.deb ...
```

Se uma mensagem de erro ocorrer como a exibida abaixo, não se desespere, a falta de configuração gera este erro e este processo será realizado mais adiante.

```
● isc-dhcp-server.service - LSB: DHCP server
 Loaded: loaded (/etc/init.d/isc-dhcp-server; generated)
 Active: failed (Result: exit-code) since Tue 2020-06-16 10:13:33 -03; 15ms ago
 Docs: man:systemd-sysv-generator(8)
 Process: 878 ExecStart=/etc/init.d/isc-dhcp-server start (code=exited, status=1/FAILURE)

Jun 16 10:13:31 debian dhcpcd[890]: bugs on either our web page at www.isc.org or in the README file
Jun 16 10:13:31 debian dhcpcd[890]: before submitting a bug. These pages explain the proper
Jun 16 10:13:31 debian dhcpcd[890]: process and the information we find helpful for debugging.
Jun 16 10:13:31 debian dhcpcd[890]:
Jun 16 10:13:31 debian dhcpcd[890]: exiting.
Jun 16 10:13:33 debian isc-dhcp-server[878]: Starting ISC DHCPv4 server: dhcpdcheck syslog for diagn
ostics. ... failed!
Jun 16 10:13:33 debian isc-dhcp-server[878]: failed!
Jun 16 10:13:33 debian systemd[1]: isc-dhcp-server.service: Control process exited, code=exited, sta
tus=1/FAILURE
Jun 16 10:13:33 debian systemd[1]: isc-dhcp-server.service: Failed with result 'exit-code'.
Jun 16 10:13:33 debian systemd[1]: Failed to start LSB: DHCP server.
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10) ...
Processing triggers for systemd (241-7~deb10u4) ...

```

O erro é a falta de configuração, mas aproveitando que estamos aqui, fica como conhecimento que o erro será lançado em **/var/log/syslog** conforme imagem abaixo. Sempre que tiver dificuldade, acompanhe este arquivo.

```
Jun 16 10:41:52 debian dhcpcd[1282]: Configuration file errors encountered -- exiting
Jun 16 10:41:52 debian systemd[1]: isc-dhcp-server.service: Failed with result 'exit-code'.
Jun 16 10:41:52 debian dhcpcd[1282]:
Jun 16 10:41:52 debian systemd[1]: Failed to start LSB: DHCP server.
Jun 16 10:41:52 debian dhcpcd[1282]: If you think you have received this message due to a bug rather
Jun 16 10:41:52 debian dhcpcd[1282]: than a configuration issue please read the section on submitting
Jun 16 10:41:52 debian dhcpcd[1282]: bugs on either our web page at www.isc.org or in the README file
Jun 16 10:41:52 debian dhcpcd[1282]: before submitting a bug. These pages explain the proper
Jun 16 10:41:52 debian dhcpcd[1282]: process and the information we find helpful for debugging.
Jun 16 10:41:52 debian dhcpcd[1282]:
Jun 16 10:41:52 debian dhcpcd[1282]: exiting.
```

### 19.2.3 Arquivos de Configuração do Serviço DHCP

O primeiro arquivo que se deve configurar é o arquivo `dhcpcd.conf`, neste arquivo será definido o plano de endereçamento da rede interna bem como parâmetros que permite a customização do serviço para a organização. Para editar utilize o editor de texto conforme exemplo abaixo.

1. `sudo nano /etc/dhcp/dhcpcd.conf`

```
usuario@debian:~$
usuario@debian:~$ sudo nano /etc/dhcp/dhcpcd.conf
```

Existem as etapas básicas necessárias para definir as configurações de propriedades globais para configurar um servidor DHCP. Para especificar o tempo de concessão padrão e máximo, localize os parâmetros `default-lease-time` e `max-lease-time` no arquivo de configuração e altere seus valores.

Em resumo, o tempo de concessão do DHCP é a quantidade de tempo em minutos/segundos que um dispositivo de rede pode usar um endereço IP específico em uma rede. O endereço IP é reservado para esse dispositivo até a reserva expirar.

O tempo de concessão indica quanto tempo um dispositivo tem permissão para usar esse endereço IP. Agora, isso é importante porque a quantidade de endereços IP disponíveis é limitada em uma rede. O tempo máximo de concessão (max-lease-time) é o tempo máximo de concessão que você terá se vocês pedir um tempo, digamos 10000 segundos e o tempo máximo de concessão é (digamos 6000 segundos, você terá um tempo de concessão de 6000 segundos).

O tempo de concessão padrão (default-lease-time) é o tempo de concessão que você terá se não o fizer solicitar de tempo de locação específico. Geralmente, você define a concessão padrão tempo para o tempo de concessão que você gostaria que todos usassem, e o máximo tempo de concessão para o tempo de concessão mais alto que você está preparado para aceitar. Estão perfeitamente aceitável tê-los configurados da mesma maneira.

Para tornar o servidor DHCP o servidor DHCP oficial dos clientes, remova o comentário da seguinte linha no arquivo de configuração (removendo o caractere #): autoritativo;

```
dhcp-server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/dhcp/dhcpd.conf *
option definitions common to all supported networks...
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

The ddns-updates-style parameter controls whether or not the server will
attempt to do a DNS update when a lease is confirmed. We default to the
behavior of the version 2 packages ('none', since DHCP v2 didn't
have support for DDNS.)
ddns-update-style none;

If this DHCP server is the official DHCP server for the local
network, the authoritative directive should be uncommented.
authoritative;

Use this to send dhcp log messages to a different log file (you also
have to hack syslog.conf to complete the redirection).
log-facility local7;

No service will be given on this subnet, but declaring it helps the

dhcp-server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/dhcp/dhcpd.conf *
This declaration allows BOOTP clients to get dynamic addresses,
which we don't really recommend.

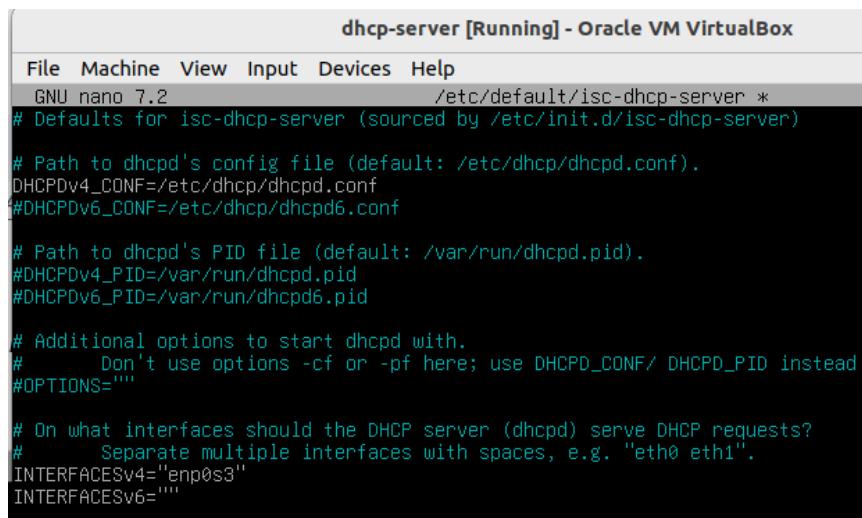
subnet 10.254.239.32 netmask 255.255.255.224 {
 range dynamic-bootp 10.254.239.40 10.254.239.60;
 option broadcast-address 10.254.239.31;
 option routers rtr-239-32-1.example.org;
}
subnet 192.168.200.0 netmask 255.255.255.0 {
 range 192.168.200.60 192.168.200.254;
 option routers 192.168.200.1;
 option domain-name-servers 8.8.8.8;
}

A slightly different configuration for an internal subnet.
#subnet 10.5.5.0 netmask 255.255.255.224 {
```

Veja acima que `autoritativo` foi removido o comentário e ativo, e o log será registrado a partir deste momento em outro arquivo de log (não mais no syslog) pois foi removido o comentário da linha `log-facility local17`.

Para estudos deste livro, vou reservar do endereço 192.168.200.1 até o endereço 192.168.200.59 para práticas com IP estático e 192.168.200.60 até 192.168.200.254 para IP dinâmico. O atributo `range` recebe o primeiro endereço e o último endereço de um range de endereços.

Mas este arquivo só será utilizado se for informado em outro arquivo, vamos então editar o `/etc/default/isc-dhcp-server`.



```

dhcp-server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/default/isc-dhcp-server *
Defaults for isc-dhcp-server (sourced by /etc/init.d/isc-dhcp-server)

Path to dhcpcd's config file (default: /etc/dhcp/dhcpd.conf).
DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
#DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf

Path to dhcpcd's PID file (default: /var/run/dhcpcd.pid).
#DHCPDv4_PID=/var/run/dhcpcd.pid
#DHCPDv6_PID=/var/run/dhcpcd6.pid

Additional options to start dhcpcd with.
Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

On what interfaces should the DHCP server (dhcpcd) serve DHCP requests?
Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="enp0s3"
INTERFACESv6=""

```

Em `INTERFACEv4` vamos colocar o rótulo definido pelo Linux para a nossa interface interna da organização, conforme visto no comando `ip address`.

Também informamos o arquivo de configuração de subredes (configurado anteriormente), veja que `DHCPDv4_CONF` está apontando para o arquivo `/etc/dhcp/dhcpd.conf`. Inicie o serviço usando o seguinte comando:

1. `sudo systemctl start isc-dhcp-server.service`

**ATENÇÃO: se já estiver iniciado (rodando) você deve usar `stop` e só de depois o `start`.**

Para saber se está tudo OK, basta executar o comando:

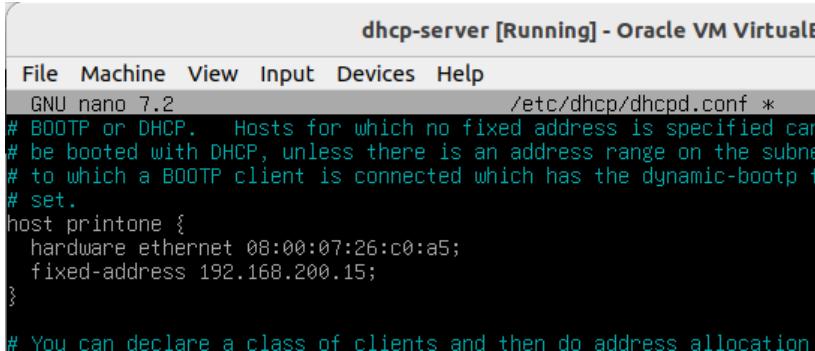
1. `sudo systemctl status isc-dhcp-server.service`

```
usuario@debian:~$ sudo systemctl status isc-dhcp-server.service
● isc-dhcp-server.service - LSB: DHCP server
 Loaded: loaded (/etc/init.d/isc-dhcp-server; generated)
 Active: active (running) since Tue 2020-06-16 11:15:15 -03; 1min 21s ago
 Docs: man:systemd-sysv-generator(8)
 Process: 454 ExecStart=/etc/init.d/isc-dhcp-server start (code=exited, status=0/SUCCESS)
 Tasks: 1 (limit: 2354)
 Memory: 2.7M
 CGroup: /system.slice/isc-dhcp-server.service
 └─466 /usr/sbin/dhcpd -4 -q -cf /etc/dhcp/dhcpd.conf enp0s8

Jun 16 11:15:13 debian dhcpd[463]: All rights reserved.
Jun 16 11:15:13 debian dhcpd[463]: For info, please visit https://www.isc.org/software/dhcp/
Jun 16 11:15:13 debian dhcpd[466]: Internet Systems Consortium DHCP Server 4.4.1
Jun 16 11:15:13 debian dhcpd[466]: Copyright 2004-2018 Internet Systems Consortium.
Jun 16 11:15:13 debian dhcpd[466]: All rights reserved.
Jun 16 11:15:13 debian dhcpd[466]: For info, please visit https://www.isc.org/software/dhcp/
Jun 16 11:15:13 debian dhcpd[466]: Wrote 0 leases to leases file.
Jun 16 11:15:13 debian dhcpd[466]: Server starting service.
Jun 16 11:15:15 debian isc-dhcp-server[454]: Starting ISC DHCPv4 server: dhcpd.
Jun 16 11:15:15 debian systemd[1]: Started LSB: DHCP server.
usuario@debian:~$ _
```

#### 19.2.4 Fixando um endereço para um Host

Alguns equipamentos precisam ter um endereço para manutenção ou para configuração, o clássico exemplo é o de impressoras, no qual os computadores precisam mapear por endereços. Porém a configuração na impressora física é um obstáculo, então costuma-se reservar no DHCP o endereço para o dispositivo usando o MAC Address, conforme exemplo abaixo.



```
dhcp-server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/dhcp/dhcpd.conf *
BOOTP or DHCP. Hosts for which no fixed address is specified can
be booted with DHCP, unless there is an address range on the subnet
to which a BOOTP client is connected which has the dynamic-bootp f
set.
host printone {
 hardware ethernet 08:00:07:26:c0:a5;
 fixed-address 192.168.200.15;
}
You can declare a class of clients and then do address allocation.
```

## 19.3 Configurando Debian Cliente

Em um segundo Linux, um Debian terminal vamos garantir que a interface está configurada para buscar na rede um serviço DHCP, para isso vamos abrir para edição o arquivo `/etc/network/interfaces` conforme figura abaixo.

```
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp
```

Se a configuração `iface` estiver como `inet dhcp` está tudo OK, senão, configure conforme figura acima. Digite o comando `ip address` no cliente para saber se o mesmo obteve IP por meio do DHCP.

CLIENT-DHCP [Running] - Oracle VM VirtualBox

```
File Machine View Input Devices Help
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
 link/ether 08:00:27:ic:81:77 brd ff:ff:ff:ff:ff:ff
 inet 192.168.200.60/24 brd 192.168.200.255 scope global dynamic enp0s3
 valid_lft 591sec preferred_lft 591sec
 inet6 fe80::a00:27ff:fe1c:8177/64 scope link dadfailed tentative
 valid_lft forever preferred_lft forever
usuario@debian:~$
```

Para saber se o `option routers` foi utilizado para configurar automaticamente o Gateway do cliente, digite o comando `ip route` conforme imagem abaixo.

CLIENT-DHCP [Running] - Oracle VM VirtualBox

```
File Machine View Input Devices Help
usuario@debian:~$ ip route
default via 192.168.200.1 dev enp0s3
192.168.200.0/24 dev enp0s3 proto kernel scope link src 192.168.200.60
usuario@debian:~$
```

Veja que a primeira linha `default via 192.168.200.1 dev enp0s3` indica que o Gateway foi configurado com sucesso pelo serviço de Network. Para testar, utilize o comando Ping (supondo que a prática de Gateway foi realizada neste cenário).

1. ping 192.168.200.1
2. ping 8.8.8.8
3. ping google.com.br

```
root@debian:~# ping 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data.
64 bytes from 192.168.200.1: icmp_seq=1 ttl=64 time=0.424 ms
64 bytes from 192.168.200.1: icmp_seq=2 ttl=64 time=0.596 ms
64 bytes from 192.168.200.1: icmp_seq=3 ttl=64 time=0.646 ms
64 bytes from 192.168.200.1: icmp_seq=4 ttl=64 time=0.577 ms
^C
```

## 19.3 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

### 19.5.1 Prática 57t90021 01: Validando o serviço DHCP

Nesta prática o aluno deve ter a expertise para configurar o servidor DHCP em um ambiente com Gateway já configurado, para isso deve realizar a prática de Gateway antes (**ATENÇÃO, UTILIZE ESPAÇOS, NÃO UTILIZE TAB**).

1. O max-lease-time deve ser 6000;
2. default-lease-time deve ser 400;
3. Configure uma subnet no endereço 192.168.200.0/24 com range 192.168.200.100 até 192.168.200.250;
4. O router deve ser o endereço 192.168.200.1;
5. O DNS Server deve ser o 8.8.8.8;
6. Configure o arquivo isc-dhcp-server para apontar para este dhcpd.conf em IPV4;
7. Informe a interface de rede adequada (rede interna da organização);
8. Inicie/Reinic peace;

Execute o comando aied conforme comando listagem abaixo.

1. sudo aied validar

Caso tenha certeza que deseja continuar, pressione s de sim.

O output esperado será similar ao exibido abaixo, pois pode existir alterações no layout de saída do comando ao longo do tempo.

### 19.5.2 Prática 57t90021 02: Cliente DHCP

Nesta prática o aluno deve ter a expertise para configurar um cliente DHCP e se conectar ao servidor DHCP pela rede interna da organização.

1. Em uma máquina cliente, conecte a única interface de rede na rede interna (a mesma do servidor), caso tenha dúvidas a aula de Gateway explica este procedimento;

2. No arquivo /etc/network/interfaces informe que a interface de rede deverá obter IP por dhcp;
3. Reinicie o cliente;
4. Quando voltar o sistema, envie ICMP para 8.8.8.8

Execute o comando aied conforme comando listagem abaixo.

1. sudo aied validar

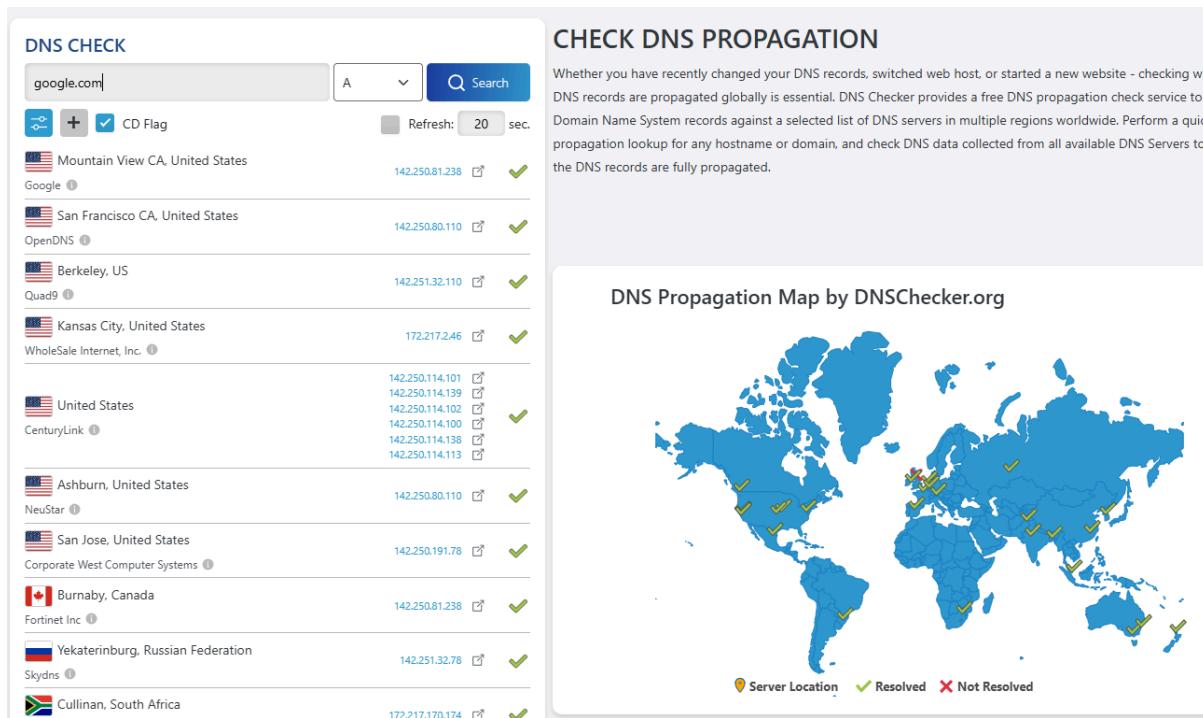
Caso tenha certeza que deseja continuar, pressione s de sim.

O output esperado será similar ao exibido abaixo, pois pode existir alterações no layout de saída do comando ao longo do tempo.

## 20 Serviço DNS com Bind9

O protocolo Domain Name System (DNS) é um protocolo que permite a tradução de nomes por endereços IPv4 e IPv6. Isso garante que pessoas comuns naveguem na Internet usando nomes, tal como **google.com**, **aied.com.br**, **universidadegratuita.com.br**, etc. O DNS é como a lista telefônica da Internet, e ele simplifica o processo de busca por sites específicos por meio de aplicações clientes de alto nível.

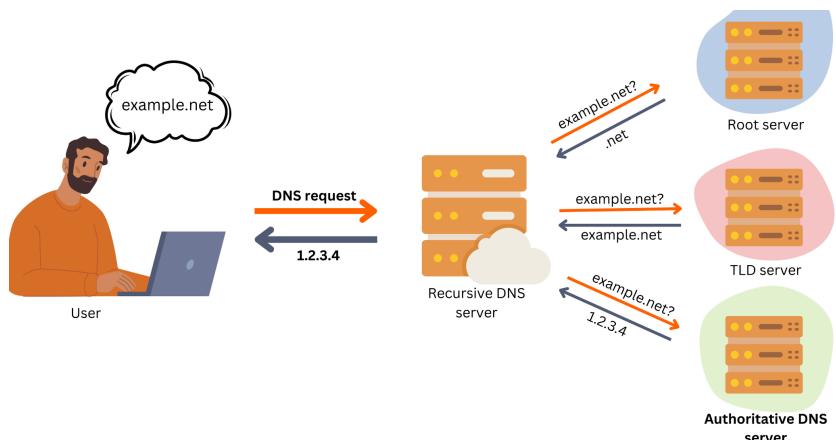
Outra grande vantagem do serviço de DNS é que ele pode descentralizar mais a Internet, como exemplo vou usar **google.com**, uma empresa gigante que para sua existência precisa descentralizar seus data centers pelo mundo, sendo assim é esperado que em diferentes lugares do mundo a tradução do domínio **google.com** retornará IPs diferentes. Para provar isso é fácil, vamos recorrer ao DNS Checker, conforme figura abaixo.



Veja que em San Francisco o endereço IP que foi retornado pela tradução foi **142.250.80.110** enquanto para Kansas foi **172.217.2.46**.

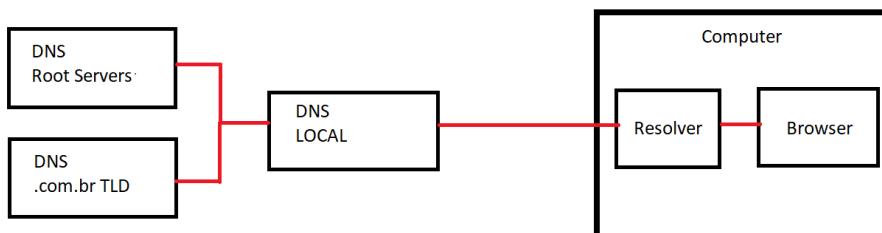
Quando um cliente DNS faz uma solicitação DNS usando um nome de domínio ou host, como **aied.com.br**, uma série de funções conecta essa solicitação ao endereço IP correspondente. Essas funções fornecem autenticação de endereços IP (IPv4 e IPv6) e tornam o uso da Internet mais acessível, traduzindo nomes de domínio personalizáveis em endereços numéricos complexos.

Os registros de recursos DNS são armazenados em servidores DNS autoritativos, também conhecidos como servidores de nomes autoritativos. Eles contêm informações relacionadas ao domínio, incluindo por quanto tempo um servidor manterá os registros DNS em cache, um período conhecido como tempo de vida (TTL).

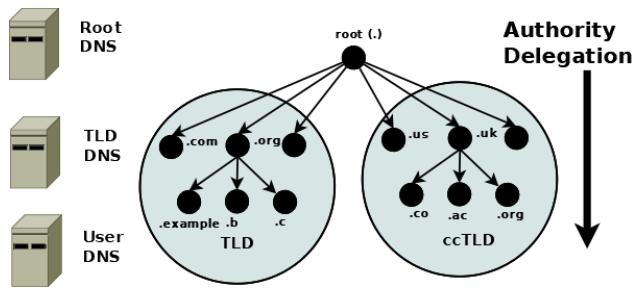


Operações de DNS, por exemplo, consultas e operações de manutenção de zona, por padrão usam porta 53. Por motivos de desempenho, as consultas utilizam o protocolo UDP com um limite de bloco de 512 bytes. Na figura acima temos uma pesquisa recursiva clássica de DNS com UDP, primeiro a requisição é enviado ao servidor primário de DNS, utiliza-se o IP que está configurado com DNS primário na máquina do cliente. Logo em seguida a mensagem é enviada para os Root Servers, alguns grandes serviços de tradução mantidos por órgãos, que informa quais IPs dos servidores locais, no caso do Brasil é o Registro.br e em seguida é consultado o servidor local da empresa ou prestadora de serviço de DNS/hospedagem.

No computador do usuário encontra-se a aplicação de alto nível como já dito, e nesta aplicação o cliente já digitou um domínio, tal como **aied.com.br**, como este domínio não é acessível por nome, e sim por um IP, então o Browser (exemplo de aplicação de alto nível) consulta um serviço interno de resolução de nomes, no caso do GNU/Linux o Resolver. O Resolver por sua vez pode responder diretamente (se já foi consultado anteriormente este domínio) ou consultar um serviço local, que pode ser local e privado, na internet e privado ou um público.



Esse servidor DNS se não sabe, então consulta os servidores DNS responsáveis pelo domínio .com.br e os servidores raiz (que são 13 servidores). No caminho inverso os servidores realizam um processo de aprendizagem para evitar fazer todo esse caminho sempre, e afinal, se um usuário acessou o domínio **aied.com.br** uma vez, pode acessar outra vez em um futuro próximo. A imagem acima pode ser melhor visualizada no esquema abaixo.



## 20.1 Programa Bind9

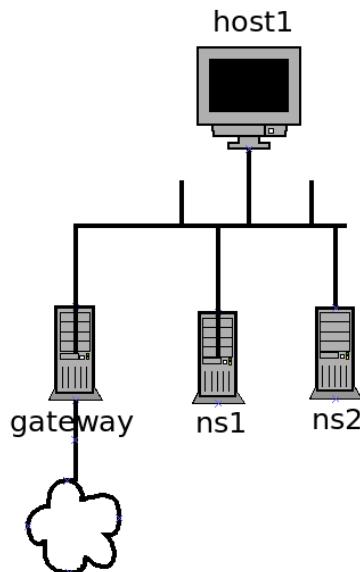
Uma parte importante do gerenciamento da configuração e da infraestrutura do servidor inclui manter uma maneira fácil de pesquisar interfaces de rede e endereços IP por nome, configurando um Domain Name Server (DNS). O uso de nomes de domínio qualificados (FQDNs) se faz fundamental em uma organização, em vez de endereços IP, para especificar endereços de rede facilita a configuração de serviços e aplicativos e aumenta a capacidade de manutenção dos arquivos de configuração.

Configurar seu próprio DNS Server para sua rede privada é uma ótima maneira de melhorar o gerenciamento de seus servidores. Com DNS local também é possível reduzir o número de requisições para a rede mundial, pois a tradução de nomes de domínios passa a ser local, ou seja, dentro de nossa rede. O DNS também pode ser configurado para que seja realizada uma consulta inversa, ou seja, um máquina na rede mundial pode conhecer uma máquina interna da organização, cito como exemplo o uso de www interno na organização.

Neste capítulo será configurado o DNS interno, usando o software de servidor de nomes BIND (Bind9) no Debian 12, que pode ser usado por seus servidores para resolver nomes de hosts privados, endereços IP privados e domínio externo. Isso fornece uma maneira central de gerenciar seus nomes de hosts internos e endereços IP privados e reduzir o consumo de serviço de rede mundial.

Para realizar esta prática, você precisará da infraestrutura a seguir:

- Um novo servidor Debian para servir como servidor DNS primário, ns1
- Um segundo servidor Debian para servir como servidor DNS secundário, ns2
- Um servidor gateway utilizado para prover uma rede interna com acesso a internet, gateway
- Uma máquina com qualquer OS, vamos usar por padrão o Debian, será usado para testar, host1



Para os fins deste livro, assumimos o seguinte:

- Temos dois servidores que serão designados como nossos servidores de nomes DNS. Iremos nos referir a eles como **ns1** e **ns2**.
- Temos uma VM cliente que usarão a infraestrutura DNS que criamos para validar. Vamos chamá-la de **host1**.
- Todos esses VMs existem na mesma sub-rede. Assumiremos que esta rede é a **faculdade.aied.com.br** (fictício, AIED é apenas um site qualquer).
- Todos esses servidores têm rede privada habilitada (e estão na 192.168.200.0/24).
- Todos os servidores estão conectados a um projeto executado em **aied.com.br**.
- Como nosso sistema DNS é totalmente interno e privado, você não precisa adquirir um nome de domínio.

Para configurar nome de máquina e endereço IP, utilize a tabela abaixo.

VM	Função	Nome	IP
ns1	Servidor DNS Primário (master)	<b>ns1.faculdade.aied.com.br</b>	192.168.200.10
ns2	Servidor DNS Secundário (slave)	<b>ns2.faculdade.aied.com.br</b>	192.168.200.11
host1 <sup>91</sup>	Host para teste	<b>host1.faculdade.aied.com.br</b>	192.168.200.2
gateway	Gateway da rede	<b>gateway.faculdade.aied.com.br</b>	192.168.200.1

## 20.2 Configurando o nome das máquinas

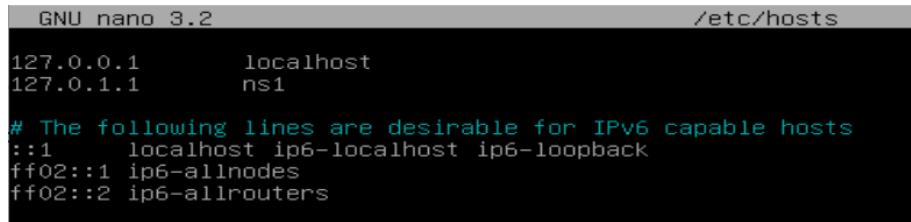
Na máquina **ns1** edite o arquivo **/etc/hostname** informando **ns1** conforme figura abaixo.

<sup>91</sup> A única máquina que irá consultar o NS1 e NS2 nesta prática



```
GNU nano 3.2 /etc/hostname
ns1
```

Também edite o arquivo de **/etc/hosts** informando o novo nome da máquina.



```
GNU nano 3.2 /etc/hosts
127.0.0.1 localhost
127.0.1.1 ns1

The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Isso é necessário pois inúmeros serviços resolvem o nome da máquina local pelo nome ao invés de localhost. Configure da mesma forma as máquinas: **host1**, **ns2** e **gateway**.

## 20.3 Instalando recursos em ns1 e ns2

Em ambos os servidores DNS, **ns1**, **ns2**, atualize o apt cache do pacote com um update, e depois instale o serviço bind9, conforme listagem abaixo, isso deve ser feito tanto em **ns1** quanto em **ns2**.

1. sudo apt update -y
2. sudo apt install bind9 bind9utils bind9-doc -y

## 20.4 Configurando Vincular ao Modo IPv4 em ns1 e ns2

Antes de continuar, vamos definir o BIND para o modo IPv4, já que nossa rede privada usa exclusivamente IPv4. Em ambos os servidores **ns1** e **ns2**, edite o arquivo bind9 de configurações padrão digitando<sup>92</sup>:

1. sudo nano /etc/default/bind9

Veja como deve digitar no arquivo.



```
GNU nano 3.2 /etc/default/bind9
#
run resolvconf?
RESOLVCONF=no
#
startup options for the server
OPTIONS="-u bind -4"
```

Salve e feche o arquivo quando terminar. Reinicie o BIND para implementar as mudanças:

1. sudo systemctl restart bind9

<sup>92</sup> No Debian 12 passou a não existir, mas crie

## 20.5 Configurando o servidor DNS primário em ns1

A configuração do BIND consiste em vários arquivos, que são incluídos no arquivo de configuração principal **named.conf**. Esses nomes de arquivo começam com named porque esse é o nome do processo que o BIND executa (abbreviação de “daemon de nome de domínio”). Começaremos configurando o arquivo de opções.

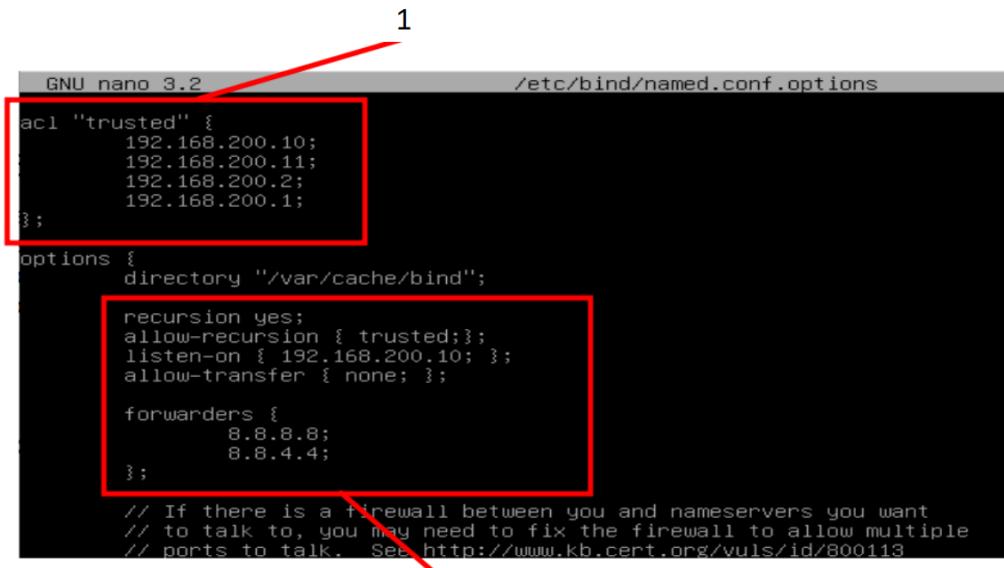
### 20.5.1 Configurando o Arquivo de Opções

No **ns1** , abra o arquivo **named.conf.options** para edição:

```
1. sudo nano /etc/bind/named.conf.options
```

Digite a seguinte informação no documento:

1



```
GNU nano 3.2 /etc/bind/named.conf.options

acl "trusted" {
 192.168.200.10;
 192.168.200.11;
 192.168.200.2;
 192.168.200.1;
};

options {
 directory "/var/cache/bind";

 recursion yes;
 allow-recursion { trusted; };
 listen-on { 192.168.200.10; };
 allow-transfer { none; };

 forwarders {
 8.8.8.8;
 8.8.4.4;
 };

 // If there is a firewall between you and nameservers you want
 // to talk to, you may need to fix the firewall to allow multiple
 // ports to talk. See http://www.kb.cert.org/vuls/id/800113
}
```

2

Onde:

1. Lista acl de permissão chamada trusted;
2. Opções de configuração do bind9;

Acima do bloco options existente, crie um novo bloco ACL (lista de controle de acesso) chamado “**trusted**”. É aqui que definiremos uma lista de clientes dos quais permitiremos consultas DNS recursivas (ou seja, seus servidores que estão na mesma rede que **ns1**). Usando nosso exemplo de endereços IP privados, adicionaremos **ns1** , **ns2** , **host1** e **gateway** à nossa lista de clientes.

Abaixo da diretiva **directory**, adicione as linhas de configuração destacadas (e substitua no endereço IP ns1 adequado ) para que se pareça com isto:

- **recursion yes**: permite consultas recursivas;
- **allow-recursion {trusted;}**: permite consultas recursivas de clientes "confiáveis";

- **listen-on {192.168.200.10;}**: endereço IP privado ns1 - ouvir apenas na rede privada;
- **allow-transfer {none;}**: desabilitar transferências de zona por padrão. Veja como é fácil obter dados de uma [organização por uma falha de configuração neste ponto](#).

Quando terminar, salve e feche o arquivo **named.conf.options**. A configuração acima especifica que apenas seus próprios servidores (os “trusted”) serão capazes de consultar seu servidor DNS para domínios externos.

## 20.5.2 Configurando o Arquivo Local

No servidor ns1, abra o arquivo **named.conf.local** para edição:

```
1. sudo nano /etc/bind/named.conf.local
```

Além de alguns comentários, o arquivo deve estar vazio. Aqui, especificaremos nossas zonas direta e reversa. As zonas DNS designam um escopo específico para gerenciar e definir registros DNS. Uma vez que nossos domínios estarão todos dentro do subdomínio “faculdade.aied.com.br”, usaremos isso como nossa zona de encaminhamento. Como os endereços IP privados de nossos servidores estão cada um no espaço IP 192.168.200.0/24, configuraremos uma zona reversa para que possamos definir pesquisas reversas dentro desse intervalo.

Adicione a zona de encaminhamento com as seguintes linhas, substituindo o nome da zona pelo seu próprio e pelo endereço IP privado do servidor DNS secundário na diretiva **allow-transfer**.



```

GNU nano 3.2 /etc/bind/named.conf.local
zone "faculdade.aied.com.br" {
 type master;
 file "/etc/bind/zones/db.faculdade.aied.com.br";
 allow-transfer { 192.168.200.11; };
};

zone "200.168.192.in-addr.arpa" {
 type master;
 file "/etc/bind/zones/db.192.168.200";
 allow-transfer { 192.168.200.11; };
};

```

Onde:

1. Configuração da zona para busca de nomes;
2. Configuração reversa da zona;

Supondo que nossa sub-rede privada seja 192.168.200.0/24, adicione a zona reversa com as seguintes linhas, observe que o nome da nossa zona reversa começa com “200.168.192” que é a reversão do octeto de “192.168.200” pois a máscara é 255.255.255.0. Se seus servidores abrangem várias sub-redes privadas, mas estão no mesmo datacenter, certifique-se de especificar uma zona adicional e um arquivo de zona para cada sub-rede distinta. Quando terminar de adicionar todas as zonas desejadas, salve e saia do arquivo

named.conf.local. Agora que nossas zonas estão especificadas no BIND, precisamos criar os arquivos de zona direta e reversa correspondentes.

### 20.5.3 Criação do arquivo de zona de encaminhamento

O arquivo de zona de encaminhamento é onde definimos os registros de DNS para pesquisas de DNS de encaminhamento. Ou seja, quando o DNS recebe uma consulta de nome, “host1.faculdade.aied.com.br” por exemplo, ele vai olhar no arquivo de zona de encaminhamento para resolver o endereço IP privado correspondente do host1.

Vamos criar o diretório onde nossos arquivos de zona residirão. De acordo com nossa configuração named.conf.local , esse local deve ser /etc/bind/zones:

```
1. sudo mkdir /etc/bind/zones
```

Vamos nos basear em nosso arquivo de zona de encaminhamento padrão, no arquivo db.local, copie-o para o local adequado com os seguinte comando:

```
1. sudo cp /etc/bind/db.local /etc/bind/zones/db.faculdade.aied.com.br
```

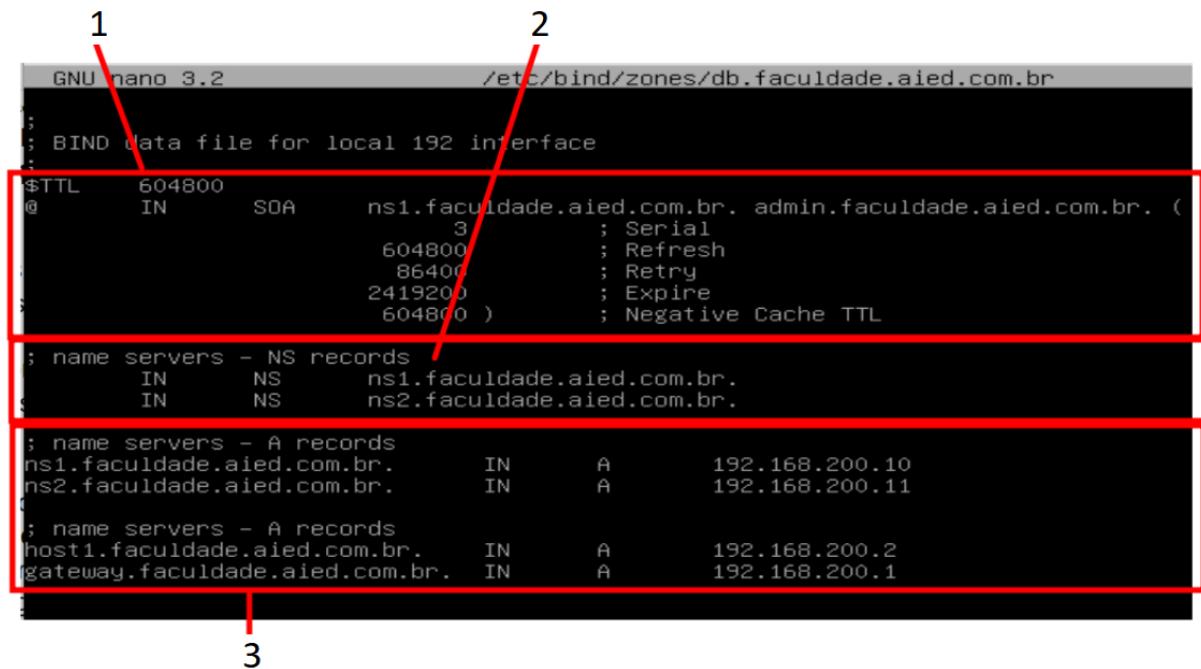
Agora vamos editar nosso arquivo de zona de encaminhamento:

```
1. sudo nano /etc/bind/zones/db.faculdade.aied.com.br
```

Na primeira parte será editado o registro SOA, substituindo o primeiro “localhost” pelo FQDN de **ns1** e, em seguida, substitua “**root.localhost**” por “**admin.faculdade.aied.com.br**”. Cada vez que você edita um arquivo de zona, é necessário incrementar o valor serial antes de reiniciar o processo named, vamos iremos incrementar para “3”.

Em seguida, exclua os três registros no final do arquivo (após o registro SOA). Se você não tiver certeza de quais linhas excluir. No final do arquivo, adicione os registros do servidor de nomes com as seguintes linhas (substitua os nomes pelos seus). Observe que a segunda coluna especifica que esses são registros “NS”

Agora, adicione os registros A para seus hosts que pertencem a esta zona. Isso inclui qualquer servidor cujo nome desejamos terminar com “**.faculdade.aied.com.br**” (substitua os nomes e endereços IP privados). Usando nossos nomes de exemplo e endereços IP privados, adicionaremos registros A para **ns1** , **ns2** e **host1** da seguinte forma:



```

GNU nano 3.2 /etc/bind/zones/db.faculdade.aied.com.br

; BIND Data file for local 192 interface

$TTL 604800
@ IN SOA ns1.faculdade.aied.com.br. admin.faculdade.aied.com.br. (
 3 ; Serial
 604800 ; Refresh
 86400 ; Retry
 2419200 ; Expire
 604800) ; Negative Cache TTL

; name servers - NS records
IN NS ns1.faculdade.aied.com.br.
IN NS ns2.faculdade.aied.com.br.

; name servers - A records
ns1.faculdade.aied.com.br. IN A 192.168.200.10
ns2.faculdade.aied.com.br. IN A 192.168.200.11

; name servers - A records
host1.faculdade.aied.com.br. IN A 192.168.200.2
gateway.faculdade.aied.com.br. IN A 192.168.200.1

```

Onde:

1. Registro SOA para o domínio **faculdade.aied.com.br**;
2. Endereços do serviço Domain Name Server (NS);
3. Endereços IPv4 e nome de computadores;

Salve e feche o **db.faculdade.aied.com.br**.

#### 20.5.4 Criando o (s) arquivo (s) de zona reversa

Os arquivos de zona reversa são onde definimos os registros PTR de DNS para pesquisas reversas de DNS. Ou seja, quando o DNS recebe uma consulta por endereço IP, “192.168.200.2” por exemplo, ele vai procurar no(s) arquivo(s) de zona reversa para resolver o FQDN correspondente, “**host1.faculdade.aied.com.br**” neste caso .

Em **ns1** , para cada zona reversa especificada no arquivo `named.conf.local`, crie um arquivo de zona reversa. Vamos basear nossos arquivos de zona reversa no arquivo `db.127` de zona. Copie-o para o local adequado com o seguinte comando:

```
1. sudo cp /etc/bind/db.127 /etc/bind/zones/db.192.168.200
```

Edite o arquivo da zona reversa que corresponde às zonas reversas definidas em **named.conf.local**:

```
1. sudo nano /etc/bind/zones/db.192.168.200
```

Da mesma maneira que o arquivo de zona de encaminhamento, você deseja editar o registro SOA e incrementar o valor serial.

No final do arquivo, adicione os registros do servidor de nomes com as seguintes linhas (substitua os nomes pelos seus). Observe que a segunda coluna especifica que esses são registros “NS”.

Em seguida, adicione registros PTR para todos os seus servidores cujos endereços IP estão na sub-rede do arquivo de zona que você está editando. Em nosso exemplo, isso inclui todos os nossos hosts porque estão todos na rede 192.168.200.0/24. Observe que a primeira coluna consiste nos três últimos octeto dos endereços IP privados dos seus servidores na ordem inversa. Certifique-se de substituir nomes e endereços IP privados para corresponder aos seus servidores.

```

1
2
3
GNU nano 3.2
/etc/bind/zones/db.192.168.200

; BIND reverse data file for local loopback interface
;

$TTL 604800
@ IN SOA faculdade.aied.com.br. admin.faculdade.aied.com.br. (
 3 ; Serial
 604800 ; Refresh
 86400 ; Retry
 2419200 ; Expire
 604800) ; Negative Cache TTL

; name servers
IN NS ns1.faculdade.aied.com.br.
IN NS ns2.faculdade.aied.com.br.

; PTR records
10 IN PTR ns1.faculdade.aied.com.br. ; 192.168.200.10
11 IN PTR ns2.faculdade.aied.com.br. ; 192.168.200.11
12 IN PTR gateway.faculdade.aied.com.br. ; 192.168.200.1
13 IN PTR host1.faculdade.aied.com.br. ; 192.168.200.2

```

Onde:

1. Registro SOA para o domínio `faculdade.aied.com.br`;
2. Endereços do serviço Domain Name Server (NS);
3. Endereço PTR para consulta de nomes baseado no IP.

Salve e feche o arquivo de zona reversa (repita esta seção se precisar adicionar mais arquivos de zona reversa).

### 20.5.5 Verificando a sintaxe de configuração do BIND

Execute o seguinte comando `named-checkconf` para verificar a sintaxe dos arquivos `named.conf`:

```
1. sudo named-checkconf
```

Se tudo estiver certo, nada será exibido.

```
usuario@ns1:~$ sudo named-checkconf
usuario@ns1:~$
```

Se os arquivos de configuração nomeados não tiverem erros de sintaxe, você retornará ao prompt do shell e não verá mensagens de erro. O comando **named-checkzone** pode ser usado para verificar a exatidão de seus arquivos de zona. Seu primeiro argumento especifica um nome de zona e o segundo argumento especifica o arquivo de zona correspondente, ambos definidos em **named.conf.local**.

Por exemplo, para verificar o `faculdade.aied.com.br` Configuração da zona de encaminhamento, execute o seguinte comando (altere os nomes para corresponder à zona de encaminhamento e arquivo):

```
1. sudo named-checkzone facultade.aied.com.br /etc/bind/zones/db.faculdade.aied.com.br
```

Um **OK** deverá ser exibido.

```
usuario@ns1:~$ sudo named-checkzone facultade.aied.com.br /etc/bind/zones/db.faculdade.aied.com.br
zone facultade.aied.com.br/IN: loaded serial 3
OK
usuario@ns1:~$
```

E para verificar o “`192.168.200.in-addr.arpa` ”configuração da zona reversa, execute o seguinte comando (altere os números para corresponder à sua zona reversa e arquivo):

```
1. sudo named-checkzone 192.168.200.in-addr.arpa /etc/bind/zones/db.192.168.200
```

Um **OK** deverá ser exibido.

```
usuario@ns1:~$ sudo named-checkzone 192.168.200.in-addr.arpa /etc/bind/zones/db.192.168.200
zone 192.168.200.in-addr.arpa/IN: loaded serial 3
OK
usuario@ns1:~$
```

Quando todos os seus arquivos de configuração e zona não contêm erros, você deve estar pronto para reiniciar o serviço BIND9.

```
1. sudo systemctl restart bind9
```

## 20.6 Configurando o servidor DNS secundário **ns2**

Na maioria dos ambientes, é uma boa ideia configurar um servidor DNS secundário que responderá às solicitações se o primário ficar indisponível. Felizmente, o servidor DNS secundário é muito mais fácil de configurar. Lembre-se que deve definir o nome **ns2** para a máquina secundária e dar o IP `192.168.200.11/24`, um serviço semelhante ao realizado no **ns1**. No **ns2**, edite o arquivo **named.conf.options**:

```
1. sudo nano /etc/bind/named.conf.options
```

Na parte superior do arquivo, adicione a ACL com os endereços IP privados de todos os seus servidores confiáveis.

```
GNU nano 3.2 /etc/bind/named.conf.options

acl "trusted" {
 192.168.200.10;
 192.168.200.11;
 192.168.200.1;
 192.168.200.2;
};

options {
 directory "/var/cache/bind";

 recursion yes;
 allow-recursion {trusted; };
 listen-on {192.168.200.11; };
 allow-transfer { none; };
 forwarders {
 8.8.8.8;
 8.8.4.4;
 };

 // If there is a firewall between you and nameservers you want
 // to talk to, you may need to fix the firewall to allow multiple
 // ports to talk. See https://www.iana.org/assignments/dns-ports
 // for further information.
};


```

Salve e feche o arquivo **named.conf.options**. Este arquivo deve ser exatamente igual ao arquivo **named.conf.options** da máquina **ns1**, exceto que deve ser configurado para escutar no endereço IP privado do **ns2**. Agora edite o arquivo **named.conf.local**:

1. sudo nano /etc/bind/named.conf.local

Defina as zonas slave que correspondem às zonas mestras no servidor DNS primário. Observe que o tipo é “**slave**”, o arquivo não contém um caminho e há uma diretiva **master** que deve ser definida para o endereço IP privado do servidor DNS primário. Se você definiu várias zonas reversas no servidor DNS primário, certifique-se de adicioná-las aqui:

```
GNU nano 3.2 /etc/bind/named.conf.local

zone "faculdade.aied.com.br" {
 type slave;
 file "db.faculdade.aied.com.br";
 masters { 192.168.200.10; };
};

zone "192.168.200.in-addr.arpa" {
 type slave;
 file "db.192.168.200";
 masters { 192.168.200.10; };
};


```

Agora salve e feche o arquivo **named.conf.local**. Execute o seguinte comando para verificar a validade dos seus arquivos de configuração:

1. sudo named-checkconf

Depois de verificar, reinicie o BIND9:

1. sudo systemctl restart bind9

Agora você tem servidores DNS primários e secundários para nome de rede privada e resolução de endereço IP.

## 20.7 Cliente Debian host1

Para realizar o teste será necessário utilizar o comando nslookup que está no pacote dnsutils, para isso execute a instalação abaixo.

- ```
1. sudo apt update -y
2. sudo apt install dnsutils -y
```

Nos servidores Debian Linux, você pode editar o arquivo **/etc/network/interfaces**:

- ```
1. sudo nano /etc/network/interfaces
```

Dentro, encontre a linha **dns-nameservers**, em seguida, acrescente seus próprios servidores de nome antes da lista que está lá atualmente. Abaixo dessa linha, adicione uma opção **dns-search** apontada para o domínio base de sua infraestrutura, que em nosso caso, seria **faculdade.aied.com.br**:

```
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
#allow-hotplug enp0s3
auto enp0s3
iface enp0s3 inet static
 address 192.168.200.2
 netmask 255.255.255.0
 gateway 192.168.200.1
 network 192.168.200.0
 broadcast 192.168.200.255
 dns-nameservers 192.168.200.10
 dns-search faculdade.aied.com.br
```

Reinicialize a máquina. O próximo passo é alterar o arquivo **/etc/resolv.conf** para ter como único **nameserver** o endereço **192.168.200.10**. Para testar no host1, basta usar o comando **nslookup** (está instalado no pacote dnsutils).

```

usuario@debian:~$ nslookup gateway.faculdade.aied.com.br
Server: 192.168.200.10
Address: 192.168.200.10#53

Name: gateway.faculdade.aied.com.br
Address: 192.168.200.1

usuario@debian:~$
```

Onde:

1. IP do servidor DNS que respondeu ao nslookup;
2. Nome e IP da pesquisa;

Veja que quem respondeu foi o servidor **192.168.200.10**. Ao fazer uma solicitação para um domínio externo, quem nos responde também é o servidor **192.168.200.10**, ou seja, o servidor **192.168.200.10** aprendeu e nos responde internamente na próxima vez.

```

usuario@debian:~$ nslookup google.com
Server: 192.168.200.10
Address: 192.168.200.10#53

Non-authoritative answer:
Name: google.com
Address: 172.217.162.206
Name: google.com
Address: 2800:3f0:4001:818::200e

usuario@debian:~$
```

Em uma máquina fora desta rede (sem a influência do **ns1** e **ns2**), veja que a resposta é dada por um servidor externo.

```

usuario@debian:~$ nslookup google.com
Server: 181.213.132.2
Address: 181.213.132.2#53

Non-authoritative answer:
Name: google.com
Address: 172.217.30.110
Name: google.com
Address: 2800:3f0:4001:802::200e

usuario@debian:~$
```

## 20.8 Práticas do capítulo de DNS com GNU/Linux

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

### 20.8.1 Prática 1ae1b810 01: Validando a configuração do servidor **ns1**

Nesta prática o aluno deve ter a expertise para configurar o servidor DNS.

1. Execute a configuração do servidor ns1;

Execute o comando aied conforme comando listagem abaixo.

1. sudo aied validar 1ae1b810 checkpoint01

```
usuario@ns1:/tmp$ sudo aied validar 1ae1b810 checkpoint01
Ação que será executada: validar

Prática: Prática da aula DNS - Configurando o servidor NS1
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: cat /etc/hostname
Será enviado o OUTPUT do comando: cat /etc/hosts
Será enviado o OUTPUT do comando: ping -c 1 192.168.200.1
Será enviado o OUTPUT do comando: cat /etc/bind/named.conf.options
Será enviado o OUTPUT do comando: named-checkconf
Será enviado o OUTPUT do comando: named-checkzone facultade.aied.com.br /etc/bind/zones/db.faculdade.aied.com.br
Será enviado o OUTPUT do comando: named-checkzone 192.168.200.in-addr.arpa /etc/bind/zones/db.192.168.200

Deseja continuar? (s para SIM) _
```

Digite **s** caso tenha certeza que nenhuma informação afetará sua segurança, o resultado deverá ser semelhante ao abaixo (pode-se adicionar novas validações ao longo do tempo):

```
++++++ RESULTADO ++++++
1 - Comando: ip address
1.1 Validar por regex /inet s+192.168.200.10/
1.2 Validar por regex /brd s+ d+. d+. 255/

2 - Comando: cat /etc/hostname
2.1 Validar por text ns1

3 - Comando: cat /etc/hosts
3.1 Validar por regex /127.0.1.1(.*)ns1/

4 - Comando: ping -c 1 192.168.200.1
4.1 Validar por regex / s+0% s+packet s+loss/

5 - Comando: cat /etc/bind/named.conf.options
5.1 Validar por regex /acl(.*)"(.*)trusted/
5.2 Validar por regex /recursion(.*)yes/
5.3 Validar por regex /allow-recursion(.*)trusted/
5.4 Validar por regex /listen-on(.*)192.168.200.10/

6 - Comando: named-checkconf
6.1 Validar por text

7 - Comando: named-checkzone facultade.aied.com.br /etc/bind/zones/db.faculdade.aied.com.br
7.1 Validar por regex /OK/

8 - Comando: named-checkzone 192.168.200.in-addr.arpa /etc/bind/zones/db.192.168.200
8.1 Validar por regex /OK/

Total de acertos: 12 do total de 12 validações equivalente a 100%.
Identificação: 8c913dc775
AIED v(7)

usuario@ns1:/tmp$
```

## 20.8.2 Prática 1ae1b810 01: Validando a configuração do servidor **ns2**

Nesta prática o aluno deve ter a expertise para configurar o servidor DNS.

1. Execute a configuração do servidor ns2;

Execute o comando aied conforme comando listagem abaixo.

1. sudo aied validar 1ae1b810 checkpoint02

```
usuario@ns2:/tmp$ sudo aied validar 1ae1b810 checkpoint02
Ação que será executada: validar

Prática: Prática da aula DNS - Configurando o servidor NS2
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: cat /etc/hostname
Será enviado o OUTPUT do comando: cat /etc/hosts
Será enviado o OUTPUT do comando: ping -c 1 192.168.200.1
Será enviado o OUTPUT do comando: cat /etc/bind/named.conf.options
Será enviado o OUTPUT do comando: named-checkconf

Deseja continuar? (s para SIM)
```

Digite **s** caso tenha certeza que nenhuma informação afetará sua segurança, o resultado deverá ser semelhante ao abaixo (pode-se adicionar novas validações ao longo do tempo):

```
++++++ RESULTADO ++++++
1 - Comando: ip address
1.1 Validar por regex /inet s+192.168.200.11/
1.2 Validar por regex /brd s+ d+. d+. d+.255/

2 - Comando: cat /etc/hostname
2.1 Validar por text ns2

3 - Comando: cat /etc/hosts
3.1 Validar por regex /127.0.1.1(.*)ns2/

4 - Comando: ping -c 1 192.168.200.1
4.1 Validar por regex / s+0% s+packet s+loss/

5 - Comando: cat /etc/bind/named.conf.options
5.1 Validar por regex /acl(.*)"(.*)trusted/
5.2 Validar por regex /recursion(.*)yes/
5.3 Validar por regex /allow-recursion(.*)trusted/
5.4 Validar por regex /listen-on(.*)192.168.200.11/

6 - Comando: named-checkconf
6.1 Validar por text

Total de acertos: 10 do total de 10 validações equivalente a 100%.
Identificação: 52b252d9b6
AIED v(7)
```

### 20.8.3 Prática 1ae1b810 03: Validando a configuração no **host1**

Nesta prática o aluno deve ter a expertise para configurar o cliente DNS.

1. No host1;
2. Configure o arquivo /etc/network/interfaces conforme tópico de cliente;
3. Configure o arquivo /etc/resolv.conf no cliente;

```
1. sudo aied validar 1ae1b810 checkpoint03
```

```
usuario@host1:~$ sudo aied validar 1ae1b810 checkpoint03
Ação que será executada: validar

Prática: Prática da aula DNS - Configurando e testando o cliente host1
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: cat /etc/hostname
Será enviado o OUTPUT do comando: cat /etc/hosts
Será enviado o OUTPUT do comando: ping -c 1 192.168.200.10
Será enviado o OUTPUT do comando: nslookup gateway.faculdade.aied.com.br
Será enviado o OUTPUT do comando: nslookup google.com

Deseja continuar? (s para SIM) _
```

Digite **s** caso tenha certeza que nenhuma informação afetará sua segurança, o resultado deverá ser semelhante ao abaixo (pode-se adicionar novas validações ao longo do tempo):

```
++++++ RESULTADO ++++++
1 - Comando: ip address
1.1 Validar por regex /inet s+192.168.200.2/
1.2 Validar por regex /brd s+ d+. d+. d+.255/

2 - Comando: cat /etc/hostname
2.1 Validar por text host1

3 - Comando: cat /etc/hosts
3.1 Validar por regex /127.0.1.1(.*)host1/

4 - Comando: ping -c 1 192.168.200.10
4.1 Validar por regex / s+0% s+packet s+loss/

5 - Comando: nslookup gateway.faculdade.aied.com.br
5.1 Validar por regex /Server :(.*)192.168.200.10/
5.2 Validar por regex /Address :(.*)192.168.200.1/

6 - Comando: nslookup google.com
6.1 Validar por regex /Server :(.*)192.168.200.10/

Total de acertos: 8 do total de 8 validações equivalente a 100%.
Identificação: ab0b27b2e1
AIED v(7)
```

Questões de segurança para bind9

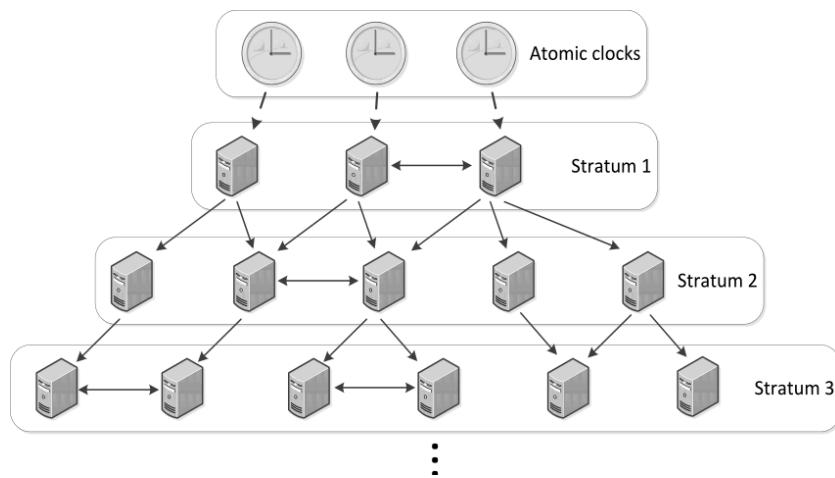
## 21 Banco de Dados

## 22 Serviço NTP

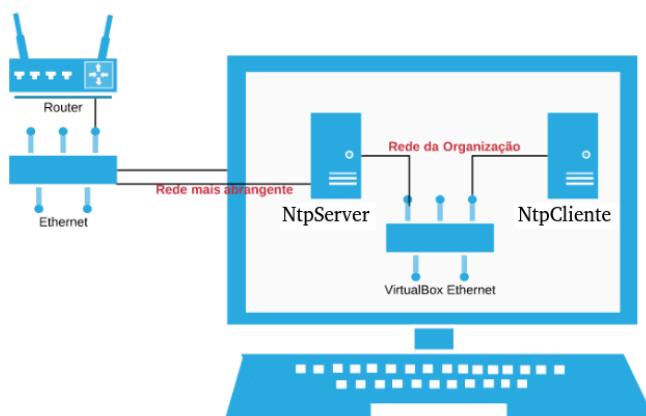
NTP é um protocolo de rede usado para sincronizar todos os relógios do sistema em uma rede. Isso permite que todos os sistemas em uma rede tenham o mesmo horário. O NTP faz isso entrando em contato com vários outros servidores de horário na Internet. O NTP usa a porta UDP 123 para se comunicar com clientes e outros servidores NTP.

O serviço NTP está apoiado em uma arquitetura em árvore, na qual existem camadas, do topo para a base da árvore é aumentada a abrangência em número de clientes direto. Na camada 0 encontramos os servidores NTP que se regulam com uma das duas opções:

- Sistema de Posicionamento Global (GPS);
- Relógio atômico;



O objetivo desta aula é configurar um servidor NTP privado com GNU/Linux dentro da rede local para poder disponibilizar o serviço NTP Date para as estações e configurar uma estação cliente.



Neste cenário o servidor NtpServer terá 2 interfaces de rede, isso é preciso pois ele utiliza-se de um serviço na rede mundial para se atualizar, já o NtpCliente deverá consultar o servidor de Date Time do NtpServer. **Atenção, faça toda a aula de Gateway para poder iniciar esta aula.**

Então a configuração de rede no NtpServer será:

```
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp

auto enp0s8
iface enp0s8 inet static
 address 192.168.200.1
 netmask 255.255.255.0
 network 192.168.200.0
 broadcast 192.168.200.255
```

Já o NtpCliente terá apenas uma interface de rede apontada para rede interna do VirtualBox, e terá a seguinte configuração de rede.

```
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.2
 netmask 255.255.255.0
 network 192.168.200.0
 broadcast 192.168.200.255
 gateway 192.168.200.1
 dns-nameservers 8.8.8.8
```

Reinic peace ambas as máquinas e execute o comando ping para validar a conectividade das duas máquinas configuradas.

## 22.1 Recursos do capítulo

Todo o conteúdo externo ao texto pode ser obtido no link abaixo.



O material deste capítulo pode ser obtido neste link.

## 22.2 Instalação do servidor NTP (máquina NtpServer)

Se você estiver executando um único sistema ou uma pequena rede local com poucos hosts, configurar um servidor NTP dedicado pode ser considerado um exagero. No entanto, para uma rede grande, é recomendado configurar um servidor NTP privado, servindo apenas clientes LAN de forma privada.

A instalação do servidor NTP é fácil como a execução de um comando:

```
1. sudo apt update -y
2. sudo apt install ntp -y
```

Após a instalação, confirme se o servidor NTP está instalado e funcionando:

```
1. sudo systemctl status ntp.service
```

## 22.3 Configuração do servidor NTP

O servidor NTP vem pré-configurado e pronto para uso. A configuração do seu servidor NTP pode ser gerenciada via arquivo de configuração **/etc/ntp.conf**. A configuração do servidor NTP padrão depende do grupo de servidores `debian.pool.ntp.org`:

```
1. pool 0.debian.pool.ntp.org iburst
2. pool 1.debian.pool.ntp.org iburst
3. pool 2.debian.pool.ntp.org iburst
4. pool 3.debian.pool.ntp.org iburst
```

Dependendo de sua localização, você pode editar o grupo de servidores NTP acima para qualquer servidor NTP conhecido mais próximo de sua localização. Como alternativa, use `pool.ntp.org` subgrupos específicos de cada país. Por exemplo, para limitar o subgrupo do servidor NTP aos Estados Unidos, edite o seu `/etc/ntp.conf` com:

```
1. pool 0.us.pool.ntp.org iburst
2. pool 1.us.pool.ntp.org iburst
3. pool 2.us.pool.ntp.org iburst
4. pool 3.us.pool.ntp.org iburst
```

Isso funciona para a maioria dos países. Por exemplo, se você estiver na Austrália, o `0.debian.pool.ntp.org` será `0.au.pool.ntp.org`, para a Índia `0.in.pool.ntp.org` e assim por diante. Agora, é possível que seja necessário alojar servidores NTP dentro de sua rede

local, seja pelo seu tamanho ou seja problemas de servidores transparentes fora de seu domínio.

### 22.3.1 Brasil br.pool.ntp.org

Para usar esta zona de pool específica, adicione o seguinte ao seu arquivo `ntp.conf`:

1. server 0.br.pool.ntp.org
2. server 1.br.pool.ntp.org
3. server 2.br.pool.ntp.org
4. server 3.br.pool.ntp.org

Conforme imagem abaixo.

```
GNU nano 3.2 /etc/ntp.conf

You do need to talk to an NTP server or two (or three).
#server ntp.your-provider.example

pool.ntp.org maps to about 1000 low-stratum NTP servers. Your
pick a different set every time it starts up. Please consider
pool: <http://www.pool.ntp.org/join.html>
#pool 0.debian.pool.ntp.org iburst
#pool 1.debian.pool.ntp.org iburst
#pool 2.debian.pool.ntp.org iburst
#pool 3.debian.pool.ntp.org iburst

server 0.br.pool.ntp.org
server 1.br.pool.ntp.org
server 2.br.pool.ntp.org
server 3.br.pool.ntp.org
```

Na maioria dos casos, é melhor usar `pool.ntp.org` para encontrar um servidor NTP (ou `0.pool.ntp.org`, `1.pool.ntp.org`, etc, se você precisar de vários nomes de servidor). O sistema tentará encontrar os servidores disponíveis mais próximos para você usando uma técnica de serviço transparente.

### 22.3.2 Restringindo acesso ao NTP Server local

Caso queira manter um serviço NTP você pode querer restringir o acesso ao seu servidor NTP apenas para clientes específicos da LAN. Por exemplo, adicionando a linha abaixo em seu arquivo de configuração NTP `/etc/ntp.conf`, o servidor irá restringir o uso de NTP apenas para `192.168.200.0` rede com máscara `255.255.255.0`.

```
restrict 192.168.200.0 mask 255.255.255.0 nomodify notrap
```

Sempre que é feita qualquer alteração sobre o arquivo /etc/ntp.conf o administrador de rede deve reiniciar o serviço:

```
sudo systemctl restart ntp.service
```

Para permitir que seu servidor seja iniciado após a execução da reinicialização:

```
sudo systemctl enable ntp.service
```

Caso use firewall ou regras Iptables, deve então adicionar na lista de regras a porta 123 conforme listagem abaixo e **execute as regras**.

1. iptables -A OUTPUT -p udp --dport 123 -j ACCEPT
2. iptables -A INPUT -p udp --sport 123 -j ACCEPT

## 22.4 Monitorando

Uma ferramenta muito útil para monitorar o serviço NTP (no NtpServer) é o ntpmon que está no pacote ntpsec, para instalar é simples, basta:

1. sudo apt install ntpsec -y

Para executar no terminal o monitoramento basta digitar:

1. sudo ntpmon

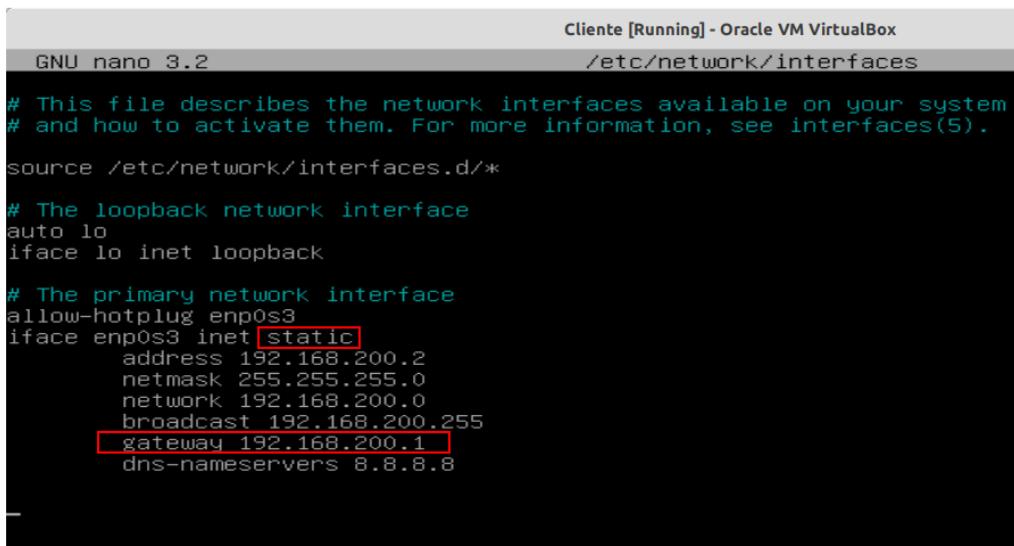
O programa monitora em tempo real a comunicação entre os elementos NTP (**OBS: a imagem abaixo foi retirada de um print após todo o ambiente está configurado, ou seja, prática pronta**).

remote	refid	st	t	when	poll	reach	delay	offset	jitter
0.debian.pool.n	.POOL.	16	p	-	256	0	0.0000	0.0000	0.0001
1.debian.pool.n	.POOL.	16	p	-	256	0	0.0000	0.0000	0.0001
2.debian.pool.n	.POOL.	16	p	-	256	0	0.0000	0.0000	0.0001
3.debian.pool.n	.POOL.	16	p	-	256	0	0.0000	0.0000	0.0001
+c.ntp.br	200.160.7.186	2	u	21	64	3	40.4194	-7.4056	7.5268
+a.ntp.br	200.160.7.186	2	u	20	64	3	11.9173	-7.4586	8.3575
#45.11.105.223	200.160.7.197	2	u	21	64	3	128.0695	50.0262	7.3651
#45.11.105.253	200.160.7.197	2	u	19	64	3	152.4524	62.0636	7.9796
-t1.time.ir2.yah	212.82.106.33	2	u	76	64	2	198.7389	-2.3463	2.9805
-bras-base-toroo	97.183.206.88	2	u	17	64	3	176.4766	-11.1002	8.8853
#196.200.160.123	96.180.207.109	2	u	20	64	3	261.3084	-0.6136	7.5846
#185.32.222.237	195.176.26.206	2	u	18	64	3	209.2480	-7.6964	9.2616
#2a0b:4341:1500:	200.160.7.197	2	u	17	64	3	130.4349	52.7748	8.0727
*a.st1.ntp.br	.ONBR.	1	u	19	64	3	12.0952	-6.8076	7.9790
#br-zone.ntp4.rb	131.188.3.220	2	u	17	64	3	239.9858	-7.9777	8.3960
+a.ntp.br	200.160.7.186	2	u	16	64	3	13.5348	-6.1703	6.6065
-time.cloudflare	10.97.10.21	3	u	19	64	3	12.6636	-15.7913	6.3916
any.time.nl	.INIT.	16	u	-	64	0	0.0000	0.0000	0.0001
-181.215.89.100	69.89.207.99	2	u	18	64	3	147.6438	-15.9555	9.0849
stratum2-1.ntp.	.INIT.	16	u	-	64	0	0.0000	0.0000	0.0001
#85.199.214.101	.GPS.	1	u	15	64	3	226.4690	-5.8688	7.0128
ns1.blazing.de	213.172.96.14	2	u	10	64	1	224.8199	-17.0207	1.1357
-euphoric.ca	128.233.154.245	2	u	15	64	3	221.2702	-12.8266	6.6184
-38.229.62.9	172.16.21.35	2	u	18	64	3	123.2638	-9.0366	8.4658
ntpd ntpsec-1.1.3	2019-11-18T06:04:00Z						Updated: 2021-06-16T12:26:40 (1)		
lstint avgint rstr r m v	count rport remote address								
0 0.046	0 . 6 2	1729	32860	localhost					
4 4.70	d0 . 3 4	16	123	192.168.200.2					
10 2.00	d0 . 4 4	6	123	ns1.blazing.de					
15 11	d0 . 4 4	7	123	85.199.214.101					
15 11	d0 . 4 4	7	123	euphoric.ca					
17 11	d0 . 4 4	7	123	a.ntp.br					
17 11	d0 . 4 4	7	123	br-zone.ntp4.rbx-fr.hosts.301-moved.de					
17 11	d0 . 4 4	7	123	bras-base-toroon2638w-grc-85-174-88-103-					
17 11	d0 . 4 4	7	123	2a0b:4341:1500:123::					
18 11	d0 . 4 4	7	123	185.32.222.237					

Veja que na listagem acima uma máquina com o IP 192.168.200.2 já realizou 16 requisições NTP.

## 22.5 Configuração do Cliente

//todo: Wellington colocar a valicação do dpkg -i | grep ntpdate, e explicar lá o motivo disso.  
Configure o arquivo /etc/network/interfaces com IP conforme figura abaixo.



```

Cliente [Running] - Oracle VM VirtualBox
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.2
 netmask 255.255.255.0
 network 192.168.200.0
 broadcast 192.168.200.255
 gateway 192.168.200.1
 dns-nameservers 8.8.8.8

```

Para configurar o cliente para usar o servidor NTP, primeiro você precisa instalar o ntpdate pacote:

```

sudo apt update -y
sudo apt install ntpdate -y

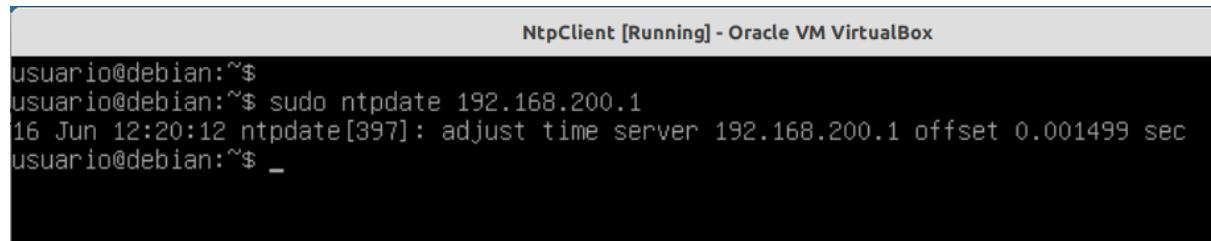
```

Depois de instalado, tente consultar seu novo servidor NTP. Dado que o servidor NTP pode ser resolvido através do nome do host 192.168.200.1 execute:

```

sudo ntpdate 192.168.200.1

```



```

NtpClient [Running] - Oracle VM VirtualBox
usuario@debian:~$ sudo ntpdate 192.168.200.1
16 Jun 12:20:12 ntpdate[397]: adjust time server 192.168.200.1 offset 0.001499 sec
usuario@debian:~$ _

```

## 22.5.1 Criando um script de inicialização

Uma forma garantida de executar o comando de atualização de datas no início do Linux é criar um arquivo de serviço no Systemd, o processo é simples, crie um novo arquivo em /etc/systemd/system chamado nptdate.service conforme imagem abaixo.

```
GNU nano 3.2 /etc/systemd/system/ntpdate.service

[Unit]
Description=Ntpdate
After=network.target

[Service]
ExecStart=ntpdate -u 192.168.200.1

[Install]
WantedBy=multi-user.target
```

Agora ative o script para inicialização com a sequência de comandos abaixo.

1. sudo systemctl daemon-reload
2. sudo systemctl enable ntpdate.service
3. sudo systemctl start ntpdate.service
4. sudo systemctl status ntpdate.service

## 22.6 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

### 22.6.1 Prática i0721ax 01: Configurando o serviço NTP

Nesta prática o aluno deve ter a expertise para configurar instalar e configurar o serviço NTP.

1. Configure a interface de rede enp0s8 do servidor com os seguintes dados:
  - a. Endereço: 192.168.200.1;
  - b. Rede: 192.168.200.0
  - c. Broadcast: 192.168.200.255;
  - d. Máscara: 255.255.255.0;
2. Instale o npt e o ntpsec;
3. Configure o arquivo /etc/ntp.conf com o pool BR;
4. Confirme que o serviço ntp.service está ativo para ser iniciado com o Servidor;

Execute o comando aied (**NO SERVIDOR**) conforme comando listagem abaixo no servidor NTP.

1. sudo aied validar i0721ax checkpoint01

```
usuario@debian:~$ sudo aied validar i0721ax checkpoint01
Ação que será executada: validar

Prática: Validação da prática de NTP, instalando o serviço NTP
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: cat /etc/ntp.conf
Será enviado o OUTPUT do comando: ps aux | grep /usr/sbin/ntpd | grep -v grep
Será enviado o OUTPUT do comando: dpkg -l | grep ntp
```

Deseja continuar? (s para SIM) s

```
++++++ RESULTADO ++++++
1 - Comando: ip address
1.1 Validar por regex /inet s+192.168.200.1/
1.2 Validar por regex /brd s+ d+. d+. 255/

2 - Comando: cat /etc/ntp.conf
2.1 Validar por text 0.br.pool.ntp.org
2.2 Validar por text 1.br.pool.ntp.org
2.3 Validar por text 2.br.pool.ntp.org
2.4 Validar por text 3.br.pool.ntp.org

3 - Comando: ps aux | grep /usr/sbin/ntpd | grep -v grep
3.1 Validar por text /ntp.conf

4 - Comando: dpkg -l | grep ntp
4.1 Validar por text ntp
4.2 Validar por text ntpsec

Total de acertos: 9 do total de 9 validações equivalente a 100%.
Identificação: b84e3955d9
AIED v(7)
```

## 22.6.2 Prática i0721ax 02: Configurando o cliente

Nesta prática o aluno deve ter a expertise para configurar instalar e configurar o cliente.

1. Configure a interface enp0s3 do cliente para se conectar por static;
2. Reinicie;
3. Faça a instalação do ntpdate com apt;
4. Configure a interface de rede enp0s3 do servidor com os seguintes dados:
  - a. Endereço: 192.168.200.2;
  - b. Rede: 192.168.200.0
  - c. Broadcast: 192.168.200.255;
  - d. Máscara: 255.255.255.0;
5. Reinicie a máquina cliente;
6. Crie um arquivo ntpdate.service em /etc/systemd/system/ conforme tópico 22.5.1;
7. Reinicie o servidor;

Execute o comando aied (**NO SERVIDOR**) conforme comando listagem abaixo no servidor NTP.

1. sudo aied validar i0721ax checkpoint02

```
usuario@debian:~$ sudo aied validar i0721ax checkpoint02
Ação que será executada: validar

Prática: Validação da prática de NTP, validando que o cliente acessou
Será enviado o OUTPUT do comando: ntpq -nc mrulist
```

```
Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
```

```
++++++ RESULTADO ++++++
1 - Comando: ntpq -nc mrulist
1.1 Validar por text 192.168.200.2

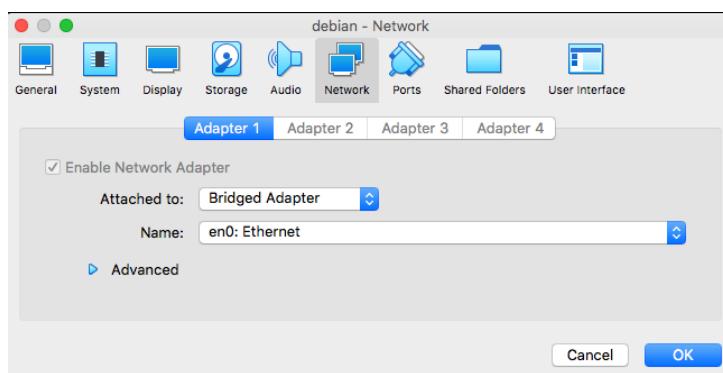
Total de acertos: 1 do total de 1 validações equivalente a 100%.
Identificação: b84e3955d9
AIED v(7)
```

## 23 Samba (finalizado)

Samba é um serviço de compartilhamento de recursos de armazenamento em sistemas secundários e também serviço de impressão. Toda distribuição Linux pode ser um servidor de arquivos em rede.

Para esta prática será necessário apenas uma única máquina virtual ligado a sua rede, a idéia é configurar um GNU/Linux virtual na sua rede e pelo seu sistema operacional Host é feito o acesso ao serviço SMB.

Será preciso para esta prática apenas uma Máquina Virtual, então no VirtualBox garanta que a Nova Virtual Machine está com o Adaptador 1 em modo Bridge conforme figura abaixo.



A configuração desta interface no GNU/Linux (arquivo `/etc/network/interfaces`) deve estar como DHCP para obter IP automaticamente pelo router físico, após garantir esta configuração veja o IP adquirido por dhcp com o comando `ip address` conforme figura abaixo.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group d
000
 link/ether 08:00:27:c2:47:d2 brd ff:ff:ff:ff:ff:ff
 inet 192.168.0.15/24 brd 192.168.0.255 scope global dynamic enp0s3
 valid_lft 35985sec preferred_lft 35985sec
 inet6 2804:14c:bf41:9a4e:a00:27ff:fea2:47d2/64 scope global dynamic mngtmpaddr
 valid_lft 86385sec preferred_lft 71985sec
 inet6 fe80::a00:27ff:fea2:47d2/64 scope link
 valid_lft forever preferred_lft forever
usuario@debian:~$
```

No caso da rede interna do autor deste texto o Router passou como resposta a requisição DHCP o IP 192.168.0.15 e naturalmente na sua infraestrutura este número pode ser diferente, anote o IP atribuído no seu caso.

Agora de seu computador pessoal (real), usando o comando **ping** confirme acesso ao servidor, conforme figura abaixo, veja que usei o IP que está na figura acima.

```
[MacBook-Pro:~ wellington$ ping 192.168.0.15
PING 192.168.0.15 (192.168.0.15): 56 data bytes
64 bytes from 192.168.0.15: icmp_seq=0 ttl=64 time=0.495 ms
64 bytes from 192.168.0.15: icmp_seq=1 ttl=64 time=0.311 ms
64 bytes from 192.168.0.15: icmp_seq=2 ttl=64 time=0.464 ms
64 bytes from 192.168.0.15: icmp_seq=3 ttl=64 time=0.364 ms
64 bytes from 192.168.0.15: icmp_seq=4 ttl=64 time=0.466 ms
64 bytes from 192.168.0.15: icmp_seq=5 ttl=64 time=0.399 ms
64 bytes from 192.168.0.15: icmp_seq=6 ttl=64 time=0.351 ms
64 bytes from 192.168.0.15: icmp_seq=7 ttl=64 time=0.947 ms
64 bytes from 192.168.0.15: icmp_seq=8 ttl=64 time=0.338 ms
```

Pronto, o ambiente está preparado para receber a configuração do Samba.  
falar sobre nfs e RPC

## 23.1 Recursos do capítulo

Todo o conteúdo externo ao texto pode ser obtido no link abaixo.



☰ O material deste capítulo pode ser [obtido neste link](#).

## 23.2 Configurando o servidor Samba

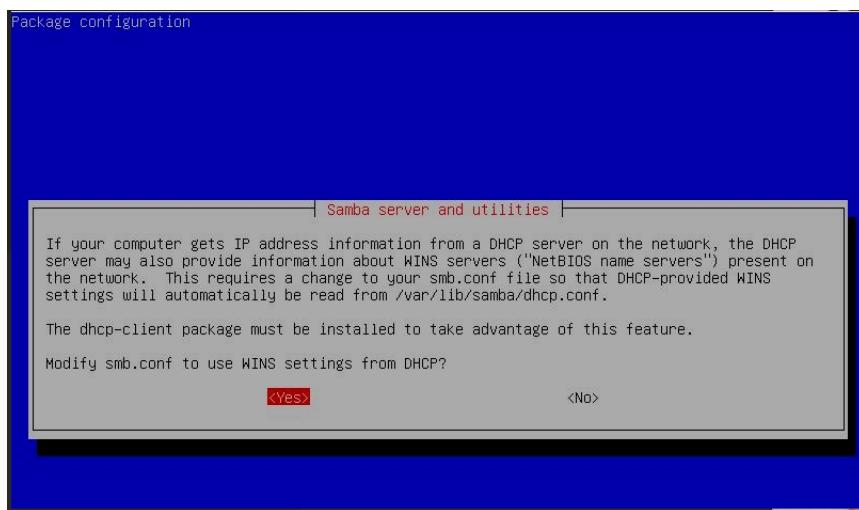
Agora na Virtual Machine execute um update, conforme listagem abaixo.

```
1. sudo apt update -y
```

Nesta mesma máquina Virtual que será configurada instale os pacotes do samba e suas dependências, conforme imagem abaixo.

```
usuario@debian:~$ sudo apt install libcups2 samba samba-common
```

O Cups está sendo adicionado pois o Samba também permite que usuários acessem o serviço de impressão.



**ATENÇÃO:** Se alguma pergunta for feita em relação ao DHCP cliente, responda Yes.

O próximo passo é copiar o arquivo de configuração do Samba para um diretório temporário para que possamos alterar com segurança, um erro aqui pode ser difícil de reverter se o aluno está aprendendo.

Comum em vez de jogar em /tmp fazer isso:

```
1. cp /etc/samba/smb.conf /etc/samba/smb.conf.backup
```

Agora vamos editar o arquivo de configuração do Samba, para isso com um editor de texto edite o seguinte arquivo: /etc/samba/smb.conf

```
usuario@debian:~$ sudo nano /etc/samba/smb.conf
[sudo] password for usuario:
```

O arquivo é grande, então procure um comentário chamado "# ##### Authentication #####" conforme figura abaixo.

Se a linha **security** não existir, digite ela. Vamos passar como parâmetro que a segurança será feita por usuário da máquina, conforme figura abaixo.

```
#####
Authentication
security = user
```

Você pode reiniciar o serviço do Samba com uma das opções:

1. sudo reboot
2. **sudo /etc/init.d/smbd stop** e depois **sudo /etc/init.d/smbd start**
3. **sudo systemctl restart smbd.service**

No meu exemplo usei a primeira opção, onde estou reiniciando a máquina. Logo após iniciar o sistema operacional, digito o comando `systemctl status smbd.service`, conforme figura abaixo.

```
usuario@debian:~$ /etc/init.d/smbd status
● smbd.service - Samba SMB Daemon
 Loaded: loaded (/lib/systemd/system/smbd.service; enabled; vendor
 Active: active (running) since Tue 2020-06-09 10:16:01 -03; 28s ag
 Docs: man:smbd(8)
 man:samba(7)
 man:smb.conf(5)
 Process: 430 ExecStartPre=/usr/share/samba/update-apparmor-samba-pr
CESS)
 Main PID: 439 (smbd)
 Status: "smbd: ready to serve connections..."
 Tasks: 4 (limit: 1150)
 Memory: 17.3M
 CGroup: /system.slice/smbd.service
 ├─439 /usr/sbin/smbd --foreground --no-process-group
 ├─441 /usr/sbin/smbd --foreground --no-process-group
 ├─442 /usr/sbin/smbd --foreground --no-process-group
 └─443 /usr/sbin/smbd --foreground --no-process-group
usuario@debian:~$
```

Veja que o serviço está ativo e rodando. O próximo passo é criar um diretório único para todos poderem guardar arquivos lá dentro, será um diretório público igual ao que existe na Fatec Zona Leste onde os alunos acessam de forma promíscua. Vamos criar o diretório `/home/publico` conforme figura abaixo.

```
usuario@debian:~$ sudo mkdir /home/publico
[sudo] password for usuario:
usuario@debian:~$ chgrp users /home/publico
chgrp: changing group of '/home/publico': Operation not permitted
usuario@debian:~$ sudo chgrp users /home/publico
usuario@debian:~$
```

Mas por ter sido criado com sudo o diretório terá como dono o root, queremos que todos os usuários que pertençam ao grupo `users` também tenham acesso, para isso vamos alterar o grupo do diretório com o comando `chgrp` conforme figura acima. (na figura acima, paguei o mico de esquecer o sudo, não precisa errar necessariamente)

Agora vamos dar as permissões de acesso para os usuários, o dono root terá acesso total ao diretório, podendo ler, escrever e listar para isso usei o 7, mesmo digo para o grupo `users`. Já para os outros, estes podem apenas listar e ler que é o 5.

```
usuario@debian:~$ sudo chmod 775 /home/publico/
usuario@debian:~$ _
```

Agora vamos voltar ao arquivo de configuração samba. Vamos então editar.

```
usuario@debian:~$
usuario@debian:~$ sudo nano /etc/samba/smb.conf _
```

Nosso objetivo agora é permitir que todos os usuários do grupo **users** possam acessar o diretório **/home/publico** e também os seus diretórios particulares (**~**).

Vamos começar comentando (caso não esteja) o parâmetro **read only**, ou seja, vamos permitir que se a regra de usuário for válida então ele possa escrever arquivos. Veja que foi usado **;** como comentário.

```
By default, the home
next parameter to 'n
; read only = yes
```

Agora vamos permitir que os usuários que estão no grupo **users** possam criar/escrever diretórios e arquivos, conforme figura abaixo.

```
This might need tweaking wh
valid users = %S
writable = yes

Un-comment the following an
```

Isso garante que cada usuário que pertença ao grupo **users** possam pela rede acessar seu diretório neste máquina pela rede, mas não permite que acessem o diretório público criado, então vamos configurar o arquivo para permitir isso.

Navegue até o fim do arquivo e digite:

```
[publico]
comment = Diretório dos alunos
path = /home/publico
valid users =@users
force group = users
create mask = 0660
directory mask = 0771
writable = yes
```

A propriedade **valid users** então aponta para o grupo de usuários da máquina, ele com **force group** garante que a política de acesso será feita então pelo grupo **users**.

Já as propriedades **create mask** e **directory mask** são utilizados pelo samba para quando o usuário criar um arquivo ou diretório (respectivamente) crie com tais permissões de acesso.

Agora vamos associar os usuários no grupo users, veja que o usuário da máquina chamado usuário ele não pertence ao grupo users.

```
usuario@debian:~$ groups usuario
usuario : usuario cdrom floppy audio dip video plugdev netdev bluetooth
usuario@debian:~$ _
```

Vamos adicionar então, para isso vamos usar o comando usermod conforme figura abaixo.

**ATENÇÃO:** Alguns Linux não estão configurados para /usr/sbin/ ser um diretório de binários executáveis, por isso o caminho completo.

```
usuario@debian:~$ sudo /usr/sbin/usermod -g users usuario
usuario@debian:~$
```

Agora veja que **usuario** já pertence ao grupo **users**.

```
usuario@debian:~$ groups usuario
usuario : users cdrom floppy audio dip video plugdev netdev bluetooth
usuario@debian:~$
```

Agora só falta informar para o Samba que o determinado usuário possui uma senha de acesso pela rede, para isso o comando smbpasswd, conforme figura abaixo.

```
usuario@debian:~$ sudo smbpasswd -a usuario_
```

Informe a senha, recomendo que seja uma senha diferente da que ele utiliza na máquina por questão de segurança.

```
usuario@debian:~$ sudo smbpasswd -a usuario
New SMB password:
```

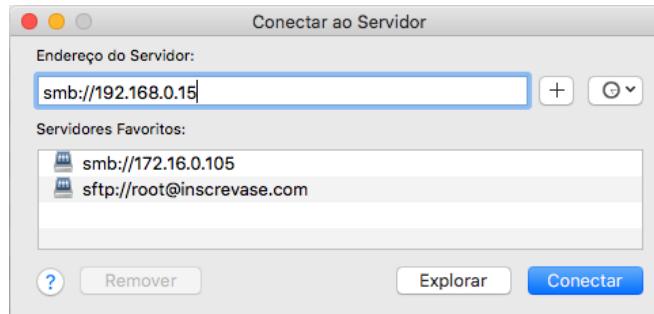
Ok, agora comando **reboot** ou **systemctl reboot**

## 23.3 Acessando o recurso

O protocolo smb de transferência de blocos de mensagem, utilizado em larga escala para transferência de arquivos na rede, por este motivo o protocolo smb é reconhecido por todos os sistemas operacionais, dando a este uma ampla penetração nos projetos de rede.

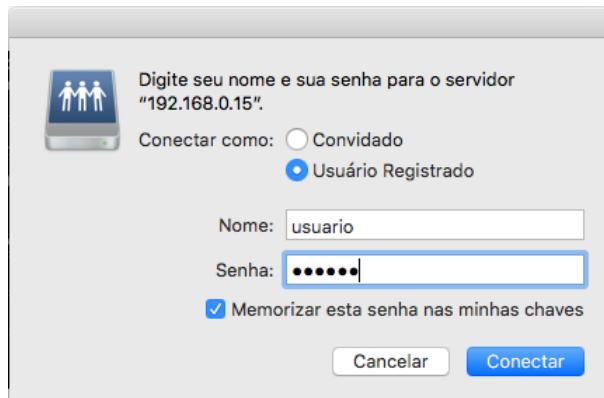
### 23.3.1 Acessando com o cliente gráfico no MAC OS

Para acessar o diretório SMB na rede a partir de um MAC OS basta abrir o gerenciador de arquivos e ir em “Conectar ao servidor” conforme figura abaixo.

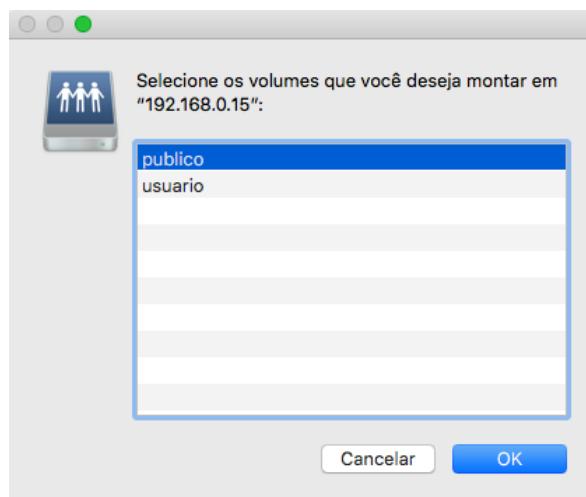


O Mac OS pergunta qual o usuário e senha, veja que é informando **usuario** e a senha **123456**, pois foi o configurado no **smbpasswd**.

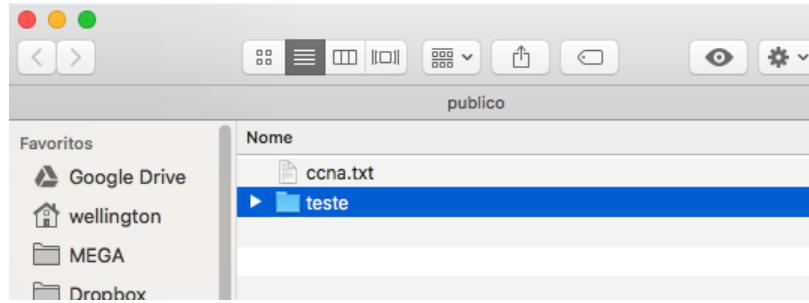
**ATENÇÃO:** Se tiver acertando usuário e senha e mesmo assim não permite acesso, então utilize **NOME\_DA\_MAQUINA\_LINUX/USUARIO\_CADASTRADO\_NO\_GRUPO\_USERS**



Veja que é possível listar o público que todos têm acesso e o diretório do usuário.

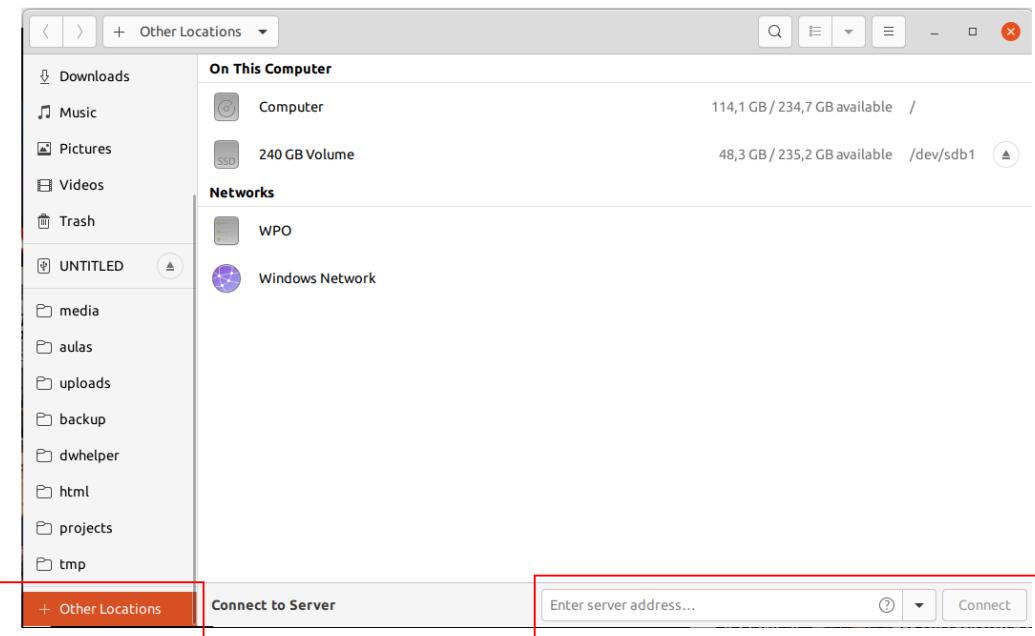


Pronto, para criar um arquivo ou diretório basta usar os recursos de seu Explorer de arquivo.



### 23.3.2 Acessando com o cliente gráfico no Ubuntu GNU/Linux

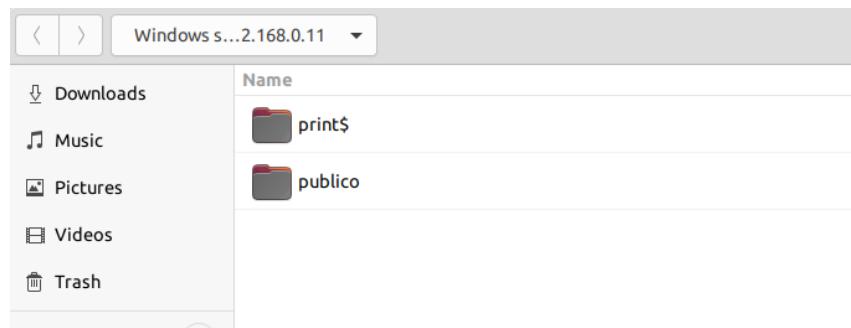
O Ubuntu possui no File Explorer a opção de montar diretórios em rede, para isso na opção “Other Locations” deve-se informar o IP do servidor Samba.



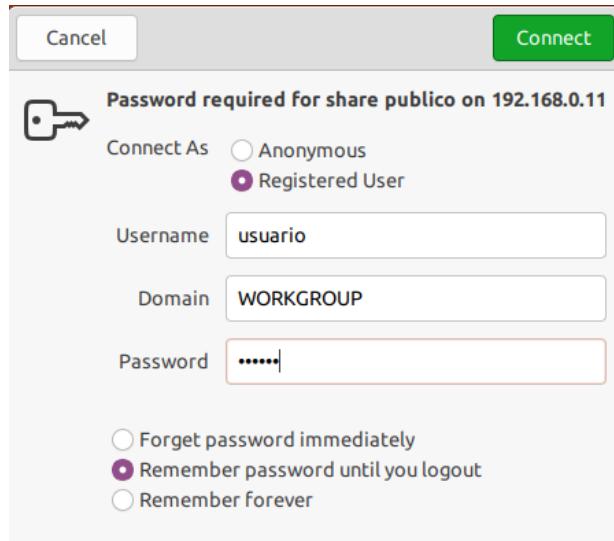
Deve-se utilizar o protocolo smb:// e o IP do servidor Samba.



Os diretórios serão listados, ao entrar no diretório público a solicitação de usuário e senha será obrigatória.



Informe o usuário cadastrado no **smbpasswd**.



Pronto, seu diretório na rede está pronto para o uso.

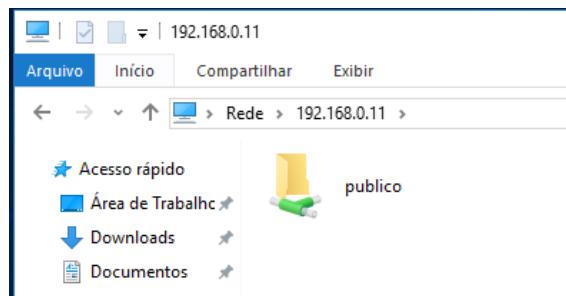
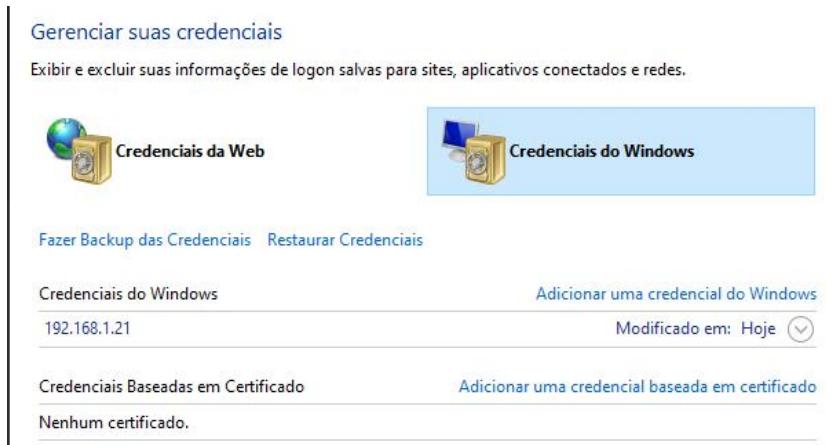
### 23.3.3 Acessando com o cliente gráfico no Windows

Já no Windows, abra o Windows Explorer e digite `\192.168.0.11` (isso porque o ip da máquina virtual é este)

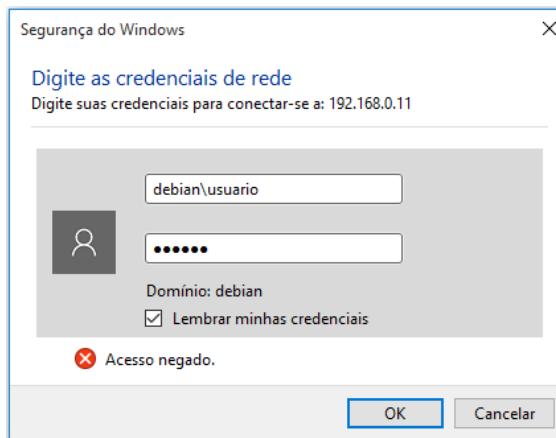
Windows tem que ativar descoberta de windows

**Atenção Windows 10:**

<https://techjourney.net/cannot-connect-to-cifs-smb-samba-network-shares-shared-folders-in-windows-10/>



Como a rede não possui um controlador de domínio, o Windows exige que se utilize o nome da máquina e o nome do usuário conforme figura abaixo.



Ao clicar em Ok, está pronto para o uso.

## 23.4 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

### 23.4.1 Prática 927a3ik 01: Configurando o serviço SMB

Nesta prática o aluno deve ter a expertise para configurar o servidor Samba.

1. Execute a configuração seguindo o tópico 23.2;

Execute o comando aied conforme comando listagem abaixo.

1. sudo aied validar 927a3ik checkpoint01

Para validar o teste 5.1 o aluno deve entrar no diretório teste em público e manter lá, aí então realiza o teste lá na VM do Samba.

```
usuario@debian:~$ sudo aied validar 927a3ik checkpoint01
Ação que será executada: validar

Prática: Prática da aula de Samba - Configurando o Samba
Será enviado o OUTPUT do comando: groups usuario
Será enviado o OUTPUT do comando: cat /etc/samba/smb.conf
Será enviado o OUTPUT do comando: ls -l /home/
Será enviado o OUTPUT do comando: systemctl status smbd.service --no-pager

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: groups usuario
1.1 Validar por text users

2 - Comando: cat /etc/samba/smb.conf
2.1 Validar por regex / s*security/
2.2 Validar por regex / ; s*read s*only/

3 - Comando: ls -l /home/
3.1 Validar por regex /root(.*)users(.*)publico/

4 - Comando: systemctl status smbd.service --no-pager
4.1 Validar por text active (running)
4.2 Validar por text enabled;

Total de acertos: 6 do total de 6 validações equivalente a 100%.
Identificação: 3d41f9cb85
AIED v(7)
```

### 23.4.2 Prática 927a3ik 02: Criando um diretório e acessando

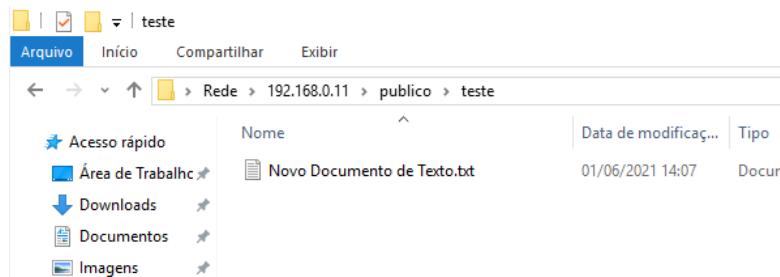
Nesta prática o aluno deve ter a expertise para acessar pela rede o serviço Samba

1. Crie um novo diretório utilizando um cliente (Windows, Linux ou MAC), este novo diretório dentro de público deve se chamar “teste”;
2. Pela interface cliente, entre no diretório e crie qualquer arquivo, se mantenha neste diretório para realizar o teste aied abaixo;

Execute o comando aied conforme comando listagem abaixo.

1. sudo aied validar 927a3ik checkpoint02

Para validar o teste 1.1 o aluno deve entrar no diretório teste em público e manter lá, aí então realizar o teste lá na VM do Samba.



Veja abaixo uma saída esperada.

```
usuario@debian:~$ sudo aiед validar 927a3ik checkpoint02
Ação que será executada: validar

Prática: Prática da aula de Samba - Acessando o serviço Samba
Será enviado o OUTPUT do comando: smbstatus

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO ++++++
1 - Comando: smbstatus
1.1 Validar por regex / /home /publico(.*)teste/

Total de acertos: 1 do total de 1 validações equivalente a 100%.
Identificação: 3d41f9cb85
AIED v(7)
```

falar de rpc e como desativar

## 24 Proxy Squid-cache

O Squid (acessível pela url <http://www.squid-cache.org/>) é um poderoso servidor proxy utilizado para controlar o acesso à Internet a partir da rede local mantendo os interesses da organização, tais interesses são:

- Cache;
- Controle sobre o tráfego;
- Fazer proxy reverso (mas neste material o proxy reverso será feito em NGINX);

Esta solução é posicionada entre o cliente Browser e o serviço que geralmente é HTTP, HTTPS ou FTP, e estes serviços estão quase em toda sua totalidade na rede mundial.

Além de servir como um servidor proxy, o Squid é usado principalmente para armazenar em cache páginas da web visitadas com frequência de um servidor da web. Portanto, quando um usuário solicita uma página de um servidor da web, as solicitações passam primeiro pelo servidor proxy para verificar se o conteúdo solicitado está disponível. Isso reduz a carga do servidor e o uso de largura de banda e acelera a entrega de conteúdo, melhorando assim a experiência do usuário.

O Squid também pode ser usado para se tornar anônimo ao navegar na Internet. Por meio do proxy Squid, podemos acessar o conteúdo restrito de um determinado país.

colocar vantagens

### 24.1 Recursos do capítulo

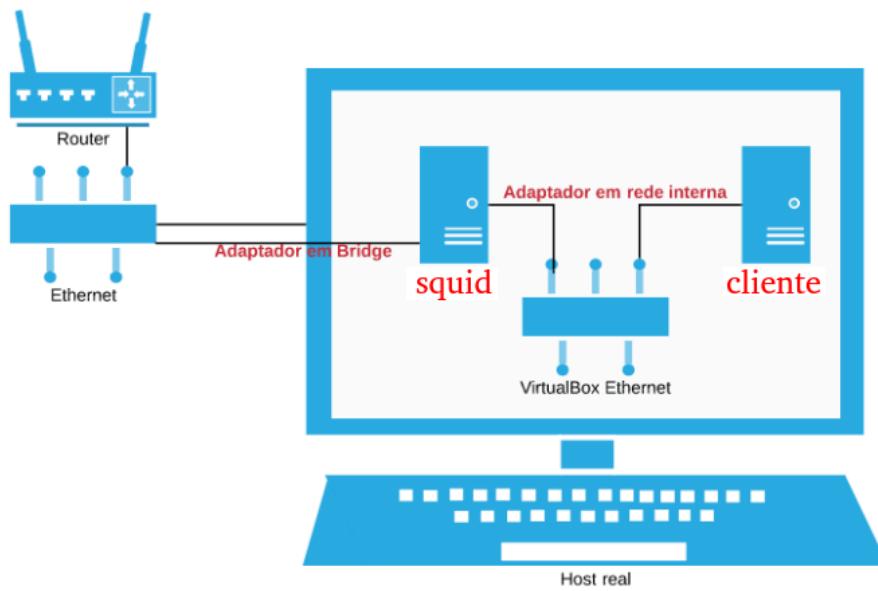
Todo o conteúdo externo ao texto pode ser obtido no link abaixo.



O material deste capítulo pode ser [obtido neste link](#).

### 24.2 Cenário

Conforme já demonstrado o serviço Squid está entre o Browser cliente e o serviço HTTP (vamos assumir que estamos só trabalhando com HTTP/HTTPS), logo será preciso 2 máquinas virtuais, uma somente terminal que será chamado de **squid** e outra gráfica que será chamada de **cliente**.

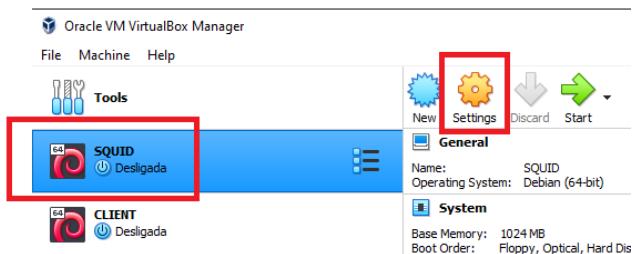


A equipe de TI deverá escolher uma das três possíveis formas de abordar os clientes, são:

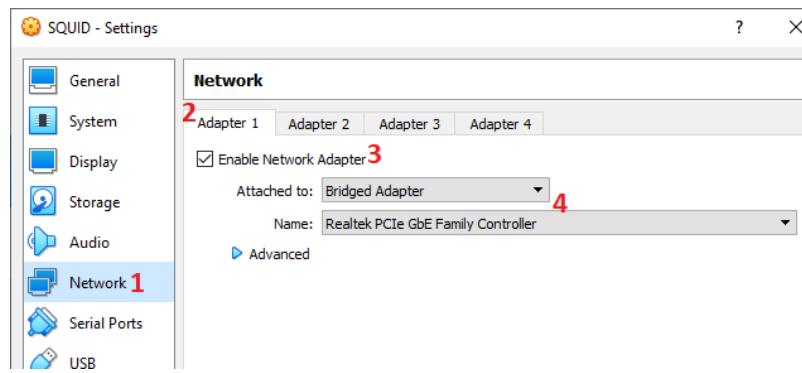
- **ABORDAGEM 1:** Em cada cliente é manualmente configurado o serviço Proxy;
- **ABORDAGEM 2:** As informações de proxy serão passadas automaticamente por DHCPack;
- **ABORDAGEM 3:** Configuração de proxy transparente para serviço http;

Para as três abordagens será necessário uma máquina terminal com Squid, esta deverá ter 2 adaptadores de rede, o primeiro adaptador em modo Bridge (ver capítulo 1 caso tenha dúvidas) e o segundo adaptador em modo Rede Interna.

A máquina cliente gráfica ou terminal deverá ter apenas um Adaptador de rede e ele deve estar configurado em modo Rede Interna, e na mesma rede virtual do squid. Comece criando uma nova máquina virtual terminal, será a nossa máquina SQUID.



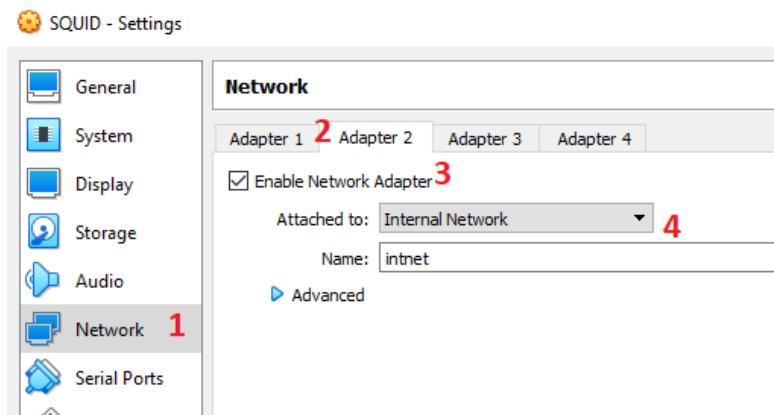
Acesse as configurações desta máquina virtual, conforme figura acima, veja que nas opções vamos precisar acessar a aba Network.



Onde:

1. Selecione a aba Network;
2. Selecione a aba Adapter 1;
3. Habilite o Adapter 1, ou seja, conecte no cabo virtual;
4. Selecione Bridged Adapter como opção de conexão e selecione a placa de rede da qual seu computador acessa a rede local física;

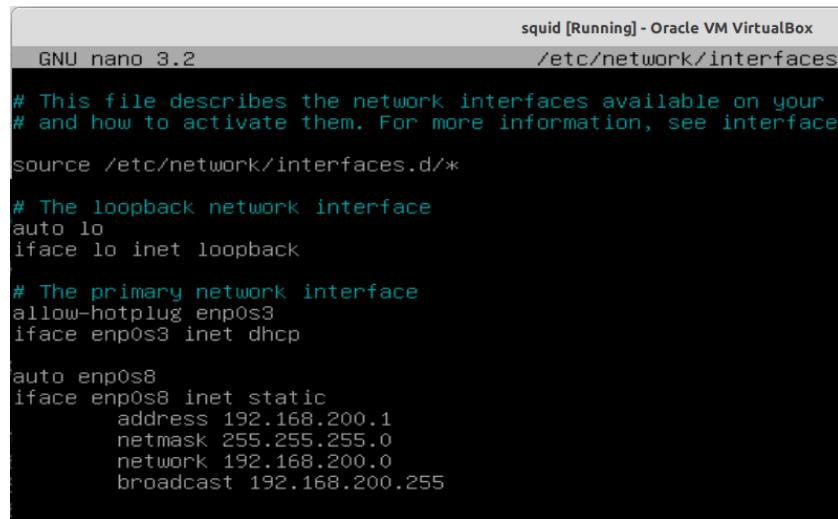
Na segunda interface de rede do SQUID, coloque esta placa em modo Rede Interna, é por essa interface que os clientes vão acessar a Internet.



Onde:

1. Selecione a aba Network;
2. Abra a aba Adapter 2;
3. Habilite a placa de rede, conectando ela na rede virtual;
4. Selecione a opção **Internal Network**.

Inicie a máquina virtual SQUID e edite o arquivo `/etc/network/interfaces`, o objetivo é realizar a configuração lógica da rede, conforme figura abaixo.



```

squid [Running] - Oracle VM VirtualBox
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

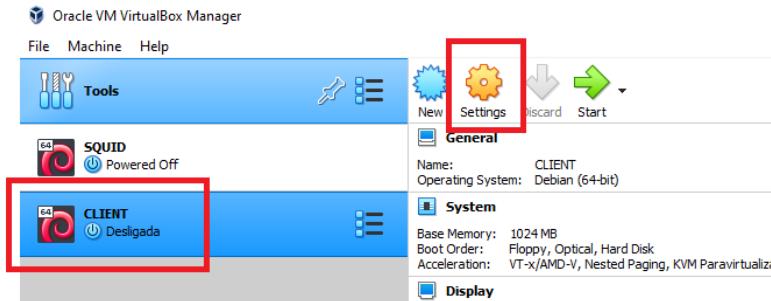
The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp

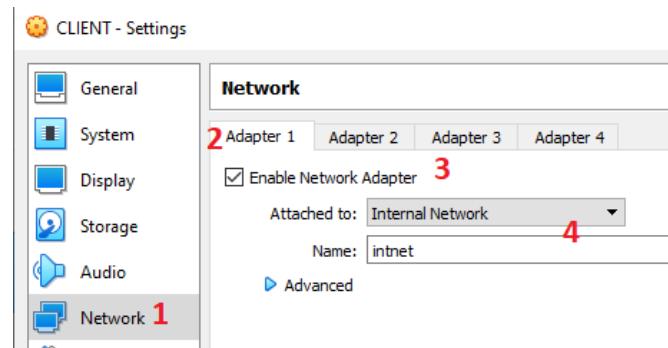
auto enp0s8
iface enp0s8 inet static
 address 192.168.200.1
 netmask 255.255.255.0
 network 192.168.200.0
 broadcast 192.168.200.255

```

Já a máquina virtual CLIENT terá apenas 1 interface de rede e esta interface deverá ser ligada na Internal Network.



Veja na figura abaixo.



Onde:

5. Selecione a aba Network;
6. Abra a aba Adapter 1;
7. Habilite a placa de rede, conectando ela na rede virtual;
8. Selecione a opção **Internal Network**.

Inicie a máquina CLIENT e realize a configuração lógica conforme figura abaixo. Vamos utilizar a Abordagem 2 na prática, então o IP será obtido por meio de DHCP Server.

```
CLIENT [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/network/interfaces
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp
```

Reinic peace tanto o CLIENT quanto o SQUID.

### 24.3 Instalando o squid

O processo de instalação é simples, mas lembre-se de executar o update antes de qualquer coisa.

1. sudo apt update -y
2. sudo apt install squid -y

A versão do material é a versão 5.7 do squid, conforme figura abaixo. Cada versão do SQUID o arquivo de configuração é diferente.

```
SQUID [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/squid/squid.conf
WELCOME TO SQUID 5.7
#
This is the documentation for the Squid configuration file
This documentation can also be found online at:
http://www.squid-cache.org/Doc/config/
#
You may wish to look at the Squid home page and wiki for t
FAQ and other documentation:
```

Como o arquivo é grande, recomendo que faça uma cópia, conforme comando da listagem abaixo.

1. sudo cp /etc/squid/squid.conf /etc/squid/squid.conf.old

Caso cometa erros, ou é possível o retorno da versão original ou com o comando **diff** você pode localizar seus erros.

### 24.4 Configurando a permissão de acesso ao Proxy

O processo de instalação não auto-configura, então é preciso agora que seja feita a configuração de abrangência do serviço proxy, essa abrangência então deve ser aplicada

na sua política de permissão de acesso ao serviço proxy, na prática temos 3 possibilidades amplamente utilizada:

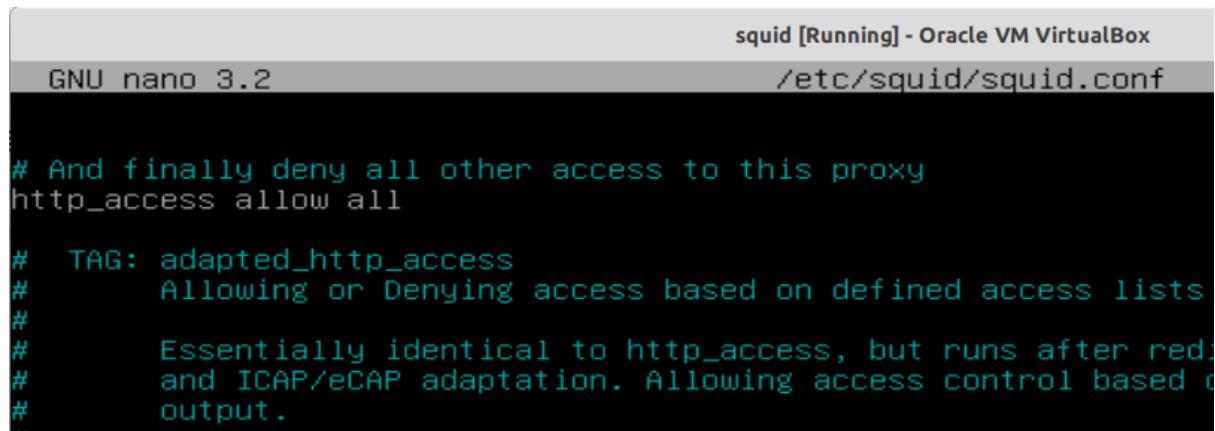
- todos que tiverem acesso ao servidor podem usar o proxy;
- permissão para uma rede específica;
- permissão para um ou mais máquinas específicas;

#### 24.4.1 Permissão para todos que podem acessar o servidor

Primeiro passo é permitir acesso ao serviço de proxy HTTP, digamos que qualquer pessoa possa ter acesso ao proxy, então no arquivo /etc/squid/squid.conf localize uma linha que contém a string “**http\_access deny all**” e mude para “**http\_access allow all**”, também caso a linha esteja comentada, remova o símbolo de # do início da linha.

1. sudo nano /etc/squid/squid.conf

**Atenção:** no arquivo do squid 5.7 a tag especificada acima está na linha 1555.



```
GNU nano 3.2 squid [Running] - Oracle VM VirtualBox
/etc/squid/squid.conf

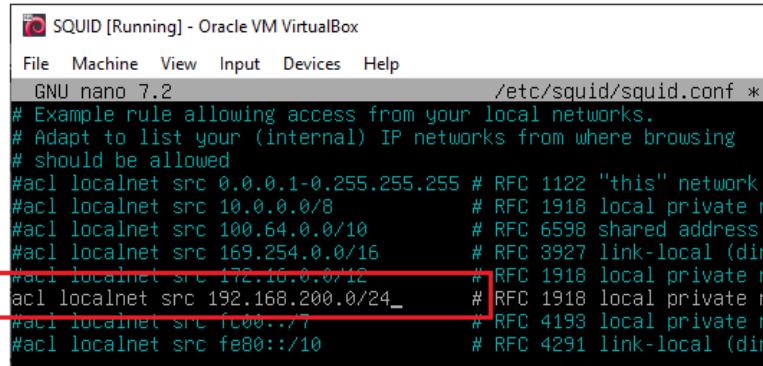
And finally deny all other access to this proxy
http_access allow all

TAG: adapted_http_access
Allowing or Denying access based on defined access lists
#
Essentially identical to http_access, but runs after redi-
and ICAP/eCAP adaptation. Allowing access control based on
output.
```

#### 24.4.2 Permitindo acesso apenas para uma determinada rede/subrede

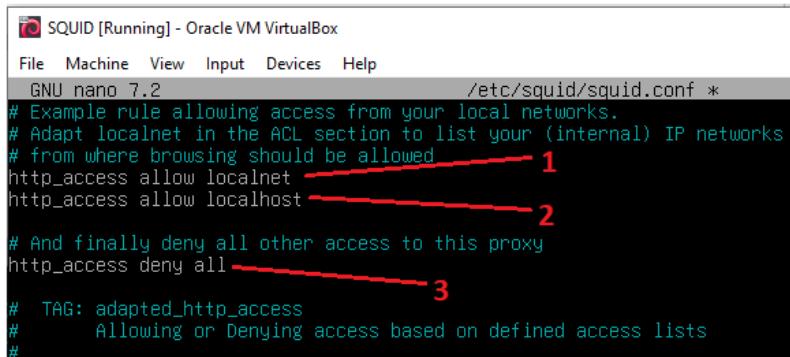
Permitiremos apenas que o host local e os dispositivos de nossa rede local (LAN) usem o Squid para um controle mais preciso, para isso deve-se retirar o comentário em “**http\_access allow localnet**” e garantir que tanto localnet e localhost tenham acesso conforme figura abaixo.

No primeiro passo devemos definir quais computadores vão fazer parte da acl, para isso navegue até a linha 1338 do arquivo, e edite a acl para a rede específica do exercício que é a rede 192.168.200.0/24, conforme figura abaixo.



```
GNU nano 7.2 /etc/squid/squid.conf *
Example rule allowing access from your local networks.
Adapt to list your (internal) IP networks from where browsing
should be allowed
#acl localnet src 0.0.0.1-0.255.255.255 # RFC 1122 "this" network
#acl localnet src 10.0.0.0/8 # RFC 1918 local private network
#acl localnet src 100.64.0.0/10 # RFC 6598 shared address
#acl localnet src 169.254.0.0/16 # RFC 3927 link-local (dir
#acl localnet src 172.16.0.0/12 # RFC 1918 local private network
acl localnet src 192.168.200.0/24 # RFC 1918 local private network
#acl localnet src fc00::/7 # RFC 4193 local private network
#acl localnet src fe80::/10 # RFC 4291 link-local (dir
```

Comente todas as demais regras, afinal não queremos um squid tão permissivo. O segundo passo é liberar a **acl** chamada **localnet**, para isso navegue até a linha 1551.



```
GNU nano 7.2 /etc/squid/squid.conf *
Example rule allowing access from your local networks.
Adapt localnet in the ACL section to list your (internal) IP networks
from where browsing should be allowed 1
http_access allow localnet
http_access allow localhost 2
And finally deny all other access to this proxy
http_access deny all 3
TAG: adapted_http_access
Allowing or Denying access based on defined access lists
```

Onde:

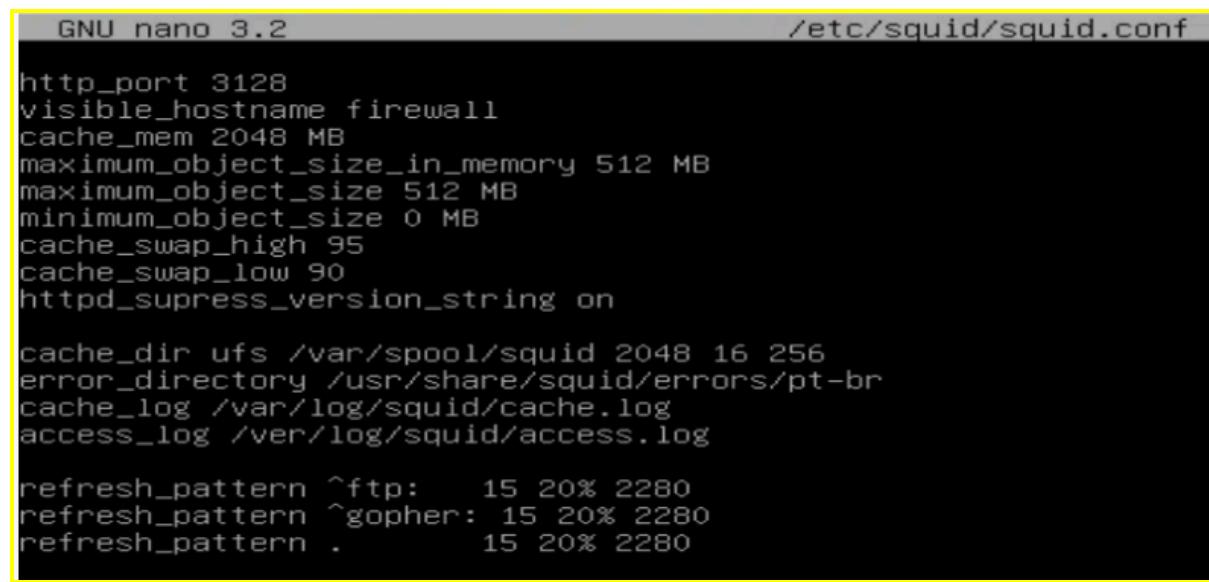
1. Permitindo que qualquer computador em localnet acesse o squid;
2. Permitindo que a máquina local acesse o squid;
3. Negando todo o resto.

As regras serão executadas na sequência, então evita-se distribuir estas regras ao longo do arquivo e por isso navegamos até a linha 1551, e será executada no formato top-down onde encontrou a regra, executa-se e para.

## 24.5 Ativando cache

Na linha 3635. nao fazer ainda, estou trabalhando aqui....

```
#Default:
No disk cache. Store cache objects only in memory.
#
Uncomment and adjust the following to add a disk cache directory.
cache_dir ufs /var/spool/squid 100 16 256
#
TAG: store_dir_select_algorithm
How Squid selects which cache_dir to use when the response
object will fit into more than one
```



```
GNU nano 3.2 /etc/squid/squid.conf

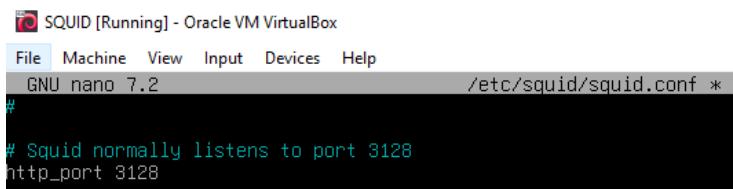
http_port 3128
visible_hostname firewall
cache_mem 2048 MB
maximum_object_size_in_memory 512 MB
maximum_object_size 512 MB
minimum_object_size 0 MB
cache_swap_high 95
cache_swap_low 90
httpd_supress_version_string on

cache_dir ufs /var/spool/squid 2048 16 256
error_directory /usr/share/squid/errors/pt-br
cache_log /var/log/squid/cache.log
access_log /var/log/squid/access.log

refresh_pattern ^ftp: 15 20% 2280
refresh_pattern ^gopher: 15 20% 2280
refresh_pattern . 15 20% 2280
```

## 24.6 Alterando a porta serviço proxy

Caso seja necessário alterar a porta (caso raro) é possível alterar o valor de `http_port` que contém o número da porta para servidores proxy Squid. O número da porta padrão é **3218**. Essa configuração fica aproximadamente na linha 2006 do arquivo, conforme figura abaixo.



```
GNU nano 7.2 /etc/squid/squid.conf *
#
Squid normally listens to port 3128
http_port 3128
```

## 24.7 Habilitando comunicação por portas para clientes

As configurações de portas permitidas estão no próprio arquivo, logo abaixo da definição de `localnet` (aproximadamente linha 1342), trata-se de um conjunto de regras ACL com as portas, conforme figura abaixo.

```

SQUID [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/squid/squid.conf

acl SSL_ports port 443
acl Safe_ports port 80 # http
acl Safe_ports port 21 # ftp
acl Safe_ports port 443 # https
acl Safe_ports port 70 # gopher
acl Safe_ports port 210 # wais
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http

```

Caso queira uma configuração mais restritiva, comente as linhas fechando portas.

## 24.8 Controles de acesso no Squid com ACL

Uma tarefa muito comum é garantir que as regras organizacionais sejam aplicadas, para isso deve-se aplicar regras de segurança e bem estar na rede, digamos que em um local de trabalho é inapropriado o uso de sites (não vou escrever a palavra para não ser bloqueado), já sabem.

Ao carregar o arquivo de configuração, o Squid processa todas as linhas **acl** (diretivas) na memória como testes que podem ser executados em qualquer transação de solicitação. Os tipos de testes são baseados no que se quer acessar.

O Squid conhece os seguintes tipos de elementos ACL, segue os principais:

- src:** endereços IP de origem (cliente);
- dst:** endereços IP de destino (servidor);
- myip:** o endereço IP local da conexão de um cliente;
- arp:** correspondência de endereço Ethernet (MAC);
- srcdomain:** nome de domínio de origem (cliente);
- dstdomain:** nome de domínio de destino (servidor);
- port:** número da porta de destino (servidor);
- method:** método de solicitação HTTP (get, post, etc);
- http\_status:** status da resposta HTTP (200 302 404 etc.).

Existem várias listas de acesso diferentes, as principais são:

**http\_access:** Permite que clientes HTTP (navegadores) accessem a porta HTTP. Esta é a lista de controle de acesso principal;

**http\_reply\_access:** Permite que clientes HTTP (navegadores) recebam a resposta à sua solicitação. Isso restringe ainda mais as permissões fornecidas por **http\_access** e destina-se principalmente ao uso em conjunto com **rep\_mime\_type acl** para bloquear diferentes tipos de conteúdo;

**cache:** Define respostas que não devem ser armazenadas em cache.

**url\_rewrite\_access:** controla quais solicitações são enviadas por meio do pool de redirecionadores;

**snmp\_access:** Controla o acesso do cliente SNMP ao cache;

**htcp\_access**: Controla quais máquinas remotas são capazes de fazer solicitações HTCP;

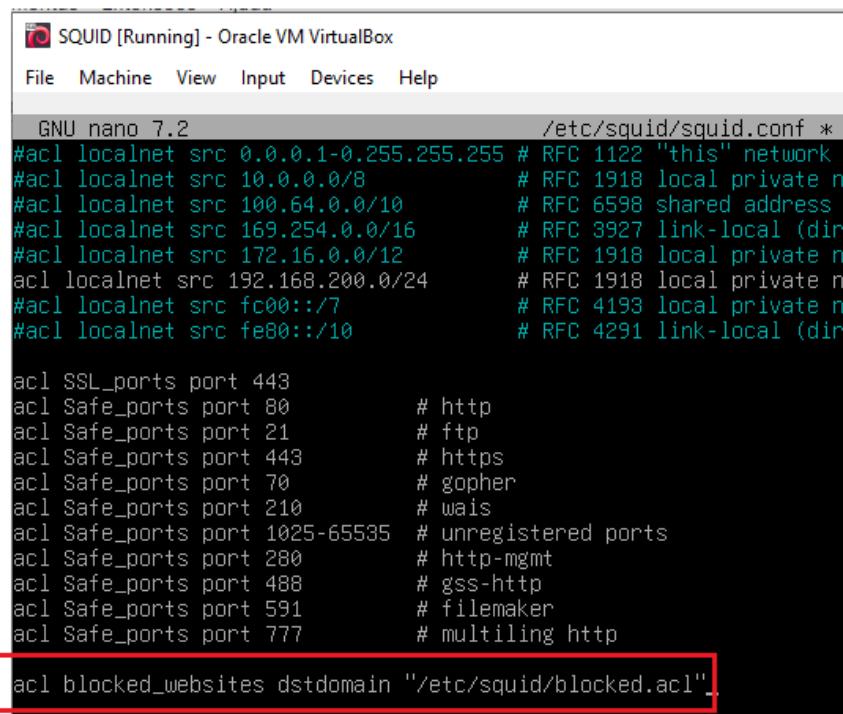
**htcp\_clr\_access**: Controla quais máquinas remotas são capazes de fazer solicitações HTCP CLR;

**request\_header\_access**: controla quais cabeçalhos de solicitação são removidos ao violar o protocolo HTTP;

**respond\_header\_access**: controla quais cabeçalhos de resposta são removidos da entrega ao cliente ao violar o protocolo HTTP.

#### 24.8.1 Bloqueando sites

Uma ação muito requisitada é o bloqueio de sites a partir de uma lista privada de domínios, para isso devemos começar criando uma acl, no exemplo da figura abaixo a acl se chamará **blocked\_websites**, você pode usar o nome que quiser para suas acls, mas para esta prática vamos padronizar com este nome, para isso navegue até a linha 1354 (aproximadamente) e edite a nova acl conforme figura abaixo.



```

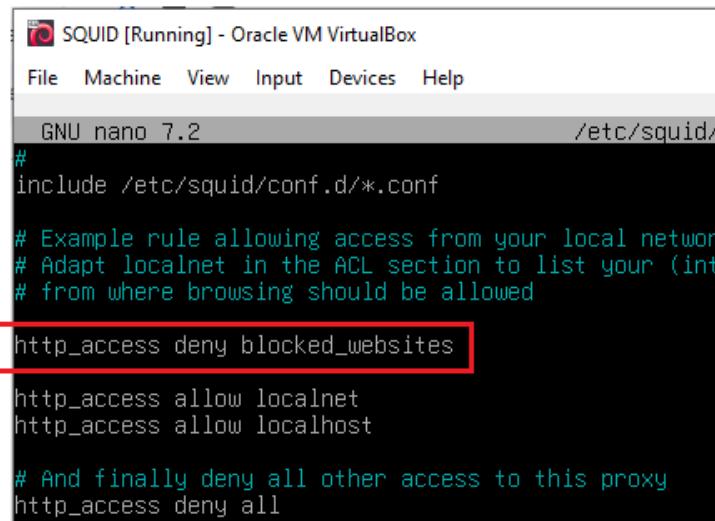
GNU nano 7.2 /etc/squid/squid.conf *
#acl localnet src 0.0.0.1-0.255.255.255 # RFC 1122 "this" network (local)
#acl localnet src 10.0.0.0/8 # RFC 1918 local private network
#acl localnet src 100.64.0.0/10 # RFC 6598 shared address space
#acl localnet src 169.254.0.0/16 # RFC 3927 link-local (directly
#acl localnet src 172.16.0.0/12 # RFC 1918 local private network
acl localnet src 192.168.200.0/24 # RFC 1918 local private network
#acl localnet src fc00::/7 # RFC 4193 local private network
#acl localnet src fe80::/10 # RFC 4291 link-local (directly

acl SSL_ports port 443
acl Safe_ports port 80 # http
acl Safe_ports port 21 # ftp
acl Safe_ports port 443 # https
acl Safe_ports port 70 # gopher
acl Safe_ports port 210 # wais
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http

acl blocked_websites dstdomain "/etc/squid/blocked.ac1"

```

Navegue até as regras, aproximadamente linha 1551 e utilize a regra deny para a acl **blocked\_websites**, conforme figura abaixo. Declaramos em dois locais pois deve-se manter a organização, sempre as regras allow e deny próximas para ficar fácil localizar a sequencia de decisões.



```
GNU nano 7.2 /etc/squid/
#
include /etc/squid/conf.d/*.conf

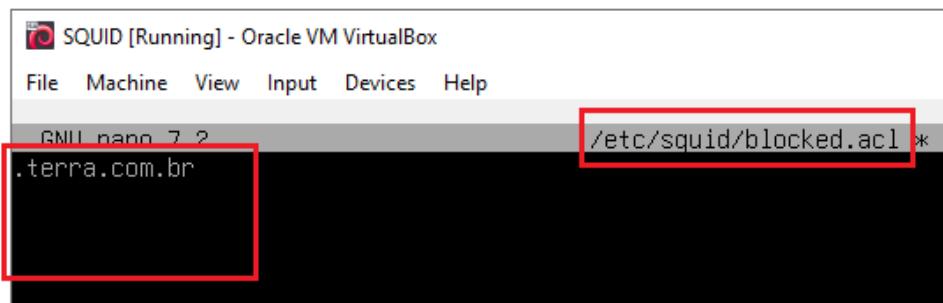
Example rule allowing access from your local network
Adapt localnet in the ACL section to list your (int
from where browsing should be allowed

http_access deny blocked_websites

http_access allow localnet
http_access allow localhost

And finally deny all other access to this proxy
http_access deny all
```

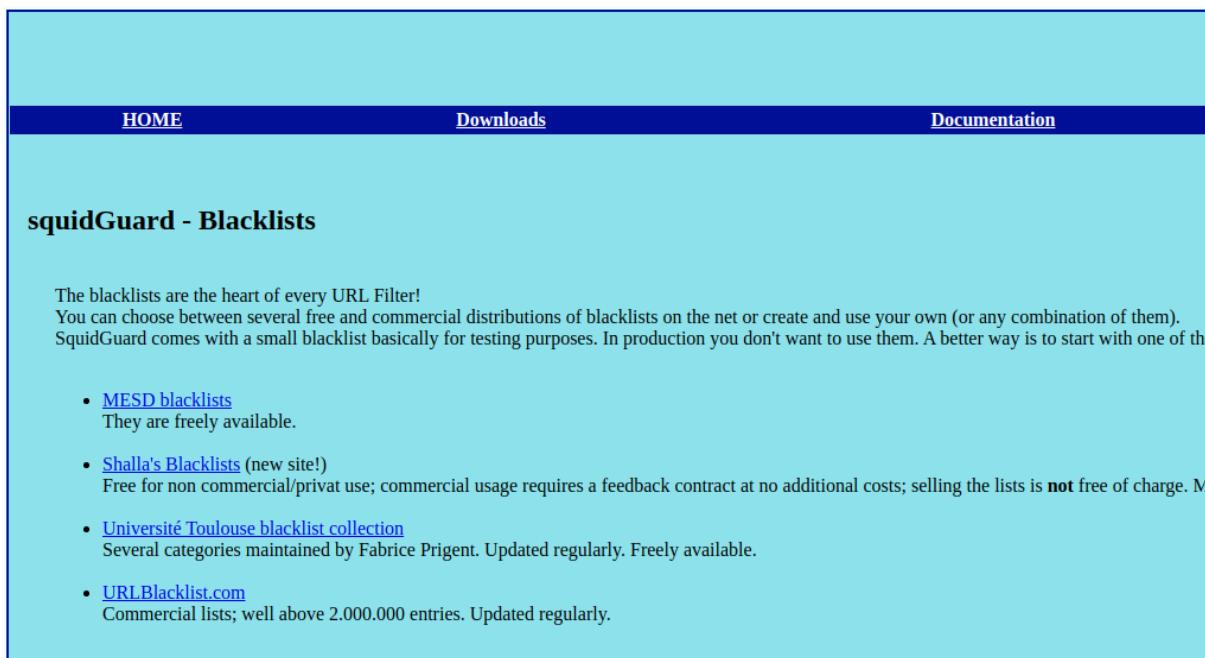
No próximo passo, será criada a lista que não existe ainda, para isso com o nano crie o novo arquivo /etc/squid/blocked.acl e adicione o domínio .terra.com.br conforme figura abaixo. Estou usando este domínio apenas para fins de demonstração e por acreditar que este domínio “**tem uma vida longa e próspera**”.



```
GNU nano 7.2 /etc/squid/blocked.acl *
.terra.com.br
```

Também é muito comum se aplicar regras e listas já desenvolvidas por comunidades, isso pois as comunidades possuem uma força de trabalho muito superior à de uma organização, na figura abaixo temos o exemplo do site SquidGuard<sup>93</sup>.

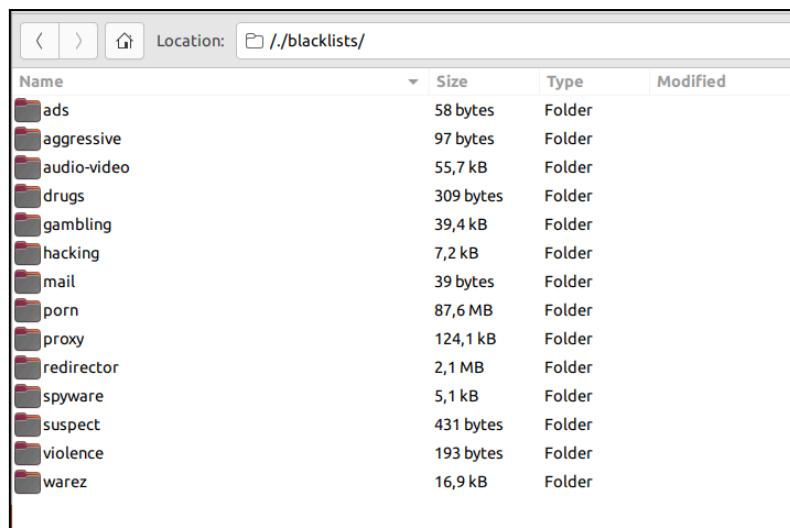
<sup>93</sup> Acessível pela url: <http://www.squidguard.org/blacklists.html>



The screenshot shows a web page with a dark blue header bar containing three white text links: "HOME", "Downloads", and "Documentation". Below the header, the page title "squidGuard - Blacklists" is displayed in a bold, dark font. A text block follows, stating: "The blacklists are the heart of every URL Filter! You can choose between several free and commercial distributions of blacklists on the net or create and use your own (or any combination of them). SquidGuard comes with a small blacklist basically for testing purposes. In production you don't want to use them. A better way is to start with one of the following blacklists." Below this text is a bulleted list of four blacklists:

- [MESD blacklists](#)  
They are freely available.
- [Shalla's Blacklists](#) (new site!)  
Free for non commercial/privat use; commercial usage requires a feedback contract at no additional costs; selling the lists is **not** free of charge. M
- [Université Toulouse blacklist collection](#)  
Several categories maintained by Fabrice Prigent. Updated regularly. Freely available.
- [URLBlacklist.com](#)  
Commercial lists; well above 2.000.000 entries. Updated regularly.

Sim, o site é feio que doi o olho, mas tem as melhores listas, conforme a imagem abaixo.



Name	Size	Type	Modified
ads	58 bytes	Folder	
aggressive	97 bytes	Folder	
audio-video	55,7 kB	Folder	
drugs	309 bytes	Folder	
gambling	39,4 kB	Folder	
hacking	7,2 kB	Folder	
mail	39 bytes	Folder	
porn	87,6 MB	Folder	
proxy	124,1 kB	Folder	
redirector	2,1 MB	Folder	
spyware	5,1 kB	Folder	
suspect	431 bytes	Folder	
violence	193 bytes	Folder	
warez	16,9 kB	Folder	

A maior das listas e a mais utilizada é a lista de sites p0rn0gr@PH1c05<sup>94</sup>, que está na pasta, para adicionar esta lista basta descompactar o arquivo e configurar conforme feito no exemplo deste tópico, é muito simples.

#### 24.8.2 Criando uma lista de sites permitidos

Da mesma forma, podemos criar um novo arquivo para armazenar os endereços IP de clientes permitidos que usarão o proxy Squid.

Agora especifique os endereços IP que deseja permitir e salve o arquivo. Agora crie uma nova linha acl no arquivo de configuração principal e permita o acesso ao acl usando a diretiva http\_access. Essas etapas são mostradas abaixo:

1. acl allowed\_websites dstdomain "/etc/squid/allow.acl"
2. http\_access allow allowed\_websites

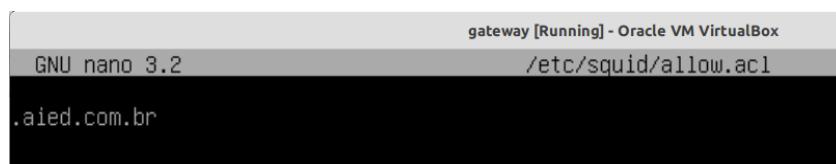


```
GNU nano 3.2 /etc/squid/squid.conf

acl blocked_websites dstdomain "/etc/squid/blocked.acl"
http_access deny blocked_websites

acl allowed_websites dstdomain "/etc/squid/allow.acl"
http_access allow allowed_websites
```

sudo nano /etc/squid/allow.acl



```
GNU nano 3.2 /etc/squid/allow.acl

.aied.com.br
```

<sup>94</sup> Ofuscado para que este material não seja banido;

## 24.9 Configurando Proxy + DHCP server

Para reduzir o esforço de configuração de estações clientes, a melhor abordagem é passar as configurações de proxy no DHCP Pack, para isso antes de tudo o administrador deve instalar dois programas extras no servidor SQUID, conforme listagem abaixo.

```
1. sudo apt update -y
2. sudo apt install apache2 -y
3. sudo apt install isc-dhcp-server -y
```

A ideia é no DHCPACK passar uma URL de um arquivo **.pac** que deverá estar dentro do apache, para reduzir o consumo de memória, nesta prática o aluno pode manter no mesmo servidor o serviço apache, o DHCP e o proxy.

Com o nano crie o seguinte arquivo **/var/www/html/proxy.pac**

```
1. function FindProxyForURL(url, host) {
2. if(isPlainHostName(host))
3. return "DIRECT";
4. else
5. return "PROXY 192.168.200.1:3128";
6. }
```

Execute a [prática de DHCP](#), e edite o arquivo **/etc/dhcp/dhcpd.conf** adicionando no final do arquivo a seguinte listagem:

```
1. subnet 192.168.200.0 netmask 255.255.255.0 {
2. range 192.168.200.10 192.168.200.200;
3. option domain-name-servers 8.8.8.8;
4. }
5.
6. option wpad-url code 252 = text;
7. option wpad-url "http://192.168.200.1/proxy.pac\n";
```

Agora deve-se informar que para IPV4 haverá uma configuração DHCP e que estará disponível na interface de rede `enp0s8`, edite o arquivo: `/etc/default/isc-dhcp-server`

```
1. # Defaults for isc-dhcp-server (sourced by /etc/init.d/isc-dhcp-server)
2.
3. # Path to dhcpcd's config file (default: /etc/dhcp/dhcpd.conf).
4. DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
5. #DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf
6.
7. # Path to dhcpcd's PID file (default: /var/run/dhcpcd.pid).
8. #DHCPDv4_PID=/var/run/dhcpcd.pid
9. #DHCPDv6_PID=/var/run/dhcpd6.pid
```

```

10.
11. # Additional options to start dhcpcd with.
12. # Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
13. #OPTIONS=""
14.
15. # On what interfaces should the DHCP server (dhcpcd) serve DHCP requests?
16. # Separate multiple interfaces with spaces, e.g. "eth0 eth1".
17. INTERFACESv4="enp0s8"
18. INTERFACESv6=""
19.

```

Agora deve-se reiniciar os 3 serviços.

```

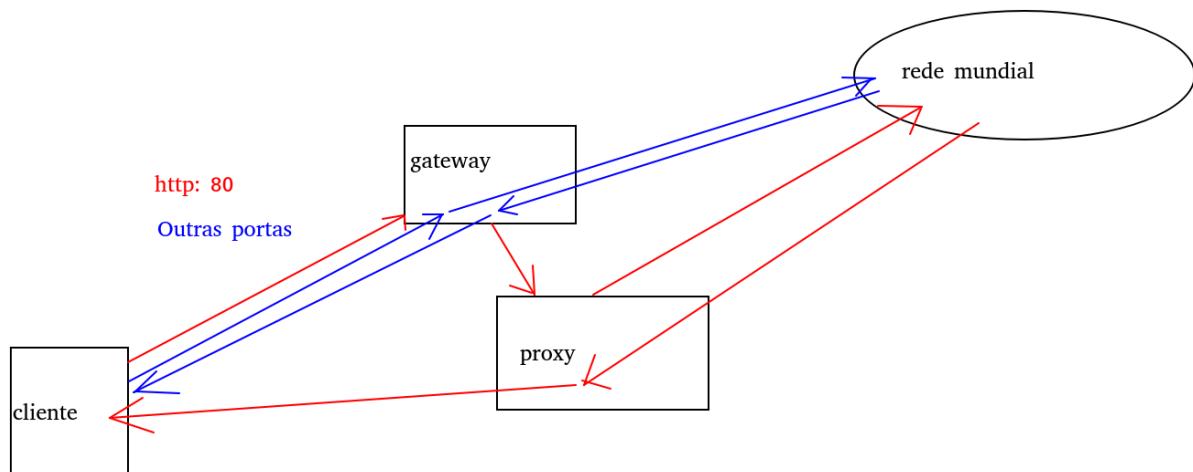
1. sudo systemctl restart apache2.service
2. sudo systemctl start isc-dhcp-server.service
3. sudo systemctl stop squid.service

```

Com esta configuração as estações clientes serão configuradas com auto dhcp e nenhuma configuração extra deverá ser feita pois o arquivo proxy.pac deverá trazer toda a configuração do servidor pelo DHCPACK.

## 24.10 Configurando Clientes por Proxy Transparente para porta 80

Proxy transparente não requer que os clientes sejam configurados, para o cliente existirá um gateway padrão e tudo deverá funcionar por este. A ideia é no servidor squid interceptar as requisições da porta 80 e redirecionar para o squid.



Para isso na configuração do servidor squid haverá uma mudança na configuração feita até este momento:

- Introdução do Gateway na configuração dos clientes;
- Ativar a diretiva net.ipv4.ip\_forward=1 no arquivo /etc/sysctl.conf;

- Criar 2 regras de IPTABLES;
- Adicionar http\_port <numero> transparent no arquivo /etc/squid/squid.conf;

No arquivo /etc/sysctl.conf ative a net.ipv4.ip\_forward=1 removendo o comentário desta linha.

```
GNU nano 3.2 /etc/sysctl.conf

Uncomment the next line to enable TCP/IP SYN cookies
See http://lwn.net/Articles/277146/
Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

Uncomment the next line to enable packet forwarding for IPv6
Enabling this option disables Stateless Address Autoconfiguration
```

Ainda no servidor, deve-se executar as duas regras de iptables na listagem, mas lembre-se que se reiniciar o servidor as regras serão perdidas, proceda conforme capítulo de Iptables para salvar.

1. sudo iptables -t nat -A PREROUTING -i enp0s8 -p tcp --dport 80 -j REDIRECT --to-ports 3129
2. sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

Agora deve-se habilitar a porta 3128 e adicionar uma outra linha informando que a porta 3128 também atuará no modo intercept, conforme imagem abaixo.

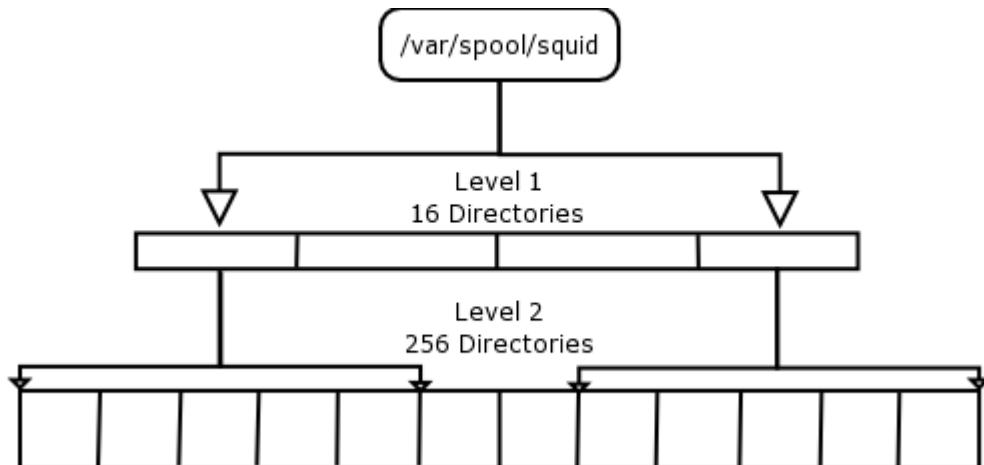
```
File Machine View Input Devices Help
GNU nano 3.2 /etc/squid/squid.conf

and an external interface we recommend you to specify the
internal address:port in http_port. This way Squid will only be
visible on the internal address.
#
#
Squid normally listens to port 3128
http_port 3128
http_port 3128 intercept
TAG: https_port
Usage: [in:lnopt [model] tls-cert=certificate.pem [options]]
```

Nesta abordagem somente o tráfego da porta 80 será redirecionado para o squid, os clientes deverão ter em sua configuração o gateway apontando para 192.168.200.1.

Falar de diretórios...

Falar de init.d



squidguard

```

OPTIONS FOR URL REWRITING

Linha 4712
url_rewrite_program /usr/bin/squidGuard

```

## 24.11 SquidGuard e listas de negação

O SquidGuard é um software redirecionador de URL, que pode ser usado para controle de conteúdo de sites que os usuários podem acessar. Ele foi escrito como um plug-in para o Squid e usa BlackLists para definir sites que o usuário não pode acessar. A filtragem do software se estende a todos os computadores em uma organização, incluindo computadores Windows e Macintosh

1. `sudo apt update -y`
2. `sudo apt install squid -y`
3. `sudo apt install squidguard -y`

Conforme imagem abaixo.

```

usuario@debian:~$
usuario@debian:~$
usuario@debian:~$ sudo apt install squid3 squidguard -y

```

Agora reinicie o computador. Quando o computador carregar o Sistema Operacional, navegue até o diretório /tmp e realize o download de uma lista, com wget, na listagem abaixo você encontra a sequência de comandos.

1. cd /tmp
2. wget  
https://web.archive.org/web/20210502020725if\_/http://www.shallalist.de/Downloads/shallalist.tar.gz
3. tar -xzf shallalist.tar.gz

A imagem abaixo demonstra a execução do wget

```
wget https://web.archive.org/web/20210502020725if_/http://www.shallalist.de/Downloads/shallalist.tar.gz
--2023-10-26 17:26:01-- https://web.archive.org/web/20210502020725if_/http://www.shallalist.de/Downloads/shallalist.tar.gz
Resolving web.archive.org (web.archive.org)... 207.241.237.3
Connecting to web.archive.org (web.archive.org) |207.241.237.3|:443... connected.
HTTP request sent, awaiting response... 200 OK [1]
Length: 9859347 (9.4M) [application/x-gzip]
Saving to: 'shallalist.tar.gz'

shallalist.tar.gz 100%[=====] 9.40M 722KB/s in 16s [2]
2023-10-26 17:26:18 (588 KB/s) - 'shallalist.tar.gz' saved [9859347/9859347] [3]
```

Onde:

1. Comando wget com a url do arquivo que será carregado;
2. Status 200 de http, indicando sucesso;
3. Tamanho do arquivo compactado.

Ao final da sequência de comandos acima, você terá então um diretório BL em /tmp com todas as listas de negação, agora é simples, agora é mover este diretório e seu conteúdo para /var/lib/squidguard/db/ conforme código abaixo.

```
usuario@debian:/tmp$ tar -xzf ./shallalist.tar.gz
usuario@debian:/tmp$ ls
BL
shallalist.tar.gz
```

1. sudo mv ./BL /var/lib/squidguard/db/

Veja que precisará do sudo.

```
usuario@debian:/tmp$ sudo mv ./BL /var/lib/squidguard/db/
[sudo] password for usuario:
usuario@debian:/tmp$
```

Agora vamos entender o diretório BL, com um ls você poderá ver que existem várias listas já segregadas em diretórios, mas é lógico nem todas as listas serão necessárias mas vamos manter elas no diretório **/var/lib/squidguard/db/BL**.

```
usuario@debian:/tmp$ ls /var/lib/squidguard/db/BL/
adv downloads government military recreation socialnet webphone
aggressive drugs hacking models redirector spyware webradio
alcohol dynamic hobby movies religion tracker webtv
anonvpn education homestyle music remotecontrol updatesites
automobile finance hospitals news ringtones urlshortener
chat fortunetelling imagehosting podcasts science violence
COPYRIGHT forum isp politics porn sex warez
costtraps gamble jobsearch radiotv shopping weapons
dating global_usage library searchengines
usuario@debian:/tmp$
```

Para este conteúdo será aplicada somente como exemplo o filtro de negação para sites pornográficos, veja quais são os arquivos que temos:

```
usuario@debian:/tmp$ ls /var/lib/squidguard/db/BL/porn/
domains urls
usuario@debian:/tmp$ _
```

Para configurar qual lista vamos utilizar, será editado o arquivo **/etc/squidguard/squidGuard.conf** conforme imagem abaixo.

```
usuario@debian:/tmp$
usuario@debian:/tmp$ sudo nano /etc/squidguard/squidGuard.conf
```

Logo no começo do arquivo é definido o diretório das listas, que por padrão é **/var/lib/squidguard/db/** conforme já utilizado até então.

```
GNU nano 3.2 /etc/squidguard/squidGuard.conf

#
CONFIG FILE FOR SQUIDGUARD
#
Caution: do NOT use comments inside { }
#
dbhome /var/lib/squidguard/db
logdir /var/log/squidguard

#
TIME RULES:
abbrev for weekdays:
s = sun, m = mon, t =tue, w = wed, h = thu, f = fri, a = sat
time workhours {
 weekly mtwhf 08:00 - 16:30
 date -*-*01 08:00 - 16:30
}
```

Ative a **dest adult** conforme listagem abaixo, aponte para os arquivos conforme figura abaixo. Fique atento a abertura e fechamento de chaves.

```
dest adult {
 domainlist BL/porn/domains
 urllist BL/porn/urls
 #
 # expressionlist BL/adult/expressions
 #
 # redirect http://admin.foo.bar.de/cgi-bin
```

Desative a acl foo-clients conforme figura abaixo, seremos bem restritivos.

```

acl {
 admin {
 pass any
 }

 # foo-clients within workhours {
 # pass good !in-addr !porn any
 # } else {
 # pass any
 #
}

```

Na última acl, a default negue a lista de destino adult usando ! e permita o resto.

```

default {
 pass !adult any
 redirect http://admin.foo.bar
}-

```

**Salve** o arquivo **/etc/squidguard/squidGuard.conf** e pronto, agora temos que informar ao squid que será usado um programa externo de rewrite, então abra com o nano o arquivo **/etc/squid/squid.conf**. Na linha aproximada 5038 (estou atuando com o squid versão 5), informe o programa para rewrite, conforme figura abaixo.

```

GNU nano 7.2 /etc/squid/squid.conf *
squid -k reconfigure.
#Default:
pinger_enable on

OPTIONS FOR URL REWRITING

url_rewrite_program /usr/bin/squidGuard
TAG: url_rewrite_program
The name and command line parameters of an admin-provided executable

```

A partir do arquivo domains e urls o squidguard deverá montar os arquivos domains.db e urls.db, para isso execute o comando da imagem abaixo.

```

usuario@debian:/tmp$ sudo squidGuard -C all
usuario@debian:/tmp$

```

Se as listas estão corretas, terá 100% de sucesso, mas quando isso é feito os arquivos serão gerados para o root, e o squid opera com o usuário proxy, para mudar o dono dos arquivos use o comando chown conforme figura abaixo.

```

sudo chown -R proxy:proxy /var/lib/squidguard/db/
sudo chmod 755 /var/lib/squidguard/db/

```

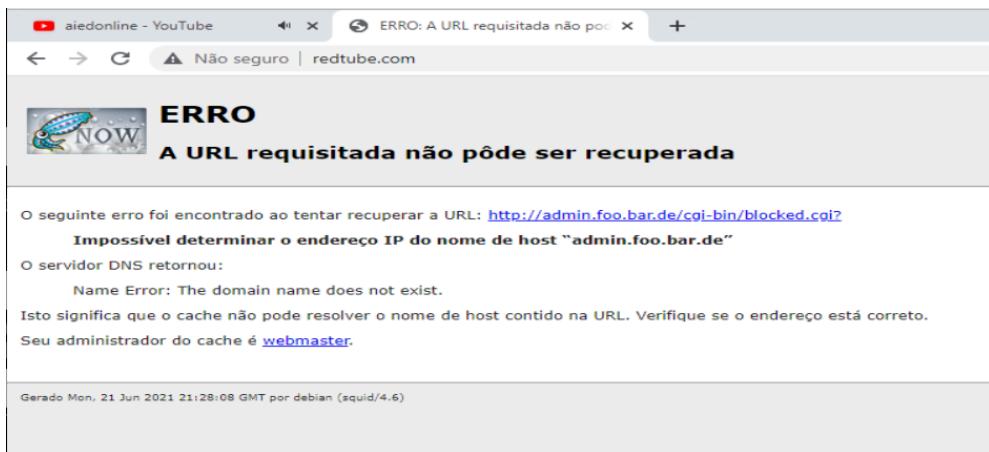
Também mude a permissão de acesso para 755, com um ls confirme o dono do arquivo e a permissão de acesso.

```

usuario@debian:/tmp$ sudo ls -l /var/lib/squidguard/db/BL/porn/
total 54452
-rw-r---- 1 proxy proxy 14362713 May 24 16:20 domains
-rw-r--r-- 1 proxy proxy 35930112 Jun 21 18:21 domains.db
-rw-r---- 1 proxy proxy 1868863 Sep 23 2020 urls
-rw-r--r-- 1 proxy proxy 3592192 Jun 21 18:21 urls.db
usuario@debian:/tmp$

```

Agora reinicie o servidor, quando o servidor estiver ativo novamente com o systemctl tenha certeza que o squid opera, pronto, em um cliente proxy interno verá que as urls clássicas estarão negadas conforme figura abaixo.



## 24.12 Validando e recarregando a configuração

Para validar o arquivo de configuração antes de reiniciar o serviço do squid, o administrador deve executar o parse do squid, para isso deve-se utilizar o comando.

1. sudo squid -k parse
2. sudo squid -k reconfigure

Veja que termina sem nenhum problema, caso haja problemas então a validação será parada e um mensagem de erro será bem destacada.

```
2021/06/15 14:50:00| Processing: http_access allow all
2021/06/15 14:50:00| Processing: http_port 3128
2021/06/15 14:50:00| Processing: coredump_dir /var/spool/squid
2021/06/15 14:50:00| Processing: refresh_pattern ^ftp: 1440 20% 10080
2021/06/15 14:50:00| Processing: refresh_pattern ^gopher: 1440 0% 1440
2021/06/15 14:50:00| Processing: refresh_pattern -i (/cgi-bin/|\.?) 0 0% 0
2021/06/15 14:50:00| Processing: refresh_pattern . 0 20% 4320
2021/06/15 14:50:00| Initializing https:// proxy context
usuario@debian:~$ _
```

Para reiniciar o serviço squid, é muito simples.

1. sudo systemctl restart squid.service

Pronto, o squid está pronto para o uso nos clientes.

## 24.13 Configurando uma máquina cliente na abordagem 1

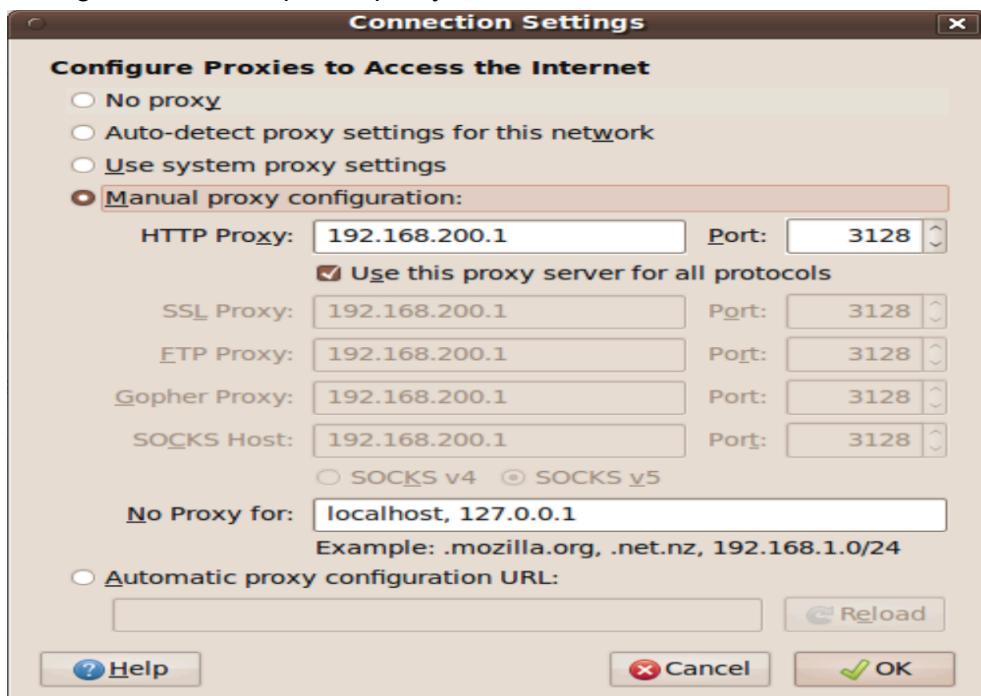
A configuração do squid é feita em um servidor e naturalmente todas as máquinas internas da organização que pertencem a política de acesso são obrigadas a acessar o serviço

proxy, algo que se deve ter em mente é que estas máquinas não possuem GATEWAY configurado em sua configuração de interface de rede.

### 24.13.1 GNU/Linux Ubuntu Gráfico

Qualquer GNU/Linux que utilizar que for gráfico será facilmente configurado, na sequência de imagens abaixo o autor está utilizando um GNU/Linux Ubuntu 9.04 gráfico pois está revivendo a nostalgia, sim o autor tem várias distros legadas.

No seu navegador (Browser) deve chegar em Configurações e em Network, configure para network os seguinte caminho para o proxy.

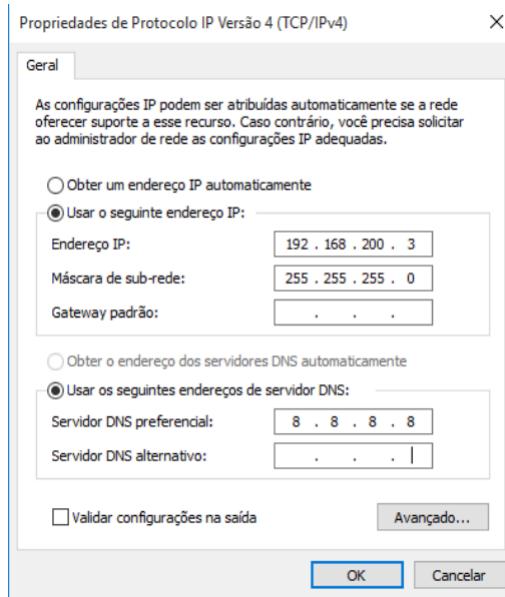


Todas as portas devem seguir a mesma configuração de IP e porta de serviço proxy na rede, conforme figura acima. Na figura abaixo é possível ver uma página aberta no navegador.

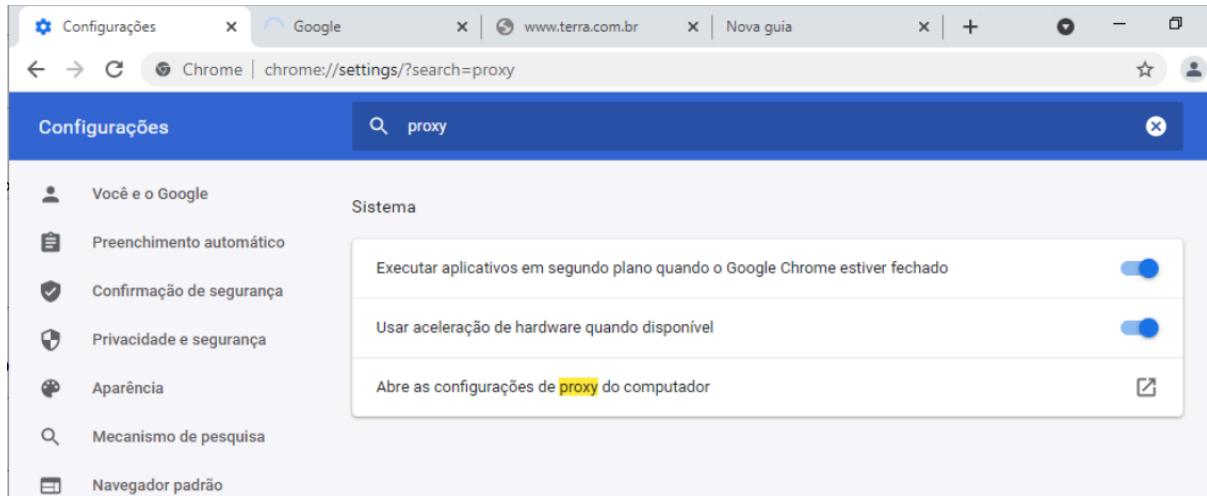


## 24.13.2 Microsoft Windows 10

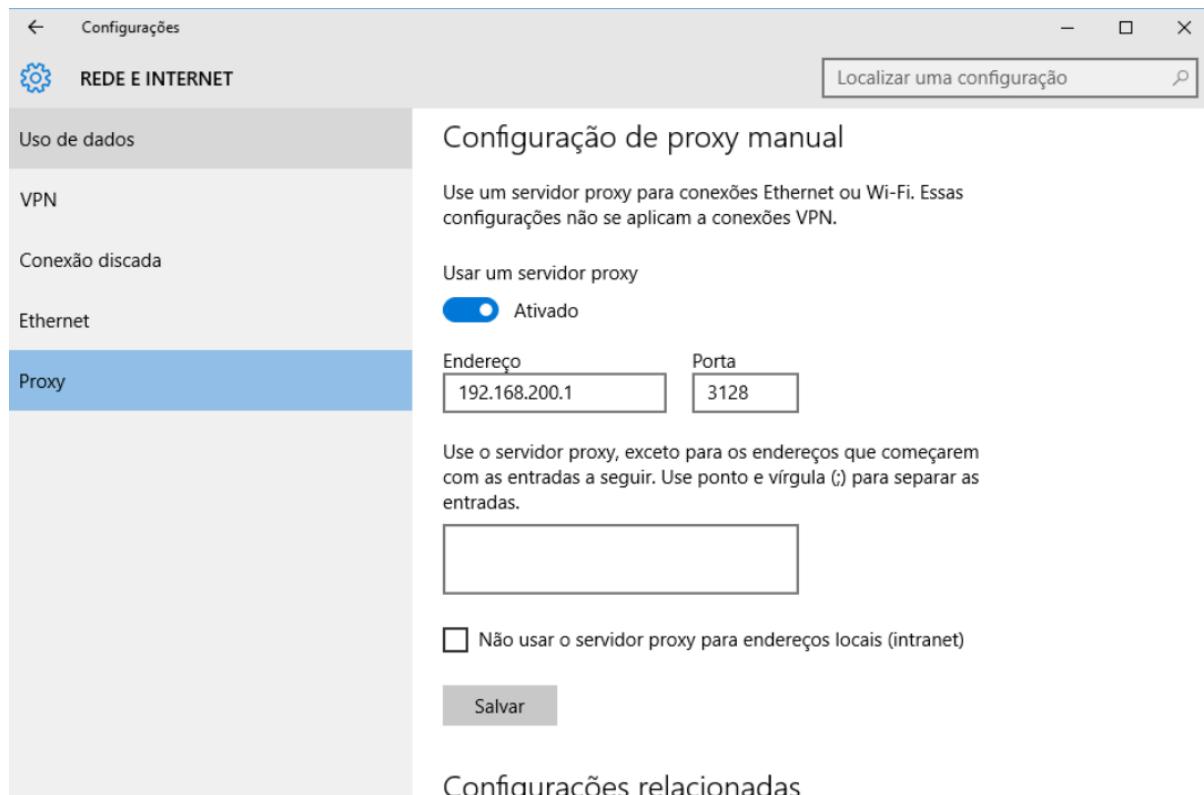
No caso do Microsoft Windows, configure o IP fixo conforme figura abaixo, não configure o Gateway.



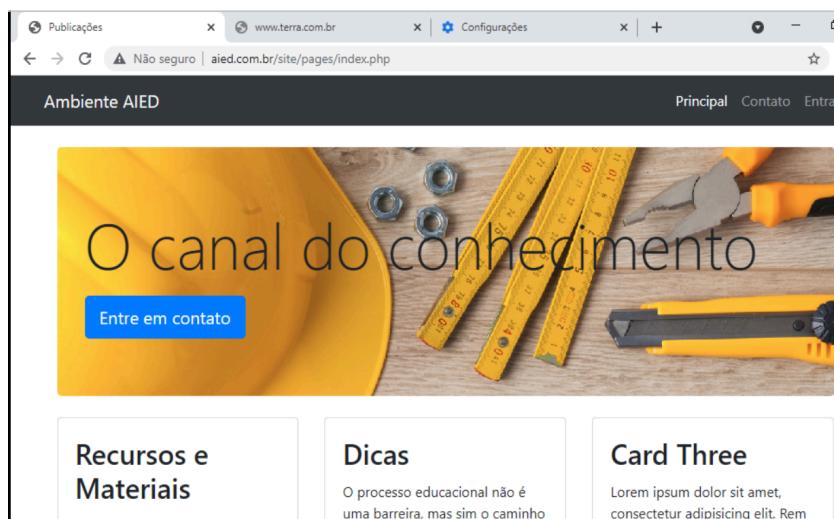
Vamos fazer um teste simples de proxy com o navegador Chrome, vai em Settings e procure por proxy.



Na opção de proxy, informe o endereço IP e a porta do serviço proxy, conforme figura abaixo.



Agora você pode desfrutar do site [www.aied.com.br](http://www.aied.com.br)



### 24.13.3 GNU/Linux Debian Terminal

A configuração de um Linux terminal é muito simples, após configurar o arquivo `/etc/network/interfaces` com uma configuração sem gateway, navegue até o diretório do usuário.

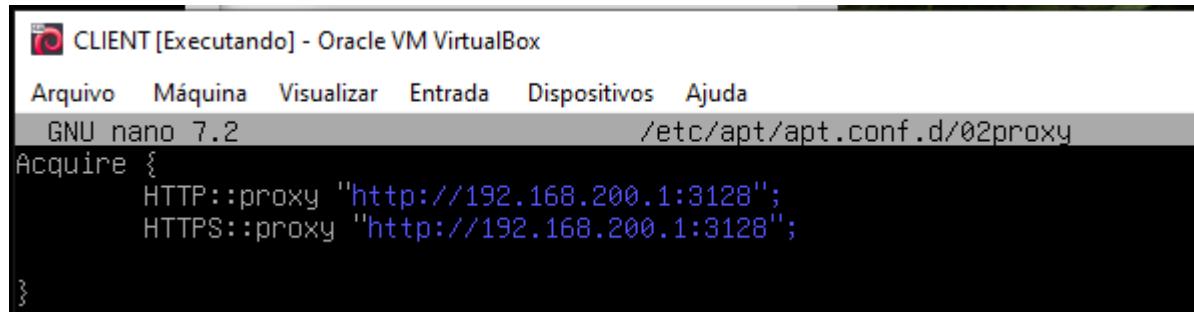
Edite um arquivo que já existe chamado `/etc/environment`, adicione no final do arquivo 3 linhas, conforme imagem abaixo.

```
export http_proxy=http://192.168.200.1:3128
export https_proxy=https://192.168.200.1:3128
export ftp_proxy=http://192.168.200.1:3128
```

Os 3 protocolos http, https e ftp estão sendo redirecionado para o proxy 192.168.200.1 porta 3128, ou seja, nosso squid.

```
usuario@debian:/tmp$ wget http://www.google.com
--2021-06-15 19:03:38-- http://www.google.com/
Connecting to 192.168.200.1:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html'

index.html [=>] 13.20K --.-KB/s in 0.009s
2021-06-15 19:03:38 (1.45 MB/s) - 'index.html' saved [13514]
usuario@debian:/tmp$ _
```



## 24.14 Configurando uma máquina cliente na abordagem 2

A configuração do squid é feita em um servidor e naturalmente todas as máquinas internas da organização que pertencem a política de acesso são obrigadas a acessar o serviço proxy, mas com esta abordagem não é necessário nenhuma configuração de proxy nos clientes, tudo será feito por DHCP, então todas as interfaces de rede dos clientes devem estar com DHCP auto.

## 24.15 Configurando uma máquina cliente na abordagem 3

A configuração do squid é feita em um servidor e naturalmente todas as máquinas internas da organização que pertencem a política de acesso mas nesta abordagem somente a porta 80 é direcionada internamente.

Os hosts internos não sabem deste redirecionamento, acreditam fielmente no Gateway e por isso nesta abordagem o gateway deve ser informado no arquivo de interfaces, mas atenção, não faça nenhuma configuração de proxy no cliente.

## 24.16 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

### 24.16.1 Prática 30x498c 01: Configurando o Squid na Abordagem 1

Nesta prática o aluno deve ter a expertise para configurar o serviço Squid;

1. Configure o arquivo de Interfaces com as seguintes configurações:
  - a. Endereço IP: 192.168.200.1
  - b. Network: 192.168.200.0
  - c. Máscara: 255.255.255.0
  - d. Broadcast: 192.168.200.255
2. Instale o squid com o uso do apt;
3. Permita que qualquer máquina tenho acesso ao serviço proxy;

Execute o comando aied conforme comando listagem abaixo **na máquina squid**.

1. sudo aied validar 30x498c checkpoint01

Após aceitar, será então validado todos os tópicos conforme figura abaixo, tudo deve estar verde.

```
usuario@debian:/tmp$ sudo aied validar 30x498c checkpoint01
Action that will be performed: validar

Practice: Validação da prática de Squid, instalando e configurando o squid
The OUTPUT of the command will be sent: ip address
The OUTPUT of the command will be sent: cat /etc/sysctl.conf
The OUTPUT of the command will be sent: iptables -L
The OUTPUT of the command will be sent: systemctl status squid.service --no-pager
The OUTPUT of the command will be sent: cat /etc/squid/squid.conf
The OUTPUT of the command will be sent: squid -k parse

Do you wish to continue? (y for YES) y

Total hits: 8 out of a total of 8. This is equivalent to 100% (hit).
Identification: 25e64062-c
AIED v(11)
```

### 24.16.2 Prática 30x498c 02: Negando redes sociais com Abordagem 1

Nesta prática o aluno deve ter a expertise para configurar criar ACL que negam algumas redes sociais.

1. Crie um arquivo chamado /etc/squid/deny\_sociais.acl
2. Neste arquivo negue acessoa o .facebook.com e .instagram.com;
3. No arquivo /etc/squid/squid.conf crie uma configuração ACL com o nome deny\_social e aponte para este arquivo;
4. negue usando deny a configuração deny\_social;

Execute o comando aied conforme comando listagem abaixo **na máquina squid**.

1. sudo aied validar 30x498c checkpoint02

```
usuario@debian:~$ sudo aied validar 30x498c checkpoint02
Action that will be performed: validar

Practice: Validação da prática de Squid, instalando negando redes sociais
The OUTPUT of the command will be sent: systemctl status squid.service --no-pager
The OUTPUT of the command will be sent: cat /etc/squid/squid.conf
The OUTPUT of the command will be sent: cat /etc/squid/deny_sociais.acl
The OUTPUT of the command will be sent: squid -k parse

Do you wish to continue? (y for YES) y

Total hits: 7 out of a total of 7. This is equivalent to 100% (hit).
Identification: 25e64062-c
AIED v(11)
```

#### 24.16.3 Prática 4a2f8c 01: Configurando o Squid para atender 192.168.200.0/24 com Abordagem 1

Nesta prática o aluno deve ter a expertise para configurar o serviço Squid, com a seguinte configuração:

1. Configure o arquivo de Interfaces com as seguintes configurações:
  - a. Endereço IP: 192.168.200.1
  - b. Network: 192.168.200.0
  - c. Máscara: 255.255.255.0
  - d. Broadcast: 192.168.200.255
2. Instale o squid com o uso do apt;
3. Permita que somente a rede **192.168.200.0/24** acesse o serviço squid;

**ATENÇÃO: Não pode ter configuração de Gateway;**

Execute o comando aied conforme comando listagem abaixo **na máquina squid**.

1. sudo aied validar 4a2f8c checkpoint01

```
usuario@debian:~$ sudo aied validar 4a2f8c checkpoint01
Action that will be performed: validar

Practice: Validação da prática de Squid, configurando squid para a rede 192.168.200.0/24
The OUTPUT of the command will be sent: ip address
The OUTPUT of the command will be sent: cat /etc/sysctl.conf
The OUTPUT of the command will be sent: iptables -L
The OUTPUT of the command will be sent: systemctl status squid.service --no-pager
The OUTPUT of the command will be sent: cat /etc/squid/squid.conf
The OUTPUT of the command will be sent: squid -k parse

Do you wish to continue? (y for YES) y

Total hits: 11 out of a total of 11. This is equivalent to 100% (hit).
Identification: 25e64062-c
AIED v(11)
```

#### 24.16.4 Prática 4a2f8c 02: Bloqueando sites Pornográficos com Abordagem 1

Nesta prática o aluno deve ter a expertise para configurar o serviço Squid e adicionar uma lista para negar sites pornográficos;

1. Realize o bloqueio de sites da lista **porn** conforme tópico sobre Squidguard deste capítulo.

Execute o comando aied conforme comando listagem abaixo **na máquina squid**.

```
1. sudo aied validar 4a2f8c checkpoint02
```

#### 24.16.5 Prática ccd6973c 01: Configurando Squid + DHCP com Abordagem 2

Nesta prática o aluno deve ter a expertise para configurar o serviço Squid e adicionar uma lista para negar sites pornográficos;

1. Configure o arquivo de Interfaces com as seguintes configurações:
  - a. Endereço IP: 192.168.200.1
  - b. Network: 192.168.200.0
  - c. Máscara: 255.255.255.0
  - d. Broadcast: 192.168.200.255
2. Instale o squid, apache2 e o DHCP Server com o uso do apt;
3. Permita que somente a rede 192.168.200.0/24 acesse o serviço squid;
4. Crie o arquivo /var/www/http/proxy.pac no apache e por systemctl garanta que o apache esteja ativo e habilitado para inicialização;
5. Configure o arquivo /etc/dhcp/dhcpd.conf, não use gateway, use a configuração de proxy passando como parâmetro opcional o arquivo /var/www/html/proxy.pac;
6. Configure o squid;
7. Reinicie o servidor;

**ATENÇÃO: Não pode ter configuração de Gateway;**

Execute o comando aied conforme comando listagem abaixo **na máquina squid**.

```
1. sudo aied validar ccd6973c checkpoint01
```

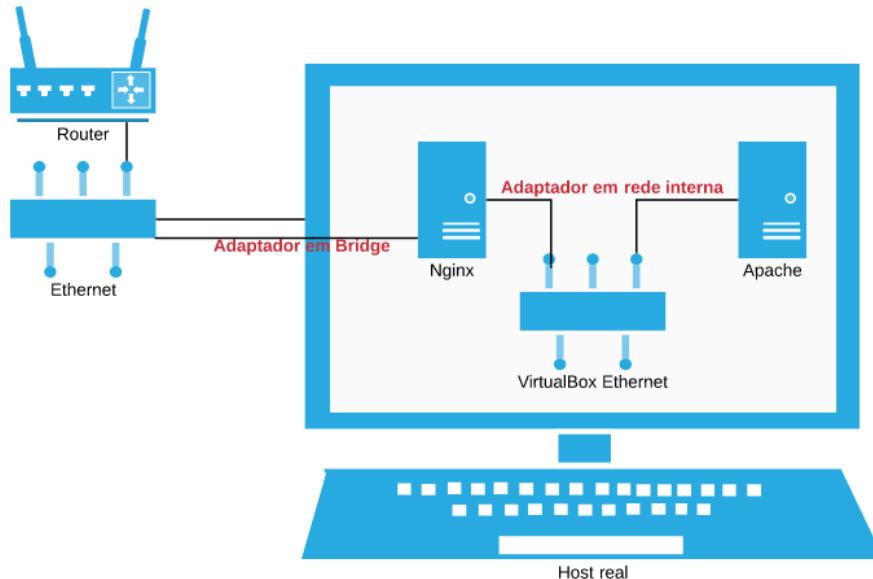
## 25 Proxy Reverso Nginx (finalizado)

Nginx é um servidor de web que também pode ser utilizado como um proxy reverso, balanceador de carga, proxy de email e serviço de cache, é um software livre e de código aberto, lançado sob os termos da Licença BSD.

Hoje em dia (data de escrita deste capítulo), uma grande fração dos servidores web usa NGINX, geralmente como um balanceador de carga pela sua simples implementação, baixo custo de recursos.

A instalação do NGINX é realizada em um GNU/Linux que está exposto na rede mundial, levando em conta que o aluno deve compreender a montagem de um Firewall por Iptables para proteger esta máquina.

Neste capítulo a rede mundial será a rede local mais abrangente e a rede interna com o Apache será a rede interna do VirtualBox, conforme esquema abaixo.

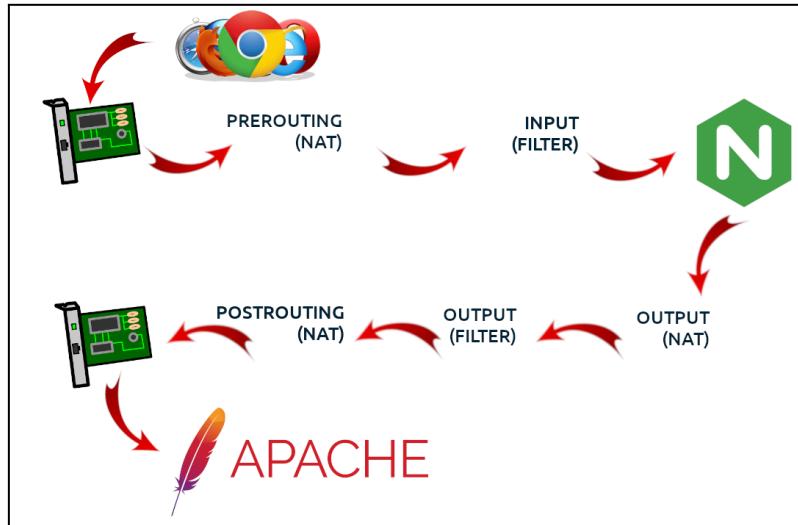


É feito assim para evitar que o aluno seja obrigado a obter um IP público só para aprender o conceito de NGINX. Esta máquina que contém o NGINX deve estar configurada segundo a aula de "[Capítulo de configuração de serviço Gateway](#)", ou seja, um mínimo de regras de Iptables para dar acesso ao Apache para a rede mundial.

Em casos bem extremos e é até recomendado o Apache não precisa de ter tal acesso a rede mundial, isso evitaria um ataque de dentro para fora, mas para esta prática sim, o Apache terá acesso a rede mundial para update e instalação de pacotes.

O NGINX fará um serviço de Proxy HTTP reverso + Cache, para isso ele deve receber as requisições na máquina Gateway (que está exposta para a rede mais abrangente).

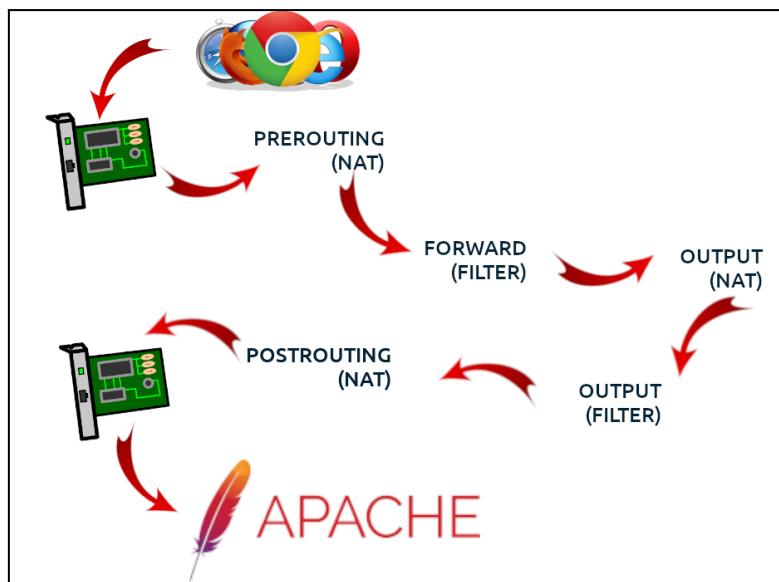
Veja que neste modelo o NGINX ele é o servidor na visão do Browser e a requisição é para ele, conforme esquema abaixo.



Na configuração com o NGINX será o cliente do Apache pois quem é o alvo do cliente real (Browser do usuário) é o NGINX, este realiza um **pass** para o Apache que está interno.

O Apache poderá estar em uma ou mais máquinas dentro da rede, mas para evitar uso de memória será configurado em apenas uma máquina virtual.

Esta arquitetura é diferente de um simples redirecionamento NAT feito por Iptables (ver figura abaixo), que por regras de redirecionamento por PREROUTING altera o destino da requisição do Browser e em FILTER é feito o FORWARD.



**Atenção:** também é possível se ter balanceamento de carga de serviços por Iptables.

Uma grande vantagem do uso do NGINX é que as máquinas que possuem o Apache podem

## 25.1 Recursos do capítulo

Todo o conteúdo externo ao texto pode ser obtido no link abaixo.



O material deste capítulo pode ser [obtido neste link](#).

## 25.2 Prática de Nginx

O objetivo desta prática é disponibilizar um serviço http utilizando a porta 80, vamos utilizar como container o Apache2 por ser simples.

O aluno deve importar duas máquinas virtuais com GNU/Linux instalado, e naturalmente sem interface gráfica. Na figura abaixo estou esquematizando a conexão de rede das máquinas bem como o nome que irão assumir neste momento.

A máquina chamada Nginx terá dois adaptadores de rede configurados conforme o capítulo “[configuração de serviço Gateway](#)”.

Já a máquina Apache, deve ter como única interface de rede configurada a Rede interna do VirtualBox, para esta máquina será adicionada um IP estático, conforme prática “[Configurando um servidor WEB com Apache](#)”.

## 25.3 Instalando e configurando o servidor Apache2

A primeira configuração que deve ser realizada é a configuração do IP da interface interna da máquina apache, para isso edite o arquivo /etc/network/interfaces conforme a imagem abaixo que representa o trecho do arquivo que interessa.

```
The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.2
 network 192.168.200.0
 netmask 255.255.255.0
 broadcast 192.168.200.255
usuario@debian:~$ _
```

Veja que a única interface da máquina está na rede interna do VirtualBox e a configuração é de um endereço de rede que está na rede 192.168.200.0/24.

Para instalar o apache proceda conforme o tópico “[Instalado e configurando Apache2](#)”

## 25.4 Instalando o Nginx

A instalação do NGINX é muito simples, na máquina Gateway execute os dois comandos da listagem abaixo.

1. sudo apt update -y
2. sudo apt install nginx -y

A linha 1 é clássica, sempre que vamos preparar uma nova máquina executamos a atualização das bibliotecas e dos resources. Na linha 2 está sendo instalada a última versão no NGINX estável e que está no PPA.

O NGINX vem configurado por padrão para assumir o serviço porta 80 da máquina Gateway, portanto deve-se remover os arquivos default, conforme listagem abaixo.

1. sudo rm /etc/nginx/sites-enabled/default
2. sudo rm /etc/nginx/sites-available/default

Agora vamos configurar um site, chamado “artigos.aied.com.br”, sim um site genérico, para isso crie um arquivo com o nano conforme:

1. sudo nano /etc/nginx/sites-available/artigos.aied.com.br.conf

Codifique conforme a listagem abaixo, **atenção**: cuidado como o Ctrl+C/Ctrl+V de máquinas Windows no servidor Linux, e, utilize espaços ao invés de TAB.

```
1. proxy_cache_path /tmp/web levels=1:2 keys_zone=web:10m inactive=10m
 max_size=1000M;
2.
3. upstream backend {
4. server 192.168.200.2:80;
5. }
6.
7. server {
8. listen 80;
9. server_name artigos.aied.com.br;
10. location / {
11. proxy_set_header x-real-IP $remote_addr;
12. proxy_set_header x-forwarded-for $proxy_add_x_forwarded_for;
13. proxy_set_header host $http_host;
14. proxy_pass http://backend;
15. proxy_cache web;
16. }
17. }
```

Na linha 1 está sendo configurado um diretório para armazenamento de cache HTTP, uma zona (diretório) com 10M chamada web no diretório /tmp/web.

Da linha 3 a linha 5 é configurado o array de servidores que deverão atender por este serviço;

A linha 8 e linha 9 definem o domínio e a porta que este serviço deverá responder, lembre-se que uma máquina com NGINX pode contar inúmeros arquivos de configuração como este, e inclusive estudando na mesma porta mas não no mesmo server\_name.

Da linha 11 até a linha 13 é passado parâmetros encontrados no HTTP para o proxy\_pass;

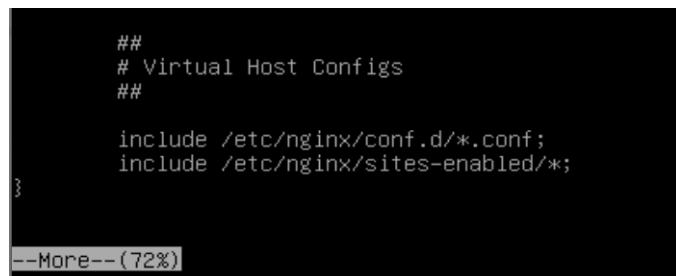
A linha 14 é o proxy\_pass;

A linha 15 define a zona cache que é criada na Linha 1 deste arquivo.

#### 25.4.1 Sites available e sites enabled

Sempre que o NGINX recebe uma requisição para um site, neste exemplo artigos.aied.com.br ele busca em memória qual o endereço do serviço HTTP/HTTPS e repassa, estas informações estão em memória mas são carregadas de arquivos de configurações conforme o exemplo já digitado.

Estes são carregados de /etc/nginx/sites-enabled/, esta configuração (ver figura abaixo) está definida em /etc/nginx/nginx.conf.



```

Virtual Host Configs

include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;
}

--More-- (72%)
```

O correto não é adicionar /etc/nginx/sites-available/ (jamais faça isso), o correto é adicionar por link simbólico um link em /etc/nginx/sites-enabled/ que aponta para o arquivo configurado em sites available.

Agora o site deve ser referenciado em sites-enabled, para isso crie um link simbólico de artigos.aied.com.br.conf em sites-available para sites-enabled, conforme Linha 1 da listagem abaixo.

1. sudo ln -s /etc/nginx/sites-available/artigos.aied.com.br.conf /etc/nginx/sites-enabled/artigos.aied.com.br.conf
- 2.
3. sudo systemctl restart nginx.service
4. sudo systemctl status nginx.service

Configuração pronta, execute as Linhas 3 e 4 da listagem abaixo para recarregar o serviço NGINX e confirmar que está tudo OK conforme figura abaixo.

```
usuario@debian:~$ sudo systemctl status nginx.service
[sudo] password for usuario:
● nginx.service - A high performance web server and a reverse proxy server
 Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
 Active: active (running) since Tue 2021-06-08 11:19:47 -03; 1h 59min ago
 Docs: man:nginx(8)
 Process: 1193 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Process: 1194 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 1195 (nginx)
 Tasks: 3 (limit: 1149)
 Memory: 3.8M
 CGroup: /system.slice/nginx.service
 ├─1195 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
 ├─1196 nginx: worker process
 └─1197 nginx: cache manager process

Jun 08 11:19:47 debian systemd[1]: Starting A high performance web server and a reverse proxy server...
Jun 08 11:19:47 debian systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid argument
Jun 08 11:19:47 debian systemd[1]: Started A high performance web server and a reverse proxy server.
```

## 25.4.2 Balanceamento de Carga

Para fazer o balanceamento de carga, no script de configuração do site deve-se definir um array de IPs e Portas em backend (ver exemplo na listagem abaixo). Você pode fazer uma rotina em Python que mantém esta lista sempre atualizada e pode fazer o Dimensionamento Horizontal no seu serviço.

```
3. upstream backend {
4. server 192.168.200.2:80;
5. server 192.168.200.3:80;
6. server 192.168.200.4:80;
7. }
```

## 25.5 Configurando o hosts da máquina real

Para ver funcionando, o aluno deveria ter dois IPs públicos, o que seria muito difícil, mas para resolver este problema o Gateway está dentro da rede do aluno na qual a rede do aluno vai “simular” a WAN, para isso vamos fazer o computador real do aluno “Traduzir” artigos.aied.com.br para o IP assumido pela interface definida no Adaptador 1 do Gateway.

Então no Gateway execute o comando ip address conforme figura abaixo.

IP definido pelo Router real do autor

```
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 08:00:27:fe:25:53 brd ff:ff:ff:ff:ff:ff
 inet 192.168.0.7/24 brd 192.168.0.255 scope global dynamic enp0s3
 valid_lft 2033sec preferred_lft 2033sec
 inet6 2804:14c:bf41:890b:a00:27ff:fe:2553/64 scope global dynamic mngtmpaddr
 valid_lft 604783sec preferred_lft 518383sec
 inet6 fe80::a00:27ff:fe:2553/64 scope link
 valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 08:00:27:eb:a0:ee brd ff:ff:ff:ff:ff:ff
 inet 192.168.200.1/24 brd 192.168.200.255 scope global enp0s8
 valid_lft forever preferred_lft forever
 inet6 fe80::a00:27ff:feeb:a0ee/64 scope link
 valid_lft forever preferred_lft forever
```

Neste caso, o autor anotou este endereço, agora, como você deve proceder para configurar seu computador real para fazer esta tradução:

### 25.5.1 Configurando o /etc/hosts no GNU/Linux e MAC OS

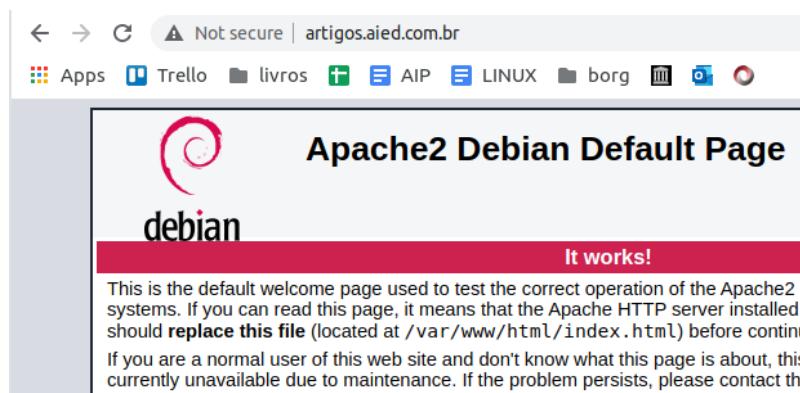
Se usa um GNU/Linux ou um MACOS, então edite o arquivo hosts conforme comando abaixo.

#### 1. sudo nano /etc/hosts

Adicione para a prática a linha abaixo, estou usando 192.168.0.7 pois conforme já explicado é o IP associado a este Gateway na minha rede em um dado momento.

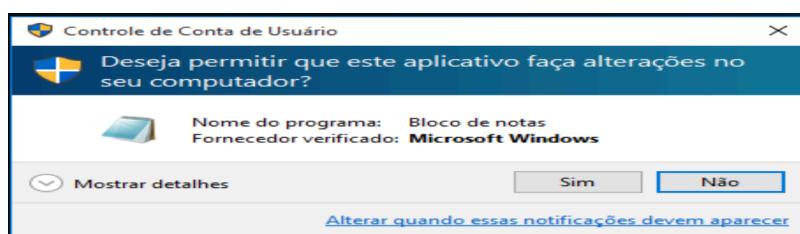
```
GNU nano 4.8
127.0.0.1 localhost
127.0.0.1 wpo
192.168.0.7 artigos.aied.com.br
The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
```

Pronto, agora abra o Browser e digite <http://artigos.aied.com.br> e veja que o NGINX repassou a requisição para a máquina apache.

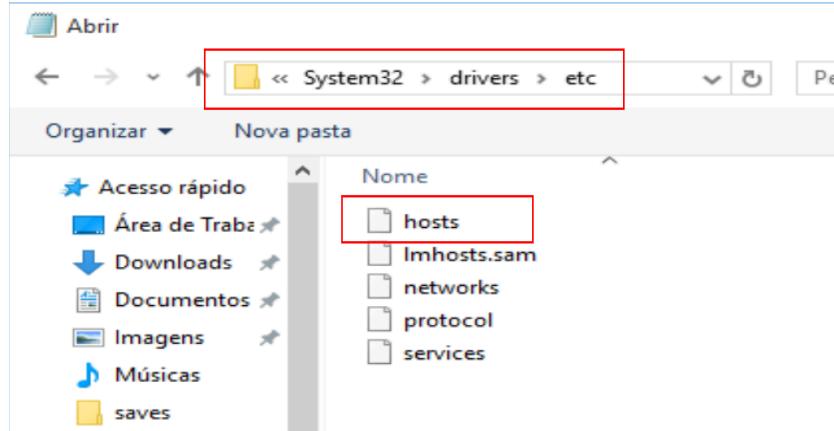


### 25.5.2 Configurando o C:\windows\system32\drivers\etc\hosts no Microsoft Windows

Se utiliza Microsoft Windows então abra um Bloco de Notas em modo “Administrador”, conforme figura abaixo.



Com o Bloco de notas aberto como Administrador, abra o arquivo **C:\Windows\System32\drivers\etc\hosts** conforme figura abaixo.



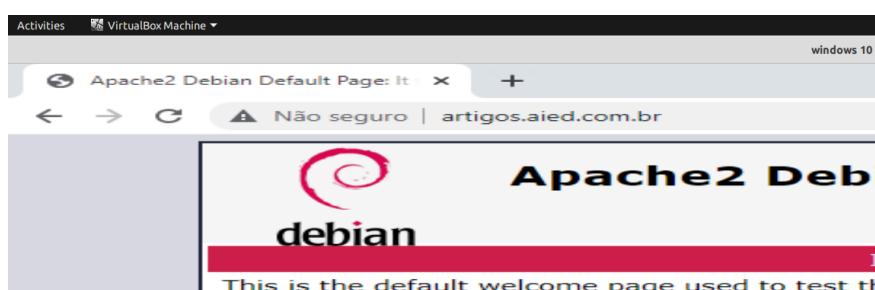
Agora adicione conforme figura abaixo a tradução, estou usando 192.168.0.7 pois é o IP definido pelo meu router em um dado momento, você conforme dito anteriormente deve utilizar o IP do Gateway definido pelo seu router.

```

hosts - Bloco de notas
Arquivo Editar Formatar Exibir Ajuda
Copyright (c) 1993-2009 Microsoft Co
#
This is a sample HOSTS file used by I
#
This file contains the mappings of IP
entry should be kept on an individual
be placed in the first column follow
The IP address and the host name shou
space.
#
Additionally, comments (such as these
lines or following the machine name)
#
For example:
#
102.54.94.97 rhino.acme.com
38.25.63.10 x.acme.com
#
localhost name resolution is handled
127.0.0.1 localhost
::1 localhost
192.168.0.7 artigos.aied.com.br

```

Abra um Browser no seu Microsoft Windows e digite: <http://artigos.aied.com.br> e veja que o NGINX repassa a requisição para o Apache.



## 25.6 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

### 25.6.1 Prática 09a378b 01: Configurando o Gateway + Nginx

Nesta prática o aluno deve ter a expertise para configurar o serviço Gateway e Nginx;

1. Proceda com a configuração “[Capítulo de configuração de serviço Gateway](#)”
2. Realize a configuração segundo o tópico “[24.4 Configurando o NGINX](#)”;
3. Reinicie a máquina

Execute o comando aied conforme comando listagem abaixo.

```
2. sudo aied validar 09a378b checkpoint01
```

```
usuario@debian:~$ sudo aied validar 09a378b checkpoint01
Ação que será executada: validar

Prática: Validação da prática de NGINX, Configurando o Gateway + Nginx
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: ping -c 1 192.168.200.2
Será enviado o OUTPUT do comando: cat /etc/sysctl.conf
Será enviado o OUTPUT do comando: cat /usr/local/sbin/gateway.sh
Será enviado o OUTPUT do comando: systemctl status gateway.service --no-pager
Será enviado o OUTPUT do comando: iptables -L
Será enviado o OUTPUT do comando: file /etc/nginx/sites-enabled/artigos.aied.com.br.conf
Será enviado o OUTPUT do comando: systemctl status nginx.service --no-pager
Será enviado o OUTPUT do comando: cat /etc/nginx/sites-available/artigos.aied.com.br.conf

Deseja continuar? (s para SIM) s
```

Após aceitar, será então validado todos os tópicos conforme figura abaixo, tudo deve estar verde.

```
++++++ RESULTADO ++++++
1 - Comando: ip address
1.1 Validar por regex /inet s+192.168.200.1/
1.2 Validar por regex /brd s+ d+. d+. d+.255/

2 - Comando: ping -c 1 192.168.200.2
2.1 Validar por regex / s+0% s+packet s+loss/

3 - Comando: cat /etc/sysctl.conf
3.1 Validar por text #net.ipv4.ip_forward=1

4 - Comando: cat /usr/local/sbin/gateway.sh
4.1 Validar por regex /iptables s+ -A s+INPUT s+ -m s+state s+ - -state s+RELATED,ESTABLISHED s+ -j s+ACCEPT/
4.2 Validar por regex /iptables s+ -A s+FORWARD s+ -m s+state s+ - -state s+RELATED,ESTABLISHED s+ -j s+ACCEPT/
4.3 Validar por regex /iptables s+ -t s+nat s+ -A s+POSTROUTING s+ -o s+enp0s3 s+ -j s+MASQUERADE/

5 - Comando: systemctl status gateway.service --no-pager
5.1 Validar por text enabled;

6 - Comando: iptables -L
6.1 Validar por text RELATED,ESTABLISHED

7 - Comando: file /etc/nginx/sites-enabled/artigos.aied.com.br.conf
7.1 Validar por text /etc/nginx/sites-enabled/artigos.aied.com.br.conf: symbolic link to /etc/nginx/sites-available.com.br.conf

8 - Comando: systemctl status nginx.service --no-pager
8.1 Validar por text active (running)
8.2 Validar por text enabled;

9 - Comando: cat /etc/nginx/sites-available/artigos.aied.com.br.conf
9.1 Validar por text proxy_cache_path /tmp/web levels=1:2 keys_zone=web:10m inactive=10m max_size=1000M;
9.2 Validar por regex /upstream s+backend s+ {/
9.3 Validar por text server 192.168.200.2:80;
9.4 Validar por regex /server_name s+artigos.aied.com.br ;/
9.5 Validar por regex /proxy_set_header s+x-real-IP s+ $remote_addr ;/
9.6 Validar por regex /proxy_set_header s+x-forwarded-for s+ $proxy_add_x_forwarded_for ;/
9.7 Validar por regex /proxy_set_header s+host s+ $http_host ;/
9.8 Validar por regex /proxy_pass s+http: / /backend ;/
9.9 Validar por regex /proxy_cache s+web ;/

Total de acertos: 21 do total de 21 validações equivalente a 100%.
Identificação: 7bb0856c22
AIED v(7)
```

## 25.6.2 Prática 09a378b 02: Configurando o serviço Apache

Nesta prática o aluno deve ter a expertise para configurar o servidor apache em uma máquina na rede interna;

1. Crie uma nova máquina virtual, que utiliza o gateway;
2. Configurar o arquivo /etc/network/interfaces com os seguintes parâmetros:
  - a. Endereço: 192.168.200.2
  - b. Máscara: 255.255.255.0
  - c. Rede: 192.168.200.0
  - d. Broadcast: 192.168.200.255
  - e. Gateway 192.168.200.1
3. Reinicie a máquina;
4. Instale o apache com o comando: sudo apt install apache2

Execute o comando aied conforme comando listagem abaixo.

3. sudo aied validar 09a378b checkpoint02

```
usuario@debian:~$ sudo aied validar 09a378b checkpoint02
Ação que será executada: validar

Prática: Pratica de lab Apache2
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: ip route
Será enviado o OUTPUT do comando: systemctl status apache2.service --no-pager

Deseja continuar? (s para SIM) s
s
++++++ RESULTADO +++++

1 - Comando: ip address
1.1 Validar por regex /inet s+192.168.200.2/
1.2 Validar por regex /brd s+192.168.200.255/

2 - Comando: ip route
2.1 Validar por regex /default s+via s+192.168.200.1 s+dev/

3 - Comando: systemctl status apache2.service --no-pager
3.1 Validar por text active (running)
3.2 Validar por text enabled;

Total de acertos: 5 do total de 5 validações equivalente a 100%.
Identificação: d09fa5c4f4
AIED v(7)
```

## 26 Serviço de mail com Postfix (finalizado)

Adicionar mais detalhes sobre serviço

Postfix software classificado como Agente de Transferência de Emails (MTA), roteia e gerencia a entrega e-mail para contas externas ao ambiente. Atualmente é usado por aproximadamente 33% dos servidores de correio na Internet.

Teve como raiz pesquisas na IBM no final da década de 90 como uma solução leve e aberta para MTA, hoje os especialistas criadores desta tecnologia atuam no Google e mantêm suporte pelo site [postfix.org](http://postfix.org).

Características da solução:

- Suporte a Ipv6;
- Suporte a MIME;
- Autenticação SASL (Simple Authentication and Security Layer);
- Canal seguro utilizando TLS;
- Suporte a diversos banco de dados:
  - MySQL
  - PostgreSQL
  - LDAP
- Verificação a listas RBL (Real-time Blackhole List);
- Extenso suporte a filtros:
  - Suporte a expressões regulares;
  - Verificação de cabeçalho;
  - Verificação no corpo da mensagem;

### 26.1 Recursos do capítulo

Todo o conteúdo externo ao texto pode ser obtido no link abaixo.



O material deste capítulo pode ser obtido neste link.

### 26.2 Instalação e Configuração Postfix

O processo de instalação é muito simples pois será realizada visando aplicações na máquina enviando e-mails, além do uso de empacotamento de alto nível a configuração inicial é realizada por meio de um wizard auto explicativo. Comece toda instalação realizando um update geral do sistema e logo após instale os dois pacotes:

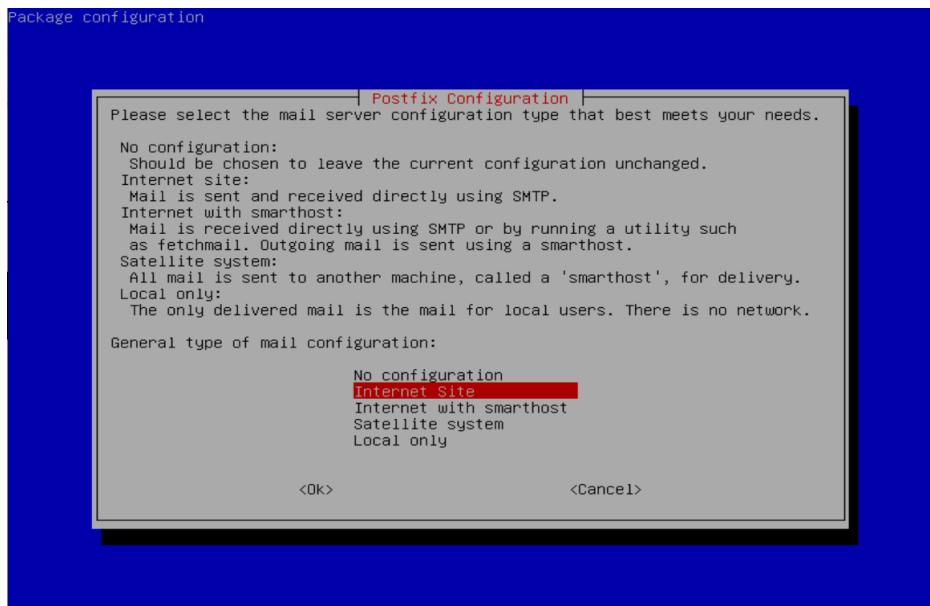
- **postfix**: O MTA de gerenciamento de email;
- **mailutils**: ferramentas auxiliares como mail que permite envio de emails por command line;

Execute os seguintes comando:

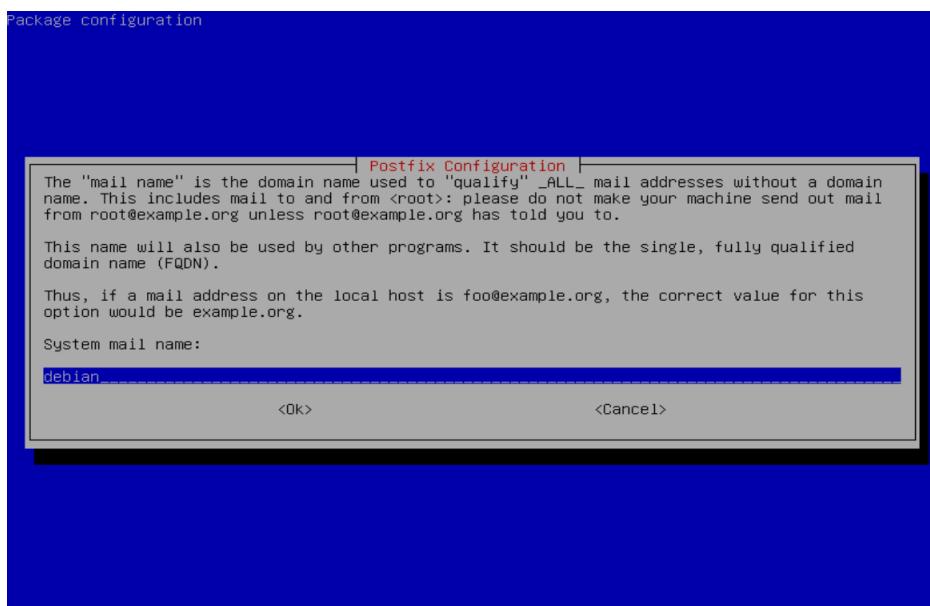
1. sudo apt update -y
2. sudo apt install postfix mailutils -y

Após a instalação do pacote postfix será carregado o wizard de configuração do serviço, na primeira etapa deve-se escolher a opção “Internet Site” pois neste material o aluno está sendo preparado para implantar um ambiente completo para atuar com serviços na Internet.

Selecione a opção **Internet Site** e pressione **Enter**.



O próximo passo é configurar o nome do domínio, vamos deixar esta configuração para depois, então pressione **Enter** e finalize a instalação.



Um utilitário de configuração para o postfix é o **postconf**, está em **/usr/sbin** um diretório que possivelmente não estão no PATH do Environment, então adicione **/usr/sbin** o PATH conforme comando abaixo.

```
usuario@debian:~$ PATH=$PATH:/usr/sbin
usuario@debian:~$ sudo postconf -e "myorigin = aied.com.br"
[sudo] password for usuario:
```

Configure o domínio de origem do e-mail com o comando acima, alterando a constante myorigin e informe o nome do host alterando o myhostname conforme figura abaixo.

```
usuario@debian:~$ sudo postconf -e "myhostname = debian.aied.com.br"
usuario@debian:~$
```

Os domínios que correspondem a \$relay\_domains são entregues com o transporte de entrega de correio \$relay\_transport. O servidor SMTP valida endereços de destinatários com \$relay\_recipient\_maps e rejeita destinatários inexistentes.

```
usuario@debian:~$ sudo postconf -e "relay_domains = aied.com.br"
usuario@debian:~$
```

O próximo passo é reiniciar o postfix e validar o status do serviço, para isso utilize o comando systemctl conforme já descrito em capítulos anteriores, veja abaixo esta operação.

```
usuario@debian:~$ sudo systemctl restart postfix.service
usuario@debian:~$ sudo systemctl status postfix.service
● postfix.service - Postfix Mail Transport Agent
 Loaded: loaded (/lib/systemd/system/postfix.service; enabled; vendor preset: enabled)
 Active: active (exited) since Thu 2021-10-21 11:36:59 -03; 9s ago
 Process: 1018 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 1018 (code=exited, status=0/SUCCESS)

Oct 21 11:36:59 debian systemd[1]: Starting Postfix Mail Transport Agent...
Oct 21 11:36:59 debian systemd[1]: Started Postfix Mail Transport Agent.
usuario@debian:~$
```

## 26.3 Testando o serviço com TELNET

Telnet é um utilitário que permite ao usuário interagir via comunicação Socket com qualquer aplicação, para isso precisa-se de um IP e uma Porta, o comando é interativo ou seja, você como cliente interage com o protocolo do servidor.

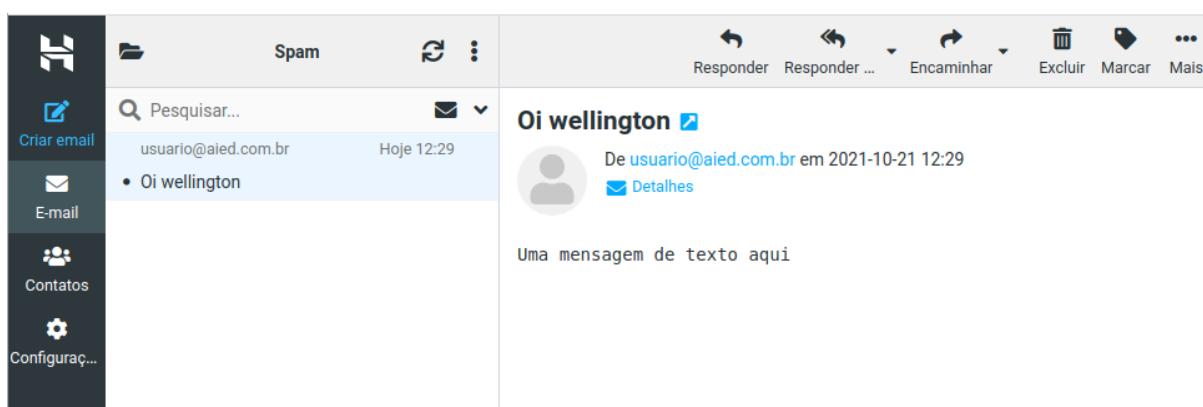
```
telnet -r usuario@debian to connect to remote host - trying to generate
usuario@debian:~$ telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 debian.aied.com.br ESMTP Postfix (Debian/GNU)
-
```

Quando você se conecta na porta 25 o Postfix responde **220 <DOMINIO>**, conforme figura abaixo. O próximo passo do protocolo é informar o **mail from** que é a conta do usuário local na máquina, deve-se informar o usuário **usuario** pois este existe nesta máquina. O postfix retorna **250 OK**.

Informa-se logo após para quem será enviado o e-mail, com **rcpt to**, e o servidor postfix retorna **250 OK**. Inicia-se o campo de dados, para isso informe **data** e o servidor retorna que **PONTO** definirá o fim da mensagem. Finaliza-se com um **PONTO** e um **quit**.

```
usuario@debian:~$ telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 debian.aied.com.br ESMTP Postfix (
mail from: usuario@aied.com.br
250 2.1.0 Ok
rcpt to: wellington@crawlerweb.com.br
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
subject: Oi wellington
Uma mensagem de texto aqui
.
250 2.0.0 Ok: queued as 99F8760813
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

O e-mail de destino é [wellington@crawlerweb.com.br](mailto:wellington@crawlerweb.com.br) então o próximo passo é abrir a interface WEB de e-mail da crawlerweb.com.br, conforme figura abaixo e entrar na conta do wellington. Veja que o e-mail chegou no destino.



## 26.4 Enviando e-mail com PHP

O próximo passo é utilizar o PHP para enviar e-mails, será um exemplo muito simples, mas para isso será necessário instalar o Apache2 e o PHP. Execute o comando abaixo.

1. sudo apt install apache2 php -y

Vamos criar então um script PHP, para isso com o nano crie um arquivo conforme comando abaixo.

1. sudo nano /var/www/html/email.php

Edite o script abaixo, no qual algumas variáveis são criadas o que poderia ser provenientes de um formulário, e invoca-se o comando PHP mail() para o mecanismo interno interagir com o postfix.

```
GNU nano 3.2 /var/www/html/email.php

<?php

error_reporting(E_ALL);
$from = "usuario@aied.com.br";
$to = "wellington@crawlerweb.com.br";
$subject = "Uma forma por PHP";
$message = "O corpo da mensagem";
$headers = "From:" . $from;
mail($to, $subject, $message, $headers);
echo "Enviado";

?>
```

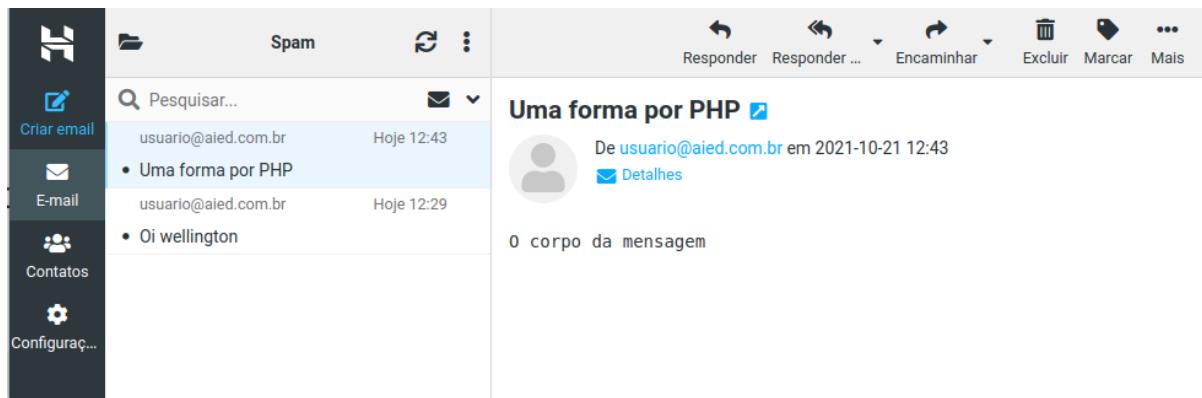
Salve o arquivo, veja que o arquivo esta devidamente no local correto, poderia criar um formulário e invocar uma interface WEB, mas não será necessário.

```
usuario@debian:/var/www/html$ ls -l
total 16
-rw-r--r-- 1 usuario usuario 258 Oct 21 12:41 email.php
-rw-r--r-- 1 root root 10701 Oct 21 12:39 index.html
usuario@debian:/var/www/html$ _
```

Outra forma de rodar um script PHP é invocar o comando php, e é isso que o apache realiza quando uma página PHP é carregada por uma interface web, então por command line podemos rodar qualquer script PHP, conforme exemplo abaixo.

```
usuario@debian:/var/www/html$ php /var/www/html/email.php
Enviadousuario@debian:/var/www/html$
usuario@debian:/var/www/html$ _
```

E-mail enviado, é possível recorrer a interface de Webmail do crawlerweb.com.br (e-mail de destino) para confirmar o recebimento.



## 26.5 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

### 26.5.1 Prática ptf0001 01: Configurando o servidor de e-mail postfix

Nesta prática o aluno deve ter a expertise para configurar o serviço Postfix para o domínio aied.com.br;

1. Realize a instalação do postfix;
2. configure o nome do domínio aied.com.br;
3. configure o nome do host no postfix como debian.aied.com.br;
4. reinicie o serviço postfix;

Execute o comando aied conforme comando listagem abaixo.

1. sudo aied validar ptf0001 checkpoint01

```
usuario@debian:~$ sudo aied validar ptf0001 checkpoint01
Ação que será executada: validar

Prática: Pratica da aula de Postfix - instalando postfix
Será enviado o OUTPUT do comando: dpkg -l | grep postfix
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: systemctl status postfix.service --no-pager
Será enviado o OUTPUT do comando: cat /etc/postfix/main.cf
```

Após aceitar, será então validado todos os tópicos conforme figura abaixo, tudo deve estar verde.

```
Deseja continuar? (s para SIM) s
s1 - Comando: dpkg -l | grep postfix
1.1 Validar por regex /ii s+postfix/
2 - Comando: ip address
2.1 Validar por regex /inet s+ d+. d+. d+. d+/
3 - Comando: systemctl status postfix.service --no-pager
3.1 Validar por text 0/SUCCESS
3.2 Validar por text enabled;
4 - Comando: cat /etc/postfix/main.cf
4.1 Validar por regex /myhostname s+ = s+debian.aied.com.br/
4.2 Validar por regex /myorigin s+ = s+aied.com.br/

Total de acertos: 6 do total de 6 validações equivalente a 100%.
Identificação: 2aebd76d0c
ATED v(8)
```

### 26.5.2 Prática ptf0001 02: Enviar um e-mail com telnet

Nesta prática o aluno deve ter a expertise para usar o comando telnet para enviar um e-mail para [wellington@crawlerweb.com.br](mailto:wellington@crawlerweb.com.br)

6. Digite o comando **sudo locale-gen "en\_US.UTF-8"**
7. Execute o comando **script**
8. Agora por telnet envie uma mensagem utilizando os seguintes dados:
  - a. Sendo enviado de: [usuario@aied.com.br](mailto:usuario@aied.com.br)
  - b. Sendo enviado para: [naoimportaweb@cyberframework.online](mailto:naoimportaweb@cyberframework.online)
  - c. Assunto: Olá wellington
  - d. Corpo da mensagem: Sou o <seu nome> e estou fazendo sua prática, gosto do seu canal
9. Finalize o envio com PONTO e logo em seguida quit
10. Execute o comando **exit**

Execute o comando aied conforme comando listagem abaixo.

1. **sudo aied validar ptf0001 checkpoint02**

```
usuario@debian:~$ sudo aied validar ptf0001 checkpoint02
Ação que será executada: validar

Prática: Pratica da aula de Postfix - Usando telnet para enviar e-mail
Será enviado o OUTPUT do comando: cat /home/usuario/typescript
```

Após aceitar, será então validado todos os tópicos conforme figura abaixo, tudo deve estar verde.

```
Deseja continuar? (s para SIM) s
s1 - Comando: cat /home/usuario/typescript
1.1 Validar por text telnet
1.2 Validar por text 220 debian.aied.com.br
1.3 Validar por regex /mail s+from s*: s*usuario@aied.com.br/
1.4 Validar por regex /rcpt s+to: s*wellington@crawlerweb.com
1.5 Validar por text data
1.6 Validar por regex /subject/
1.7 Validar por regex /queued/

Total de acertos: 7 do total de 7 validações equivalente a 100%.
Identificação: 2aebd76d0c
AIED v(8)
```

### 26.5.3 Prática ptf0001 03: Enviando e-mail com PHP

Nesta prática o aluno deve ter a expertise para enviar um e-mail com PHP;

1. Instale o PHP e o Apache 2;
2. Crie um script chamado email.php no diretório `/var/www/html/`, este script deve mandar um e-mail com os seguintes dados:
  - a. Sendo enviado de: [usuario@aied.com.br](mailto:usuario@aied.com.br)
  - b. Sendo enviado para: [wellington@aied.com.br](mailto:wellington@aied.com.br)
  - c. Assunto: Agora com PHP;
  - d. Corpo da mensagem: Sou o <seu nome> e estou adorando PHP
3. Execute o script usando o comando `php`;

Execute o comando aied conforme comando listagem abaixo.

1. sudo aied validar ptf0001 checkpoint03

```
usuario@debian:~$ sudo aied validar ptf0001 checkpoint03
Ação que será executada: validar

Prática: Pratica da aula de Postfix - Enviando com PHP
Será enviado o OUTPUT do comando: cat /var/www/html/email.php
Será enviado o OUTPUT do comando: cat /var/log/mail.log
```

Após aceitar, será então validado todos os tópicos conforme figura abaixo, tudo deve estar verde.

```
Deseja continuar? (s para SIM) s
s1 - Comando: cat /var/www/html/email.php
1.1 Validar por regex /mail ((.*?) ,(.*)? ,(.*)? ,(.*)?)/
2 - Comando: cat /var/log/mail.log
2.1 Validar por text wellington@crawlerweb.com.br

Total de acertos: 2 do total de 2 validações equivalente a 100%.
Identificação: 2aebd76d0c
AIED v(8)
```

## 27 Serviço FTP (finalizado)

O daemon VSFTPD (Very Secure FTP Daemon) é um clássico daemon ftp que disponibiliza na rede um serviço de troca de arquivos por protocolo FTP nas portas 21 e 20.

A simplicidade do protocolo FTP ganhou cenário com o advento das empresas de hospedagem, é considerado seguro se comparado ao baixo consumo de recursos e de pouca necessidade de conhecimento para o uso, mas se comparado ao SFTP é considerado inseguro. É um servidor FTP seguro, estável e rápido compatível com os sistemas operacionais Linux/UNIX (já o protocolo é compatível com qualquer arquitetura ou sistema operacional).

### 27.1 Recursos do capítulo

Todo o conteúdo externo ao texto pode ser obtido no link abaixo.



O material deste capítulo pode ser obtido neste link.

### 27.2 O protocolo FTP

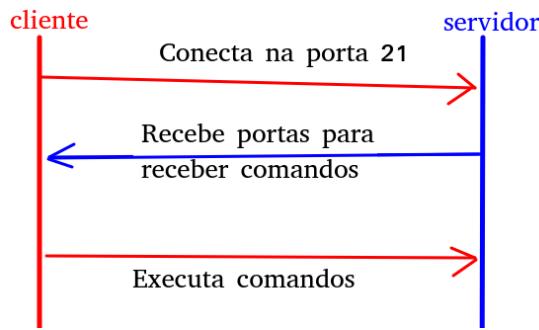
O protocolo FTP (File Transfer Protocol) é amplamente utilizado como um protocolo simples de transferência de arquivos entre computadores, principalmente em situações onde o usuário que acessa o serviço não possui muita relação com quem presta o serviço (se comparado ao SSH).

Os usuários de FTP podem se autenticar com um protocolo de entrada de texto não criptografado, normalmente na forma de um nome de usuário e senha, mas podem se conectar anonimamente se o servidor estiver configurado para permitir isso. Para uma transmissão segura que protege o nome de usuário e a senha e criptografa o conteúdo, o FTP [geralmente é protegido por SSL/TLS \(FTPS\)](#) ou substituído por [SSH File Transfer Protocol \(SFTP\)](#).

O FTP pode ser executado no modo ativo ou passivo, o que determina como a conexão de dados é estabelecida. Em ambos os casos, o cliente cria uma conexão de controle TCP.

#### 27.2.1 Comunicação em modo passivo

No FTP passivo os clientes usam a conexão de controle para enviar um comando PASV ao servidor FTP e recebe um endereço IP e a porta do servidor de comandos, então o cliente volta a se comunicar por este IP e esta porta informada.



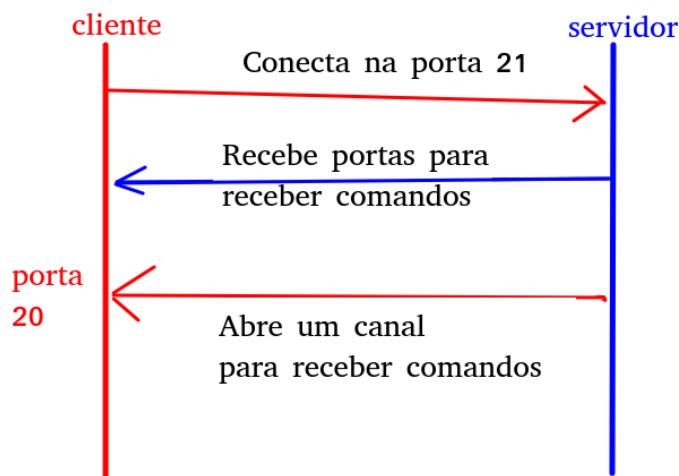
No servidor vsftpd (será estudado neste capítulo) a configuração é simples, basta ativar o flag **pasv\_enable** e informar qual o range de portas ativas do servidor para receber requisições do cliente com os comandos FTP.

```
GNU nano 3.2 /etc/vsftpd.conf
pasv_enable=YES
pasv_min_port=40000
pasv_max_port=40100
```

Nesta configuração o FIREWALL será configurado habilitando a porta 21 e as portas 40000->40100 no servidor, em apenas 1 ponto da rede. Será demonstrado no tópico [Análise de Comunicação](#) que mesmo o servidor tendo a configuração a decisão em modo passivo ou ativo parte do cliente.

### 27.2.2 Comunicação em modo ativo

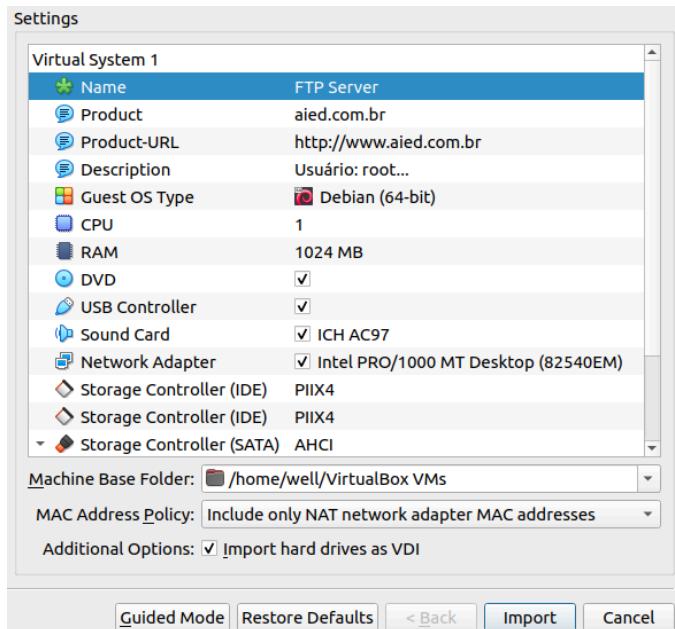
Neste modelo o cliente inicia a conexão com a porta 21 do servidor, por não passar o comando PASV então o servidor julga que a conexão é ativa. **Então o servidor se conecta ao cliente pela porta 20 do cliente.**



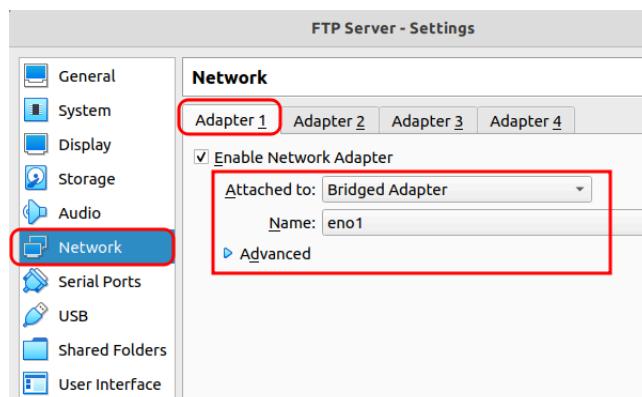
Neste cenário tanto deve-se trabalhar o Firewall do Servidor quando DOS CLIENTES, e por isso a configuração se torna difícil pois deve-se ir em cada cliente liberando porta, e liberar portas em cliente vejo que é uma falha gravíssima de segurança.

## 27.3 Preparando o ambiente

Neste procedimento será implementado um serviço FTP na própria rede interna do aluno, esta rede do aluno será considerada a rede mundial. Será necessário apenas uma máquina virtual com o Debian 10 instalado conforme [Capítulo 1](#).



O próximo passo é selecionar a rede em que este computador terá acesso, escolha a opção do Adaptador 1 e Attached selecione Bridge Adapter. O campo name é o nome da interface de rede física que é diferente da imagem abaixo.



## 27.4 Instalando o serviço FTP

Inicie a máquina virtual, e o primeiro passo é realizar um update da máquina com apt, logo em seguida instale o vsftpd conforme listagem abaixo.

1. sudo apt update -y
2. sudo apt install vsftpd -y

## 27.5 Configuração do serviço

A configuração básica refere-se a liberação de recursos atuando em valores em arquivo de configuração, recomenda-se que antes de alterar este arquivo faça uma cópia de segurança.

1. sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.old

Agora abra com o nano o arquivo **/etc/vsftpd.conf** e edite cada atributo descrito abaixo conforme este conteúdo:

- Altere o valor de **listen** para YES, isso fará com que o serviço seja inicializado a partir de um script de inicialização de sistema init ou systemd, e também fará do processo como um daemon de serviço;
- Desative o **IPv6** pois o serviço estará disponível só no IPv4, para isso **listen\_ipv6** passe para o valor NO;
- Por padrão o serviço FTP vem configurado para somente leitura, remova o comentário de **write\_enable** ativando a possibilidade de escrita;
- Por padrão vem desabilitado a opção **chroot\_local\_user**, este parâmetro quando ativo permite que usuários possam listar seus diretórios pessoais, caso queira permitir que além de ver o diretório o usuário possa escrever também, ative o parâmetro **allow\_writeable\_chroot**;

O arquivo então recebeu alterações nos seguintes valores:

1. listen=YES
2. listen\_ipv6=NO
3. write\_enable=YES
4. chroot\_local\_user=YES
5. allow\_writeable\_chroot=YES

**ATENÇÃO:** no Debian 12 tem que adicionar **local\_enable=YES** na listagem acima.

Agora reinicie o serviço vsftpd utilizando **systemctl restart** e veja se o serviço está ativo utilizando o comando **systemctl status** conforme figura abaixo.

```
usuario@debian:~$ sudo systemctl restart vsftpd.service
usuario@debian:~$ sudo systemctl status vsftpd.service
● vsftpd.service - vsftpd FTP server
 Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
 Active: active (running) since Thu 2021-10-28 12:03:31 -03; 5s ago
 Process: 676 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/SUCCESS)
 Main PID: 677 (vsftpd)
 Tasks: 1 (limit: 1149)
 Memory: 596.0K
 CPU: 0.000 CPU(s) (idle)
 CGroup: /system.slice/vsftpd.service
 └─677 /usr/sbin/vsftpd /etc/vsftpd.conf

Oct 28 12:03:31 debian systemd[1]: Starting vsftpd FTP server...
Oct 28 12:03:31 debian systemd[1]: Started vsftpd FTP server.
usuario@debian:~$
```

## 27.6 Acessando o serviço FTP

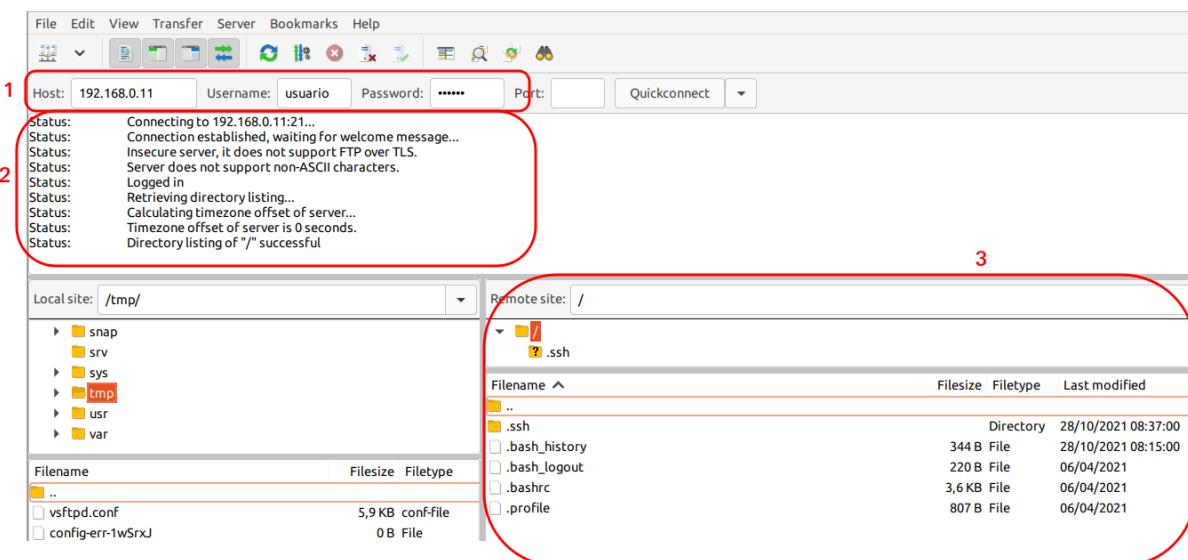
O primeiro passo é descobrir qual é o IP obtido pela máquina virtual por intermédio do DHCP, no caso da rede do autor o IP atribuído foi o ip 192.168.0.11, na casa do leitor este número vai variar dependendo da configuração de sua rede.

```
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo
 link/ether 08:00:27:fc:d0:aa brd ff:ff:ff:ff:ff:ff
 inet 192.168.0.11/24 brd 192.168.0.255 scope global dynamic e
 valid_lft 2200sec preferred_lft 2200sec
 inet6 2804:14c:bf41:9ee5:a00:27ff:fe:fc:d0aa/64 scope global d
 valid_lft 604753sec preferred_lft 518353sec
 inet6 fe80::a00:27ff:fe:fc:d0aa/64 scope link
 valid_lft forever preferred_lft forever
usuario@debian:~$ _
```

### 27.6.1 Cliente FTP gráfico FILEZILLA

No host (máquina real) o aluno deve instalar o cliente de FTP Filezilla, Filezilla cliente está disponível para Windows, Linux e MacOS, faça a instalação e inicie o cliente FTP.

A interface é semelhante em todos os Sistemas Operacionais, informe o IP da virtual machine em Host (campo 1) e o usuário do Linux e naturalmente sua senha, ao clicar em “Quickconnect” o log de conexão verbose será exibido no campo 2.



No campo 3 será carregado uma lista de diretórios e arquivos do usuário do Linux, esta é uma configuração em que cada usuário do Linux poderá acessar sua área privada.

## 27.6.2 Cliente FTP Command Line

Existem inúmeras ferramentas FTP, inclusive comandos que sem nenhuma interface gráfica se comunica com serviços FTP por meio do protocolo, isso decorre dos fatos:

- FTP é um protocolo extremamente simples;
- É uma solução amplamente utilizada;

Será utilizado o comando `ftp` no GNU/Linux, para iniciar uma conexão basta informar o IP do servidor FTP conforme figura abaixo.

```
well@wpo:~$ ftp 192.168.0.11
Connected to 192.168.0.11.
220 (vsFTPd 3.0.3)
Name (192.168.0.11:well): usuario
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Apó realizar a autenticação, conforme figura acima o terminal deverá ficar ativo em modo interativo aguardando comandos, conforme figura abaixo.

```
well@wpo:~$ ftp 192.168.0.11
Connected to 192.168.0.11.
220 (vsFTPd 3.0.3)
Name (192.168.0.11:well): usuario
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
ftp>
ftp> █
```

Na listagem abaixo, temos as opções do comando `ftp`:

- `cd`: Changes the current working directory on the FTP host server.
- `cwd`: Changes the current directory to the specified remote directory.
- `dir`: Requests a directory of files uploaded or available for download.
- `get`: Downloads a single file.
- `ls`: Requests a list of file names uploaded or available for download.
- `mget`: Interactively downloads multiple files.
- `mput`: Interactively uploads multiple files.
- `open`: Starts an FTP connection.
- `pasv`: Tells the server to enter passive mode, in which the server waits for the client to establish a connection rather than attempting to connect to a port the client specifies.
- `put`: Uploads a single file.
- `pwd`: Queries the current working directory.
- `ren`: Renames or moves a file.
- `site`: Executes a site-specific command.
- `type`: Sets the file transfer mode:

- ASCII
- Binary

## 27.7 Configurando um servidor no modo Passivo

A configuração por parte do servidor é simples, basta alterar a flag **pasv\_enable** para YES e informar qual o range de portas que serão utilizadas, estas portas deverão ser abertas no firewall da rede e naturalmente criada regras de roteamento iptables para a máquina com FTP Server.

```
pasv_enable=YES
pasv_min_port=40000
pasv_max_port=40100
```

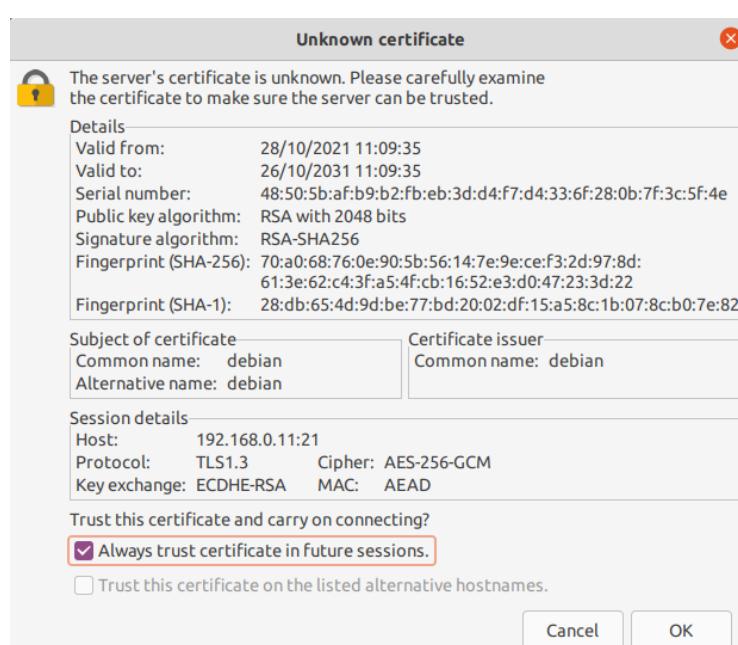
Mas saiba que a opção entre modo ativo ou passivo é do cliente.

## 27.8 Uso de certificado ssl

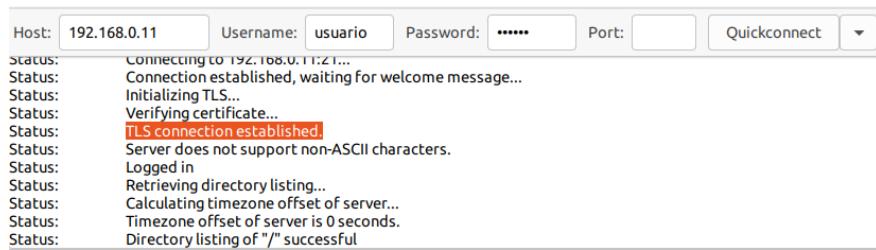
O uso de certificado digital é fundamental para se manter uma comunicação criptografada de tal forma que a descriptografia seja quase que impossível para um hacker com equipamentos normais. A configuração parte do servidor, no arquivo de configuração do vsftpd ative certificado utilizando YES para **ssl\_enable**.

```
encrypted connections.
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
ssl_enable=YES
```

Para demostrar, será utilizado o Filezilla, após realizar a conexão com o servidor (antes de enviar usuário e senha) é trocado dados sobre certificado, conforme visto na figura abaixo.



Já na figura abaixo é possível ver na área de log da ferramenta que é informado o uso de TLS como certificado, a explanação sobre os pacotes trafegando criptografados na rede será descrito no próximo tópico.



## 26.9 Recomendações de Segurança

trocar banner

portas

proteção por firewall

usuários e senhas

## 27.9 Análise da comunicação FTP

Como toda comunicação TCP o FTP inicia a conexão com um handshake de 3 vidas clássicas (ver figura abaixo parte 1).

Time	Source	Destination	Protocol	Length	Info
12625 2.996983729	192.168.0.9	192.168.0.11	TCP	74	52270 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
12628 2.997434048	192.168.0.11	192.168.0.9	TCP	74	21 → 52270 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA...
12629 2.997495314	192.168.0.9	192.168.0.11	TCP	66	52270 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=882985190 ...
12645 3.001806489	192.168.0.11	192.168.0.9	FTP	86	Response: 220 (vsFTPd 3.0.3)
12646 3.001861919	192.168.0.9	192.168.0.11	TCP	66	52270 → 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 TSval=882985194...
419610 182.768716647	192.168.0.9	192.168.0.11	FTP	80	Request: USER usuario
419611 182.769229189	192.168.0.11	192.168.0.9	TCP	66	21 → 52270 [ACK] Seq=21 Ack=15 Win=65280 Len=0 TSval=30677041...
419614 182.769688243	192.168.0.11	192.168.0.9	FTP	100	Response: 331 Please specify the password.
419615 182.769722763	192.168.0.9	192.168.0.11	TCP	66	52270 → 21 [ACK] Seq=15 Ack=55 Win=64256 Len=0 TSval=88316496...
426482 187.984731626	192.168.0.9	192.168.0.11	FTP	79	Request: PASS 123456
426489 188.015104399	192.168.0.11	192.168.0.9	FTP	89	Response: 230 Login successful.

1

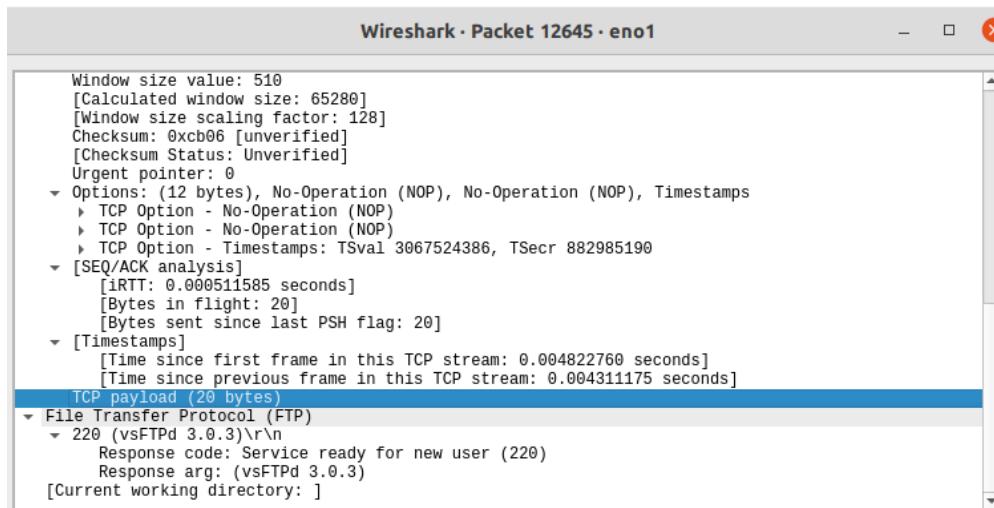
2

A máquina cliente está utilizando o comando ftp por command line e está no IP 192.168.0.9 e a máquina com vsftpd está respondendo pela porta 21 na máquina com ip 192.168.0.11.

### 27.9.1 Conexão ativa

Na conexão ativa após o Login quando é realizado o primeiro comando o servidor responde não a requisição do cliente no mesmo SOCKET, mas ele inicia um novo SOCKET no qual inicia-se no servidor e está apontado para o cliente.

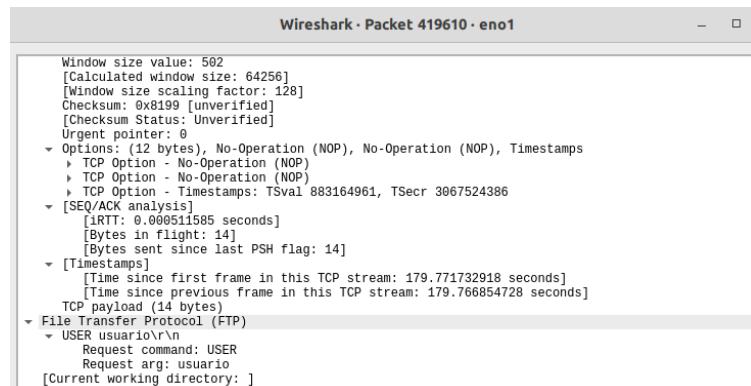
No início, após o Handshake um pacote FTP é enviado do servidor para o cliente neste SOCKET iniciado pelo cliente, veja na imagem acima a posição do pacote 12645 e abaixo o conteúdo deste envelope.



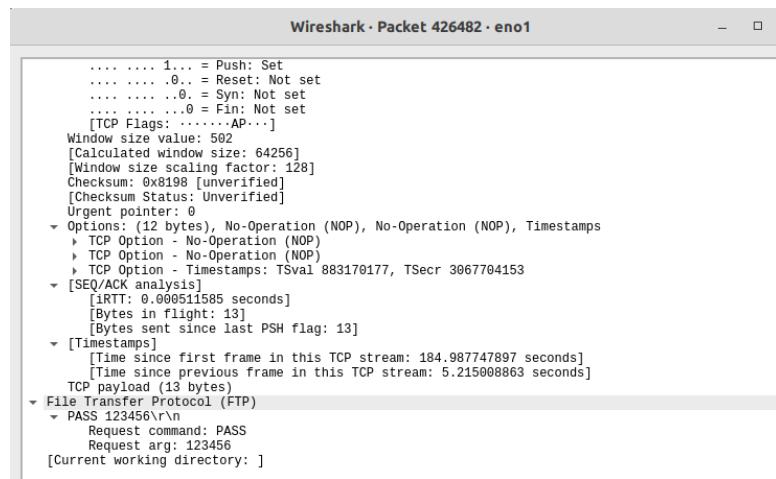
Nesta primeira resposta é enviado o código 220 indicando que o servidor FTP está ativo e esperando autenticação.

Time	Source	Destination	Protocol	Length	Info
12625 2.996983729	192.168.0.9	192.168.0.11	TCP	74	52270 - 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
12628 2.997434040	192.168.0.11	192.168.0.9	TCP	74	21 - 52270 [SYN, ACK] Seq=1 Ack=1 Win=65160 Len=0 MSS=1460 SA...
12629 2.997495314	192.168.0.9	192.168.0.11	TCP	66	52270 - 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=882985190 ...
12645 3.001806489	192.168.0.11	192.168.0.9	FTP	86	Response: 220 (vsFTPd 3.0.3)
12646 3.001861919	192.168.0.9	192.168.0.11	TCP	66	52270 - 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 TSval=882985194...
419610 182.768716647	192.168.0.9	192.168.0.11	FTP	80	Request: USER usuario
419611 182.769229189	192.168.0.11	192.168.0.9	TCP	66	21 - 52270 [ACK] Seq=21 Ack=15 Win=65280 Len=0 TSval=30677041...
419614 182.769688243	192.168.0.11	192.168.0.9	FTP	100	Response: 331 Please specify the password.
419615 182.769722763	192.168.0.9	192.168.0.11	TCP	66	52270 - 21 [ACK] Seq=15 Ack=55 Win=64256 Len=0 TSval=88316496...
426482 187.984731626	192.168.0.9	192.168.0.11	FTP	79	Request: PASS 123456
426489 188.015194390	192.168.0.11	192.168.0.9	FTP	89	Response: 230 Login successful.

Na parte 2 da figura acima temos a sequência de autenticação (não é a passiva), inicia-se com o cliente enviando o comando USER e em seguida o usuário cadastrado no Linux 192.168.0.11, então ao enviar o usuário o servidor responde com o código 331 pedindo que o cliente envie o password, pois este usuário possui um password (não é anônimo).



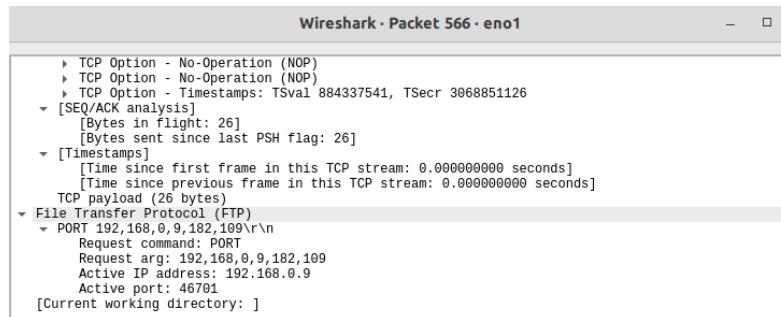
Repare que no pacote 426482 o cliente envia o comando PASS e o password do usuário, como texto e isso é muito criticado, veremos mais à frente como é a comunicação com certificado e vou voltar a falar sobre isso.



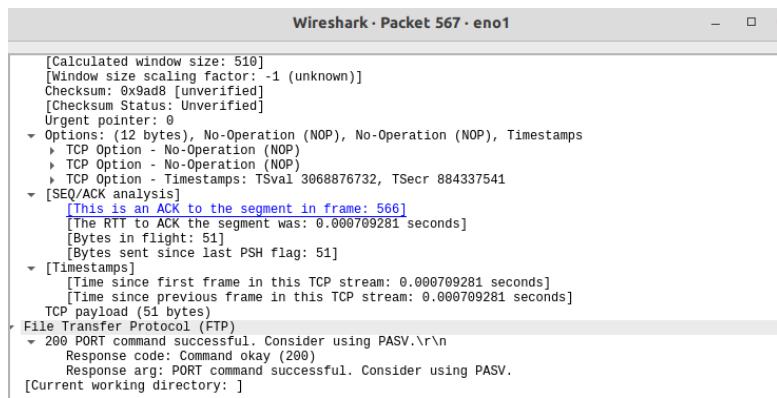
O próximo passo será envio do comando LIST, como o cliente ftp não tem a porta para comunicação então o primeiro passo é solicitar a porta com um request PORT, conforme pacote 566 na figura abaixo.

Time	Source	Destination	Protocol	Length	Info	
566 6.172550276	192.168.0.9	192.168.0.11	FTP	92	Request: PORT 192.168.0.9,182,109	1
567 6.173259557	192.168.0.11	192.168.0.9	FTP	117	Response: 200 PORT command successful. Consider using PASV.	
568 6.173304784	192.168.0.9	192.168.0.11	TCP	66	34894 → 21 [ACK] Seq=27 Ack=52 Win=502 Len=0 TSval=884337542 ...	2
569 6.173449066	192.168.0.9	192.168.0.11	FTP	72	Request: TST	
570 6.174313198	192.168.0.11	192.168.0.9	TCP	74	35561 → 46701 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK PERM=...	
571 6.174373853	192.168.0.9	192.168.0.11	TCP	74	46781 → 35561 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460...	
572 6.174667676	192.168.0.11	192.168.0.9	TCP	66	35561 → 46701 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3968876...	3
573 6.174933718	192.168.0.11	192.168.0.9	FTP	105	Response: 150 Here comes the directory listing.	
574 6.174972752	192.168.0.9	192.168.0.11	TCP	66	34894 → 21 [ACK] Seq=23 Ack=91 Win=502 Len=0 TSval=884337544 ...	

O servidor está atuando com modo ativo então o servidor envia uma mensagem informando que seria interessante usar o modo passivo (1 na figura acima), como está em modo ativo o próximo passo é o servidor iniciar uma nova conexão SOCKET (ver na figura acima em 2). Com a nova conexão estabelecida então o servidor envia a resposta conforme visto no pacote 573 na figura acima.



Na figura acima temos o pedido de PORT e na figura abaixo a resposta informando qual informa que é recomendável o uso da forma passiva de conexão.



## 27.9.2 Conexão passiva

A conexão modo ativo não teve tanto uso quanto a conexão passiva, visto que já foi descrito que na ativa há a necessidade de abrir regras de firewall nas máquinas clientes elevando o risco das estações de trabalho, principalmente quando o público é externo. Na conexão passiva vimos que o cliente se conecta e autentica na porta 21 e negocia uma outra porta para execução de comandos.

Neste cenário a manutenção do firewall é mais fácil, partindo do pressuposto que o cliente está devidamente autenticado, o recorte abaixo mostra a solicitação por parte do cliente do uso do modo passivo no pacote 55. Já no pacote 56 o servidor responde que aceita o modo passivo e envia a uma nova porta para conexão, no range de portas definido no arquivo de configuração do vsftpd.

55 5.931022382	192.168.0.9	192.168.0.11	FTP	72 Request: PASV	1
56 5.931772793	192.168.0.11	192.168.0.9	FTP	116 Response: 227 Entering Passive Mode (192,168,0,11,156,97).	
57 5.931818054	192.168.0.9	192.168.0.11	TCP	66 45390 - 21 [ACK] Seq=7 Ack=51 Win=502 Len=0 TSval=902656651 T...	
58 5.932299297	192.168.0.9	192.168.0.11	FTP	75 Request: LIST -a	
59 5.932491051	192.168.0.9	192.168.0.11	TCP	74 41783 - 40033 [SYN] Seq=0 Win=65535 MSS=1460 SACK_PERM=...	2
60 5.935553988	192.168.0.11	192.168.0.9	TCP	74 40033 - 41783 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460...	
61 5.935623189	192.168.0.9	192.168.0.11	TCP	66 41783 - 40033 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=9026566...	
62 5.936156484	192.168.0.11	192.168.0.9	FTP	105 Response: 150 Here comes the directory listing.	3
63 5.936292645	192.168.0.9	192.168.0.11	TCP	66 45390 - 21 [ACK] Seq=16 Ack=90 Win=502 Len=0 TSval=902656655 ...	
64 5.936598124	192.168.0.11	192.168.0.9	FTP-DATA	1148 FTP Data: 1082 bytes (PASV) (LIST -a)	
65 5.936651586	192.168.0.9	192.168.0.11	TCP	66 41783 - 40033 [ACK] Seq=1 Ack=1083 Win=68432 Len=0 TSval=9026...	
66 5.936718593	192.168.0.11	192.168.0.9	TCP	66 40033 - 41783 [FIN, ACK] Seq=1083 Ack=1083 Win=68432 Len=0 TSval=9026...	

No modo passivo (ver figura acima bloco 2) um novo SOCKET é iniciado do cliente para o servidor na nova porta, informada pelo servidor, é nesta direção que está a diferença entre o modo passivo e ativo, segue-se então o trâmite normal.

## 27.9.3 Uso de certificados

No próximo exemplo será utilizado Certificado para criptografia de mensagem, vimos que a senha é transferida sem certificado em forma de texto plano, para ativar o certificado no arquivo de configuração do servidor FTP deve-se definir YES no flag `ssl_enable`.

```
encrypted connections.
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
ssl_enable=YES
```

Quando se conecta com o cliente utilizando certificado a troca de comandos fica cifrada a partir de um ponto, veja no pacote 232 que há o comando AUTH TLS e logo em seguida a autenticação fica indecifrável.

## 27.10 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

## 27.10.1 Prática ftp0001 01: Configurando um serviço FTP

Nesta prática o aluno deve ter a expertise para instalar e configurar um serviço FTP para os usuários do GNU/Linux;

1. Realize um update do sistema com apt;
  2. Instale o serviço FTP com apt;
  3. Configure o serviço FTP para que seja um daemon iniciado na inicialização do Sistema Operacional;
  4. Usuário da máquina virtual podem ter acesso a seus diretórios bem como podem escrever arquivos nestes diretórios;

Execute o comando aied conforme comando listagem abaixo.

1. sudo aied validar ftp0001 checkpoint01

```
usuario@debian:~$ sudo aied validar ftp0001 checkpoint01
Ação que será executada: validar

Prática: Pratica da aula de FTP - instalando e configurando o FTP
Será enviado o OUTPUT do comando: systemctl status vsftpd.service --no-pager
Será enviado o OUTPUT do comando: dpkg -l | grep vsftpd
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: cat /etc/vsftpd.conf

Deseja continuar? (s para SIM) _
```

Após aceitar, será então validado todos os tópicos conforme figura abaixo, tudo deve estar verde.

```

Deseja continuar? (s para SIM) s
s1 - Comando: systemctl status vsftpd.service --no-pager
1.1 Validar por text 0/SUCCESS
1.2 Validar por text enabled;
2 - Comando: dpkg -l | grep vsftpd
2.1 Validar por regex /ii s+vsftpd/
3 - Comando: ip address
3.1 Validar por regex /inet s+d+. d+. d+. d+/
4 - Comando: cat /etc/vsftpd.conf
4.1 Validar por text listen=YES
4.2 Validar por text listen_ipv6=NO
4.3 Validar por text write_enable=YES
4.4 Validar por text chroot_local_user=YES
4.5 Validar por text allow_writeable_chroot=YES

Total de acertos: 9 do total de 9 validações equivalente a 100%.
Identificação: ad2c5925c2
AIED v(8)

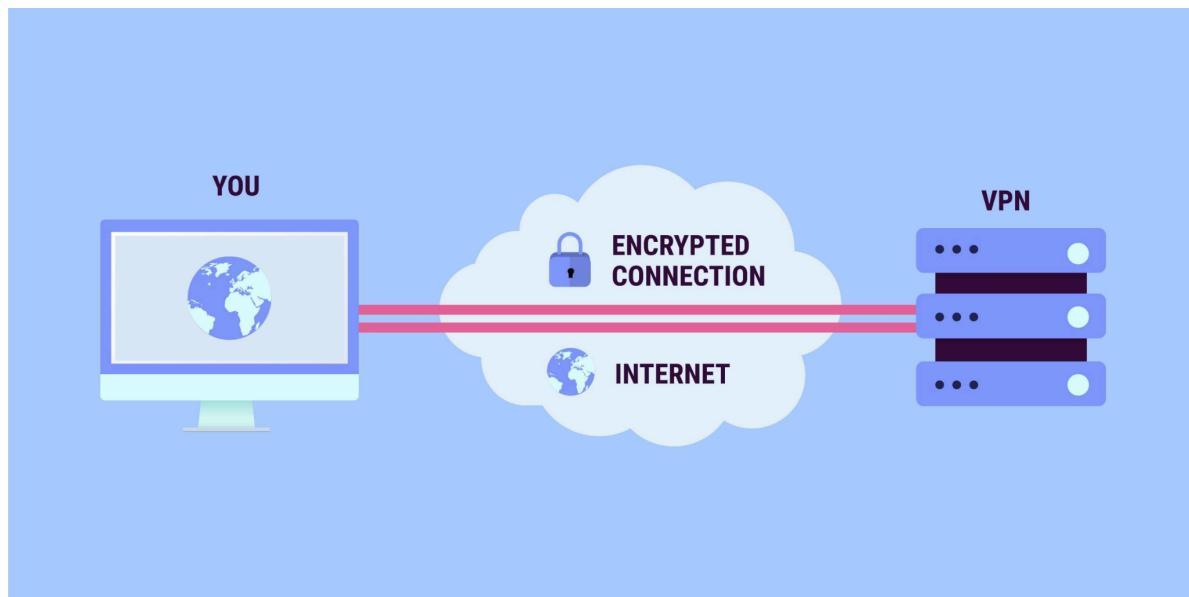
usuario@debian:~$ _

```

## 28 OpenVPN

VPN significa "segurança e privacidade", ou seja, proporciona um ambiente seguro para estabelecer uma conexão de rede protegida ao usar redes públicas. As VPNs criptografam seu tráfego e permite um caminho seguro na Internet, e também disfarçam sua identidade online quando bem configuradas, o que torna mais difícil para terceiros rastrear suas atividades online e roubar dados, a criptografia ocorre em tempo real.

Os dados não criptografados podem ser visualizados por qualquer pessoa que tenha acesso à rede e queira vê-los. Com uma VPN, os hackers e criminosos cibernéticos não podem decifrar esses dados. Sendo uma ferramenta fundamental para a segurança empresarial.



VPN está em alta, é um serviço obrigatório em organizações e agora presente no cotidiano de pessoas simples, que precisam de VPN para:

- **Acesso ao conteúdo regional:** o conteúdo regional da web nem sempre é acessível de qualquer lugar, os serviços e sites geralmente contêm conteúdo que só pode ser acessado em certas partes do mundo. As conexões padrão usam servidores locais no país para determinar sua localização, isso significa que você não pode acessar o conteúdo de sua casa durante uma viagem e não pode acessar o conteúdo internacional de sua casa, agora com o spoofing de localização VPN, você pode mudar de um servidor para outro país e efetivamente “mudar” sua localização;
- **Transferência segura de dados:** se você trabalha remotamente, pode precisar acessar arquivos importantes na rede da sua empresa. Por razões de segurança, este tipo de informação requer uma conexão segura, para obter acesso à rede, geralmente é necessária uma conexão VPN e os serviços VPN se conectam a servidores privados e usam métodos de criptografia para reduzir o risco de vazamento de dados.

Além do dito sobre a VPN o autor enxerga este recurso como um importante elemento na arquitetura de uma TI, geralmente uma organização possui uma zona desmilitarizada ou uma Intranet com serviços em teoricamente um ambiente seguro. Porém com a popularização do Home Office esses serviços de Intranet estão sendo disponibilizados na Internet. Geralmente um serviço WEB desenvolvido para uma Intranet não passa pelo rigor de segurança que passa uma aplicação para Internet, e por isso sempre são presas fáceis para os Hackers. Durante a crise sanitária de 2019 o autor recebeu um contato para atuar neste cenário a cada 3 dias, nesta crise as empresas simplesmente mandaram os funcionários para Home Office e colocaram os serviços de Intranet na Internet.

Com a VPN os funcionários em Home Office não precisam acessar um serviço na Internet, eles podem diretamente por VPN ingressar na Intranet e utilizar os serviços sem os expor na Internet.

## 28.1 Recursos do capítulo

Todo o conteúdo externo ao texto pode ser obtido no link abaixo.



O material deste capítulo pode ser obtido neste link.

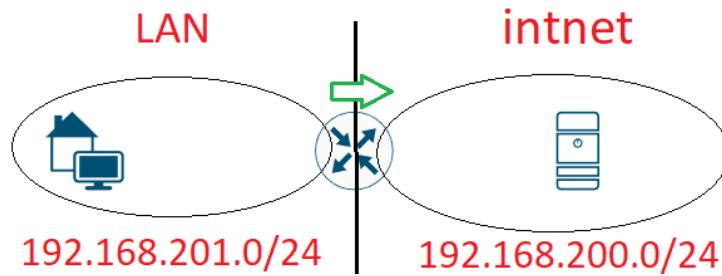
## 28.2 O ambiente

A configuração será muito simples visto que o aluno já possui uma bagagem de conhecimento excepcional, principalmente se seu ambiente já é Linux, será um pouco complicado se usar Microsoft Windows por exigir instalação de Putty e Winscp, conforme visto no capítulo [Openssh-Server](#).

Basicamente vamos ter três redes, a rede local que servirá de exemplo para uma Internet, a rede privada 192.168.200.0/24 que é a rede da empresa, é a Intranet e por último a rede virtual do tunelamento chamada 192.168.201.0/24, nesta rede de tunelamento todos os clientes estarão conectados. Na rede 192.168.200.0/24 temos um Host que terá um serviço

de Intranet. Já na sua rede local vamos ter uma máquina para simular um cliente na Internet.

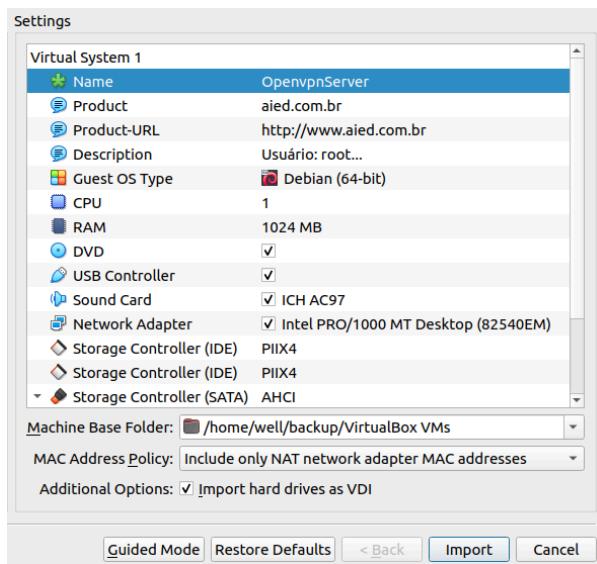
O objetivo desta prática é mostrar que a máquina na rede local vai ingressar na rede privada 192.168.201.0/24 e obter um IP, sendo assim ela poderá enviar ICMP para a máquina na Intranet e naturalmente provar a conectividade das duas máquinas por intermédio do roteamento forward da máquina **OpenvpnServer**.



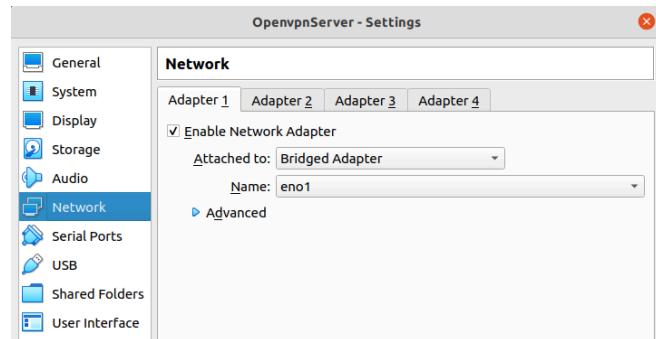
A planificação de IPs e redes será:

Virtual Machine		Network	Interface	IP/Mask
<b>OpenvpnServer</b>		LAN	enp0s3	Por DHCP
<b>OpenvpnServer</b>		intnet	enp0s8	192.168.200.1/24
<b>ClientVPN</b>		LAN	enp0s3	Por DHCP
<b>HostIntranet</b>		intnet	enp0s3	192.168.200.2/24

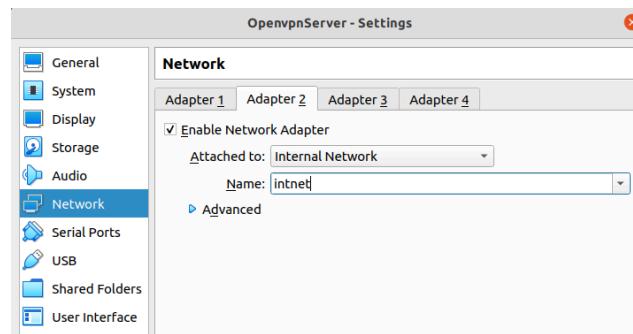
A configuração da máquina virtual **OpenvpnServer** é simples, importe um Debian GNU/Linux conforme figura abaixo. (Na rede privada será permitido acesso só por vpn)



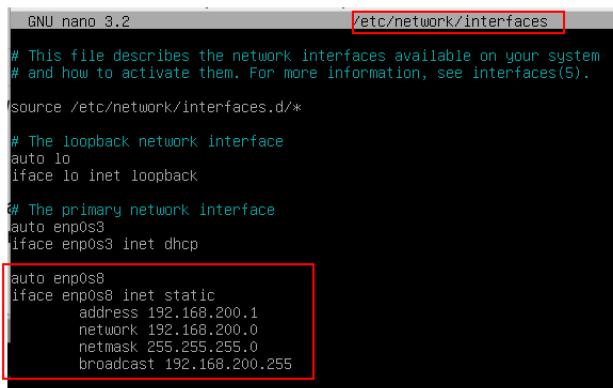
Após importar, configure o primeiro Adaptador de rede para Bridge Adapter conforme figura abaixo.



Esta interface será utilizada para o cliente se conectar neste servidor, é a rede local da casa do leitor que fará o papel de rede pública por motivos de simplificação. Já o segundo adaptador de rede deve estar configurado em “Internal Network” e a rede selecionada deverá ser intnet.

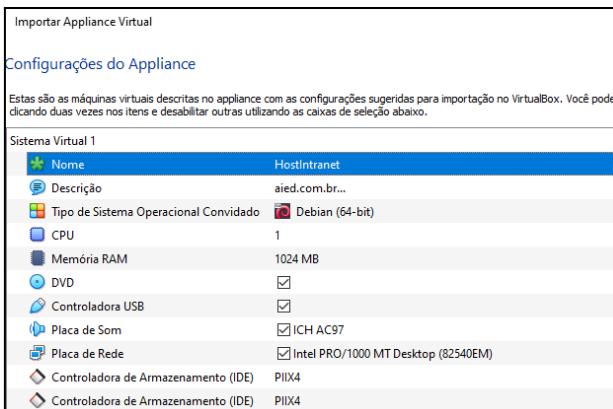


A rede **intnet** é a rede privada, a rede que seria a rede de uma empresa. Inicie o Servidor **OpenvpnServer**, edite o arquivo **/etc/network/interfaces** com o nano, informe os dados de rede da interface **enp0s8**.

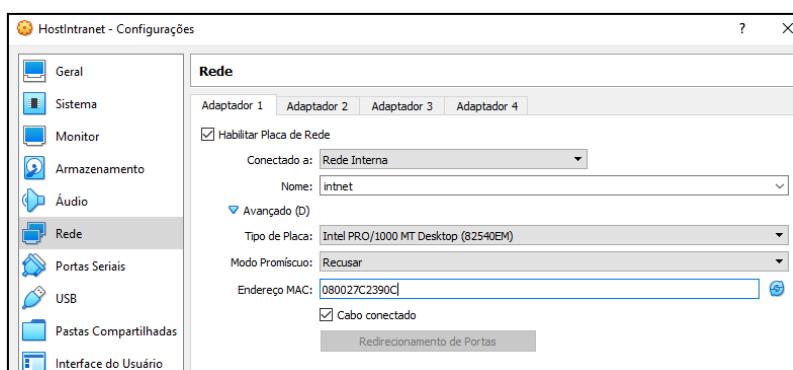


```
GNU nano 3.2 /etc/network/interfaces
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).
source /etc/network/interfaces.d/*
The loopback network interface
auto lo
iface lo inet loopback
The primary network interface
auto enp0s3
iface enp0s3 inet dhcp
auto enp0s8
iface enp0s8 inet static
 address 192.168.200.1
 network 192.168.200.0
 netmask 255.255.255.0
 broadcast 192.168.200.255
```

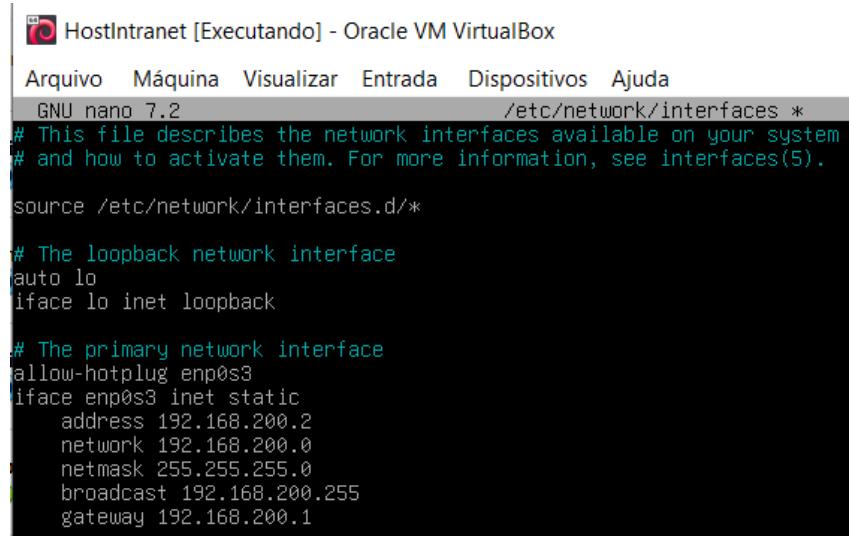
Esta rede 192.168.200.0/24 é a rede interna na qual seria localizado os equipamentos protegidos de uma rede corporativa, tal como um banco de dados, servidor de API, etc .. **Reinicie** esta máquina. Para exemplificar será implementada uma máquina sem serviço, basta um ICMP para garantir que é possível acessar. Crie uma nova Virtual Machine conforme parâmetros abaixo.



Esta máquina terá apenas 1 interface de rede, no Adaptador 1 escolha **Internal Network** e nome da rede **intnet**.



Ligue a máquina virtual e altere seu endereço IP no arquivo `/etc/network/interfaces`, conforme figura abaixo.



```

HostIntranet [Executando] - Oracle VM VirtualBox

Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
GNU nano 7.2 /etc/network/interfaces *
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

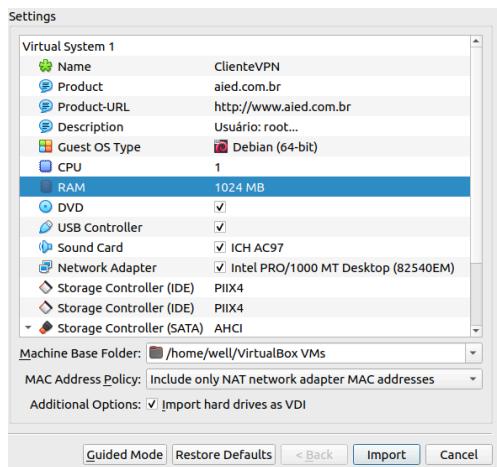
source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

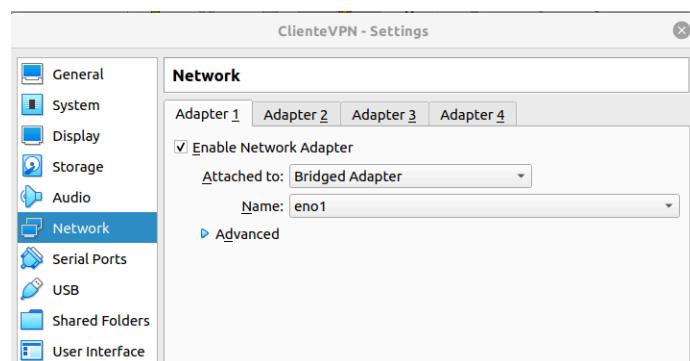
The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.2
 network 192.168.200.0
 netmask 255.255.255.0
 broadcast 192.168.200.255
 gateway 192.168.200.1

```

Após salvar o arquivo, **Reinicie** esta máquina. Para utilizar, por normatização será necessário criar uma terceira máquina virtual, caso use em seu host real um GNU/Linux não será necessário essa criação, o processo é simples, importe uma nova máquina virtual e a chame de **ClienteVPN** conforme imagem abaixo.



Após importação, no Adapter 1 selecione Bridge Adapter conforme figura abaixo, lembre-se de selecionar sua interface de rede que está ligada na rede LAN e altere seu MAC ADDRESS.



Inicie a máquina virtual **ClienteVPN**, e altere as configurações de rede conforme figura abaixo.

```

File Machine View Input Devices Help
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
auto enp0s3
iface enp0s3 inet dhcp

```

Após salvar o arquivo, reinicie esta máquina.

## 28.3 Easy-rsa

Vamos precisar do RSA, o Easy-RSA é um comando para gerenciar artefatos no padrão X.509 PKI, conhecido como infraestrutura de chave pública. Uma PKI é baseada na noção de confiar em uma autoridade específica para autenticar um par remoto. O código é escrito em shell POSIX de plataforma neutra, permitindo o uso em uma ampla variedade de sistemas operacionais. Algumas características do Easy-RSA:

- O Easy-RSA é capaz de gerenciar várias PKIs, cada uma com sua própria configuração independente, diretório de armazenamento e manipulação de extensão X.509.
- Várias opções de formatação de nome de assunto (campo X.509 DN) são suportadas. Para VPNs, isso significa que apenas uma configuração mais limpa de commonName pode ser usada.
- Um único back-end é usado em todas as plataformas suportadas, garantindo que nenhuma plataforma fique de fora dos recursos avançados.
- O suporte X.509 do Easy-RSA inclui CRL, CDP, atributos keyUsage/eKu e recursos adicionais.

Agora será feita a instalação dos pacotes necessários, para isso no terminal do servidor digite os comandos abaixo.

1. sudo apt update -y
2. sudo apt install easy-rsa -y
3. sudo apt install openvpn -y

Para listar os diretórios do easy-rsa navegue até /etc/share/easy-rsa/ e utilize o comando ls.

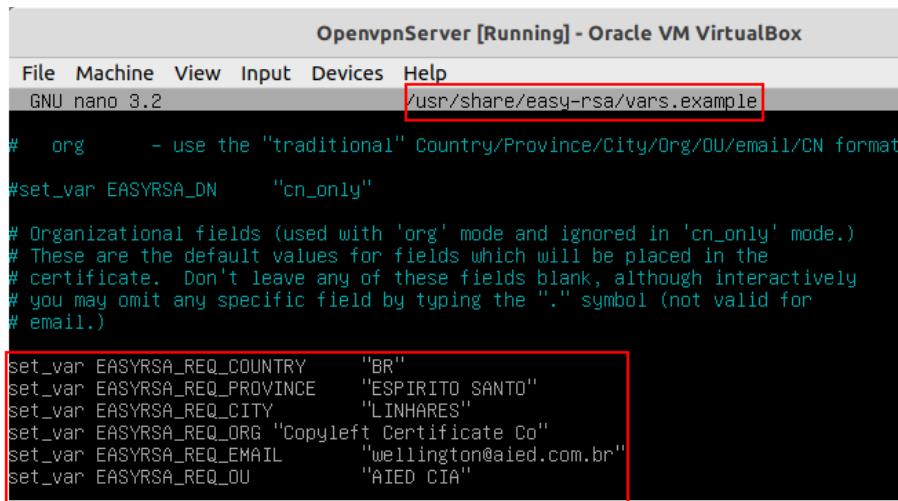
```

usuario@debian:~$ cd /etc/share/easy-rsa/
usuario@debian:/etc/share/easy-rsa$ ls -l
total 72
-rwxr-xr-x 1 root root 48730 Feb 8 2019 easyrsa
-rw-r--r-- 1 root root 4651 Feb 8 2019 openssl-easyrsa.cnf
-rw-r--r-- 1 root root 8576 Feb 8 2019 vars.example
drwxr-xr-x 2 root root 4096 Nov 18 10:44 x509-types
usuario@debian:/etc/share/easy-rsa$

```

## 28.4 Criando certificados para uso no OpenVPN

O primeiro objetivo é criar uma CA e a partir da CA criar certificados para os clientes. Mas antes será necessário informar alguns valores padrões para os certificados gerados nesta máquina, edite o arquivo `/usr/share/easy-rsa/vars.example`, informe estado, cidade e dados da corporação, conforme figura abaixo.



```

OpenvpnServer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 3.2
/usr/share/easy-rsa/vars.example

org - use the "traditional" Country/Province/City/Org/OU/email/CN format
#set_var EASYRSA_DN "cn_only"

Organizational fields (used with 'org' mode and ignored in 'cn_only' mode.)
These are the default values for fields which will be placed in the
certificate. Don't leave any of these fields blank, although interactively
you may omit any specific field by typing the "." symbol (not valid for
email.)

set_var EASYRSA_REQ_COUNTRY "BR"
set_var EASYRSA_REQ_PROVINCE "ESPIRITO SANTO"
set_var EASYRSA_REQ_CITY "LINHARES"
set_var EASYRSA_REQ_ORG "Copyleft Certificate Co"
set_var EASYRSA_REQ_EMAIL "wellington@aied.com.br"
set_var EASYRSA_REQ_OU "AIED CIA"

```

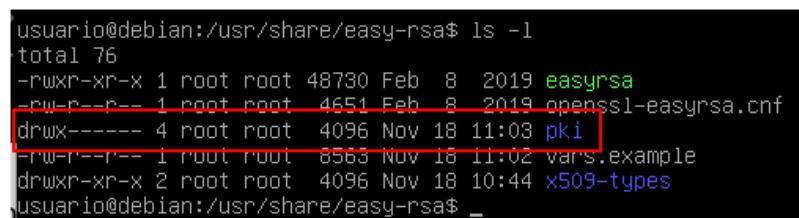
Alterado estas variáveis que serão usadas como base para emissão dos certificados auto-assinados, mas para ser utilizado então renomeie este arquivo para vars, conforme comando abaixo.

1. `sudo mv /usr/share/easy-rsa/vars.example /usr/share/easy-rsa/vars`

O próximo passo é criar o PKI, que será uma estrutura bem organizada de chaves, certificados e arquivos Diffie–Hellman, para isso execute o comando abaixo.

1. `cd /usr/share/easy-rsa/`
2. `sudo ./easysrsa init-pki`

Agora com um simples LS veja que foi criado um novo diretório chamado pki em `/easy-rsa/`.



```

usuario@debian:/usr/share/easy-rsa$ ls -l
total 76
-rwxr-xr-x 1 root root 48730 Feb 8 2019 easyrsa
-rw-r--r-- 1 root root 4651 Feb 8 2019 openssl-easyrsa.cnf
drwx----- 4 root root 4096 Nov 18 11:03 pki
-rw-r--r-- 1 root root 6563 Nov 18 11:02 vars.example
drwxr-xr-x 2 root root 4096 Nov 18 10:44 x509-types
usuario@debian:/usr/share/easy-rsa$

```

Agora será criado o primeiro certificado que será utilizado para criar os demais, então crie o CA com o `easysrsa` conforme comando abaixo.

1. `cd /usr/share/easy-rsa/`
2. `sudo ./easysrsa build-ca`

Será solicitado uma senha, como é um ambiente de estudo recomendo usar 123456 e manter o padrão de senha 123456, mas em ambiente profissional, jamais utilize.

```
usuario@debian:/usr/share/easy-rsa$ sudo ./easyrsa build-ca
Using SSL: openssl OpenSSL 1.1.1d 10 Sep 2019
Enter New CA Key Passphrase:
Re-Enter New CA Key Passphrase:
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Can't load /usr/share/easy-rsa/pki/.rnd into RNG
140398673888384:error:2406F079:random number generator:RAND_load_file:Cannot open
d/randfile.c:98:Filename=/usr/share/easy-rsa/pki/.rnd
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Common Name (eg: your user, host, or server name) [Easy-RSA CA]:openvpn
```

Utilize o Common Name openvpn, se associado ao X.500 pode-se definir o nome comum, após sucesso, veja que um path do arquivo ca.crt.

```
CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/usr/share/easy-rsa/pki/ca.crt
usuario@debian:/usr/share/easy-rsa$
```

## FALHA NO DEBIAN 12

No Debian 11 se chamar o comando passando o caminho completo (conforme imagem abaixo) irá ser executado normalmente, sem nenhuma falha, mas no Debian 12 se passar o caminho completo conforme figura abaixo ocorrerá o erro da imagem abaixo.

No texto adaptei para usar caminho relativo, mas nas imagens estou usando caminho completo e absoluto pois já tinha as imagens do Debian 11, e espero que no Debian 13 se resolva o problema.

```
usuario@debian:/usr/share/easy-rsa$ sudo /usr/share/easy-rsa/easyrsa gen-req vpnwell nopass
[sudo] password for usuario:
Sorry, try again.
[sudo] password for usuario:
Found: /usr/share/easy-rsa/vars
Found: /usr/share/easy-rsa/vars

Easy-RSA error:
Conflicting 'vars' files found.

Priority should be given to your PKI vars file:
* /usr/share/easy-rsa/pki/vars

Host: nix | Linux | /bin/bash
```

Então para resolver o problema acima deverá ser executado a seguinte sequência de

comandos:

```
cd /usr/share/easy-rsa/
./easyrsa gen-req vpnwell nopass
```

Um arquivo com extensão .crt é um arquivo de certificado de segurança usado por serviços seguros para estabelecer conexões seguras de um servidor para um cliente. Os arquivos CRT estão no formato ASCII e podem ser abertos em qualquer editor de texto para visualizar o conteúdo do arquivo de certificado, ele segue o padrão de certificado X.509 que define a estrutura do certificado. Ele define os campos de dados que devem ser incluídos no certificado SSL (exemplo), o CRT pertence ao formato PEM de certificados que são arquivos codificados em Base64 ASCII.

```
usuario@debian:/usr/share/easy-rsa$ sudo cat /usr/share/easy-rsa/pki/ca.crt
-----BEGIN CERTIFICATE-----
MIIDPzCCAiegAwIBAgIUE7vChfqgx/dsH7gn0MPEXvrspsdwDQYJKoZIhvcNAQEL
BQAwEjEQMA4GA1UEAwuHb3B1bn2wbJAeFw0yMTExMTgxNDEwNDdaFw0zMTExMTYx
NDEwNDdaMBIxEDA0BgNVBAMMB29wZk52cG4wgEiMA0GCSqGSIb3DQEBAQUA4IB
DuWuggEKaoIBAQDRk01rwI240mHeVr3p6oMn41U00vhcbvB1zKS+kAFJPMx04xRQ
idKhI72Qo3KY0k2wCvL+I8c1b99c8EYU1zTxbhhsD94K0Nx9SwC1/W1JJSWd+ch
FEsIQnRYRR1vsHUyzVHSMQNT+inYu32xApC8u17dTunDP/JVAzBQueLLc7zBjd
vUb2S8ekDQfIyg60+9JJj1L1+14T/GC17uravn3GfRmtcoB84S74nfF++0iyLz23
Xv1uYG13xyt3L1E2/x2FWTma18VUDuaQKpdNE0bY8M1w69320Sn0I1y12vNxx2R
W3Y2N/q8YLIrGgkageVhEcbKPDn0lsWeKG9AgMBAAgjgYuwgYKwHQYDVR00BBYE
FMlut62BLMjonPV+f7bMNh360tB/MEOGA1UdIwRGMESAFM1ut62BLMjonPV+f7bM
Nh360tB/oRakFDASMRAwDgYDVQDDAdvcGVudnBughQTu8KF+qDH92wfuCfQw8Re
+uy12zAMbgNVHRMEBTADAQH/MAsGA1UdDwQEAwIBBjANBkgkhkiG9w0BAQsFAA0C
AQEPASw086j0+8kdSk0WmKddpxZM+0Bee/CtrvU972+kpH16bDETzc21bTw5++pE2
4eJrtE2Y1rElQus6jG89Pjj5DvEAP12C1H05SNDI/+wrlkGtYFQdf/seWVvyx4
x6xJCL57Y8q7xmxi1Kj1BWInRavx3BjRLIv6g8RTn99u9xp0hS9aaNtkM/c7ryca
A2G7Y0v46oAyqP/a3VJCJv+0WoxQ9Jb2B001+KvAUWJCKHcedrSaRSP+XvjCWwrp+
okPP1k040aLUtp1vb/gjXno/ia061yx4tDpQ1dHLA2C10G+P9+NrYQRQcqcbT73Y
WAoGfzKJpwTdMouX1Ggu9HYKZQ==
-----END CERTIFICATE-----
usuario@debian:/usr/share/easy-rsa$
```

No próximo passo será criado um par de chaves para que acessem este serviço, para isso será criado então um par de chaves para openvpn-server conforme figura abaixo.

## 1. sudo ./easyrsa gen-req openvpn-server nopass

Veja na imagem o passo a passo.

```
usuario@debian:/usr/share/easy-rsa$ sudo ./easyrsa gen-req openvpn-server nopass
Using SSL: openssl OpenSSL 1.1.1d 10 Sep 2019
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/usr/share/easy-rsa/pki/private/openvpn-server.key.8i6e0j2nix'

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Common Name (eg: your user, host, or server name) [openvpn-server]:

Keypair and certificate request completed. Your files are:
req: /usr/share/easy-rsa/pki/reqs/openvpn-server.req
key: /usr/share/easy-rsa/pki/private/openvpn-server.key
usuario@debian:/usr/share/easy-rsa$ _
```

Ao terminar é informado onde serão salvos as chaves para o serviço, o nome usado acima openvpn-server é apenas um nome, que para manter um padrão se utiliza o mesmo nome do serviço para facilitar a manutenção futura. O próximo passo é assinar o certificado gerado para que seja autorizado o uso do certificado neste serviço.

1. cd /usr/share/easy-rsa/
2. sudo ./easyrsa sign-req server openvpn-server

Agora precisa assinar o certificado, para isso será usada a opção sign-req.

```
usuario@debian:/usr/share/easy-rsa$ sudo ./usr/share/easy-rsa/easyrsa sign-req server openvpn-server
Using SSL: openssl OpenSSL 1.1.1d 10 Sep 2019

You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 1080 days:
subject=
 commonName = openvpn-server

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes
Using configuration from /usr/share/easy-rsa/pki/safessl-easyrsa.cnf
Enter pass phrase for /usr/share/easy-rsa/pki/private/ca.key
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName :ASN.1 12:'openvpn-server'
Certificate is to be certified until Nov 2 14:27:10 2024 GMT (1080 days)

Write out database with 1 new entries
Data Base Updated
Certificate created at: /usr/share/easy-rsa/pki/issued/openvpn-server.crt
usuario@debian:/usr/share/easy-rsa$ _
```

O certificado openvpn-server deriva de ca e está devidamente assinado, este servidor irá auto-assinar seus certificados e será uma autoridade certificadora de seus serviços, o recomendado que fosse uma terceira máquina na comunicação.

Mas antes de gerar e configurar será necessário executar o Diffie-Hellman para sortear outros dados para geração de chaves, mais detalhes sobre criptografia serão descritos no tópico [Criptografia e ferramentas](#).

1. cd /usr/share/easy-rsa/
2. sudo ./easyrsa gen-dh

O processo pode demorar, mas pelo menos o mantém informado do progresso, conforme imagem abaixo.

Isso pode ser demorado, então pode obter neste momento em paralelo um café expresso.

## 28.5 Criando certificado por Cliente

Há inúmeras formas de trabalhar com VPN, muitas empresas possuem apenas um único certificado para todos os clientes VPN e aplicam a autenticação por LDAP, outras empresas aplicam um certificado para cada cliente.

Nesta prática cada cliente terá um certificado, pois caso seja desmembrado um funcionário a revogação do certificado é um processo de fácil execução. Para criar um certificado para um determinado agente, define-se um nome para o certificado que seja possível no futuro relacionar a este agente, conforme comando abaixo o certificado está sendo gerado para o usuário well. Será usado nopass para não exigir senha de acesso, caso queira, remova este parâmetro que o usuário será obrigado a informar uma senha.

1. cd /usr/share/easy-rsa/
  2. sudo ./easyrsa gen-req vpnwell nopass

```
usuario@debian:/usr/share/easy-rsa$ sudo /usr/share/easy-rsa/easyrsa gen-req vpnwell nopass
Using SSL: openssl OpenSSL 1.1.1d 10 Sep 2019
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/usr/share/easy-rsa/pki/private/vpnwell.key.98tchz5zyL'

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Common Name (eg: your user, host, or server name) [vpnwell]:

Keypair and certificate request completed. Your files are:
req: /usr/share/easy-rsa/pki/reqs/vpnwell.req
key: /usr/share/easy-rsa/pki/private/vpnwell.key

```

É gerado um par de chaves no servidor para este cliente, conforme figura acima, repare a posição dos diretórios pois arquivos serão enviados para o cliente para que ele possa usar seu certificado. Agora temos que assinar este certificado, pois este servidor além de um servidor com Openvpn (uma coisa) também é o servidor de autoridade (que é outra coisa), essa assinatura é feita com seu **ca.crt**.

1. `cd /usr/share/easy-rsa/`

## 2. sudo ./easyrsa sign-req client vpnwell

Veja a sequência de passos.

```
usuario@debian:/usr/share/easy-rsa$ sudo /usr/share/easy-rsa/easyrsa sign-req client vpnwell
Using SSL: openssl OpenSSL 1.1.1d 10 Sep 2019

You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a client certificate for 1080 days:
subject= commonName = vpnwell

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes
Using configuration from /usr/share/easy-rsa/pki/safessl-easyrsa.cnf
Enter pass phrase for /usr/share/easy-rsa/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName :ASN.1 12:'vpnwell'
Certificate is to be certified until Nov 2 14:43:09 2024 GMT (1080 days)
Write out database with 1 new entries
Data Base Updated
Certificate created at: /usr/share/easy-rsa/pki/issued/vpnwell.crt
usuario@debian:/usr/share/easy-rsa$
```

Agora criar um arquivo para o cliente, temos que criar um diretório para o usuário chamado well, neste diretório vamos adicionar os certificados, as chaves, para isso crie o diretório **/etc/openvpn/client/vpnwell** conforme listagem abaixo e copie os arquivos.

1. sudo mkdir /etc/openvpn/client/vpnwell/
2. sudo cp /usr/share/easy-rsa/pki/ca.crt /etc/openvpn/client/vpnwell/
3. sudo cp /usr/share/easy-rsa/pki/issued/vpnwell.crt /etc/openvpn/client/vpnwell/
4. sudo cp /usr/share/easy-rsa/pki/private/vpnwell.key /etc/openvpn/client/vpnwell/
5. sudo cp /usr/share/easy-rsa/pki/dh.pem /etc/openvpn/client/vpnwell/
6. sudo cp -r /etc/openvpn/client/vpnwell /home/userlinux/

Liste o diretório do usuário e confirme a existência dos arquivos, conforme figura abaixo.

```
usuario@debian:~$ ls -l /etc/openvpn/client/vpnwell/
total 20
-rw----- 1 root root 1188 Nov 18 12:57 ca.crt
-rw----- 1 root root 424 Nov 18 12:57 dh.pem
-rw----- 1 root root 4473 Nov 18 12:57 vpnwell.crt
-rw----- 1 root root 1704 Nov 18 12:57 vpnwell.key
usuario@debian:~$ _
```

O próximo passo é criar um arquivo de configuração para ser executado pelo cliente com o mínimo de esforço (por parte dele), crie o arquivo com o nano, conforme comando abaixo.

1. sudo nano /etc/openvpn/client/vpnwell/vpnwell.ovpn

neste arquivo edite conforme listagem abaixo (não precisa editar o que está comentado).

1. client

2. dev tun
3. proto udp
- 4.
5. **# Tem que colocar na linha abaixo o IP acessível do SERVIDOR COM OPENVPN**
6. **remote 192.168.0.16 1194**
- 7.
8. ca ca.crt
9. cert vpnwell.crt
10. key vpnwell.key
- 11.
12. tls-client
13. resolv-retry infinite
14. nobind
15. persist-key
16. persist-tun

Para que o usuário não se mate configurando uma VPN, vamos deixar toda a definição que será usada pelo openvpn no cliente, um arquivo de configuração pronto. Começa-se então definindo o nome da interface virtual que será criada para o tunelamento, que será **tun**, se o cliente tiver apenas uma VPN ativa com a interface **tun**, então o nome da interface virtual será **tun0**. Saiba que o protocolo utilizado é UDP, embora seja raro ter bloqueios de UDP na rede, não posso assumir que UDP seja aceito em 100% das redes de computadores no mundo inteiro, então fica o alerta.

Geralmente as empresas contratam um domínio ou criam um subdomínio com seu endereço público (IP), no nosso caso, como é uma prática local, vamos usar o endereço da interface enp0s3 da Virtual Machine **OpenvpnServer**. Também vamos indicar que a porta é 1194. Veja na figura abaixo como obter o IP que está nesta interface específica.

```

OpenvpnServer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~$
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 08:00:27:19:2c:ea brd ff:ff:ff:ff:ff:ff
 inet 192.168.0.16/24 brd 192.168.0.255 scope global dynamic enp0s3
 valid_lft 3082sec preferred_lft 3082sec
 inet6 2804:14c:bf41:9e5:a00:27ff:fe19:ccea/64 scope global dynamic mngtmpaddr
 valid_lft 604784sec preferred_lft 518384sec
 inet6 fe80::a00:27ff:fe19:ccea/64 scope link
 valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 08:00:27:81:ba:85 brd ff:ff:ff:ff:ff:ff
 inet 192.168.200.1/24 brd 192.168.200.255 scope global enp0s8
 valid_lft forever preferred_lft forever
 inet6 fe80::a00:27ff:fe81:ba85/64 scope link

```

Salvo o arquivo .ovpn temos que salvar no diretório **/home/userlinux/vpnwell** para extrair, mas como estamos usando sudo, não se esqueça de usar o chown para trocar o dono dos arquivos, afinal o administrador deve acessar com uma conta sem privilégios.

1. cp /etc/openvpn/client/vpnwell/vpnwell.ovpn /home/userlinux/vpnwell/

## 2. sudo chown -R usuario:usuario /home/userlinux/vpnwell

Geralmente o diretório **/home/userlinux/vpnwell** é compactado em um arquivo do tipo zip, e então enviado para o cliente. Faça assim nesta prática, extraia com WinSCP ou SCP para seu computador hospedeiro, depois envie também por estas ferramentas para a máquina **ClienteVPN**.

## 28.6 Configurando o servidor Openvpn

Na configuração desta prática um usuário com certificado poderá ingressar na rede 192.168.201.0/24 e então realizar operações na rede como se estivesse fisicamente ligado a ela, o servidor com a função ativa para roteamento FORWARD então o dará acesso a rede 192.168.200.0/24 no qual temos nossa Intranet, para entrar neste exemplo não será aplicada regra de autenticação por LDAP, atividade que será adicionada no capítulo LDAP no qual o assunto VPN será novamente abordado.

Partindo para a configuração do servidor OpenVPN entre no diretório **/etc/openvpn/server/** e crie um novo arquivo chamado **server.conf** conforme listagem abaixo.

### 1. sudo nano /etc/openvpn/server/server.conf

Com o nano edite o arquivo de acordo com a seguinte listagem, fique atento à digitação, principalmente no campo **tls-cipher**.

```
1. port 1194
2. proto udp
3. dev tun
4.
5. ca /usr/share/easy-rsa/pki/ca.crt
6. cert /usr/share/easy-rsa/pki/issued/openvpn-server.crt
7. key /usr/share/easy-rsa/pki/private/openvpn-server.key
8. dh /usr/share/easy-rsa/pki/dh.pem
9.
10. # As regras abaixo serão criadas no cliente, no cliente são visíveis com o comando ip route
11. server 192.168.201.0 255.255.255.0
12. push "route 192.168.200.0 255.255.255.0"
13. push "route 192.168.201.0 255.255.255.0"
14.
15. cipher AES-256-CBC
16. tls-version-min 1.2
17. tls-cipher
 TLS-DHE-RSA-WITH-AES-256-GCM-SHA384:TLS-DHE-RSA-WITH-AES-256-CBC-SHA25
 6:TLS-DHE-RSA-WITH-AES-128-GCM-SHA256:TLS-DHE-RSA-WITH-AES-128-CBC-SHA2
 56
18.
19. keepalive 20 60
20. persist-key
```

- 21. persist-tun
- 22.
- 23. log-append /var/log/openvpn.log
- 24. verb 3

A porta utilizada pelo serviço será a 1194 e será usado o protocolo UDP (camada de Transporte), se este serviço VPN estiver protegido atrás de um FIREWALL deverá ser criada a regra permitindo a passagem de protocolos UDP para a porta 1194. No servidor será criado uma interface de rede virtual chamada **tun0** (caso só exista uma).

Da linha 5 até a linha 9 estão sendo associados as chaves privada e pública bem como o arquivo PEM do Diffie–Hellman, como todos os certificados de clientes foram concebidos a partir destes eles poderão se comunicar.

Da linha 10 até 13 temos as regras de roteamento que serão criadas no dispositivo cliente, quando o cliente se conectar será criado no cliente a interface de rede **tun0** (se tiver somente uma) e também 3 regras, a primeira regra irá rotear toda a comunicação para 192.168.201.1 para a **tun0** do cliente, já a segunda deverá rotear toda a comunicação para a rede 192.168.200.0/24 para a **tun0** do cliente e por último será criado o roteamento para a rede 192.168.201.0/24 também pela **tun0** do cliente.

A criptografia que será utilizada por padrão é AES-256-CBC, uma criptografia muito comum e com boa resistência aos supercomputadores, naturalmente o protocolo mínimo de segurança será o TLS 1.2. No campo **tls-cipher** temos separado por dois pontos opções de criptografia para que o cliente utilize, na sequência. O cliente pode não ter todas as opções, será escolhida a mais próxima do início.

Pronto, o arquivo de configuração do servidor está pronto, para criar os arquivos dos clientes é necessário se ter o IP do servidor VPN, então no servidor VPN execute o comando ip address conforme figura abaixo.

```

OpenvpnServer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~$ usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 brd :: scope host
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 08:00:27:19:2c:ea brd ff:ff:ff:ff:ff:ff
 inet 192.168.0.16/24 brd 192.168.0.255 scope global dynamic enp0s3
 valid_lft 3082sec preferred_lft 3082sec
 inets 2804:14c:bf41:9ee5:a00:27ff:fe19:2cea/64 scope global dynamic mngtmpaddr
 valid_lft 604784sec preferred_lft 518364sec
 inets fe80::a00:27ff:fe19:2cea/64 scope link
 valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 08:00:27:81:ba:85 brd ff:ff:ff:ff:ff:ff
 inet 192.168.200.1/24 brd 192.168.200.255 scope global enp0s8
 valid_lft forever preferred_lft forever
 inets fe80::a00:27ff:fe81:ba85/64 scope link

```

Mas o openvpn está configurado porém não executa por si só, tal como openssh-server, então vamos criar um arquivo de inicialização para o Systemd,

basicamente podemos criar um arquivo chamado **vpn.service** no diretório **/etc/systemd/system/** e editar conforme imagem abaixo.

```
GNU nano 7.2 /etc/systemd/system/vpn.service *
[Unit]
Description=VPN Server AIED
After=network.target

[Service]
ExecStart=openvpn --config /etc/openvpn/server/server.conf

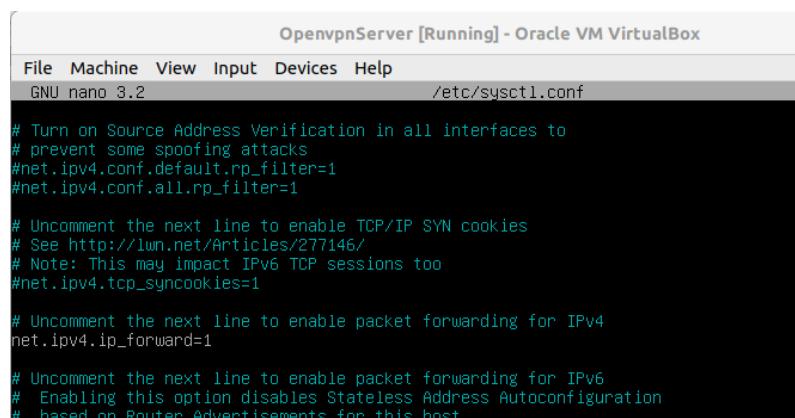
[Install]
WantedBy=multi-user.target
```

Quando o Linux for iniciado, após o evento de configuração de rede ser informado ao sistema, então será executado o comando **openvpn** passando com o parâmetro o arquivo **server.conf** editado anteriormente.

Para recarregar os arquivos de configuração pelo **systemctl** use o **daemon-reload**, ative o serviço para inicialização com o **enable** e inicie o serviço com **start**. Caso queira confirmar, use o **status**.

1. sudo systemctl daemon-reload
2. sudo systemctl enable vpn.service
3. sudo systemctl start vpn.service
4. sudo systemctl status vpn.service

Como a rede acessível será a rede 192.168.201.0/24 e a Intranet está na rede 192.168.200.0/24 temos que ativar no servidor a opção de roteamento FORWARD, para isso no arquivo **/etc/sysctl.conf** descomente a linha **net.ipv4.ip\_forward=1**, de forma que fique como na figura abaixo. Salve o arquivo.



```
OpenvpnServer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 3.2 /etc/sysctl.conf
Turn on Source Address Verification in all interfaces to
prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

Uncomment the next line to enable TCP/IP SYN cookies
See http://lwn.net/Articles/277146/
Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

Uncomment the next line to enable packet forwarding for IPv6
Enabling this option disables Stateless Address Autoconfiguration
based on Router Advertisements for this host
```

Será preciso reiniciar o servidor, neste ponto.

## 28.7 Criando um ambiente na Intranet

Na Intranet precisamos apenas de uma máquina virtual, se tiver um site web também serve, afinal a acessibilidade entre os hosts **ClienteVPN** e **HostIntranet** se dará pela camada de rede no modelo OSI. Confirme que a máquina **HostIntranet** possui só uma interface de rede está no IP 192.168.200.2/24, conforme imagem abaixo.

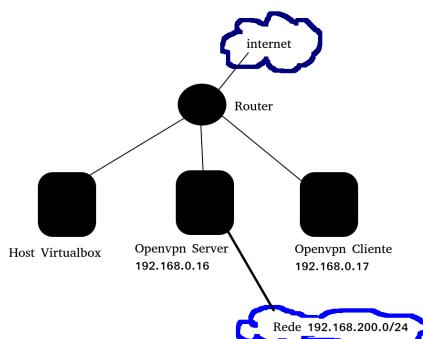
```
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 brd 0.0.0.0 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 brd 0.0.0.0 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP qlen 1000
 link/ether 08:00:27:c2:39:0c brd ff:ff:ff:ff:ff:ff
 inet 192.168.200.2/24 brd 192.168.200.255 scope global enp0s3
 valid_lft forever preferred_lft forever
 inet6 fe80::a00:27ff:fec2:390c/64 brd fe80::ff:ff:ff:ff:ff:ff scope link
 valid_lft forever preferred_lft forever
usuario@debian:~$
```

Também confirmou a conectividade do **HostIntranet** com o **OpenvpnServer** enviando protocolo **ICMP** para o endereço 192.168.200.1, conforme imagem abaixo.

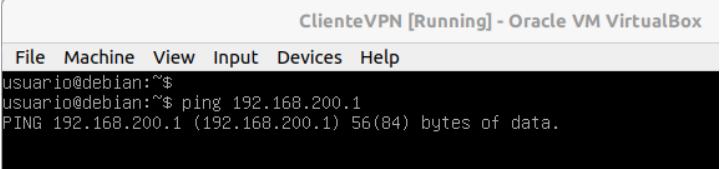
```
usuario@debian:~$ ping -c 1 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data.
64 bytes from 192.168.200.1: icmp_seq=1 ttl=64 time=0.948 ms
--- 192.168.200.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.948/0.948/0.948/0.000 ms
usuario@debian:~$ _
```

## 28.8 Utilizando em um cliente VPN

Em um mundo real o servidor VPN estará ligado diretamente em um IP público na rede mundial ou indiretamente atrás de um Firewall que este estará ligado em uma rede mundial, mas no exemplo desta prática a LAN da casa/escritório do leitor fará o papel desta rede mais abrangente, conforme esquema abaixo, já adicionei no esquema os IPs obtidos nesta prática 192.168.0.0/24 (minha casa) que está referente ao ambiente do autor, cada leitor (pessoa) encontrará outro valor.

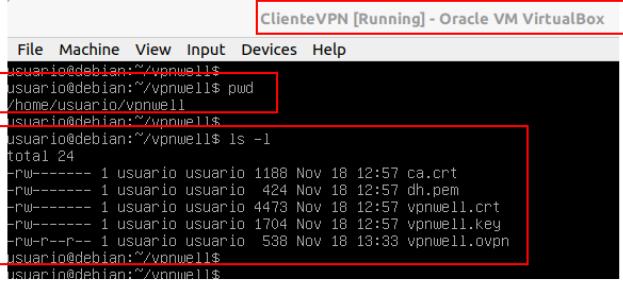


Um teste básico pode ser feito para demonstrar que o cliente não tem acesso a rede 192.168.200.0/24, um simples ping em 192.168.200.1, conforme imagem abaixo.



```
ClienteVPN [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~$ usuario@debian:~$ ping 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data.
```

Para o cliente acessar a rede 192.168.200.0/24 será necessário a configuração VPN para `vpnwell.zip`, em um mundo real os arquivos vpn são enviados para os clientes por um meio prático e seguro, tal como a instalação feita fisicamente ou por intermédio de uma mídia. Neste exemplo vou transferir os arquivos para o **ClienteVPN** por intermédio de um SSH.



```
ClienteVPN [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~/vpnwell$ usuario@debian:~/vpnwell$ pwd
/home/usuario/vpnwell
usuario@debian:~/vpnwell$ usuario@debian:~/vpnwell$ ls -l
total 24
-rw----- 1 usuario usuario 1188 Nov 18 12:57 ca.crt
-rw----- 1 usuario usuario 424 Nov 18 12:57 dh.pem
-rw----- 1 usuario usuario 4473 Nov 18 12:57 vpnwell.crt
-rw----- 1 usuario usuario 1704 Nov 18 12:57 vpnwell.key
-rw-r--r-- 1 usuario usuario 538 Nov 18 13:33 vpnwell.ovpn
usuario@debian:~/vpnwell$ usuario@debian:~/vpnwell$
```

Após transferir os arquivos instale no cliente o pacote `openvpn` conforme comando abaixo.

1. `sudo apt update -y`
2. `sudo apt install openvpn -y`

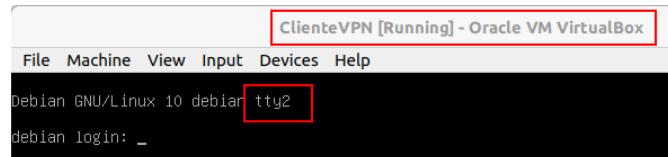
No cliente, conecte-se ao servidor VPN conforme comando abaixo (confirme se está no mesmo diretório do arquivo `vpnwell.ovpn`).

```

usuario@debian:~/vpnwell$ sudo openvpn --config ./vpnwell.ovpn
2024-05-22 20:45:19 Note: --cipher is not set. OpenVPN versions before 2.5 defaulted to BF-CBC as a
2024-05-22 20:45:19 fall-back when cipher negotiation failed in this case. If you need this fall-back please add '--data-cip
2024-05-22 20:45:19 hers-fallback BF-CBC' to your configuration and/or add BF-CBC to --data-ciphers.
2024-05-22 20:45:19 WARNING: file 'vpnwell.key' is group or others accessible
2024-05-22 20:45:19 OpenVPN 2.6.3 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] [DCO]
2024-05-22 20:45:19 library versions: OpenSSL 3.0.11 19 Sep 2023, LZO 2.10
2024-05-22 20:45:19 DCO version: N/A
2024-05-22 20:45:19 WARNING: No server certificate verification method has been enabled. See http://
2024-05-22 20:45:19 /openvpn.net/howto.html#mitm for more info.
2024-05-22 20:45:19 TCP/UDP: Preserving recently used remote address: [AF_INET]10.68.76.218:1194
2024-05-22 20:45:19 UDPv4 link local: (not bound)
2024-05-22 20:45:19 UDPv4 link remote: [AF_INET]10.68.76.218:1194
2024-05-22 20:45:19 [openvpn-server] Peer Connection Initiated with [AF_INET]10.68.76.218:1194
2024-05-22 20:45:19 TUN/TAP device tun0 opened
2024-05-22 20:45:19 net_iface_mtu_set: mtu 1500 for tun0
2024-05-22 20:45:19 net_iface_up: set tun0 up
2024-05-22 20:45:19 net_addr_ptp_v4_addr: 192.168.201.6 peer 192.168.201.5 dev tun0
2024-05-22 20:45:19 Initialization Sequence Completed

```

Como o Openvpn é bloqueador e ficará em execução, terá que acessar um segundo TTY (tela). Abra um segundo tty, com Ctrl-alt-f2 (no VirtualBox usar LEFT CTRL). Logue novamente com o usuário.



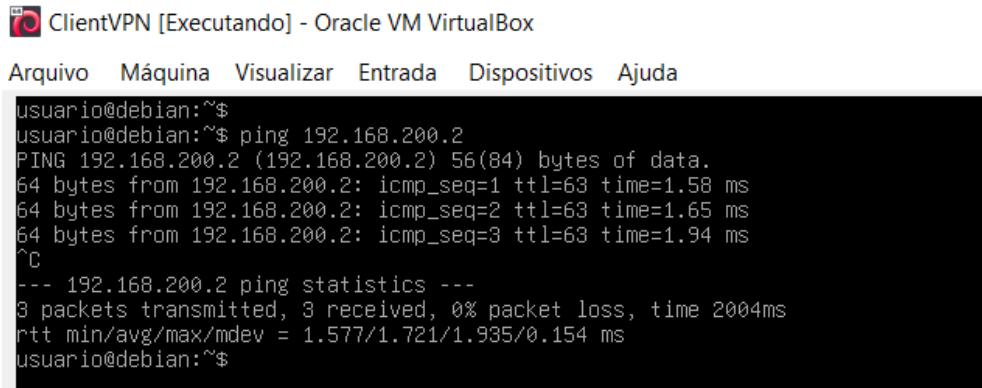
Após abrir e logar neste segundo TTY, você poderá então confirmar o sucesso da execução conforme comando ip address, confirme a criação de uma interface virtual chamada **tun0**.

```

usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
 link/ether 08:00:27:1c:31:77 brd ff:ff:ff:ff:ff:ff
 inet 10.68.76.205/23 brd 10.68.77.255 scope global dynamic enp0s3
 valid_lft 6176sec preferred_lft 6176sec
 inet6 fe80::a00:27ff:fe1c:3177/64 scope link
 valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
 link/none
 inet 192.168.201.6 peer 192.168.201.5/32 scope global tun0
 valid_lft forever preferred_lft forever
 inet6 fe80::1d0d:4d0f:84a:fd9a/64 scope link stable-privacy
 valid_lft forever preferred_lft forever
usuario@debian:~$

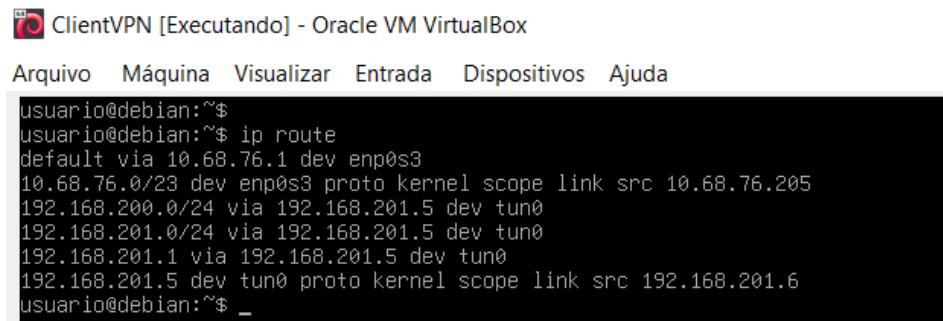
```

Já está confirmado, mas o interessante é ver algo ser transmitido para a rede 192.168.200.0/24, então um simples ICMP prova que a transmissão ocorreu, então execute novamente o ping que antes falhou do cliente na interface 192.168.200.1, conforme figura abaixo.



```
ClientVPN [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
usuario@debian:~$ usuario@debian:~$ ping 192.168.200.2
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.
64 bytes from 192.168.200.2: icmp_seq=1 ttl=63 time=1.58 ms
64 bytes from 192.168.200.2: icmp_seq=2 ttl=63 time=1.65 ms
64 bytes from 192.168.200.2: icmp_seq=3 ttl=63 time=1.94 ms
^C
--- 192.168.200.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 1.577/1.721/1.935/0.154 ms
usuario@debian:~$
```

Sucesso, temos conectividade entre a máquina fora da Intranet e a máquina dentro da Intranet, isso ocorre pois além do tunelamento há também 3 novas regras de roteamento, conforme já descrito, veja como ficam as regras.



```
ClientVPN [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
usuario@debian:~$ usuario@debian:~$ ip route
default via 10.68.76.1 dev enp0s3
10.68.76.0/23 dev enp0s3 proto kernel scope link src 10.68.76.205
192.168.200.0/24 via 192.168.201.5 dev tun0
192.168.201.0/24 via 192.168.201.5 dev tun0
192.168.201.1 via 192.168.201.5 dev tun0
192.168.201.5 dev tun0 proto kernel scope link src 192.168.201.6
usuario@debian:~$
```

Evolução:  
Ipsec e PPTP

## 29 Versionadores de código

### 29.1. Versionamento de código

## 29.2 Subversion

```
sudo apt-get update -y
sudo apt-get install apache2 -y
```

```
sudo apt-get install subversion libapache2-mod-svn libsvn-dev
```

```
sudo a2enmod dav
sudo a2enmod dav_svn
sudo service apache2 restart
```

```
/etc/apache2/mods-enabled/dav_svn.conf editar
Alias /svn /var/lib/svn
<Location /httpdoc-svn>
 DAV svn
 SVNParentPath /var/lib/svn
 AuthType Basic
 AuthName "Subversion Repository"
 AuthUserFile /etc/apache2/dav_svn.passwd
 Require valid-user
</Location>
```

```
sudo mkdir -p /var/lib/svn/
sudo svnadmin create /var/lib/svn/myrepo
sudo chown -R www-data:www-data /var/lib/svn
sudo chmod -R 775 /var/lib/svn
```

```
sudo htpasswd -cm /etc/apache2/dav_svn.passwd admin
sudo htpasswd -m /etc/apache2/dav_svn.passwd usuario
```

cliente

```
mkdir ~/projects/
cd ~/projects/
svn checkout http://192.168.3.13/httpdoc-svn/myrepo/
```

```
cd ~/projects/
svn add ./*
svn commit -m 'Uma carga de código inicial'
https://www.linode.com/docs/guides/subversion-svn-tutorial/
```

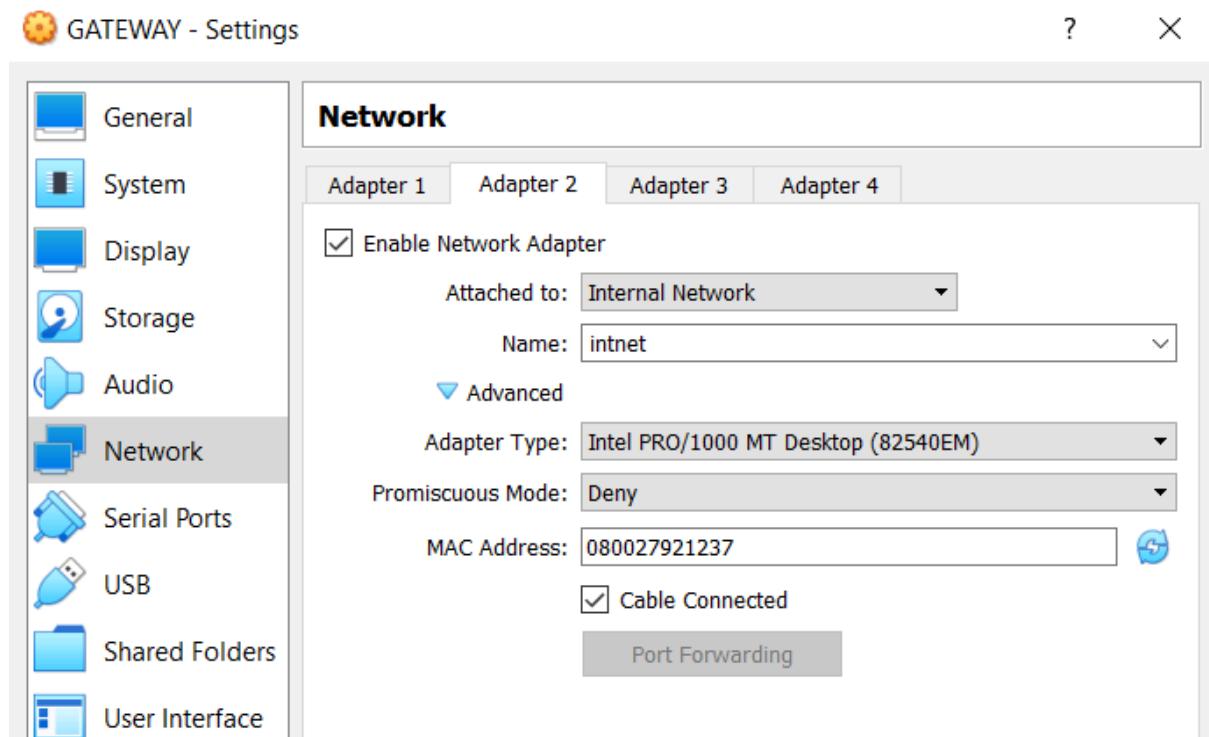
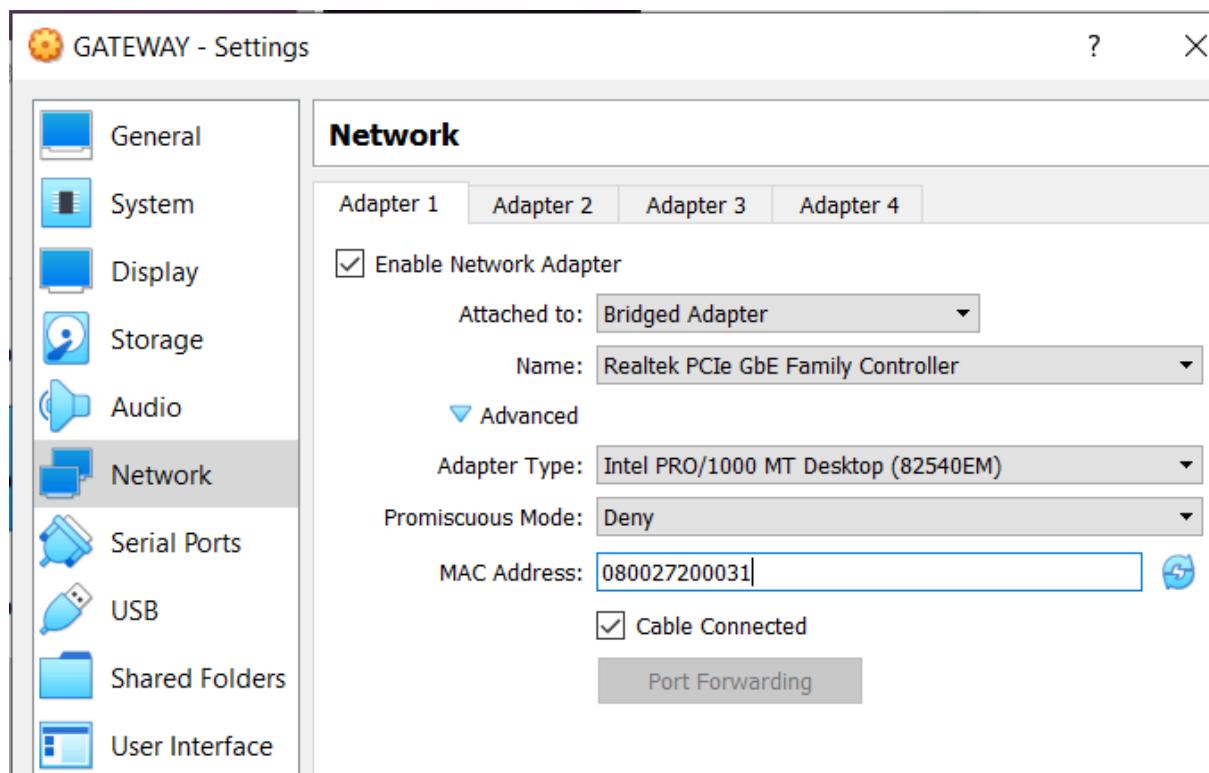
## 29.3 Git

# 30 Servidor de Virtualização

### Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported properties shown by double-clicking on the items and disable others using the check boxes below.

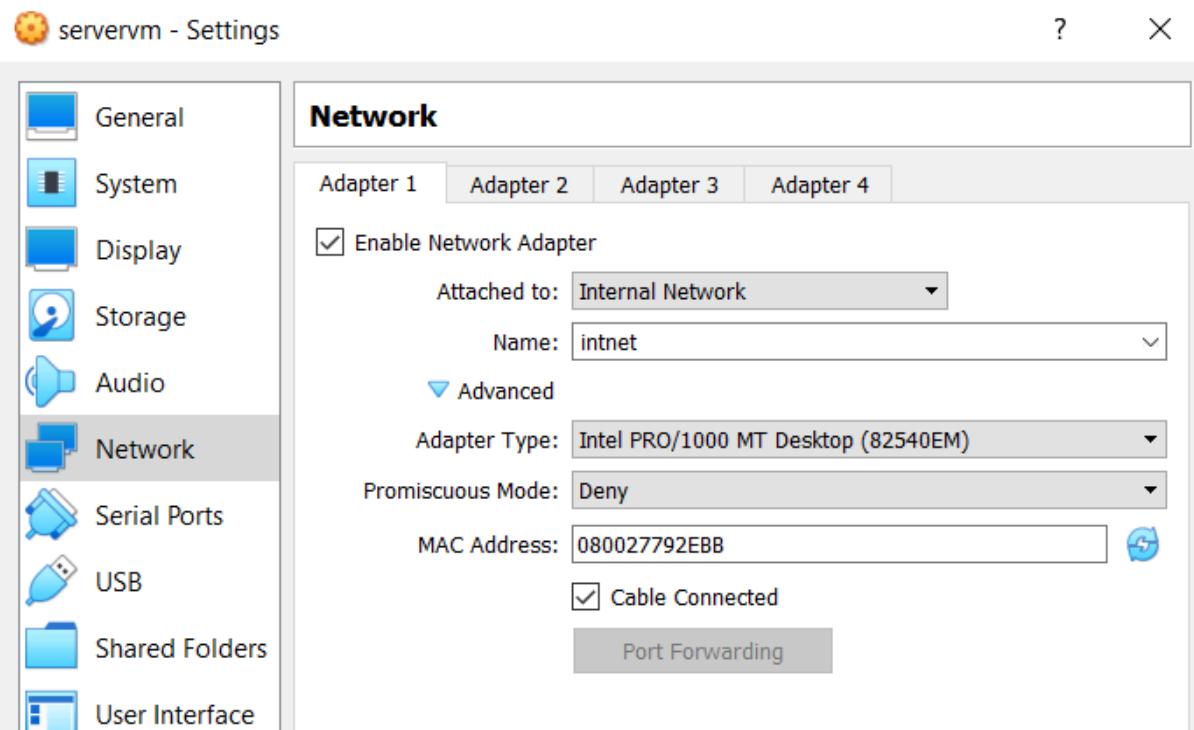
Virtual System 1	
 Name	GATEWAY
 Description	aied.com.br...
 Guest OS Type	 Debian (64-bit)
 CPU	1
 RAM	1024 MB
 DVD	<input checked="" type="checkbox"/>
 USB Controller	<input checked="" type="checkbox"/>
 Sound Card	<input checked="" type="checkbox"/> ICH AC97
 Network Adapter	<input checked="" type="checkbox"/> Intel PRO/1000 MT Desktop (82540EM)
 Network Adapter	<input checked="" type="checkbox"/> Intel PRO/1000 MT Desktop (82540EM)
 Storage Controller (IDE)	PIIX4
 Storage Controller (IDE)	PIIX4
 Storage Controller (SATA)	AHCI
 Virtual Disk Image	GATEWAY-disk001.vmdk
 Base Folder	C:\Users\dti\VirtualBox VMs
 Primary Group	/



## Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox mac properties shown by double-clicking on the items and disable others using the check boxes below.

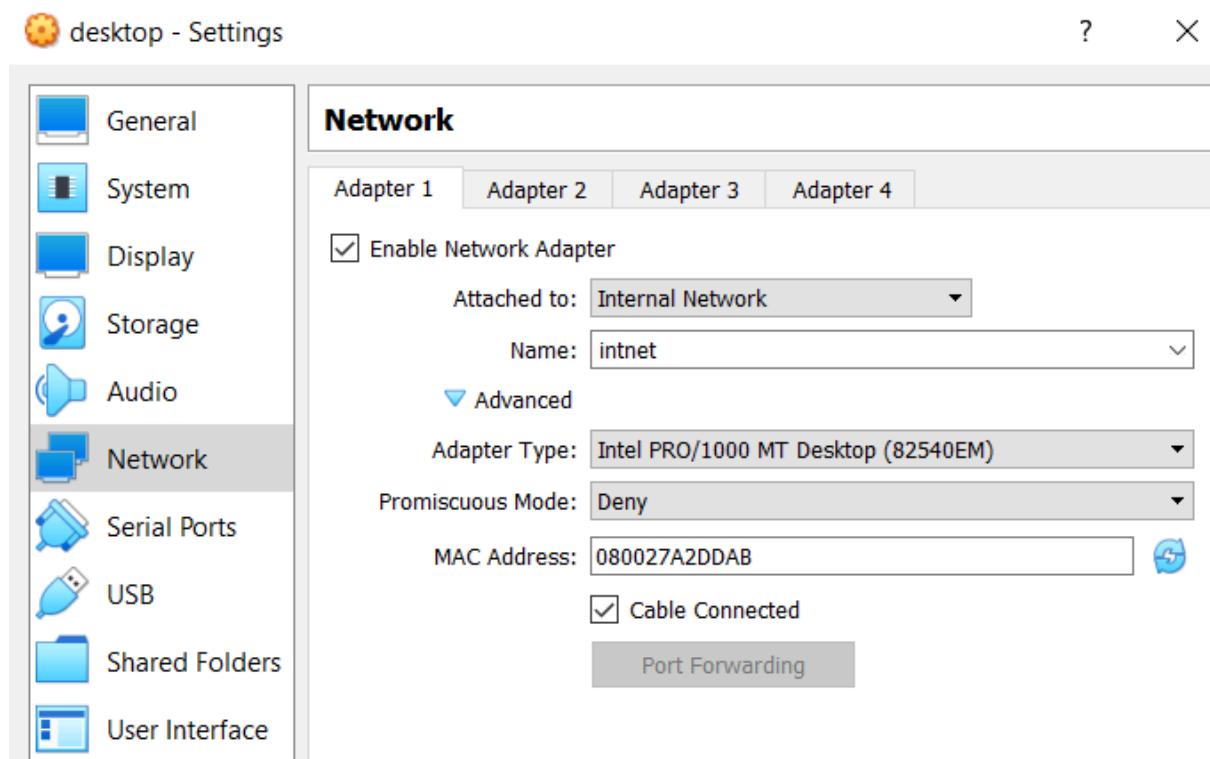
Virtual System 1	
 Name	servervm
 Description	aied.com.br...
 Guest OS Type	 Debian (64-bit)
 CPU	1
 RAM	4096 MB
 DVD	<input checked="" type="checkbox"/>
 USB Controller	<input checked="" type="checkbox"/>
 Sound Card	<input checked="" type="checkbox"/> ICH AC97
 Network Adapter	<input checked="" type="checkbox"/> Intel PRO/1000 MT Desktop (82540EM)
 Storage Controller (IDE)	PIIX4
 Storage Controller (IDE)	PIIX4
 Storage Controller (SATA)	AHCI
 Virtual Disk Image	Debian_12_64_bits_root_123456_usuario_123456-disk001.vmdk
 Base Folder	C:\Users\dti\VirtualBox VMs
 Primary Group	/



## Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machine properties shown by double-clicking on the items and disable others using the check boxes below.

Virtual System 1	
 Name	desktop
 Description	aied.com.br...
 Guest OS Type	 Debian (64-bit)
 CPU	1
 RAM	2048 MB
 DVD	<input checked="" type="checkbox"/>
 USB Controller	<input checked="" type="checkbox"/>
 Sound Card	<input checked="" type="checkbox"/> ICH AC97
 Network Adapter	<input checked="" type="checkbox"/> Intel PRO/1000 MT Desktop (82540EM)
 Storage Controller (IDE)	PIIX4
 Storage Controller (IDE)	PIIX4
 Storage Controller (SATA)	AHCI
 Virtual Disk Image	Debian_12_64_bits_root_123456_usuario_123456-disk001.vmdk
 Base Folder	C:\Users\dti\VirtualBox VMs
 Primary Group	/



---

 servervm [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

GNU nano 7.2 /etc/network/interfaces

```
This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

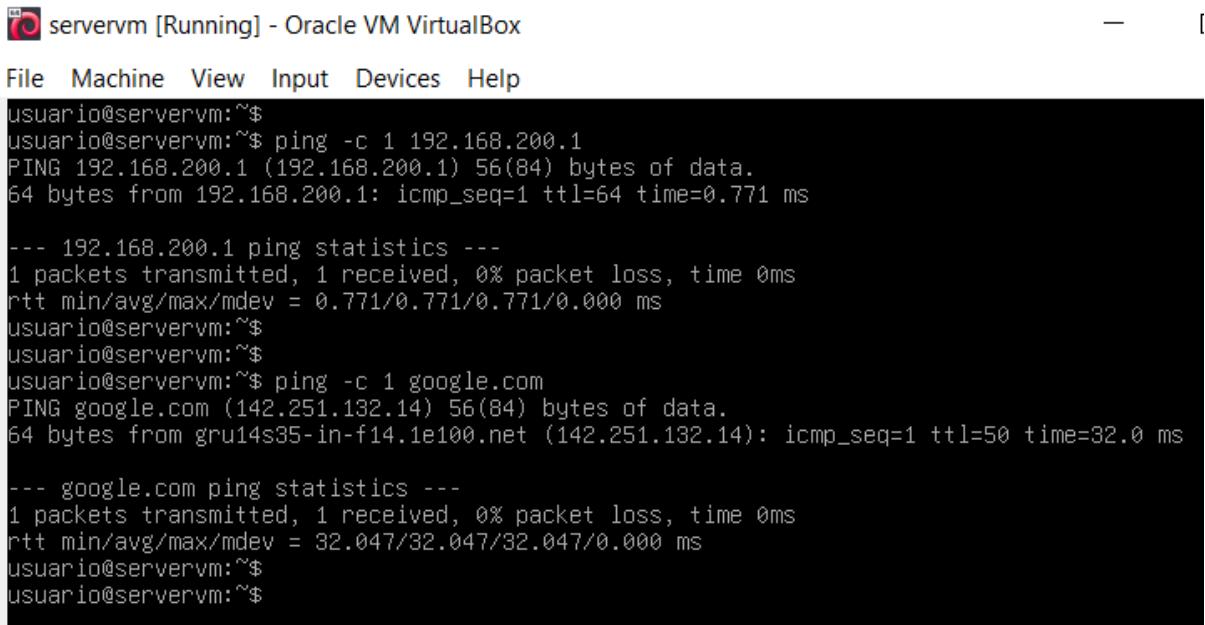
The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.15
 netmask 255.255.255.0
 network 192.168.200.0
 gateway 192.168.200.1
 dns-nameservers 8.8.8.8
```

 servervm [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

```
usuario@servervm:~$
usuario@servervm:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
 link/loopback 00:00:00:00:00 brd 00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
 link/ether 08:00:27:79:2e:bb brd ff:ff:ff:ff:ff:ff
 inet 192.168.200.15/24 brd 192.168.200.255 scope global enp0s3
 valid_lft forever preferred_lft forever
 inet6 fe80::a00:27ff:fe79:2ebb/64 scope link
 valid_lft forever preferred_lft forever
usuario@servervm:~$
```

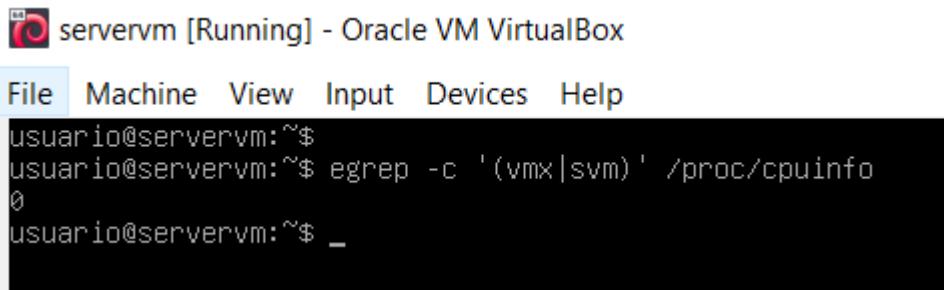


```
File Machine View Input Devices Help
usuario@servervm:~$ ping -c 1 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data.
64 bytes from 192.168.200.1: icmp_seq=1 ttl=64 time=0.771 ms

--- 192.168.200.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.771/0.771/0.771/0.000 ms
usuario@servervm:~$ ping -c 1 google.com
PING google.com (142.251.132.14) 56(84) bytes of data.
64 bytes from gru14s35-in-f14.1e100.net (142.251.132.14): icmp_seq=1 ttl=50 time=32.0 ms

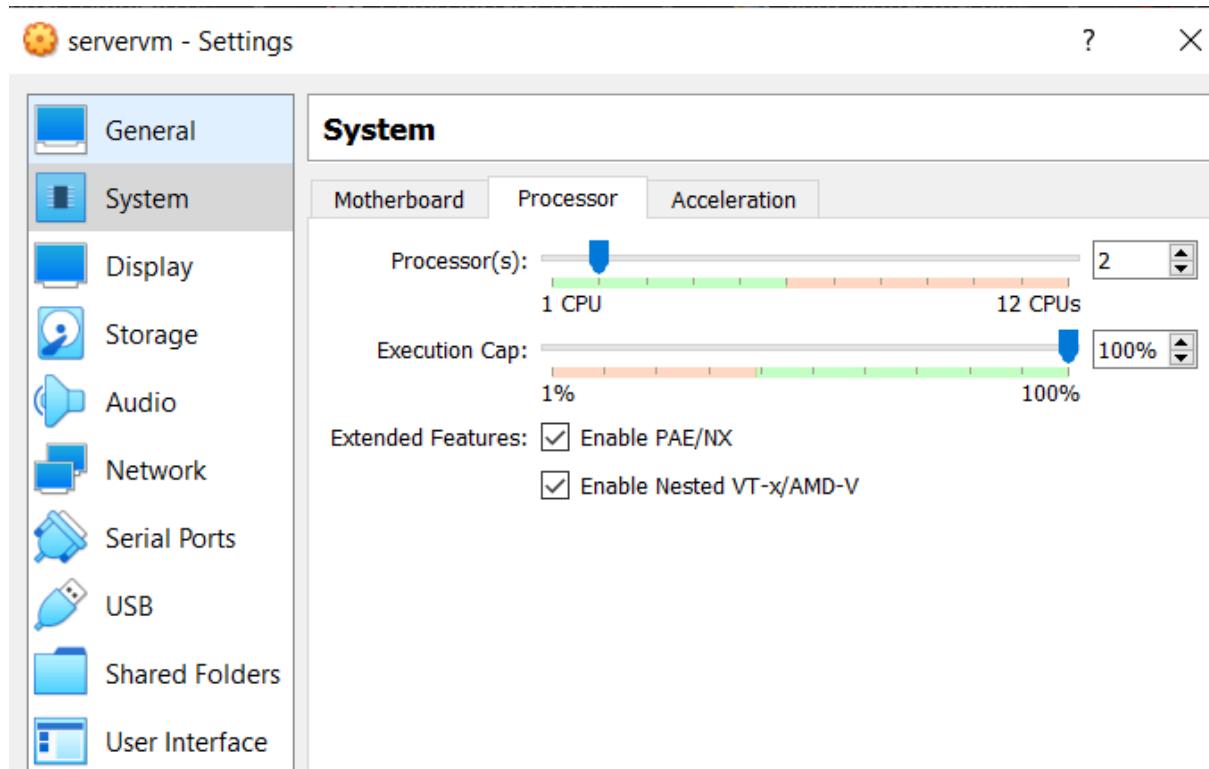
--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 32.047/32.047/32.047/0.000 ms
usuario@servervm:~$ sudo apt update -y
usuario@servervm:~$
```

sudo apt update -y



```
File Machine View Input Devices Help
usuario@servervm:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
0
usuario@servervm:~$ _
```

```
C:\Program Files\Oracle\VirtualBox>VBoxManage.exe modifyvm servervm --nested-hw-virt on
C:\Program Files\Oracle\VirtualBox>
```





# 31 Serviço de controle de Domínio OpenLDAP (atuando)

O protocolo LDAP é um protocolo de rede que provê informações de estrutura tecnológica da organização, se tornando um rico banco que é chave para organização da TI, que contém além de cadastro de usuário também cadastro de computadores, diretórios em rede, etc.

O termo LDAP significa Lightweight Directory Access Protocol como o nome indica, o LDAP foi originalmente projetado para ser um protocolo de rede que fornecia uma forma alternativa de acesso aos servidores de diretório existentes, mas à medida que a ideia do LDAP amadureceu, o termo LDAP se tornou sinônimo de um tipo específico da arquitetura do diretório.

## 31.1 X.500

Com o objetivo de organização de dados tecnológicos sobre a organização, a ITU-T desenvolve um padrão X.500, que é formado por uma série de padrões para redes de computador abordando serviço de diretório. A ISO incorpora este padrão por meio de parceria

Os protocolos definidos pelo X.500:

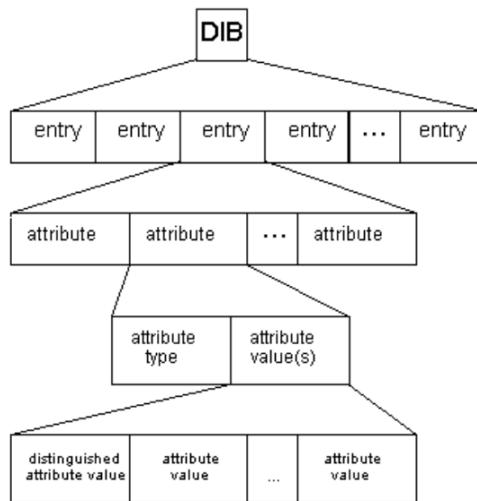
- DAP** (Directory Access Protocol)
- DSP** (Directory System Protocol)
- DISP** (Directory Information Shadowing Protocol)
- DOP** (Directory Operational Bindings Management Protocol)

Por ser usados no Modelo OSI, um número de alternativas ao DAP foram desenvolvidas para permitir que clientes de Internet pudessem acessar o diretório X.500 usando TCP/IP, a mais conhecida alternativa é o LDAP, e por permitir que o DAP e outros protocolos X.500 pudessem usar TCP/IP, LDAP é um conhecido protocolo de acesso a diretórios.

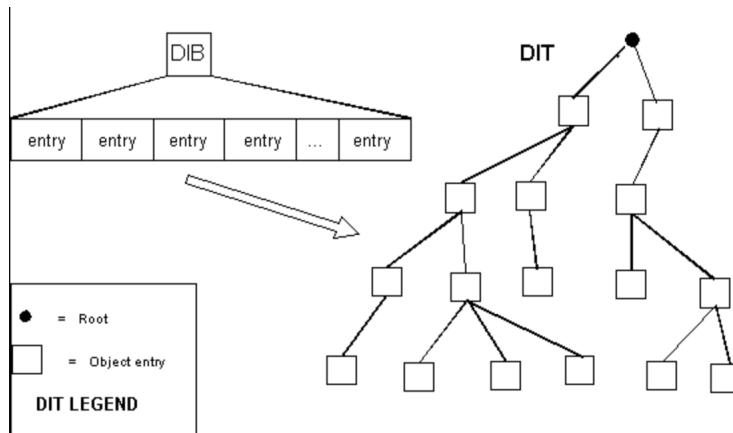
### 31.1.1 Object Entry

O conjunto completo de todas as informações contidas no diretório é conhecido como Directory Information Base (DIB), neste banco de dados há inúmeras informações, inclusive informações que não são vistas por usuários normais, estas informações estão em objetos ou "entradas" neste banco de dados, então o DIB consiste em entradas e essas entradas estão relacionadas hierarquicamente entre si.

No Directory User Information Model, uma entrada contém informações sobre um objeto de interesse para os usuários do Directory. Esses objetos (Diretório) podem estar normalmente associados a algum recurso do mundo real, tal como um computador, um diretório em rede, uma impressora, etc.. No entanto, é muito importante observar que os objetos Directory não têm necessariamente uma correspondência um-para-um com coisas do mundo real.



Objetos que possuem características semelhantes são identificados por sua classe de objeto. As entradas mantidas no DIB são estruturadas de forma hierárquica, usando uma estrutura de árvore, que é semelhante a um gráfico de estrutura usado pela maioria das organizações hierárquicas, então o DIB pode, portanto, ser representado como uma Directory Information Tree (DIT), em que cada nó na árvore representa uma entrada do diretório.



Abaixo da raiz do DIT, normalmente haverá entradas que representam países ou organizações multinacionais e, abaixo delas, entradas que representam organizações e unidades organizacionais (divisões) respectivamente, e sim, há uma relação histórica entre este protocolo e o DNS, mas não os confunda e o X.500 provê muito mais informações sobre os recursos.

Freqüentemente, será uma tarefa relativamente simples mapear a estrutura de uma organização em parte do DIT. No entanto, as organizações não precisam seguir rigidamente sua estrutura organizacional ao projetar sua estrutura DIT. Eles podem optar por uma estrutura mais plana no DIT, de forma que os navegadores não possam determinar a estrutura da organização.

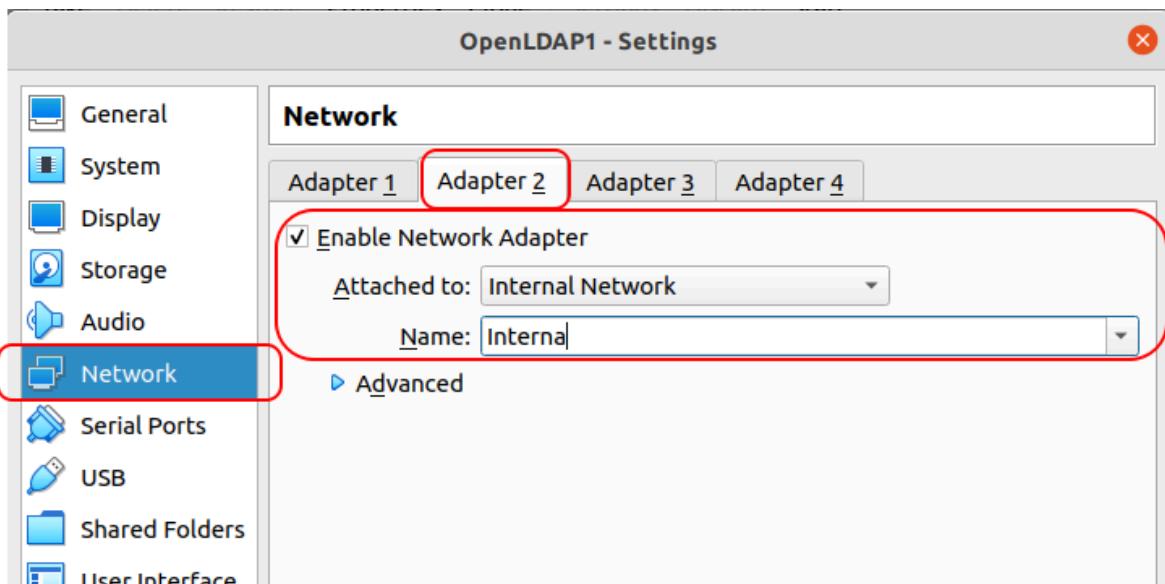
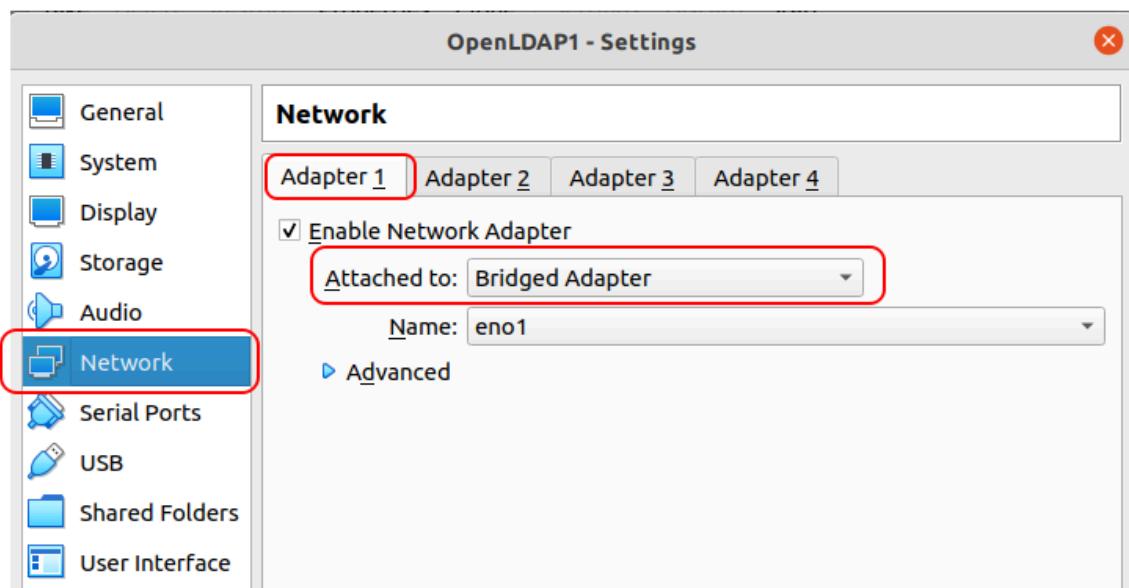
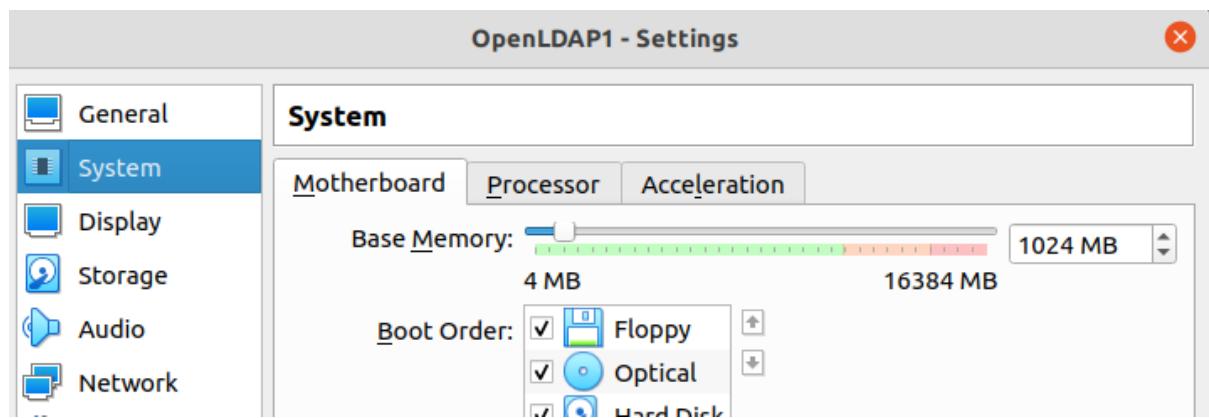
## 31.2 LDAP

### 31.2.1 Directory Service e Entry

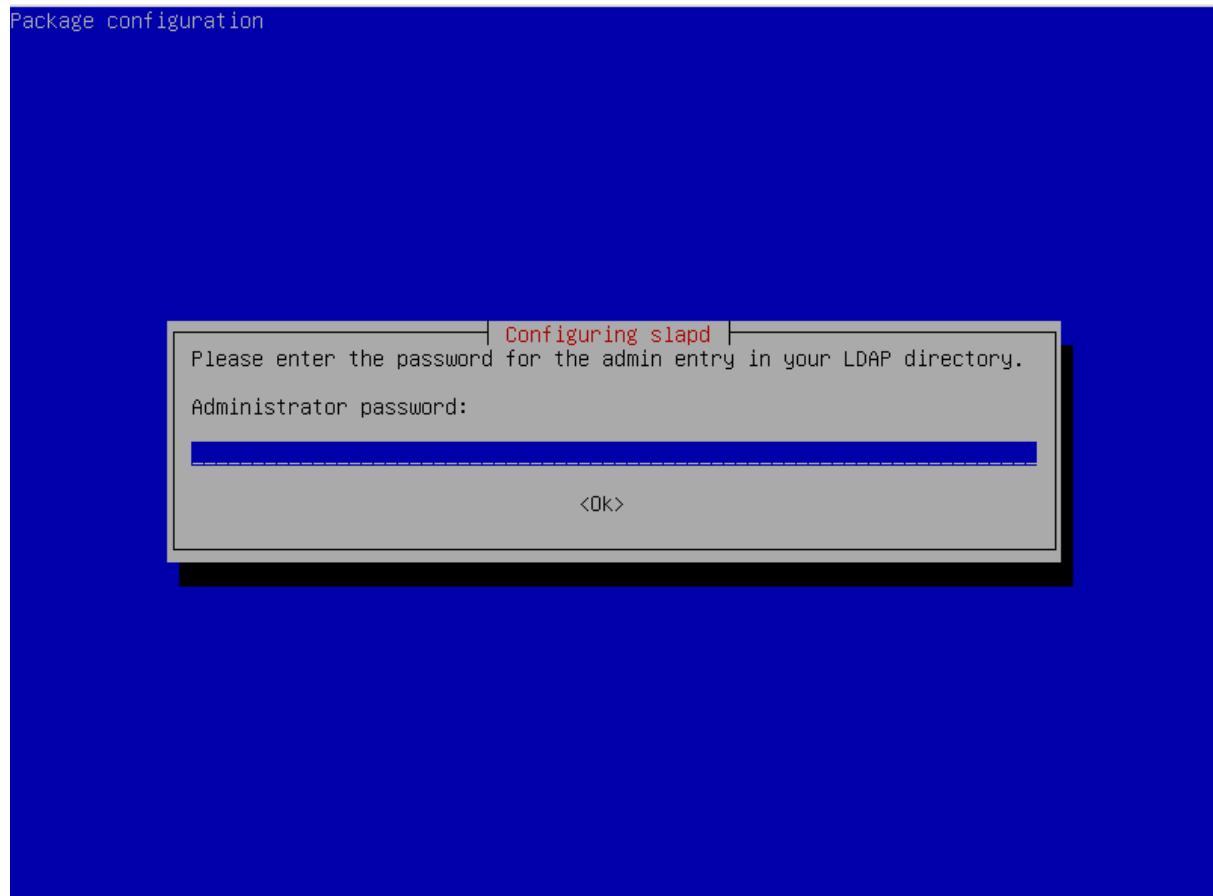
O Directory Server também é utilizado para armazenar dados de elementos da organização, mantém informações sobre algum conjunto de entidades e fornece serviços para acessar essas informações pelo protocolo de rede LDAP. Obviamente, um Directory Server também deve ter meios para adicionar, modificar e excluir informações neste banco de dados organizacional, mas, esteja atento que é um tipo de banco com pouca escrita e muita leitura.

O banco de dados do Directory Service é um banco genérico, onde é possível criar entidades e atribuir propriedades específicas, de usuários à computadores são estendíveis e ainda dá para se cadastrar até entidades que não se projetou inicialmente, há livros que diz que dá para se cadastrar até pedra.

Cada cadastro (row) neste banco é um Directory Entry e este banco é um banco hierárquico em estilo árvore, com a idéia de Entry contém Entry.



```
sudo apt-get install slapd -y
```



```
usuario@debian:~$ ls -l /etc/ldap/
total 16
-rw-r--r-- 1 root root 332 Feb 14 2021 ldap.conf
drwxr-xr-x 2 root root 4096 Feb 14 2021 sasl2
drwxr-xr-x 2 root root 4096 Nov 3 00:00 schema
drwxr-xr-x 3 openldap openldap 4096 Nov 3 00:00 slapd.d
usuario@debian:~$
```

```
usuario@debian:~$ sudo slapcat | grep dn
dn: dc=nodomain
dn: cn=admin,dc=nodomain
usuario@debian:~$ _
```

```

usuario@debian:~$ sudo slapcat
dn: dc=nodomain
objectClass: top
objectClass: dcObject
objectClass: organization
o: nodomain
dc: nodomain
structuralObjectClass: organization
entryUUID: ee60730c-d09d-103b-8dce-39b889509ce6
creatorsName: cn=admin,dc=nodomain
createTimestamp: 202111030300222
entryCSN: 20211103030022.866943Z#000000#000#000000
modifiersName: cn=admin,dc=nodomain
modifyTimestamp: 202111030300222

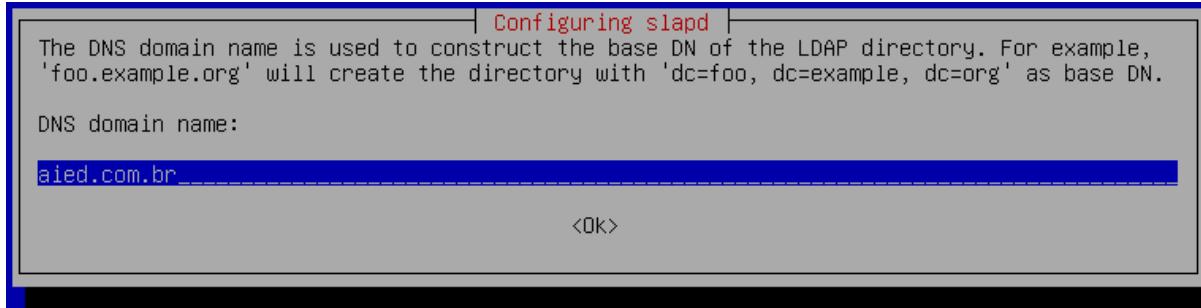
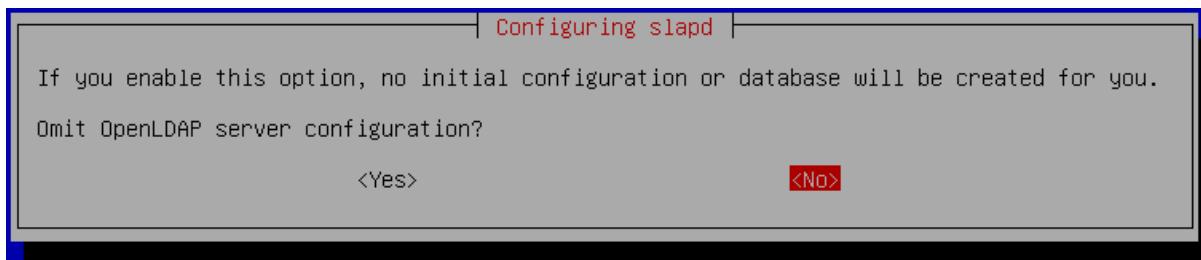
1

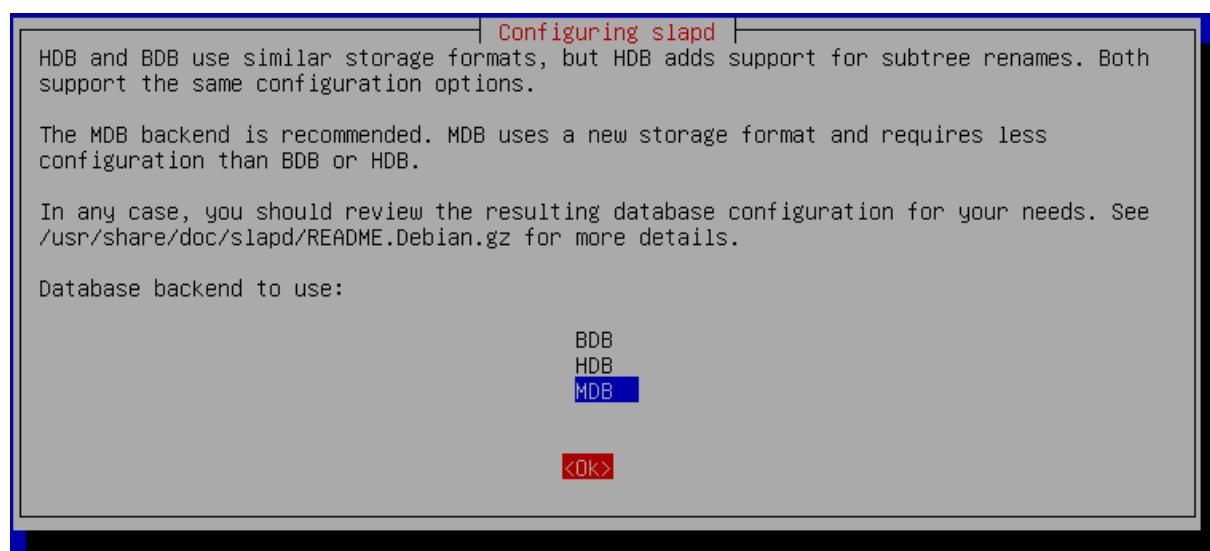
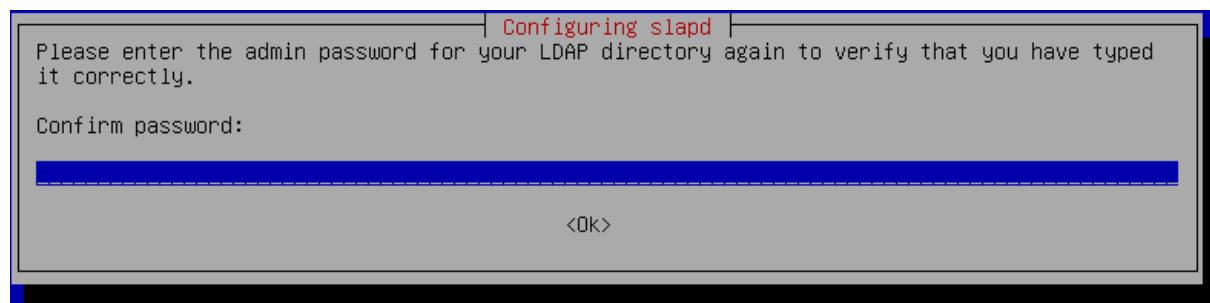
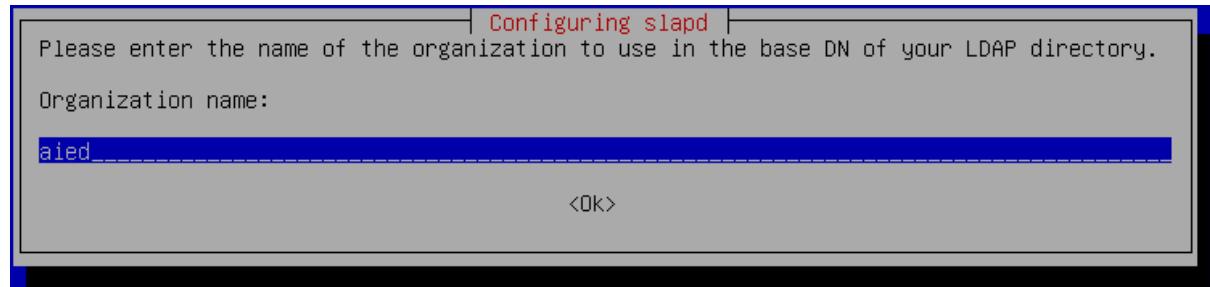
dn: cn=admin,dc=nodomain
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: e1NTSEF9WjkzSUNRTU93R322cTI4a2JDcnBGSU9Gc1BKWFaZYjY=
structuralObjectClass: organizationalRole
entryUUID: ee60b010-d09d-103b-8dcf-39b889509ce6
creatorsName: cn=admin,dc=nodomain
createTimestamp: 202111030300222
entryCSN: 20211103030022.868548Z#000000#000#000000
modifiersName: cn=admin,dc=nodomain
modifyTimestamp: 202111030300222

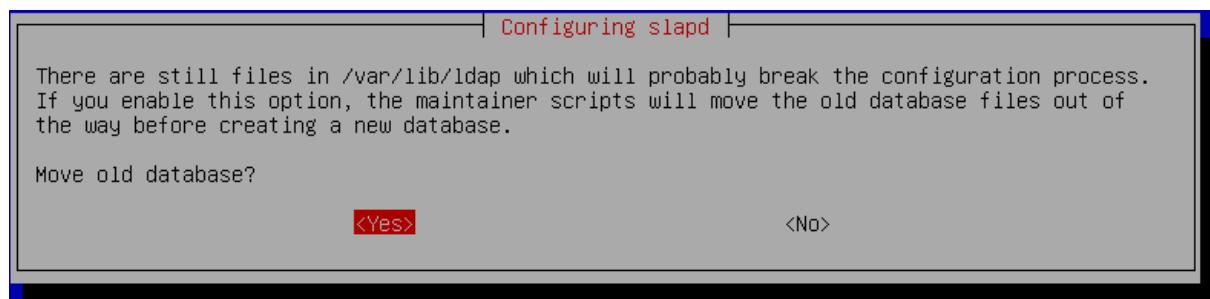
2

```

sudo dpkg-reconfigure slapd







```
usuario@debian:~$ sudo slapcat
dn: dc=aied,dc=com,dc=br
objectClass: top
objectClass: dcObject
objectClass: organization
o: aied
dc: aied
structuralObjectClass: organization
entryUUID: 94de32b6-d0a1-103b-8f91-b16ac2c8c9ef
creatorsName: cn=admin,dc=aied,dc=com,dc=br
createTimestamp: 20211103032630Z
entryCSN: 20211103032630.6830312#000000#000#000000
modifiersName: cn=admin,dc=aied,dc=com,dc=br
modifyTimestamp: 20211103032630Z

dn: cn=admin,dc=aied,dc=com,dc=br
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: e1NTSEF9TzNSeFF2aWJ2Mm9SMk5uRGx6NH1IWVQ4NXNJbDFDbHQ=
structuralObjectClass: organizationalRole
entryUUID: 94de689e-d0a1-103b-8f92-b16ac2c8c9ef
creatorsName: cn=admin,dc=aied,dc=com,dc=br
createTimestamp: 20211103032630Z
entryCSN: 20211103032630.6844452#000000#000#000000
modifiersName: cn=admin,dc=aied,dc=com,dc=br
modifyTimestamp: 20211103032630Z
```

```
usuario@debian:~$ sudo systemctl restart slapd
usuario@debian:~$ sudo systemctl status slapd
● slapd.service - LSB: OpenLDAP standalone server (Lightweight Directory Access Protocol)
 Loaded: loaded (/etc/init.d/slapd; generated)
 Loaded: loaded (/etc/init.d/slapd; generated)
 Active: active (running) since Wed 2021-11-03 00:28:11 -03; 2s ago
 Docs: man:systemd-sysv-generator(8)
 Process: 739 ExecStart=/etc/init.d/slapd start (code=exited, status=0/SUCCESS)
 Tasks: 3 (limit: 2354)
 Memory: 2.1M
 CGroup: /system.slice/slapd.service
 └─745 /usr/sbin/slapd -h ldap:/// -g openldap -u openldap -F /etc/ldap/slapd.d

Nov 03 00:28:11 debian slapd[734]: Stopping OpenLDAP: slapd.
Nov 03 00:28:11 debian systemd[1]: slapd.service: Succeeded.
Nov 03 00:28:11 debian systemd[1]: Stopped LSB: OpenLDAP standalone server (Lightweight Directory Access Protocol).
Nov 03 00:28:11 debian systemd[1]: Starting LSB: OpenLDAP standalone server (Lightweight Directory Access Protocol).
Nov 03 00:28:11 debian slapd[744]: @(#) $OpenLDAP: slapd (Feb 14 2021 18:32:34) $Debian OpenLDAP Maintainers <pkg-openldap-devel@lists.alioth.debian.org>
Nov 03 00:28:11 debian slapd[745]: slapd starting
Nov 03 00:28:11 debian slapd[739]: Starting OpenLDAP: slapd.
Nov 03 00:28:11 debian systemd[1]: Started LSB: OpenLDAP standalone server (Lightweight Directory Access Protocol).
lines 1-19/19 (END)
```

## 32 Ansible (atuando)

Lab de uma infra simples

lab de uma infra web balanceada

## LAB I Gateway completo com IPCOP (falta formatar)

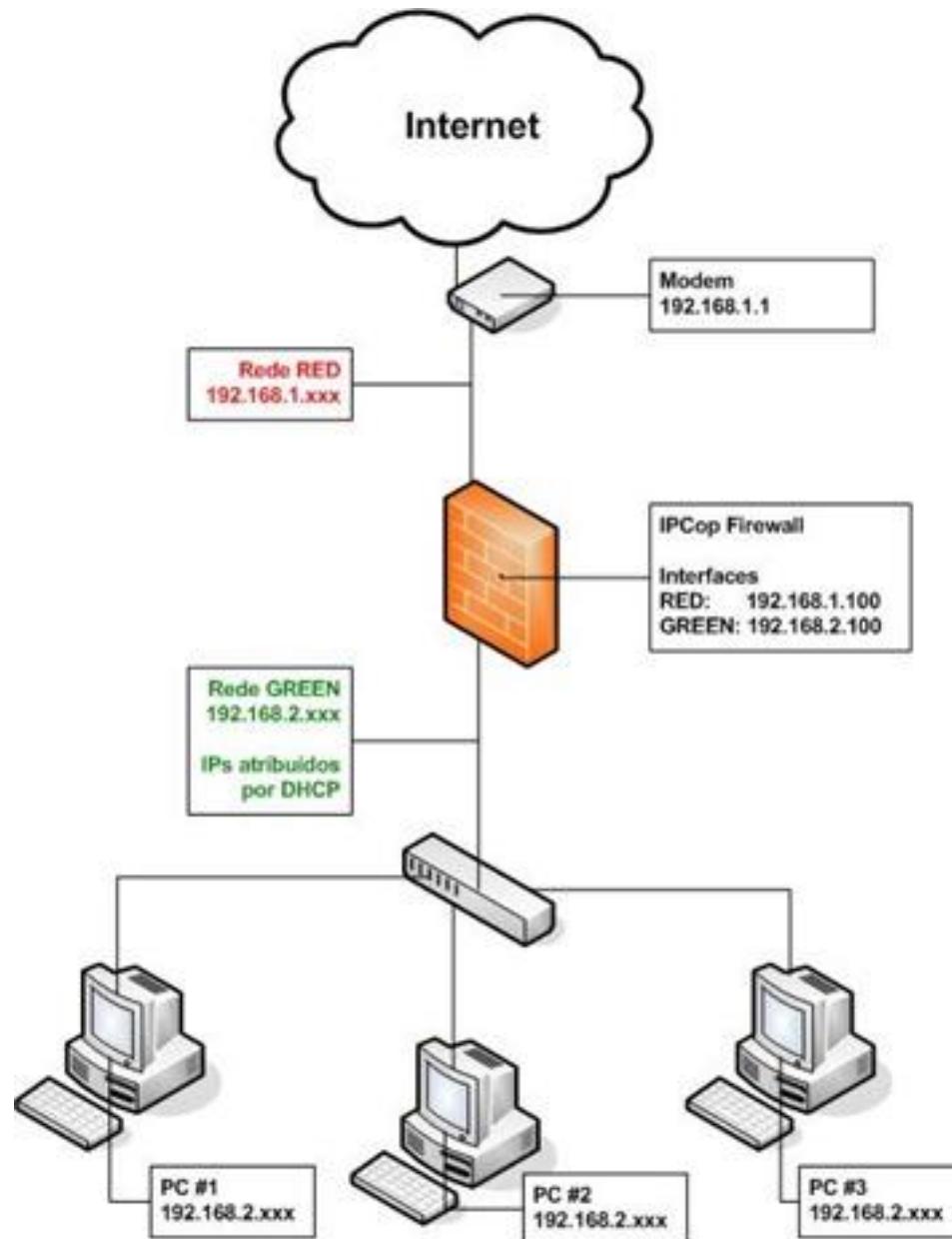
Prática da aula: <https://youtu.be/tGvCSh2dSXE>

Hoje vou mostrar como instalar o IPCop Firewall na vossa rede, seja ela doméstica ou empresarial. Para que não fiquem dúvidas em nenhum dos passos da instalação, vou mostrar mesmo passo por passo e o que fazer em cada um deles.

Como já havia referido anteriormente, para termos este sistema a funcionar precisamos do VirtualBox, mas se possui Hardware um dois computadores e um Switch:

- Processador: Pentium III
- Memória: 512MB
- Disco: 10GB
- 2 placas de rede ethernet
- Teclado, mouse e monitor para instalação

Para ficarem com uma ideia do que se pretende com este sistema fica aqui uma imagem ilustrativa de como será a configuração da nossa rede com o IPCop Firewall de modo a criarmos uma barreira entre o modem de acesso à Internet e a nossa rede local (a configuração da firewall que vou mostrar é a básica RED-GREEN):



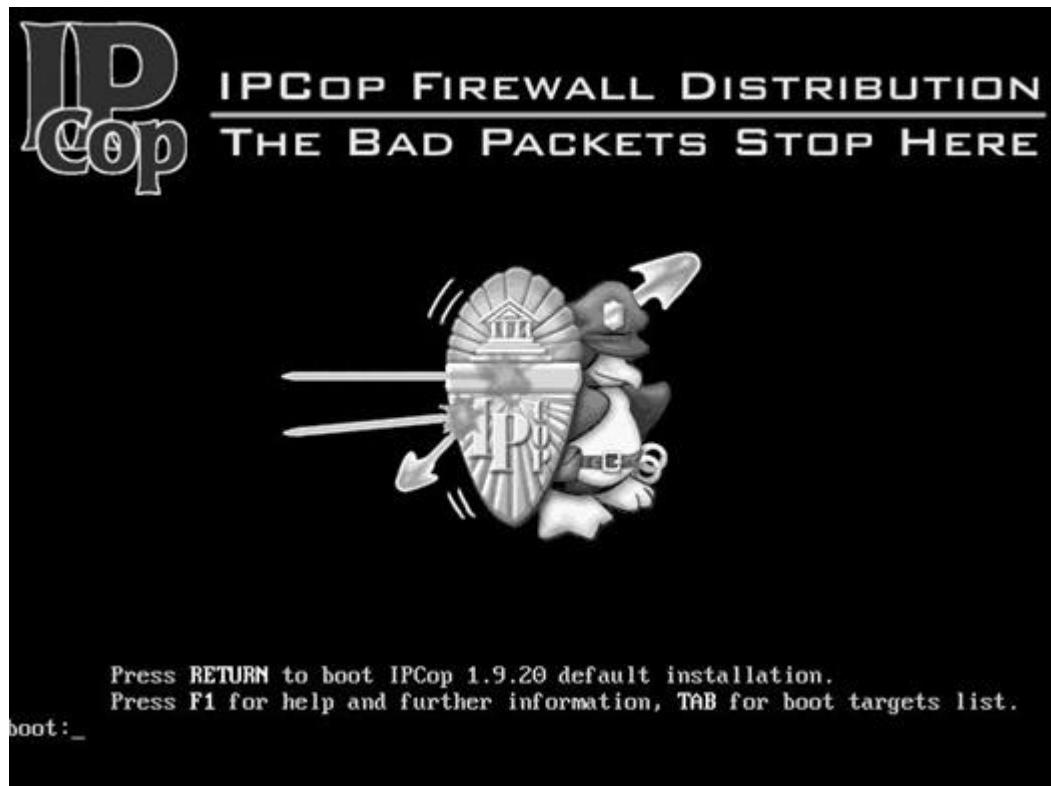
**RED:** Adaptador 1 da VirtualBox em modo Bridge Adapter

**GREEN:** Adaptador 2 da VirtualBox em modo Rede Interna

Para começar façam download do IPCop no site oficial [aqui](#). Apesar de ter saído recentemente a versão 2.0, o processo de instalação da última versão é idêntico ao que vamos apresentar.

Vai no virtualbox e cria uma VM nova, 32 bits. Coloca o CD no CD-ROM virtual.

Após o arranque ter iniciado, este é o primeiro ecrã da instalação. Para prosseguir basta carregar na tecla “Enter”



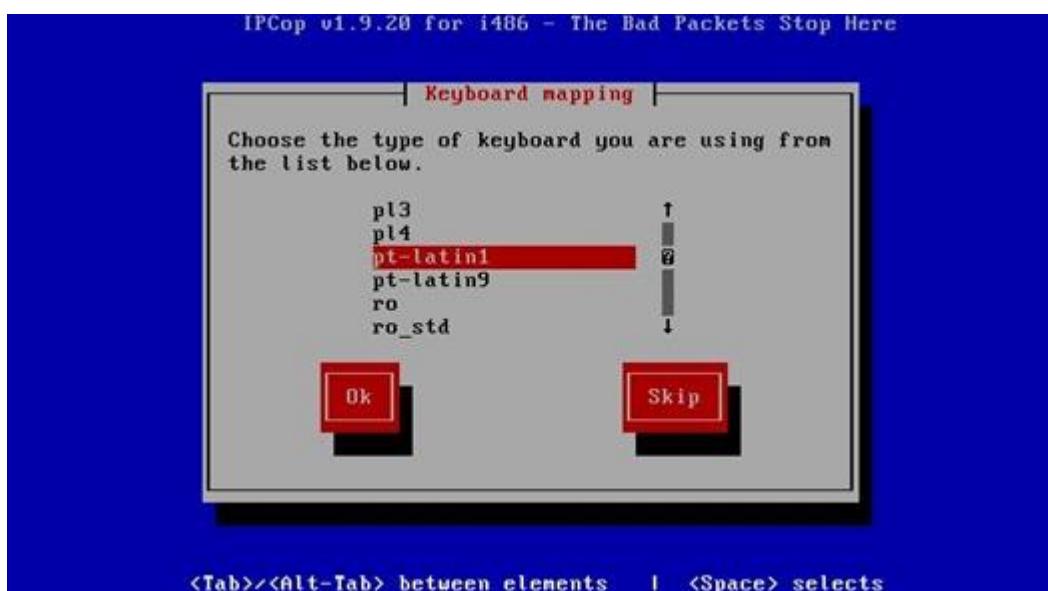
De seguida escolhemos o idioma que queremos usar no IPCop, eu optei por usar o inglês, sendo apresentado se seguida as boas vindas à instalação do IPCop:



IPCop – Boas vindas



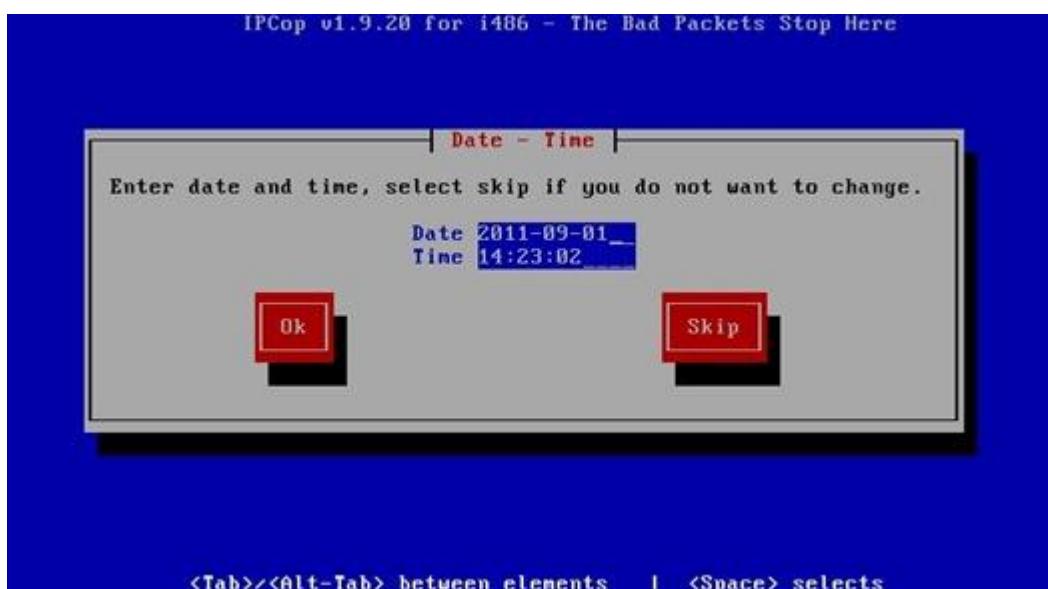
O próximo passo é escolher a configuração do teclado que vamos usar:



De seguida definir o fuso horário. No caso dos nossos leitores portugueses, devem escolher “America/Sao\_paulo”, na imagem abaixo obtida da internet está Lisboa.



Configuração da data e hora:

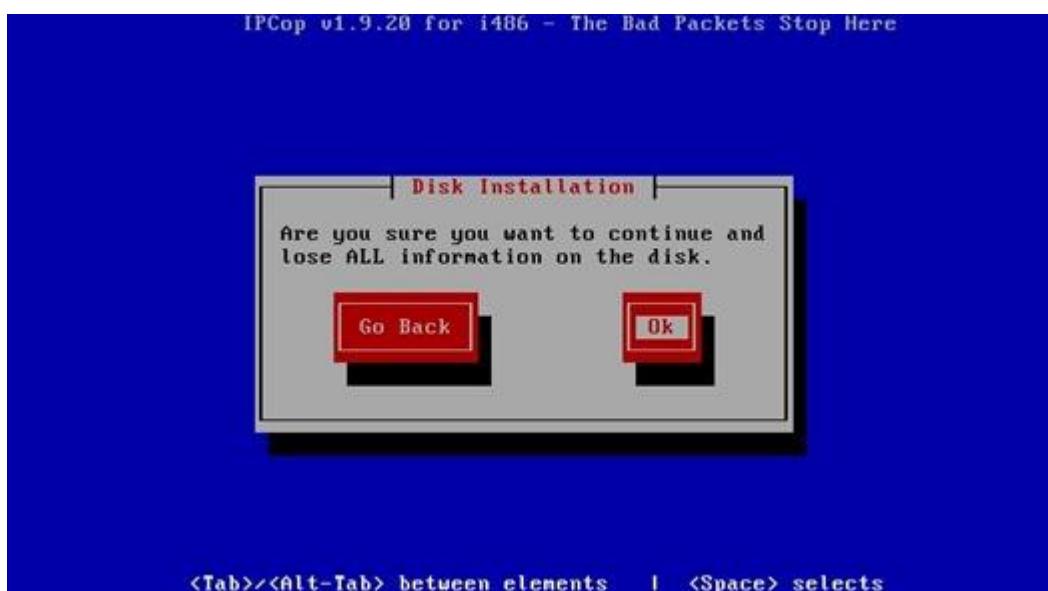


De seguida escolhemos o disco onde vamos instalar o IPCop, mas atenção, se tiverem mais que um disco ligado no PC certifiquem-se de que escolhem o disco certo, porque a instalação vai eliminar tudo o que estiver no disco que escolherem, tal como mostra o passo seguinte

**Nota: Se é a primeira vez e querem apenas testar o IPCop, recorram à virtualização usando o VMWare ou VirtualBox**



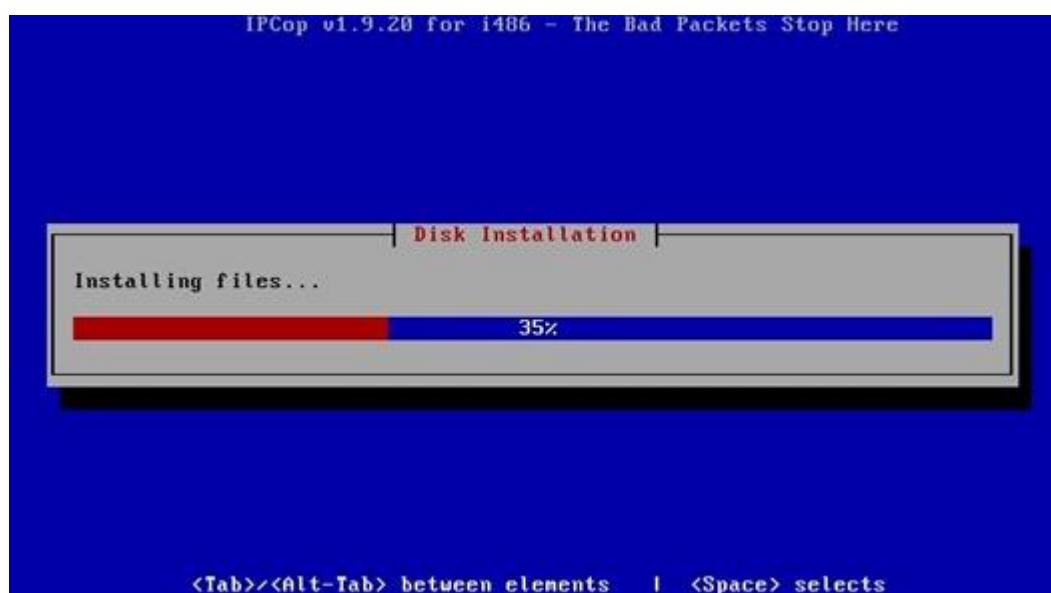
Informação que todos os dados serão removidos do disco:



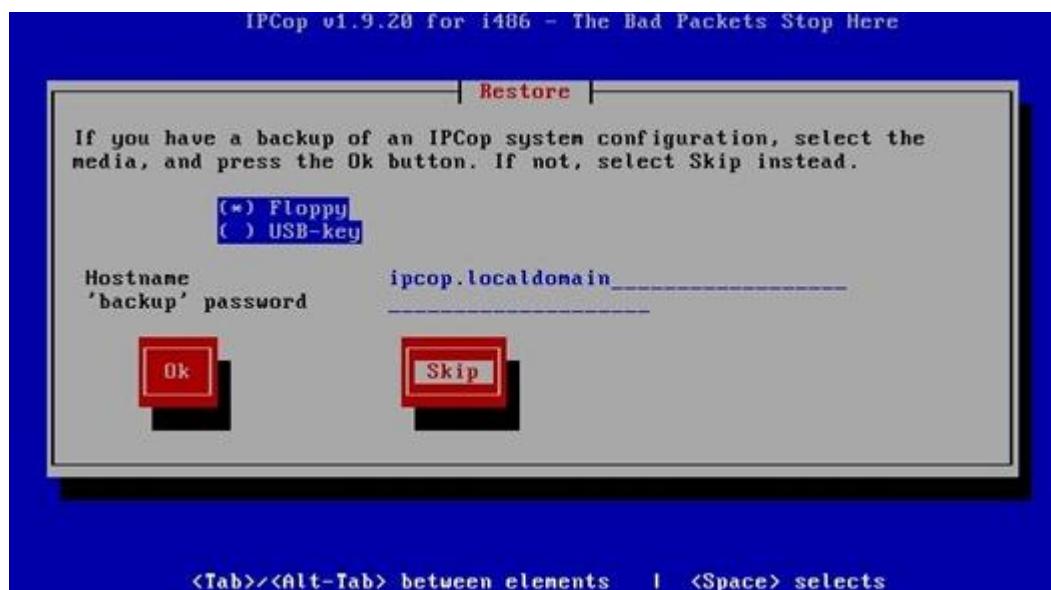
Se estiverem a usar um disco/pen USB podem também escolher o modo “Flash”. Para este tutorial vamos usar a opção Hardisk (já que possuo um disco para a instalação do sistema)



Depois é só esperar que sejam efectuadas algumas operações no disco, tais como, eliminar partições existentes, criar novas partições, formatar as partições e a instalação dos ficheiros do sistema:



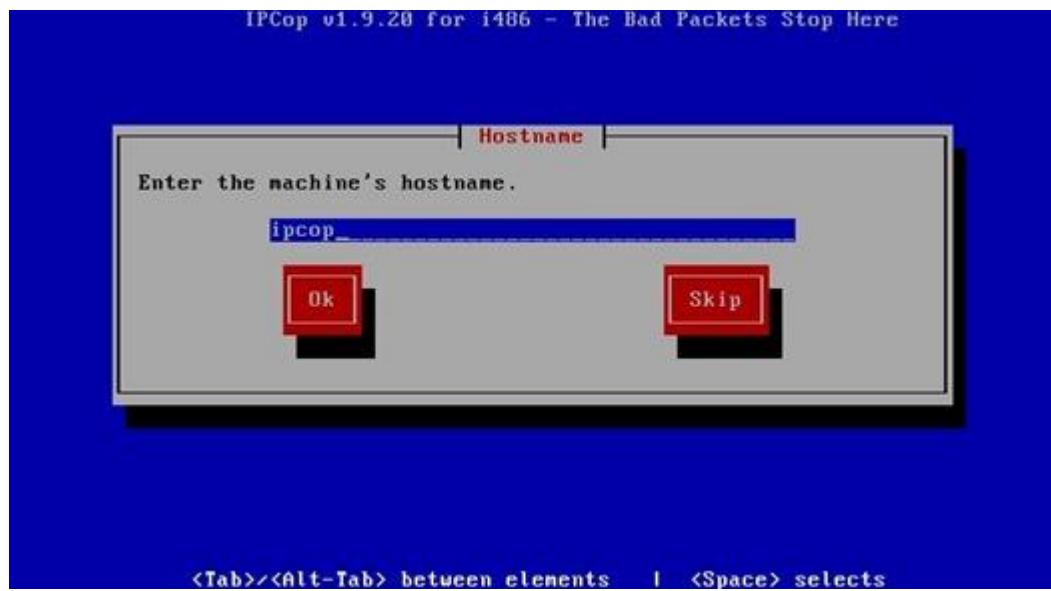
Depois de instalado o IPCop é-nos oferecida a possibilidade de efectuar um restauro de um backup da configuração, isto se já tivéssemos criado um anteriormente. Se já tivéssemos este backup os passos seguintes não seriam necessários, mas como não é o caso, vamos fazer "Skip" a este passo:



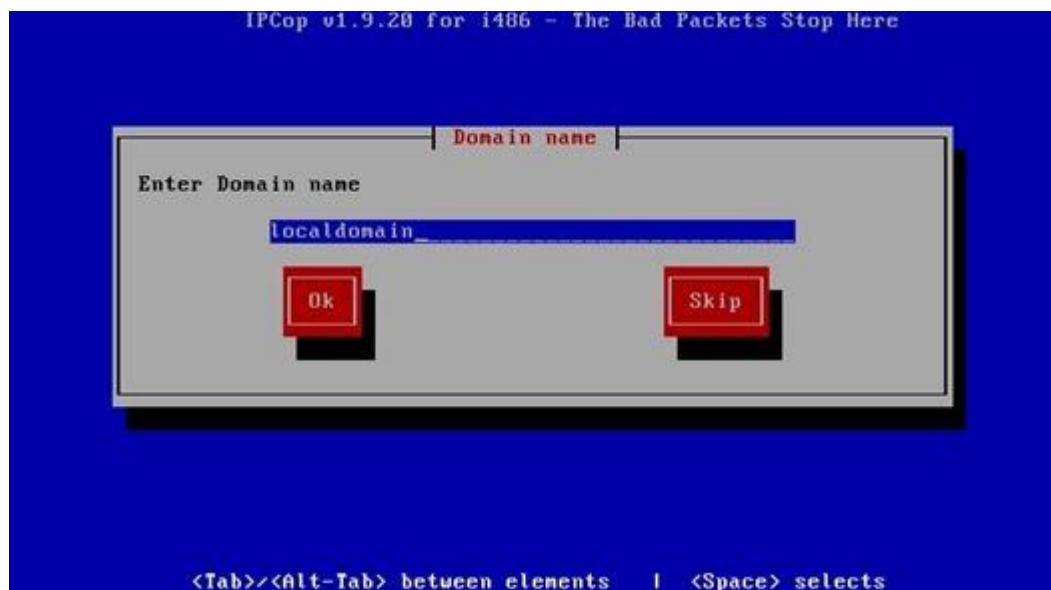
E eis que chegamos ao passo onde temos informações de como por exemplo aceder à administração web do IPCop. A partir deste ponto começamos o processo da configuração inicial do nosso IPCop, dando o nome ao PC de “ipcop” ou outro do vosso agrado:

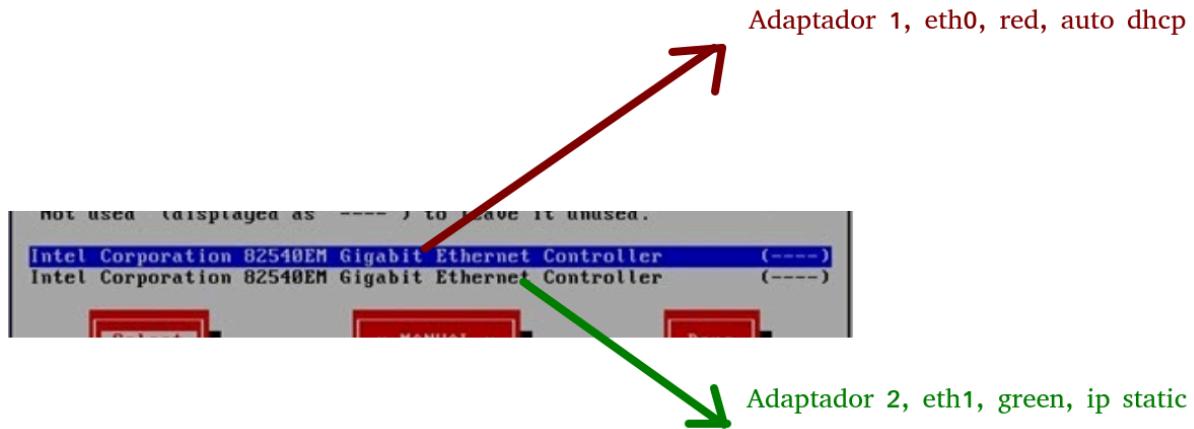


Neste passo é necessário definir um nome para a máquina



No próximo passo escrevemos o nome do domínio, se tivermos um, senão deixamos ficar tal como aparece: "localdomain".

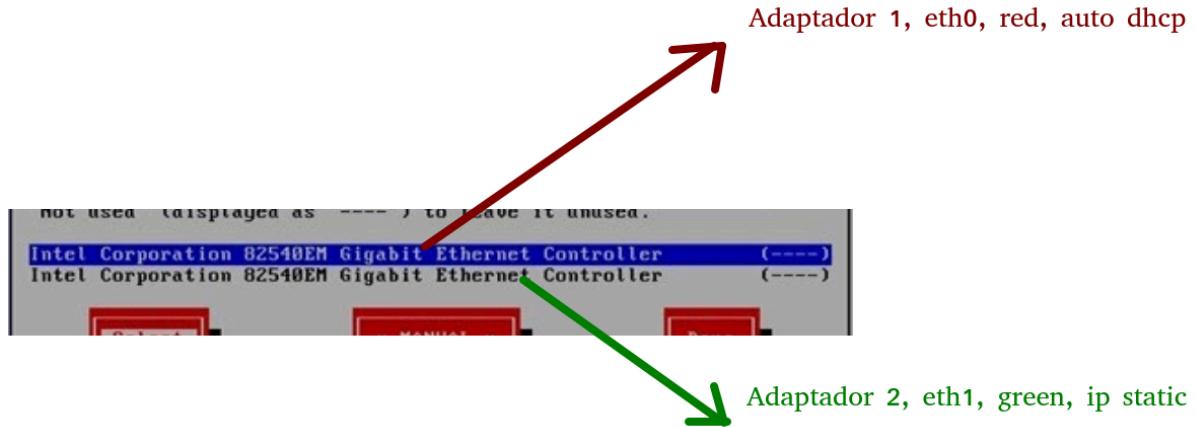




Aqui defino que esta placa de rede vai ser a nossa rede GREEN, ou seja, onde irão estar os PCs da nossa rede ligados:



Voltamos ao quadro das placas de rede e já vemos que a primeira placa já está atribuída a rede GREEN, e passamos à placa de rede seguinte:



Esta vai ser a nossa rede RED, ou seja onde vamos ligar o modem para o acesso à Internet:



Ao voltarmos ao quadro das placas de rede podemos verificar que já temos as nossas placas de rede definidas para rede GREEN e RED respectivamente:

Seleccionamos “Done” e passamos à configuração da rede GREEN onde definimos o endereço IP e máscara de rede para a placa:



De seguida fazemos o mesmo para a configuração da rede RED. Não há nenhum engano, os endereços IPs aqui configurados estão correctos. Como tenho o modem configurado com o IP 192.168.1.1 a placa de rede RED vai ficar com um IP da mesma gama 192.168.1.100 para que a comunicação seja possível. A placa de rede GREEN apesar de não estar na mesma gama 192.168.2.100 vai conseguir aceder à Internet (o IPCop encarrega-se disso) mas não vai conseguir comunicar directamente com o modem, por exemplo para aceder à configuração web do modem.

Nota2: Se tiverem outro cenário, com outro endereçamento, só têm de adaptar as vossas configuração



O passo seguinte é crucial ficar correctamente configurado, pois é aqui que conseguimos definir qual o IP de saída/entrada para o acesso à Internet:

**RED, se quiser, ficar até mais fácil colocar com DHCP.**



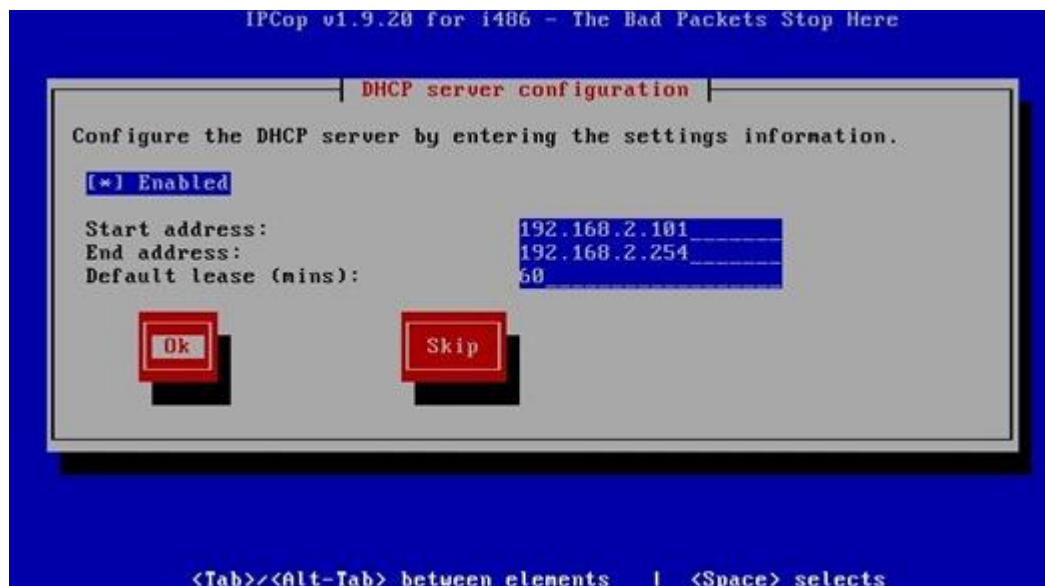
Definição dos IP's para DNS e Gateway



recomendo que o Secondary DNS seja 8.8.8.8 e que Default gateway é o gateway da sua casa.

Recomendo que seguidamente activem o Servidor DHCP para que os PCs que ligarem na rede GREEN possam receber um IP automaticamente.

Como defini o IP da placa de rede GREEN 192.168.2.100 o intervalo de endereços por DHCP tem de começar obrigatoriamente acima do IP 100 até ao limite máximo 254 (não obrigatoriamente):



De seguida definimos a password para os utilizadores root, admin e backup. Cada um destes utilizadores tem funções diferentes:

- root: aceder ao IPCop através da linha de comandos/consola
- admin: aceder ao IPCop através da administração web
- backup: efectuar/restaurar um backup na administração web

No meu caso usei sempre a mesma password em todos os utilizadores, mas podem fazer como acharem melhor.



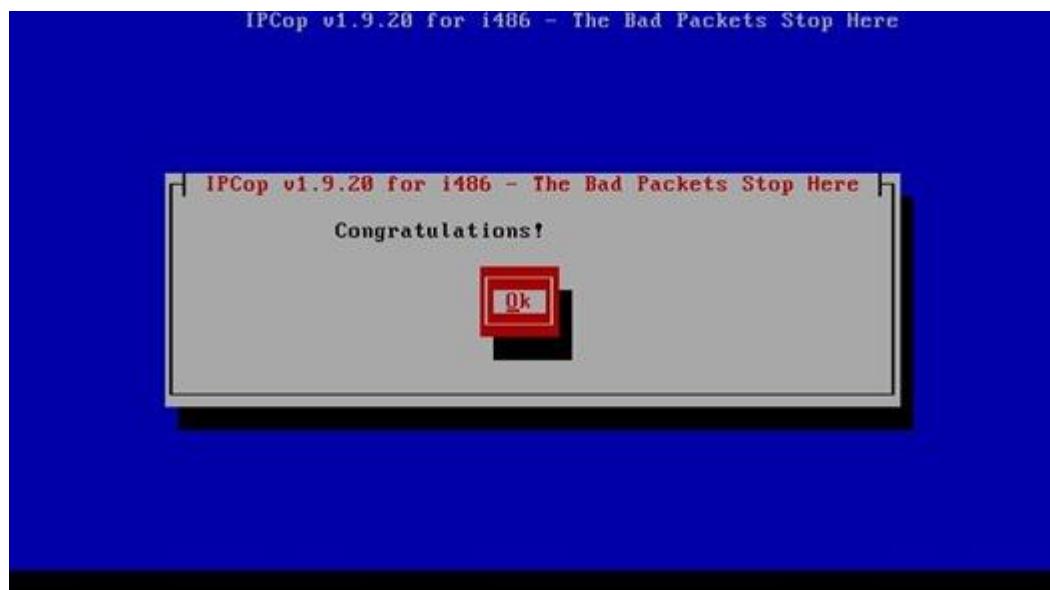
IPCop – password de admin



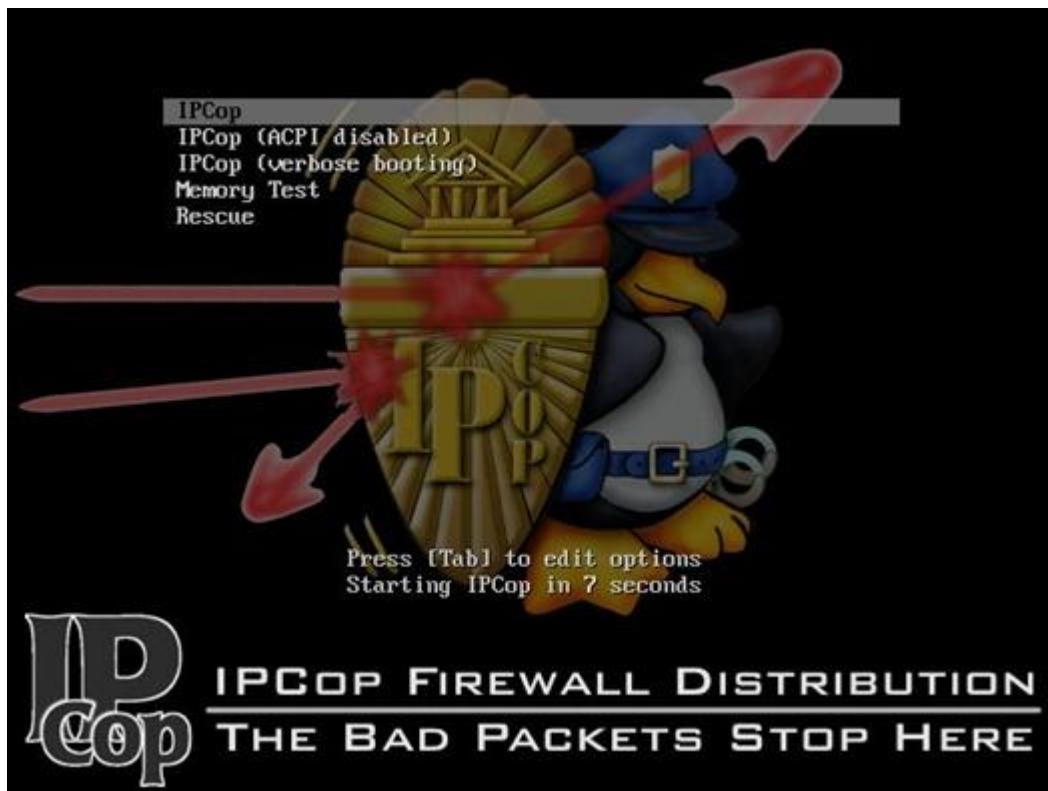
IPCop – password de backup



E chegámos ao fim da instalação/configuração do IPCop. Resta-nos reiniciar o PC e ver o IPCop a arrancar.

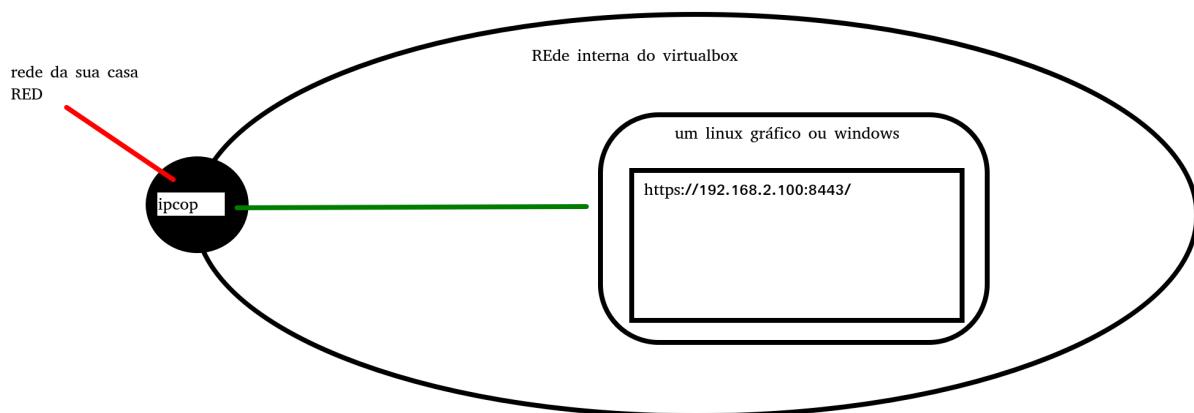


Depois de Mandarmos reiniciar o sistema , escolhemos no Grub a opção IPCop



Quando o IPCop inicia e desliga emite um sinal sonoro característico para que possamos saber que vai desligar/ligar mesmo sem termos um monitor ligado ao PC.

Após termos o sinal sonoro de que o IPCop está operacional ligamos um dos nossos PCs à rede GREEN e com o DHCP configurado na placa de rede acedemos ao seguinte endereço: <https://192.168.2.100:8443> (endereço da interface green)



Deverá aparecer um aviso de segurança a indicar que a ligação não é de confiança. Teremos de confirmar esta ligação como uma exceção, sendo

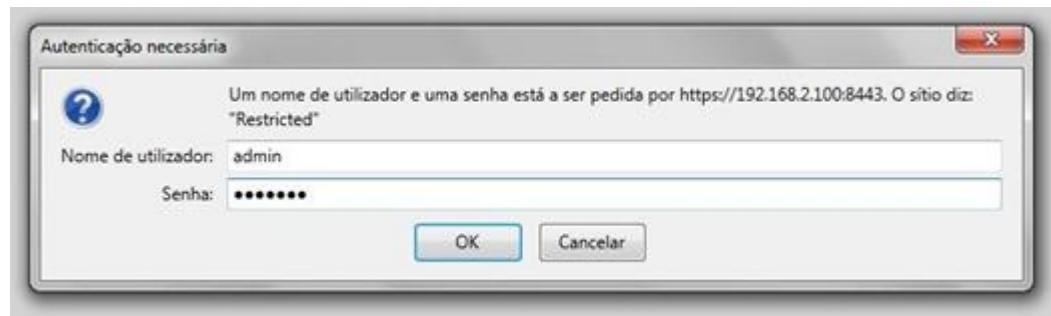
necessário efectuar este passo apenas à primeira vez que acedemos à administração web do IPCop.

No meu caso usei o browser Firefox e este passo processa-se da seguinte forma:

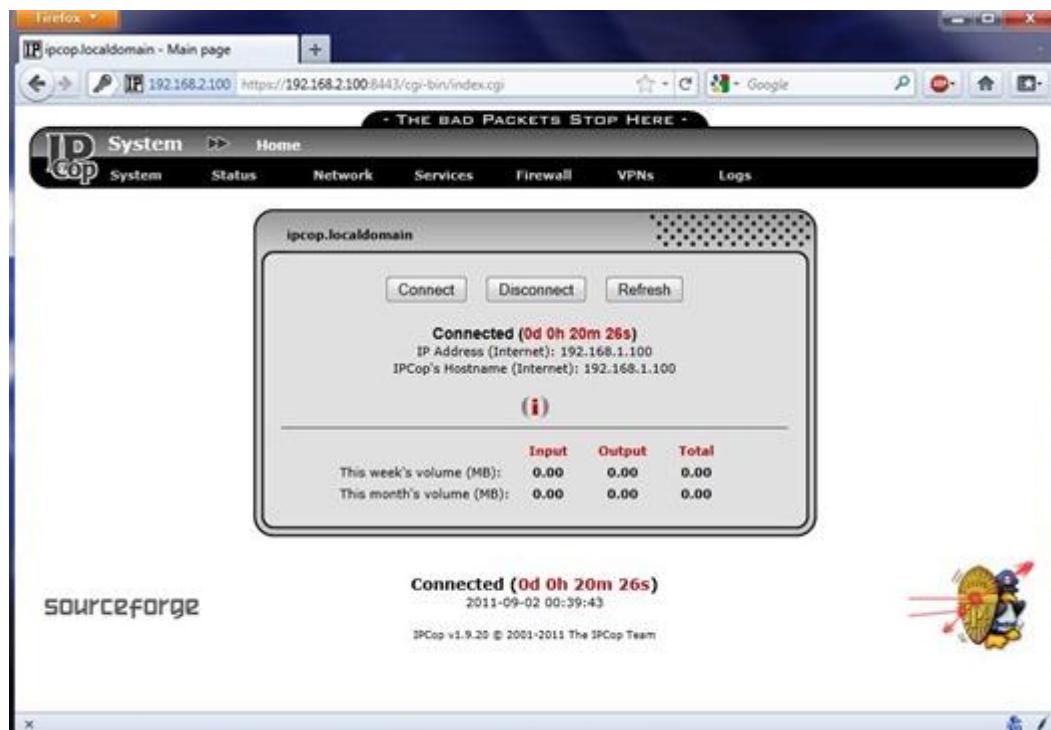




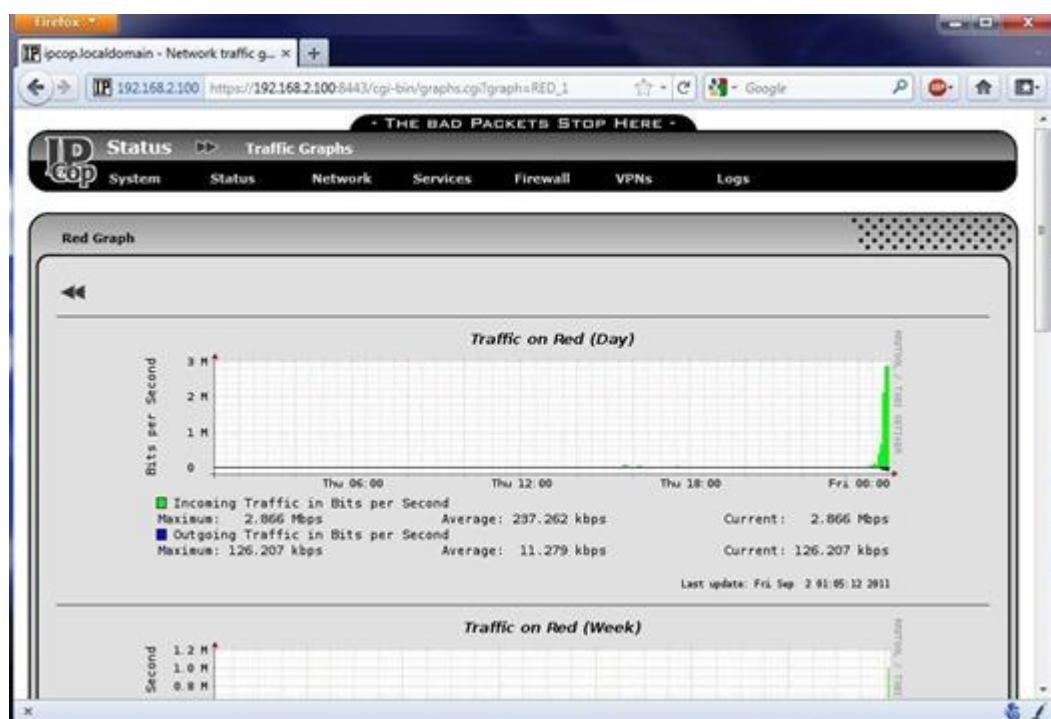
Uma vez definida esta configuração no browser aparece-nos a janela de login para a administração web do IPCop. Colocamos aqui o utilizador admin e a password definida durante a instalação:



Posto isto já estamos na administração web do IPCop, onde podemos configurar e monitorizar uma série de dados da rede estando assim mais protegidos contra intrusões não autorizadas. Aqui ficam algumas imagens do que podemos ver na administração web do IPCop:



## IPCop – Gráfico de tráfego na rede RED



Esperamos que tenham gostado deste tutorial e em caso de alguma dúvida ou questão deixem os vossos comentários. Esta é sem dúvida uma excelente solução, para quem pretende implementar um sistema de Firewall potente e gratuito. Experimentem.

## LAB II Enterprise Service Bus (Finalizado)

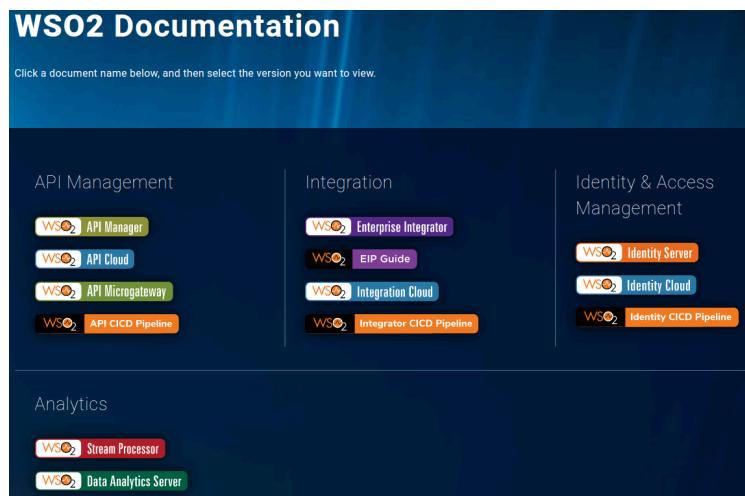
A tarefa de integrar diferentes aplicativos de software, serviços e sistemas e formar novas soluções de software a partir disso é conhecida como enterprise integration. O aplicativo de software projetado para habilitar essa tarefa é conhecido como Enterprise Service Bus (ESB).

Em geral, um ESB possui uma série de elementos que proporcionam a capacidade de integração e troca de mensagens, são funcionalidades esperadas:

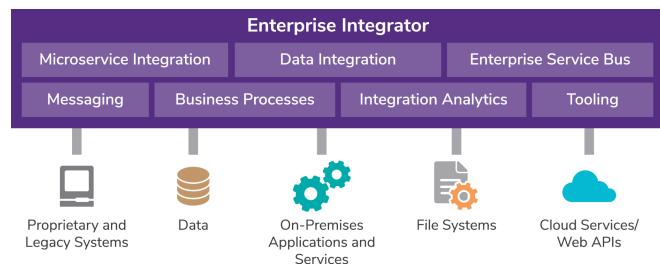
- Mediação de mensagens: Manipula o conteúdo da mensagem, direção, destino e fluxo de mensagens;
- Virtualização de serviço: envolva sistemas ou serviços existentes com novas interfaces de serviço;
- Conversão de protocolo: conecte diferentes protocolos;
- Suporte para **Enterprise Integration Patterns** (EIP): EIP é o padrão para Enterprise Integration (<http://www.eaipatterns.com/>);
- Qualidade de serviço: aplique segurança, limitação e armazenamento em cache;
- Conectando-se a sistemas legados: conectores, incluindo SAP, FIX e HL7;
- Conectores para serviços em nuvem e APIs: Salesforce, Twitter, PayPal e muitos mais;
- Orientado pela configuração: a maioria das funcionalidades é orientada pela configuração, mas não pelo código;
- Extensibilidade: Existem pontos de extensão que podem ser usados para integração com qualquer protocolo personalizado ou sistema proprietário.

WSO2 ESB foi projetado e desenvolvido desde o início pensando em alto desempenho, menor pegada de programação e o middleware de integração mais interoperável. Oferece uma ampla gama de recursos de integração e suporte de roteamento de mensagens de alto desempenho usando um mecanismo de mediação de mensagens aprimorado e otimizado, inspirado no **Apache Synapse**.

Além do custo zero, também encontramos inúmeros serviços e produtos oferecidos pela WSO2, conforme documentação oficial acessível pela url: <https://wso2.com/wso2-documentation>.



O WSO2 ESB oferece uma ampla gama de recursos de integração, desde o simples roteamento de mensagens até a integração harmoniosa de sistemas proprietários complexos. Nesta aula será instalado o Enterprise Integrator e exemplificado seu uso através de uma solução simples.



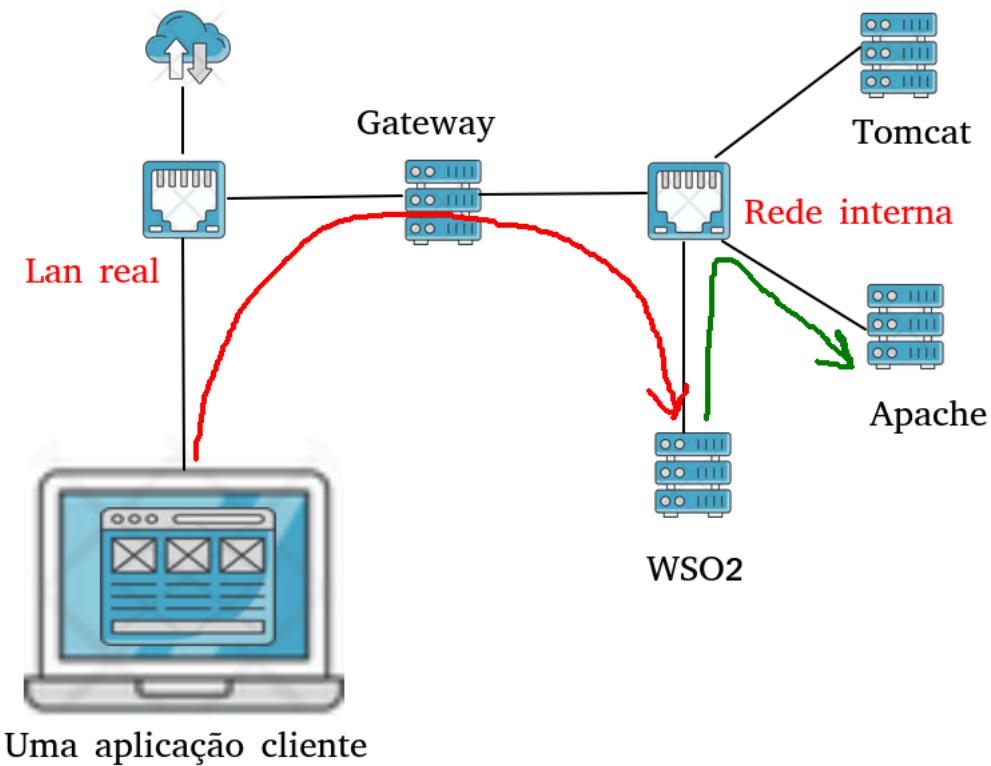
## 1 Recursos do capítulo

Todo o conteúdo externo ao texto pode ser obtido no link abaixo.

  O material deste capítulo pode ser obtido [neste link](#).

## 2 Preparando o ambiente

Neste cenário proposto temos 3 sistemas ou aplicações que se comunicam trocando dados por intermédio do WSO2, por motivo de simplificação foi adicionado ao cenário um ambiente não corporativo com o emprego de um router comum oferecido por uma operadora em uma rede caseira.



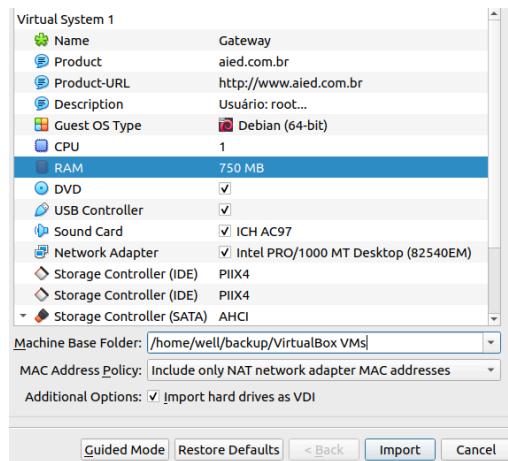
O App1 é o Postman que somente consome serviços, tem a capacidade de compreender JSON somente. Já o App2 é uma aplicação Apache que disponibiliza alguns serviços de dados em REST JSON.

**Atenção, todas as três máquinas virtuais serão AMD64, ou seja, 64 bits.**

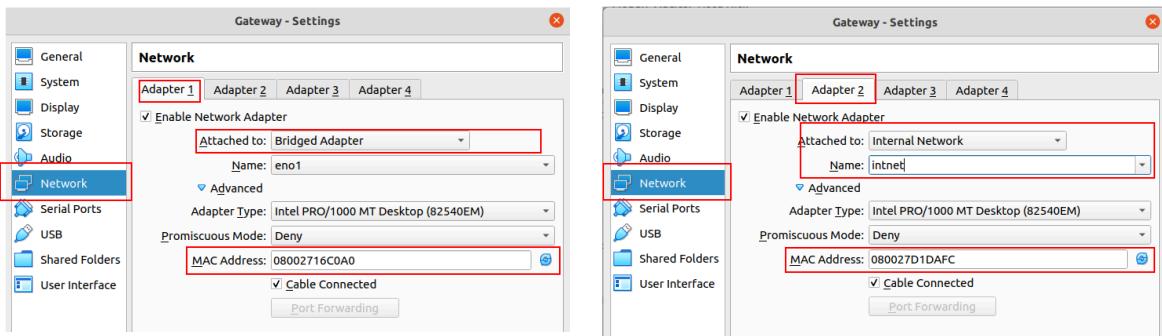
### 3 Instalando e configurando o Gateway + Iptables

Para isolar o grupo de servidores que contém os serviços será utilizado um nível, um router fazendo um papel de gateway entre 2 redes e naturalmente controlando a comunicação por Iptables. Será utilizado os capítulos [Serviço de Internet com Gateway](#) e [Iptables e Firewall](#), os dados serão aqui replicados para poder garantir que o aluno consiga instalar o complexo ambiente deste capítulo.

O primeiro computador a ser adicionado se chamará gateway e terá duas interfaces de rede conforme configuração da máquina mostrada abaixo, observe que será chamada de gateway.



Para adicionar duas interfaces de rede, após importar a máquina virtual em configurações acesse a aba de network. Configure dois adaptadores conforme as figuras abaixo.



No arquivo `/etc/sysctl.conf` habilite a propriedade `net.ipv4.ip_forward=1` conforme figura abaixo.

```
Gateway [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 3.2 /etc/sysctl.conf

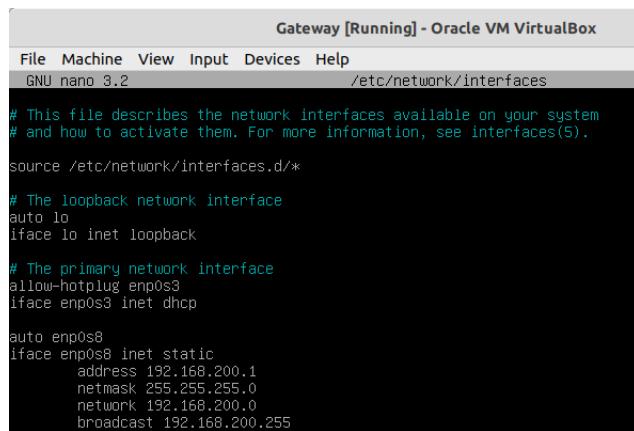
Uncomment the next two lines to enable Spoof protection (reverse-path filter)
Turn on Source Address Verification in all interfaces to
prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

Uncomment the next line to enable TCP/IP SYN cookies
See http://lwn.net/Articles/277146/
Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

Uncomment the next line to enable packet forwarding for IPv6
Enabling this option disables Stateless Address Autoconfiguration
based on Router Advertisements for this host
```

Altere o arquivo `/etc/network/interfaces` conforme figura abaixo.



```

Gateway [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp

auto enp0s8
iface enp0s8 inet static
 address 192.168.200.1
 netmask 255.255.255.0
 network 192.168.200.0
 broadcast 192.168.200.255

```

Agora com auxílio do nano crie o arquivo **/usr/local/gateway.sh** e edite o seguinte script, este script adiciona regras Iptables para a operação de Gateway incluindo mascaramento e também o roteamento dos pacotes permitidos apenas para o WSO2 que está na rede interna e por isso protegida.

```

1.#!/bin/bash
2.
3. # Limpando todas as entradas anteriores
4. iptables -t filter -F
5. iptables -t filter -X
6. iptables -t nat -F
7. iptables -t nat -X
8.
9. # Regras para o Gateway
10. iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
11. iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
12. iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
13.
14.
15. # Redirecionando o serviço do WSO2
16. iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 9443 -j DNAT --to
 192.168.200.15:9443
17. iptables -A FORWARD -p tcp -d 192.168.200.15 --dport 9443 -j ACCEPT
18.
19. iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 8280 -j DNAT --to
 192.168.200.15:8280
20. iptables -A FORWARD -p tcp -d 192.168.200.15 --dport 8280 -j ACCEPT
21.
22. # Forçando o linux a segurança, remover o comentário quando todo o ambiente estiver
 funcionando, veja a vídeo aula no Youtube
23. #iptables -P INPUT DROP
24. #iptables -P FORWARD DROP
25. #iptables -P OUTPUT DROP

```

Execute o comando chmod para permitir execução deste script, se nenhuma exceção for lançada crie um arquivo para o Systemd.

1. sudo chmod +x /usr/local/gateway.sh

Agora crie um arquivo de inicialização para rodar este script sempre que o GNU/Linux for iniciado, com nano crie o arquivo **/etc/systemd/system/gateway.service**, edite o seguinte código no arquivo.

1. [Unit]
2. Description=Gateway
3. After=network.target
- 4.
5. [Service]
6. ExecStart=/usr/local/gateway.sh
- 7.
8. [Install]
9. WantedBy=multi-user.target

Inicie o serviço gateway.service conforme listagem de código abaixo.

1. sudo systemctl daemon-reload
2. sudo systemctl enable gateway.service
3. sudo systemctl start gateway.service
4. sudo systemctl status gateway.service

Valide com o status se tudo está OK.

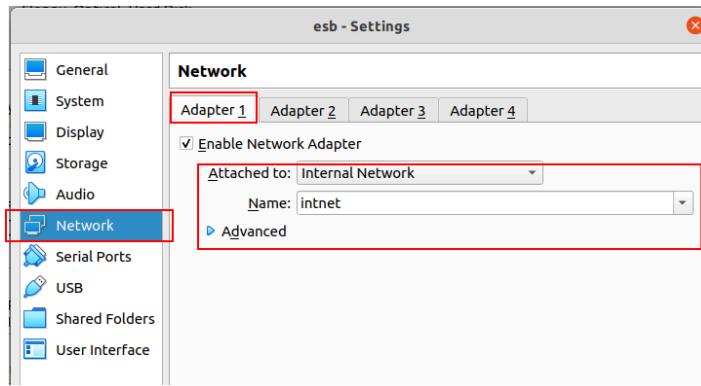
## 4 Instalando o WSO2

O WSO2 é gratuito e utiliza Java, então será obtido tanto o Java 8 quanto o WSO2, siga a configuração abaixo. Primeiro crie uma máquina virtual com no mínimo 2G de memória, se criar abaixo disso a máquina entrará em SWAP, conforme figura abaixo.

```
top - 14:37:42 up 51 min, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 70 total, 1 running, 69 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.5 us, 0.0 sy, 0.0 ni, 99.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1995.2 total, 246.0 free, 1369.3 used, 379.8 buff/cache
MiB Swap: 1022.0 total, 1022.0 free, 0.0 used. 484.1 avail Mem

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 458 wso2 20 0 3731316 1.3g 24352 S 1.7 64.8 1:33.74 java
 763 usuario 20 0 10960 3508 3076 R 0.3 0.2 0:02.27 top
 1 root 20 0 21944 9980 7800 S 0.0 0.5 0:00.80 system
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthrea
```

A interface de rede deve estar ligada na



O WSO2 é um elemento que será possível alvo de ataques, isso pois é mais valioso que apenas uma única base de dados, nele concentra-se a comunicação da organização, e por isso não pode estar exposta na rede local que neste exemplo é mais abrangente.

```
esb [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.15
 netmask 255.255.255.0
 network 192.168.200.0
 broadcast 192.168.200.255
 gateway 192.168.200.1
```

Dentro da rede interna os IPs serão todos fixados na configuração, então vamos atribuir o ip 192.168.200.15 para o servidor WSO2, conforme figura acima.

#### 4.1 Instalando o Java 8

O primeiro passo é iniciar a atualização do Debian 10, para isso deve-se executar o Update conforme listagem abaixo (linha 1). A versão que será utilizada do WSO2 Enterprise Integrate é compatível com Java 7 ou Java 8, para isso será necessário adicionar um PPA pois não é mais a versão default do Java, por isso será necessário instalar o pacote **software-properties-common** (linha 2).

1. sudo apt update
2. sudo apt install software-properties-common -y

Para adicionar um PPA é simples, utilize o comando `apt-add-repository` conforme linha 1 da listagem abaixo, o comando aponta para um nova entrada para update.

1. sudo apt-add-repository 'deb http://security.debian.org/debian-security stretch/updates main'
2. sudo apt update

A instalação do Java 8 agora é simples, basta executar um apt install conforme listagem abaixo.

```
1. sudo apt install openjdk-8-jdk -y
```

A próxima etapa é confirmar que o Java foi instalado e a versão é a 8, para isso execute o comando abaixo.

```
1. sudo update-java-alternatives -l
```

Anote em um bloco de notas o diretório do Java, conforme figura abaixo (grifado).

```
usuário@debian:~$ sudo update-java-alternatives -l
java-1.8.0-openjdk-amd64 1081 /usr/lib/jvm/java-1.8.0-openjdk-amd64
usuário@debian:~$
```

## 4.2 Instalando o WSO2

A instalação da versão 6.6.0 será realizada a partir de um Link próprio para manter compatibilidade do conteúdo com a prática, este é um ambiente de aprendizagem, em produção utilize a última versão do site WSO2.

Para realizar instale download instale o curl, ou por scp/winscp envie para /tmp/, neste material vou optar pelo curl.

```
1. sudo apt install curl -y
```

Iniciamos com um curl conforme código abaixo.

```
1. curl --output /tmp/wso2ei-linux-installer-x64-6.6.0.deb
http://www.aied.com.br/linux/download/wso2ei-linux-installer-x64-6.6.0.deb
```

**Atenção I:** se a url acima falhar, pode-se realizar o download em um computador e enviar por SCP ou WinSCP (tópico [Openssh-server](#)) utilizando a url:

<https://drive.google.com/file/d/1kY0HwhvT6yuJwlq6zPWM0sFf22ohgc8J/view?usp=sharing>

**Atenção II:** vai tomar um café neste momento.

Para saber que deu certo, execute o comando ls conforme comando abaixo.

```
1. ls -lh /tmp/
```

Repare pela imagem que o tamanho do arquivo é de 629MB, como estamos trabalhando com uma versão definida este tamanho quando correto é este.

```

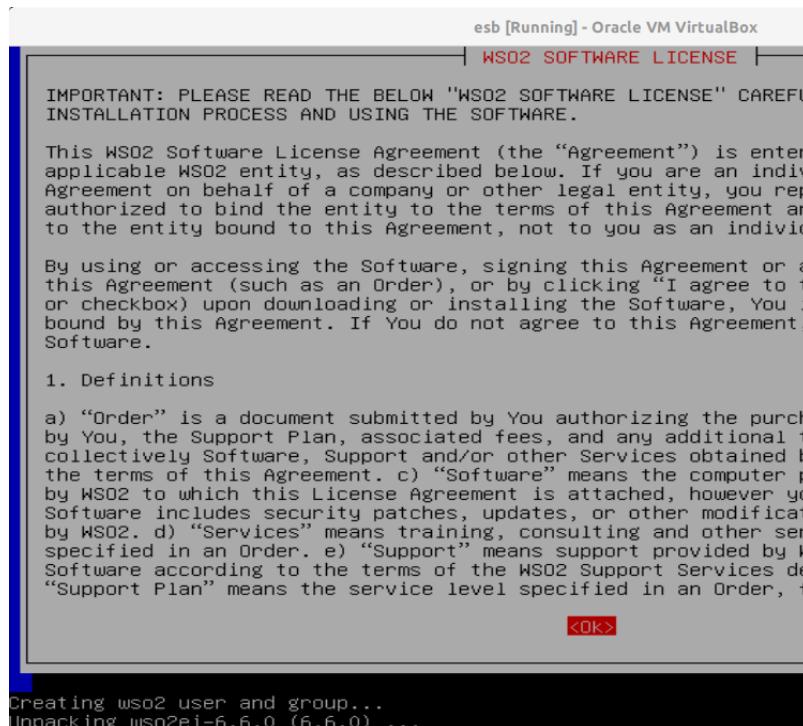
curl: (56) Recv failure: Connection timed out
usuario@debian:~$ curl --output /tmp/wso2ei-linux-installer-x64-6.6.0.deb http://www.aied.com.br/lin
ux/download/wso2ei-linux-installer-x64-6.6.0.deb
 % Total % Received % Xferd Average Speed Time Time Time Current
 Dload Upload Total Spent Left Speed
100 629M 100 629M 0 0 1199K 0 0:08:57 0:08:57 ---:-- 690k
usuario@debian:~$ ls -lh /tmp/
total 630M
drwxr-xr-x 2 root root 4.0K Aug 20 12:22 hsperfdata_root
drwx----- 3 root root 4.0K Aug 20 12:14 systemd-private-94e1c0d66bc9433e95cb6a15a4941755-syst
emd-timesyncd.service-t1DKUH
-rw-r--r-- 1 usuario usuario 630M Aug 20 14:37 wso2ei-linux-installer-x64-6.6.0.deb
usuario@debian:~$ _

```

Para fazer a instalação é simples, basta utilizar o comando dpkg conforme listagem abaixo.

1. sudo dpkg -i /tmp/wso2ei-linux-installer-x64-6.6.0.deb

Aceite os termos de licença do WSO2, conforme figura abaixo, e o processo de desempacotamento vai iniciar.



A instalação deverá gerar um diretório **/usr/lib/wso2/wso2ei/6.6.0/**, para validar isso utilize o comando file.

```
usuario@debian:/usr/lib/wso2/wso2ei/6.6.0$ ls -l
total 124
drwxrwxr-x 3 wso2 wso2 4096 Aug 20 14:43 bin
drwxrwxr-x 11 wso2 wso2 4096 Aug 20 14:43 conf
drwxrwxr-x 3 wso2 wso2 4096 Aug 20 14:43 dbscripts
drwxrwxr-x 2 wso2 wso2 4096 Aug 20 14:43 dropins
drwxrwxr-x 2 wso2 wso2 4096 Nov 22 2019 extensions
drwxrwxr-x 3 wso2 wso2 4096 Aug 20 14:40 jdk
drwxrwxr-x 2 wso2 wso2 4096 Aug 20 14:43 lib
-rw-r--r-- 1 wso2 wso2 44367 Mar 12 2020 LICENSE.txt
drwxrwxr-x 2 wso2 wso2 4096 Nov 22 2019 patches
-rw-r--r-- 1 wso2 wso2 6562 Dec 17 2019 README.txt
-rw-r--r-- 1 wso2 wso2 9946 Dec 17 2019 release-notes.html
drwxrwxr-x 9 wso2 wso2 4096 Aug 20 14:41 repository
drwxrwxr-x 3 wso2 wso2 4096 Aug 20 14:41 resources
drwxrwxr-x 8 wso2 wso2 4096 Aug 20 14:42 samples
drwxrwxr-x 2 wso2 wso2 4096 Nov 22 2019 servicepacks
drwxrwxr-x 2 wso2 wso2 4096 Aug 20 14:43 updates
drwxrwxr-x 3 wso2 wso2 4096 Aug 20 14:42 webapp-mode
drwxrwxr-x 9 wso2 wso2 4096 Aug 20 14:43 wso2
usuario@debian:/usr/lib/wso2/wso2ei/6.6.0$
```

## 4.4 Segurança do serviço

Um ambiente é seguro se já em sua instalação todas as regras de segurança são levadas às sério, mas infelizmente a WSO2 por motivos não divulgados criam um usuário inseguro para seus serviços WSO2 nesta máquina, veja que **/etc/passwd** após a instalação existe um usuário chamado wso2, com diretório errado e com shell definido.

```
usuario@debian:/usr/lib/wso2/wso2ei/6.6.0$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gna
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time Synchronization,,,,:/r
systemd-network:x:102:103:systemd Network Management,,,,:/run/
systemd-resolve:x:103:104:systemd Resolver,,,,:/run/systemd/u
messagebus:x:104:110:/nonexistent:/usr/sbin/nologin
usuario:x:1000:1000:usuario,,,,:/home/usuario:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/no
wso2:x:998:112::/home/wso2:/bin/sh
usuario@debian:/usr/lib/wso2/wso2ei/6.6.0$
```

Para corrigir abra um programa editor, pode usar o nano, abra este arquivo e aponte o diretório default do usuário para o diretório de instalação do wso2 e informe que o shell default é o /bin/false conforme figura abaixo.

```
usuario@debian:/tmp$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/var/www
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time Synchronization
systemd-network:x:102:103:systemd Network Management
systemd-resolve:x:103:104:systemd Resolver,,
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
usuario:x:1000:1000:usuario,,,:/home/usuario:
systemd-coredump:x:999:999:systemd Core Dumper
wso2:x:998:112::/usr/lib/wso2:/bin/false
```

Agora vamos validar o arquivo **/etc/shadow**, veja (figura abaixo), com senha, e isso é ruim.

```
bash: cd: comando não encontrado
usuario@debian:/usr/lib/wso2/wso2ei/6.6.0$ sudo cat /etc/shadow
[sudo] password for usuario:
root:$6$1k/hwpdoSKr549$q1z0qtI41whM8n3EI4DQ18v0Y0T6xfc2zXXpGPkRmPMw3kPx14o
x/2U20Gksx1:18723:0:99999:7:::
daemon:**:18723:0:99999:7:::
bin:**:18723:0:99999:7:::
sys:**:18723:0:99999:7:::
sync:**:18723:0:99999:7:::
games:**:18723:0:99999:7:::
man:**:18723:0:99999:7:::
lp:**:18723:0:99999:7:::
mail:**:18723:0:99999:7:::
news:**:18723:0:99999:7:::
uucp:**:18723:0:99999:7:::
proxy:**:18723:0:99999:7:::
www-data:**:18723:0:99999:7:::
backup:**:18723:0:99999:7:::
list:**:18723:0:99999:7:::
irc:**:18723:0:99999:7:::
gnats:**:18723:0:99999:7:::
nobody:**:18723:0:99999:7:::
_apt:**:18723:0:99999:7:::
systemd-timesync:**:18723:0:99999:7:::
systemd-network:**:18723:0:99999:7:::
systemd-resolve:**:18723:0:99999:7:::
messagebus:**:18723:0:99999:7:::
usuario:6rn7evvvXyzatyK8xL$9ra4Y.oGoeFZDAQLLjqilXByE5BdG2xmlnjEes1Qvy1qYXQC
vyTp1PDhyGx0K/:18823:0:99999:7:::
systemd-coredump:**:18723:0:99999:7:::
wso2:6Cx20obmjw9vNeex5$.to6/Rdey4017MW7IcQV6pVSV1sd18f1B5rSV1avi188v3/95PN
UPswEvK29m/:18859::::::
usuario@debian:/usr/lib/wso2/wso2ei/6.6.0$
```

O próximo passo então é abrir este arquivo com um editor de texto e remover a senha, substitua por uma exclamação conforme figura abaixo.

```

usuario@debian:/tmp$ sudo cat /etc/shadow
root:$6$1K/hwpdoSKr549$qi120qlI4iwhM6n3E14D
X/2U20Gksx1:18723:0:99999:7:::
daemon:*:18723:0:99999:7:::
bin:*:18723:0:99999:7:::
sys:*:18723:0:99999:7:::
sync:*:18723:0:99999:7:::
games:*:18723:0:99999:7:::
man:*:18723:0:99999:7:::
lp:*:18723:0:99999:7:::
mail:*:18723:0:99999:7:::
news:*:18723:0:99999:7:::
uucp:*:18723:0:99999:7:::
proxy:*:18723:0:99999:7:::
www-data:*:18723:0:99999:7:::
backup:*:18723:0:99999:7:::
list:*:18723:0:99999:7:::
irc:*:18723:0:99999:7:::
gnats:*:18723:0:99999:7:::
nobody:*:18723:0:99999:7:::
_apt:*:18723:0:99999:7:::
systemd-timesync:*:18723:0:99999:7:::
systemd-network:*:18723:0:99999:7:::
systemd-resolve:*:18723:0:99999:7:::
messagebus:*:18723:0:99999:7:::
usuario:6rn7evvXyzatyK8xL$9ra4Y.oGoeFZDAQL
vyTp1PDhyGxOK/:18723:0:99999:7:::
systemd-coredump:!!:18723:::::
wso2:!:18859:::::

```

O leitor deve estar atento, pois o ESB é mais valioso que um banco de dados na visão de um hacker.

## 4.5 Gerar um script de inicialização

Para iniciar o serviço WSO2, será utilizado o Systemd, para isso com o comando nano (sudo lógico) gere um novo arquivo /etc/systemd/system/esb.service.

1. sudo nano /etc/systemd/system/esb.service

Edite a seguinte configuração abaixo, repare que na linha 8 é utilizado o **path** da instalação do java obtido pelo comando **update-java-alternatives**.

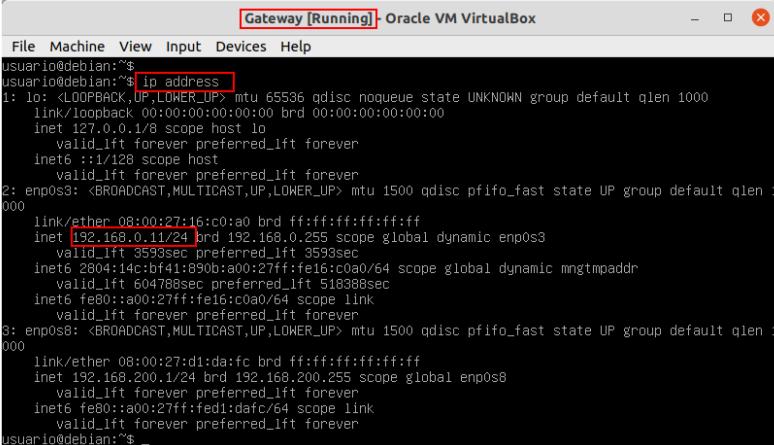
1. [Unit]
2. Description=Enterprise Service Bus WSO2
3. After=network.target
- 4.
5. [Service]
6. Type=forking
- 7.
8. Environment=JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
9. Environment='CATALINA\_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
10. Environment='JAVA\_OPTS=-Djava.awt.headless=true
-Djava.security.egd=file:/dev/.urandom'
- 11.
12. ExecStart=/usr/lib/wso2/wso2ei/6.6.0/bin/integrator.sh start > /dev/null &
13. ExecStop=/usr/lib/wso2/wso2ei/6.6.0/bin/integrator.sh stop > /dev/null &
- 14.
15. User=wso2

16. Group=wso2
17. UMask=0007
18. RestartSec=210
19. Restart=always
- 20.
21. [Install]
22. WantedBy=multi-user.target

O script de inicialização é o **/usr/lib/wso2/wso2ei/6.6.0/bin/integrator.sh**, o arquivo acima é concebido para auto inicialização, caso queira executar manualmente não utilize a opção do Systemd.

## 4.6 Utilizando o painel do Enterprise Service Bus

Em uma máquina com interface gráfica dentro da rede local (ver imagem do início do capítulo), inicie um browser de sua preferência, agora informe o IP a URL: [http://ip\\_gateway:9443/carbon/](http://ip_gateway:9443/carbon/)

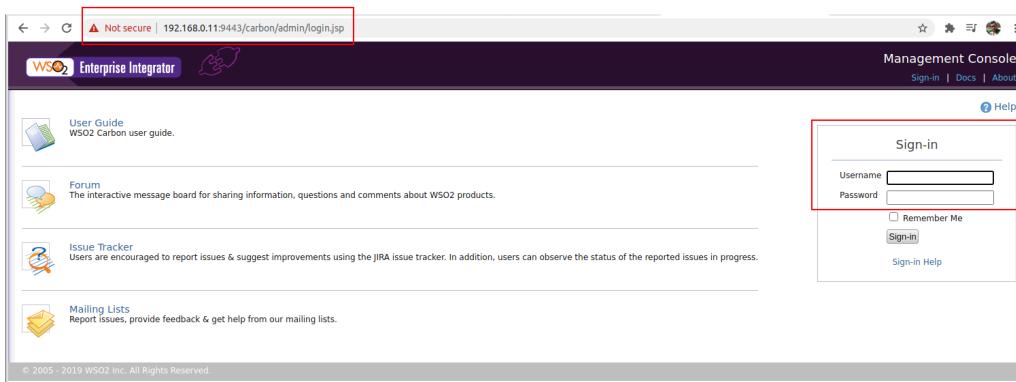


```

Gateway [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuario@debian:~$ ip address
usuario@debian:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 brd :: scope host
 valid_lft forever preferred_lft forever
2: enp0s3: <>BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 09:00:27:16:c0:a0 brd ff:ff:ff:ff:ff:ff
 inet 192.168.0.11/24 brd 192.168.0.255 scope global dynamic enp0s3
 valid_lft 3593sec preferred_lft 3593sec
 inet6 2804:14c:bf41:890b:a00:27ff:fe16:c0a/64 scope global dynamic mngtmpaddr
 valid_lft 604788sec preferred_lft 518388sec
 inet6 fe80::a00:27ff:fe16:c0a/64 scope link
 valid_lft forever preferred_lft forever
3: enp0s8: <>BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 09:00:27:d1:da:fc brd ff:ff:ff:ff:ff:ff
 inet 192.168.200.1/24 brd 192.168.200.255 scope global enp0s8
 valid_lft forever preferred_lft forever
 inet6 fe80::a00:27ff:fed1:dafc/64 scope link
 valid_lft forever preferred_lft forever
usuario@debian:~$

```

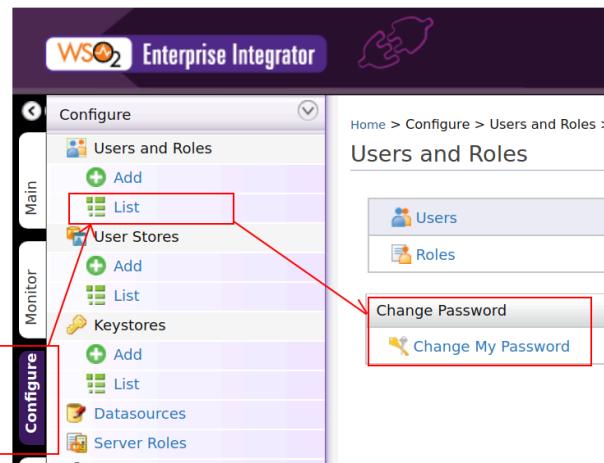
Obtenha o IP conforme comando **ip address** exibido na figura acima, e digite a url conforme imagem abaixo ([https://IP\\_DA\\_INTERFACE\\_ENP0S3\\_GATEWAY:9443/carbon/admin/login.jsp](https://IP_DA_INTERFACE_ENP0S3_GATEWAY:9443/carbon/admin/login.jsp))



### 4.6.1 Acessando a interface de manutenção

Quando se instala o WSO2 o sistema implanta uma base de dados e nesta base de dados é salvo um usuário administrador inicial chamado admin que a senha padrão é admin. Na imagem de login acima informe Username: **admin** e Password: **admin**, pressione **Sign-in**.

O primeiro procedimento é implantar uma senha muito forte, lembre-se senha forte é uma senha não vazada, veja o recurso de vídeo abaixo para compreender. Clique em **Configure** na aba lateral depois em **List** e **Change My Password**, veja na figura abaixo.



Estou usando uma senha que vou me lembrar e sei que possui elementos que dificultam uma devida força bruta.

Home > Change Password

Change Password For User: admin

Enter New Password

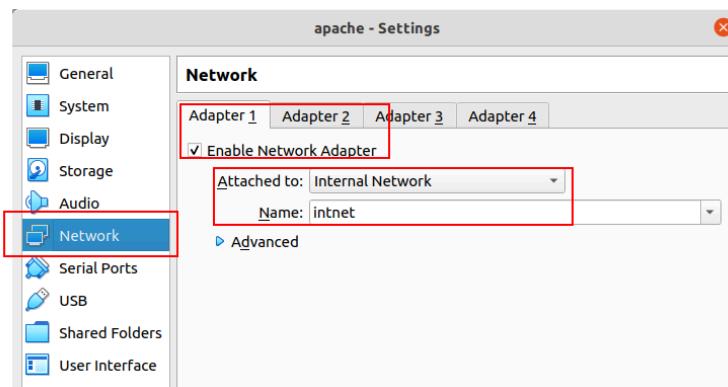
Current Password\* .....  
New Password\* .....  
Confirm New Password\* .....

Change Cancel

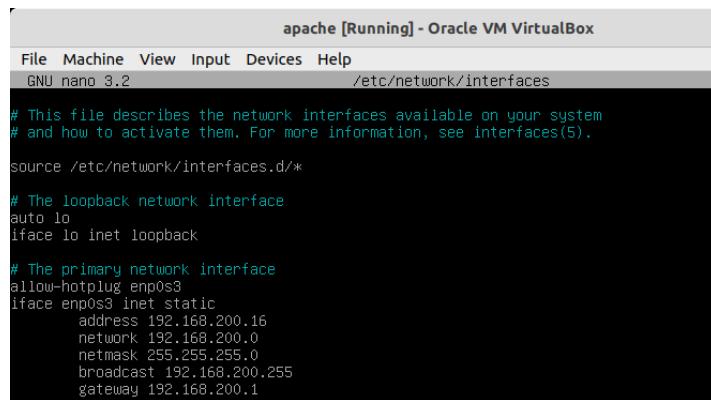
Agora o ideal é construir alguns serviços básicos para configurar o ESB.

## 5 Instalando o Apache um provedor de dados

Para configuração da máquina virtual Apache a máquina deve estar na rede interna que é protegida pelo Gateway, esta deve ser a única conexão para o mundo exterior.



Quando iniciar a máquina virtual, configure o arquivo **/etc/network/interfaces** conforme a imagem abaixo.



```

apache [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 3.2 /etc/network/interfaces

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*
The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
 address 192.168.200.16
 network 192.168.200.0
 netmask 255.255.255.0
 broadcast 192.168.200.255
 gateway 192.168.200.1

```

A opção gateway é opcional, como nada foi instalado vamos adicionar, agora instale o apache 2 e o PHP com o comando abaixo.

1. sudo apt install apache2 php -y

Agora vamos criar um serviço fictício somente para consulta, com o nano crie o arquivo **/var/www/html/pessoa.php** e edite o seguinte código.

1. <?php
2. error\_reporting(0);
3. \$data = json\_decode(file\_get\_contents('php://input'), true);
- 4.
5. echo json\_encode(array( "matricula" => \$data['matricula'], "nome" => "Wellington" ));
6. ?>

O serviço está pronto para o uso, agora o próximo passo é configurar o serviço no ESB.

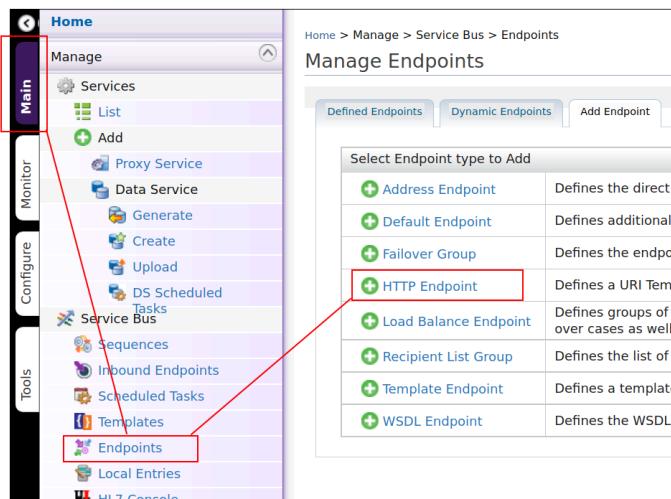
## 6 Configurando um serviço no ESB

O objetivo deste conteúdo é apenas desenvolver o conhecimento básico de ESB, curso completo sobre seu uso e sobre sua teoria pode ser localizada no link abaixo.



Curso de Enterprise Service Bus, da teoria à prática, [veja este link](#).

Primeiro passo é criar um Endpoint, para isso em **Main** acesse **Endpoints**, escolha a opção **HTTP Endpoint** conforme figura abaixo.



Entre com um nome, para este endpoint utilize o nome pessoa e informe a URL do serviço interno que está protegido atrás do gateway, é um serviço do tipo POST e recebe JSON como input e retorna JSON.

Home > HTTP Endpoint

### HTTP Endpoint

HTTP Endpoint [Switch to source view](#)

Name *	pessoa						
URI Template *	http://192.168.200.16/pessoa.php						
OPTIONS							
HTTP Method	POST						
<input checked="" type="checkbox"/> Show Advanced Options							
Endpoint Properties							
<a href="#">Add Property</a> <table border="1"> <tr> <th>Name</th> <th>Value</th> <th>Scope</th> </tr> <tr> <td>matricula</td> <td></td> <td>Synapse</td> </tr> </table>		Name	Value	Scope	matricula		Synapse
Name	Value	Scope					
matricula		Synapse					
<input type="button" value="Save &amp; Close"/> <input type="button" value="Save in Registry"/> <input type="button" value="Cancel"/>							

O próximo passo é ativar as estatísticas para obter dados de requisições.

Home > Manage > Service Bus > Endpoints

### Manage Endpoints

Defined Endpoints [Dynamic Endpoints](#) [Add Endpoint](#)

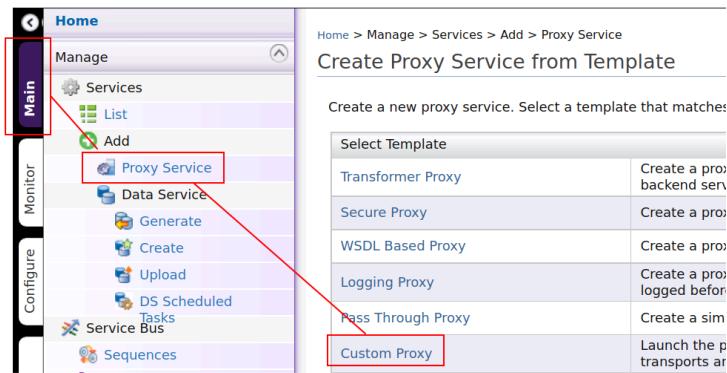
Search Endpoint

Available defined Endpoints in the Synapse Configuration > 1

Select all in this page | Select none

Select	Endpoint Name	Type	Action
<input type="checkbox"/>	pessoa	HTTP Endpoint	<a href="#">Switch Off</a> <a href="#">Disable Statistics</a> <a href="#">Edit</a>

Com o endpoint criado e configurado o próximo passo é utilizar um proxy interno para o endpoint, então em **Main** acesse **Proxy Service** e use a opção **Custom Proxy**.



Para o proxy, será utilizado o mesmo nome do endpoint, depois clique em **Next**.

Proxy Service Name\*

**General Settings**

Publishing WSDL

Service Parameters [+ Add Parameter](#)

Service Group

Do Not load service on startup

Pinned servers (separated by comma or space)

Service Description

**Transport Settings**

http

https

local

vfs

**Buttons:** Next> | Cancel

Na segunda etapa escolha o endpoint “pessoa” criado anteriormente.

Define Endpoint

None

Define Inline

Pick from Registry

Use Existing Endpoint

<Back | Next> | Cancel

Clique em **Next**, depois na terceira etapa apenas finalize a configuração.

Design [switch to source view](#)

### Step 3 of 3 - Out Sequence and Fault

**Proxy Service Name: pessoa**  
 Out sequence mediates the response messages from the proxy service.  
 Both out sequence and fault sequence are optional.

**Define Out Sequence**

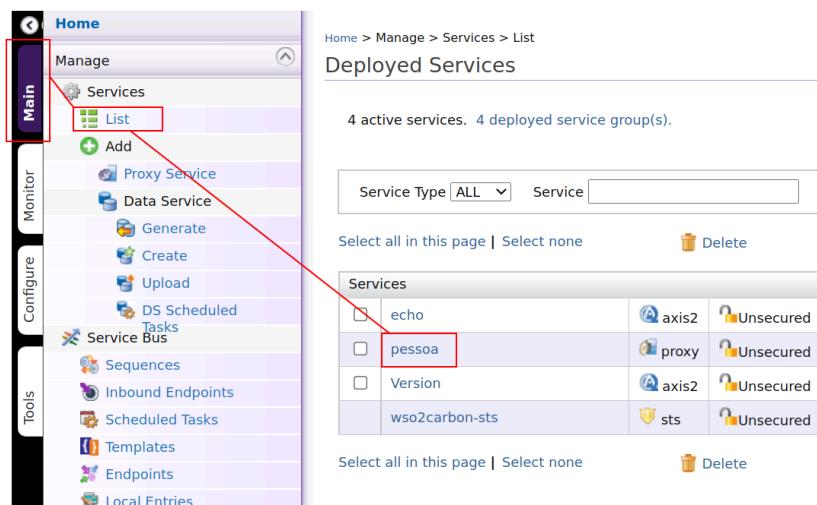
None  
 Define Inline  
 Pick from Registry  
 Use Existing Sequence

**Define Fault Sequence**

None  
 Define Inline  
 Pick from Registry  
 Use Existing Sequence

[<Back](#) [Finish](#) [Cancel](#)

O novo proxy do endpoint agora está disponível na Lista, clique sobre o proxy recém criado e para compreender as opções do serviço.



Home > Manage > Services > List

### Deployed Services

4 active services. 4 deployed service group(s).

Service Type	ALL	Service	
<input type="checkbox"/>	echo	axis2	Unsecured
<input type="checkbox"/>	pessoa	proxy	Unsecured
<input type="checkbox"/>	Version	axis2	Unsecured
<input type="checkbox"/>	wso2carbon-sts	sts	Unsecured

Três áreas na tela de edição são importantes, a primeira são definidos os endereços tanto http quanto https, repare que o nome utilizado é o mesmo do hostname da própria máquina, o interessante é trocar o nome da máquina bem como editar o hosts da própria máquina ESB.

Home > Service Dashboard

## Service Dashboard (pessoa)

Service Details	
Service Name	pessoa
Service Description	No service description found
Service Group Name	pessoa
Deployment Scope	request
Service Type	proxy

Client Operations	
Try this service	
Generate Axis2 Client	
WSDL1.1	

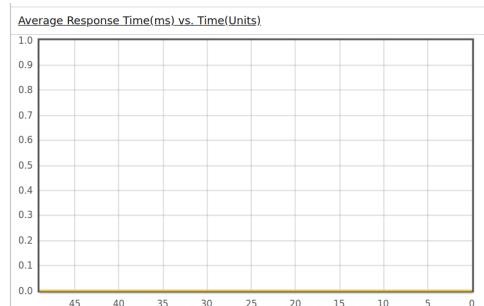
Endpoints	
<a href="http://debian:8280/services/pessoa">http://debian:8280/services/pessoa</a>	
<a href="https://debian:8243/services/pessoa">https://debian:8243/services/pessoa</a>	

Operations	
Active	<a href="#">[ Deactivate ]</a>

Quality of Service Configuration	
<b>Specific Configuration</b> <div style="display: flex; justify-content: space-between; align-items: center;"> <span> Edit</span> <span> Disable Statistics</span> <span> Enable Tracing</span> </div>	

Statistics	
Request Count	0
Response Count	0
Fault Count	0
Maximum Response Time	0 ms

Habilite as estatísticas para observar a terceira área que retorna dados de desempenho do serviço.

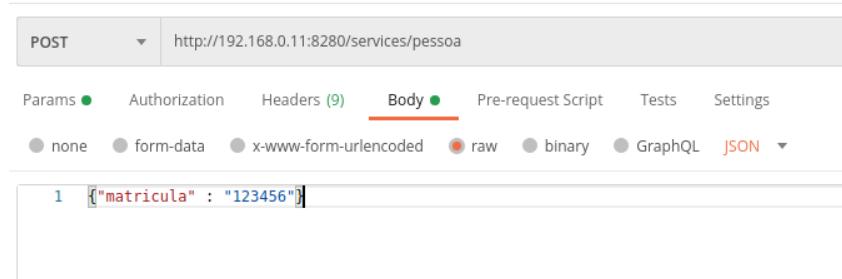


## 7 Testando o serviço

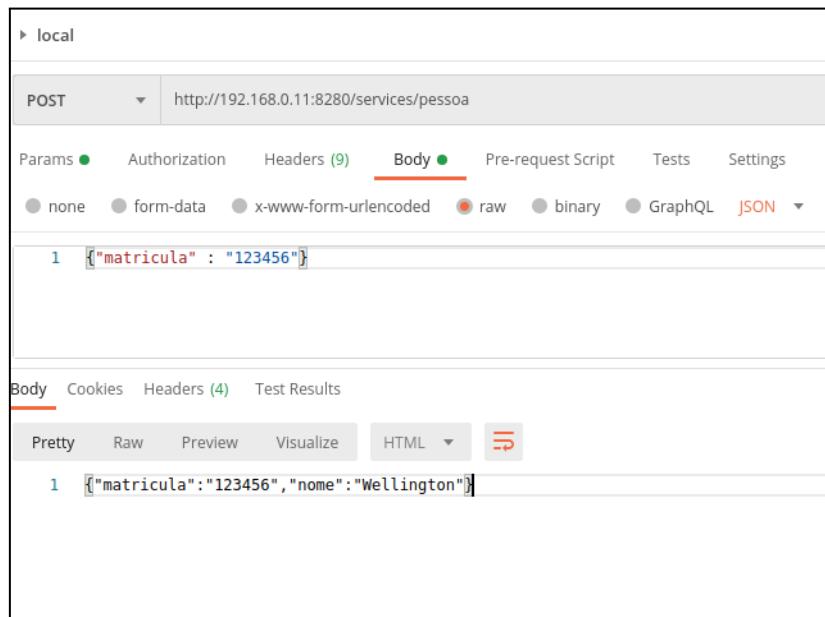
Para realizar o teste, será utilizado o **Postman**, então instale e inicie esta aplicação. Crie uma nova requisição POST, informe a URL obtida no ESB trocando a palavra debian pelo IP do gateway em sua rede local, conforme figura abaixo.

The screenshot shows the Postman interface with a POST request to <http://192.168.0.11:8280/services/pessoa>. The 'Headers' tab is highlighted with a red box. A single header 'Content-Type' is listed with the value 'application/json'.

Em Headers defina o content-type para application/json, este content-type será passado do ESB para o Apache com o serviço. Na aba Body vamos enviar um JSON com uma matrícula conforme figura abaixo.



Ao clicar no botão “Send” a requisição será enviada para o ESB que deverá processar o Endpoint enviando a requisição para o Apache, com o retorno o ESB responde ao Postman, conforme figura abaixo.



## 8 Práticas do capítulo

Nesta sessão o aluno deve realizar as práticas na sequência, caso ocorra problemas é possível que o mesmo deva ter a conhecimento para desfazer, isto já é levado em consideração.

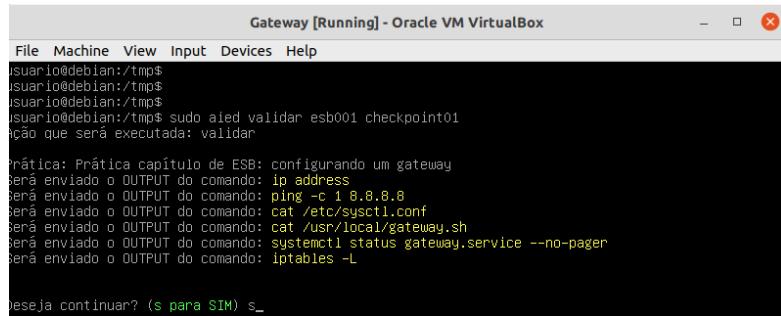
### 8.1 Prática esb001 01: Configurando o Gateway

Nesta prática o aluno deve ter a expertise para configurar o serviço Gateway bem como regras de iptables;

1. Proceda no configuração do Gateway conforme este tópico: [Preparando o ambiente](#)

Execute o comando aied conforme comando listagem abaixo.

## 1. sudo aied validar esb001 checkpoint01



```

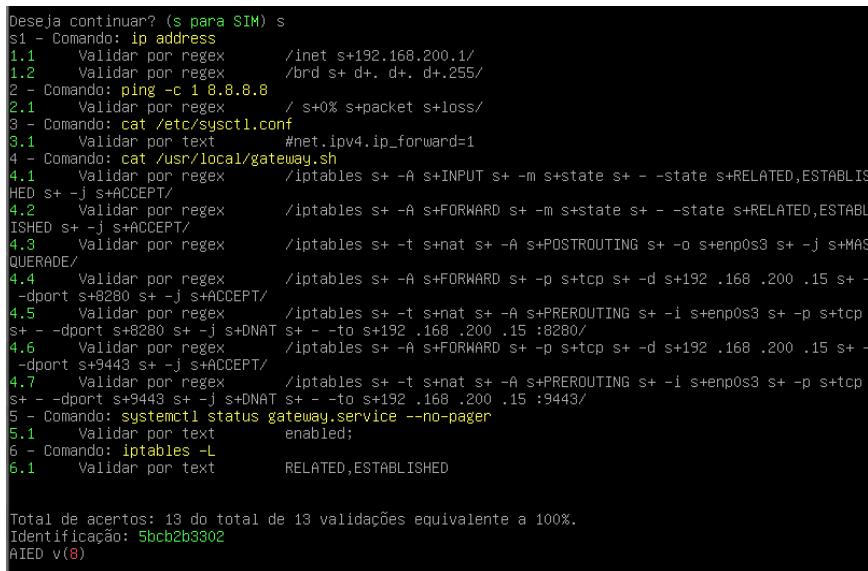
Gateway [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
usuário@debian:/tmp$
usuário@debian:/tmp$
usuário@debian:/tmp$
usuário@debian:/tmp$ sudo aied validar esb001 checkpoint01
ação que será executada: validar

Prática: Prática capítulo de ESB: configurando um gateway
será enviado o OUTPUT do comando: ip address
será enviado o OUTPUT do comando: ping -c 1 8.8.8.8
será enviado o OUTPUT do comando: cat /etc/systcl.conf
será enviado o OUTPUT do comando: cat /usr/local/gateway.sh
será enviado o OUTPUT do comando: systemctl status gateway.service --no-pager
será enviado o OUTPUT do comando: iptables -L

Deseja continuar? (s para SIM) s_

```

O resultado esperado é todos os itens aprovados conforme figura abaixo.



```

Deseja continuar? (s para SIM) s
s1 - Comando: ip address
1.1 Validar por regex /inet s+192.168.200.1/
1.2 Validar por regex /brd s+ d+. d+. 255/
2 - Comando: ping -c 1 8.8.8.8
2.1 Validar por regex /s+0% s+packet s+loss/
3 - Comando: cat /etc/systcl.conf
3.1 Validar por texto #net.ipv4.ip_forward=1
4 - Comando: cat /usr/local/gateway.sh
4.1 Validar por regex /iptables s+ -A s+INPUT s+ -m s+state s+ - -state s+RELATED,ESTABLISHED s+ -j s+ACCEPT/
4.2 Validar por regex /iptables s+ -A s+FORWARD s+ -m s+state s+ - -state s+RELATED,ESTABLISHED s+ -j s+ACCEPT/
4.3 Validar por regex /iptables s+ -t s+nat s+ -A s+POSTROUTING s+ -o s+enp0s3 s+ -j s+MASQUERADE/
4.4 Validar por regex /iptables s+ -A s+FORWARD s+ -p s+tcp s+ -d s+192 .168 .200 .15 s+ - -dport s+8280 s+ -j s+ACCEPT/
4.5 Validar por regex /iptables s+ -t s+nat s+ -A s+PREROUTING s+ -i s+enp0s3 s+ -p s+tcp s+ - -dport s+8280 s+ -j s+DNAT s+ - -to s+192 .168 .200 .15 :8280/
4.6 Validar por regex /iptables s+ -A s+FORWARD s+ -p s+tcp s+ -d s+192 .168 .200 .15 s+ - -dport s+9443 s+ -j s+ACCEPT/
4.7 Validar por regex /iptables s+ -t s+nat s+ -A s+PREROUTING s+ -i s+enp0s3 s+ -p s+tcp s+ - -dport s+9443 s+ -j s+DNAT s+ - -to s+192 .168 .200 .15 :9443/
5 - Comando: systemctl status gateway.service --no-pager
5.1 Validar por texto enabled;
6 - Comando: iptables -L
6.1 Validar por texto RELATED,ESTABLISHED

Total de acertos: 13 do total de 13 validações equivalente a 100%.
Identificação: 5bcb2b3302
AIED v(8)

```

Após aceitar, será então validado todos os tópicos conforme figura abaixo, tudo deve estar verde.

## 8.2 Prática esb001 02: Configurando o Enterprise Service Bus

Nesta prática o aluno deve ter a expertise para configurar o serviço o ESB;

1. Proceda com a instalação e configuração do ESB conforme este tópico: [Instalando o Enterprise service Bus WSO2](#)

Execute o comando aied conforme comando listagem abaixo.

## 2. sudo aied validar esb001 checkpoint02

```
usuario@debian:~$ sudo aied validar esb001 checkpoint02
Ação que será executada: validar

Prática: Instalando o ESB WS02 - Instalação e configuração
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: ping -c 1 192.168.200.1
Será enviado o OUTPUT do comando: cat /etc/systemd/system/esb.service
Será enviado o OUTPUT do comando: systemctl status esb.service --no-pager

Deseja continuar? (s para SIM)
```

O resultado esperado é todos os itens aprovados conforme figura abaixo.

```
File Machine View Input Devices Help
Deseja continuar? (s para SIM) s
1 - Comando: ip address
1.1 Validar por regex /inet s+192.168.200.15/
1.2 Validar por regex /brd s+d+, d+, d+.
1.3 Comando: ping -c 1 192.168.200.1
1.4 Validar por regex $+0% s+packet s+loss/
2 - Comando: cat /etc/systemd/system/esb.service
2.1 Validar por text [Unit]
2.2 Validar por regex /Description=(.*) n/
2.3 Validar por text After=network.target
2.4 Validar por text [Service]
2.5 Validar por text Type=forking
2.6 Validar por regex /Environment=JAVA_HOME= /usr /lib /jvm /java(.*)openjdk-(amd64|i386)
2.7 Validar por text Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
2.8 Validar por text Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/.urandom'
2.9 Validar por text ExecStart=/usr/lib/wso2/wso2el/6.0.0/bin/integrator.sh start > /dev/null
2.10 Validar por text ExecStop=/usr/lib/wso2/wso2el/6.0.0/bin/integrator.sh stop > /dev/null
2.11 Validar por text User=wso2
2.12 Validar por text Group=wso2
2.13 Validar por text Restart=always
2.14 Validar por text [Install]
2.15 Validar por regex /WantedBy=(multi-user|graphical).target/
3 - Comando: systemctl status esb.service --no-pager
3.1 Validar por text active (running)
3.2 Validar por text enabled;
```

Total de acertos: 20 do total de 20 validações equivalente a 100%.

Identificação: 95ca0d31a0

AIED v(8)

Após aceitar, será então validado todos os tópicos conforme figura abaixo, tudo deve estar verde.

### 8.3 Prática esb001 03: Configurando o serviço WEB com Apache e PHP

Nesta prática o aluno deve ter a expertise instalar e configurar o Apache e o PHP, para isso ele deve realizar toda a configuração conforme o capítulo.

A validação deve ser feita na Virtual Machine com Apache, entre na Virtual Machine e digite o seguinte comando:

```
sudo aied validar esb001 checkpoint03
```

```
File Machine View Input Devices Help
usuario@debian:~$ sudo aied validar esb001 checkpoint03
Ação que será executada: validar

Prática: Instalando o ESB WS02 - Instalação e configuração de serviço Apache
Será enviado o OUTPUT do comando: ip address
Será enviado o OUTPUT do comando: ping -c 1 192.168.200.1
Será enviado o OUTPUT do comando: ping -c 1 192.168.200.15
Será enviado o OUTPUT do comando: systemctl status apache2.service --no-pager
Será enviado o OUTPUT do comando: file /var/www/html/pessoa.php
Será enviado o OUTPUT do comando: php -v

Deseja continuar? (s para SIM) s
```

O resultado esperado é todos os itens aprovados conforme figura abaixo.

```

Deseja continuar? (s para SIM) s
si - Comando: ip address
1.1 Validar por regex /inet s+192.168.200.16/
1.2 Validar por regex /brd s+ d+. d+. d+.255/
2 - Comando: ping -c 1 192.168.200.1
2.1 Validar por regex / s+0% s+packet s+loss/
3 - Comando: ping -c 1 192.168.200.15
3.1 Validar por regex / s+0% s+packet s+loss/
4 - Comando: systemctl status apache2.service --no-pager
4.1 Validar por text active (running)
4.2 Validar por text enabled;
5 - Comando: file /var/www/html/pessoa.php
5.1 Validar por text PHP script
6 - Comando: php -v
6.1 Validar por regex /PHP s+[0-9]/

Total de acertos: 8 do total de 8 validações equivalente a 100%.
Identificação: 3f3e1141d4
AIED v(8)

usuario@debian:~$
```

## Respostas certificação

1	Correct Answer: C	50	Correct Answer: B
2	Correct Answer: AD	51	Correct Answer: B
3	Correct Answer: AC	52	Correct Answer: C
4	Correct Answer: BCE Section: System Architecture Explanation	53	Correct Answer: B
5	Correct Answer: cmdline -or- /proc/cmdline	54	Correct Answer: A
6	Correct Answer: ABD Section: System Architecture Explanation	55	Correct Answer: HISTFILE
7	Correct Answer: CD	56	Correct Answer: B
8	Correct Answer: B	57	Correct Answer: C
9	Correct Answer: dmesg -or- /bin/dmesg Section: System Architecture Explanation	58	Correct Answer: B
10	Correct Answer: D	59	Correct Answer: D
11	Correct Answer: D	60	Correct Answer: AB
12	Correct Answer: E	61	Correct Answer: B
13	Correct Answer: D	62	Correct Answer: B
14	Correct Answer: A	63	Correct Answer: B
15	Correct Answer: B	64	Correct Answer: D
16	Correct Answer: ABD	65	Correct Answer: AD
17	Correct Answer: A	66	Correct Answer: C
18	Correct Answer: D	67	Correct Answer: E
19	Correct Answer: C	68	Correct Answer: C
20	Correct Answer: /tmp -or- tmp -or- /var/tmp -or- /tmp/ -or- /var/tmp/ Section: Linux Installation and Package Management Explanatio	69	Correct Answer: D
21	Correct Answer: B	70	Correct Answer: C
22	Correct Answer: /etc/yum.repos.d	71	Correct Answer: B
		72	Correct Answer: D
		73	Correct Answer: D
		74	Correct Answer: C
		75	Correct Answer: C
		76	Correct Answer: mkdir -or- /usr/bin/mkdir Section: GNU and Unix Commands Explanation
		77	Correct Answer: CD

<p>-or- /etc/yum.repos.d/ -or- yum.repos.d -or- yum.repos.d/</p> <p>23    Correct Answer: menu.lst -or- grub.conf -or- grub.cfg Section: Linux Installation and Package Management Explanation</p> <p>24    Correct Answer: C</p> <p>25    Correct Answer: D</p> <p>26    Correct Answer: update -or- upgrade</p> <p>27    Correct Answer: swapoff -or- /sbin/swapoff</p> <p>28    Correct Answer: dpkg-reconfigure</p> <p>29    Correct Answer: C</p> <p>30    Correct Answer: B</p> <p>31    Correct Answer: C</p> <p>32    Correct Answer: D</p> <p>33    Correct Answer: B</p> <p>34    Correct Answer: B</p> <p>35    Correct Answer: D</p> <p>36    Correct Answer: HUP -or- SIGHUP -or- 1 Section: GNU and Unix Commands Explanation</p> <p>37    Correct Answer: B</p> <p>38    Correct Answer: C</p> <p>39    Correct Answer: E</p> <p>40    Correct Answer: A</p> <p>41    Correct Answer: D</p> <p>42    Correct Answer: D</p> <p>43    Correct Answer: C</p> <p>44    Correct Answer: A</p> <p>45    Correct Answer: A</p> <p>46    Correct Answer: B</p> <p>47    Correct Answer: B</p> <p>48    Correct Answer: B</p> <p>49    Correct Answer: jobs</p>	<p>78    Correct Answer: C</p> <p>79    Correct Answer: B</p> <p>80    Correct Answer: AC</p> <p>81    Correct Answer: ABC</p> <p>82    Correct Answer: B</p> <p>83    Correct Answer: CE</p> <p>84    Correct Answer: B</p> <p>85    Correct Answer: A</p> <p>86    Correct Answer: C</p> <p>87    Correct Answer: D</p> <p>88    Correct Answer: D</p> <p>89    Correct Answer: B</p> <p>90    Correct Answer: AC</p> <p>91    Correct Answer: E</p> <p>92    Correct Answer: C</p> <p>93    Correct Answer: A</p> <p>94    Correct Answer: D</p> <p>95    Correct Answer: quotacheck</p> <p>96    Correct Answer: C</p> <p>97    Correct Answer: E</p> <p>98    Correct Answer: AB</p> <p>99    Correct Answer: A</p> <p>100    Correct Answer: C</p> <p>101    Correct Answer: AC</p> <p>102    Correct Answer: D</p>

## Correcções de falhas

fx00001 Conectividade com o cliente

fx00002 -> configuração do server gateway

fx00003 - 2 placas de rede

fx00004 - iniciar serviço gateway.service

## Referências

//todo adicionar nas referencias :  
<https://www.debian.org/doc/manuals/debian-reference/ch02.pt.html>

### 9.6.3 Prática 8ab001 03: Imprimindo mas com python2

Nesta prática o aluno deverá criar um script, de acordo com o roteiro abaixo.

8. Criar um arquivo no diretório ~/ com o nome atividadeiii.py;
9. Defina o interpretador deste arquivo o Python2;
10. Faça a chamada do prin, o print deve escrever seu nome;
11. Salve o arquivo;
12. Dê a permissão de execução;

Agora execute o comando aied para validar a atividade:

3. sudo aied validar 8ab001 checkpoint03

Veja abaixo a execução da validação da atividade.

```
usuario@debian:~$ sudo aied validar 8ab001 checkpoint03
Ação que será executada: validar

Prática: Prática do capítulo de Shell Script
Será enviado o OUTPUT do comando: /home/aluno/atividadeiii.py
Será enviado o OUTPUT do comando: cat /home/usuario/atividadeiii.py
Será enviado o OUTPUT do comando: ls /home/usuario/atividadeiii.py -l

Deseja continuar? (s para SIM) s
s1 - Comando: /home/aluno/atividadeiii.py
 1.1 Validar por regex / w+/
 2 - Comando: cat /home/usuario/atividadeiii.py
 2.1 Validar por regex /print s+ "/"
 3 - Comando: ls /home/usuario/atividadeiii.py -l
 3.1 Validar por regex /-rwxr-xr-x(.*)atividadeiii.py/

Total de acertos: 3 do total de 3 validações equivalente a 100%.
Identificação: 92a575f60c
AIED v(8)
```

Usuários no documento: //\*[contains(@class, 'collab-widget-active')]

```
resize_image("./projects/blackcat/images/1.jpg");
```

## 15.9 Monitorando portas com nmap

Um scan de portas muito conhecido é o nmap, descrito no livro “[Hacker entre a luz e as trevas](#)”, trata-se de uma aplicação que explora uma série de regras de rede para localizar máquinas e portas de comunicação. Sempre é o primeiro passo para um invasor ou aplicativo mal intencionado, mas é fácil perceber quando um agente realiza um scan de portas, conforme figura abaixo inúmeras requisições sótivas abertas sem se quer ter um retorno.

15	16.833429	192.168.0.100	192.168.0.1	TCP	54 37994 → 94 [SYN] Seq=0 Win=4096 Len=0
16	16.833473	192.168.0.100	192.168.0.1	TCP	54 37994 → 419 [SYN] Seq=0 Win=4096 Len=0
17	16.833490	192.168.0.100	192.168.0.1	TCP	54 37994 → 352 [SYN] Seq=0 Win=4096 Len=0
18	16.833533	192.168.0.100	192.168.0.1	TCP	54 37994 → 9 [SYN] Seq=0 Win=4096 Len=0
19	16.833552	192.168.0.100	192.168.0.1	TCP	54 37994 → 590 [SYN] Seq=0 Win=4096 Len=0
20	16.833574	192.168.0.100	192.168.0.1	TCP	54 37994 → 426 [SYN] Seq=0 Win=4096 Len=0
21	16.833595	192.168.0.100	192.168.0.1	TCP	54 37994 → 642 [SYN] Seq=0 Win=4096 Len=0
22	16.833654	192.168.0.100	192.168.0.1	TCP	54 37994 → 443 [SYN] Seq=0 Win=3072 Len=0
23	16.833672	192.168.0.100	192.168.0.1	TCP	54 37994 → 117 [SYN] Seq=0 Win=1024 Len=0
24	16.833693	192.168.0.100	192.168.0.1	TCP	54 37994 → 1430 [SYN] Seq=0 Win=2048 Len=0
25	16.833713	192.168.0.100	192.168.0.1	TCP	54 37994 → 926 [SYN] Seq=0 Win=1024 Len=0
26	16.833734	192.168.0.100	192.168.0.1	TCP	54 37994 → 698 [SYN] Seq=0 Win=2048 Len=0
27	16.833754	192.168.0.100	192.168.0.1	TCP	54 37994 → 163 [SYN] Seq=0 Win=2048 Len=0
28	16.833775	192.168.0.100	192.168.0.1	TCP	54 37994 → 115 [SYN] Seq=0 Win=1024 Len=0
29	16.833795	192.168.0.100	192.168.0.1	TCP	54 37994 → 20005 [SYN] Seq=0 Win=2048 Len=0
30	16.833816	192.168.0.100	192.168.0.1	TCP	54 37994 → 11 [SYN] Seq=0 Win=4096 Len=0
31	16.833840	192.168.0.100	192.168.0.1	TCP	54 37994 → 650 [SYN] Seq=0 Win=1024 Len=0
32	16.833859	192.168.0.100	192.168.0.1	TCP	54 37994 → 1533 [SYN] Seq=0 Win=4096 Len=0
33	16.833880	192.168.0.100	192.168.0.1	TCP	54 37994 → 1541 [SYN] Seq=0 Win=2048 Len=0
34	16.833902	192.168.0.100	192.168.0.1	TCP	54 37994 → 3333 [SYN] Seq=0 Win=4096 Len=0
35	16.833926	192.168.0.100	192.168.0.1	TCP	54 37994 → 5555 [SYN] Seq=0 Win=3072 Len=0

Dizemos que o nmap é um scan “barulhento”, mas conforme visto abaixo seu resultado é fantástico, imagine que uma máquina possui dois serviços abertos, um é o SSH e o outro o HTTP.

```
well@wpo:~$ sudo nmap -sS 192.168.0.11
[sudo] password for well:
Starting Nmap 7.80 (https://nmap.org) at 2021-10-11 21:20 -03
Nmap scan report for 192.168.0.11
Host is up (0.00064s latency).
Not shown: 998 filtered ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
MAC Address: 08:00:27:36:26:9B (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.17 seconds
well@wpo:~$
```

Uma forma eficaz contra o mapeamento de portas por nmap é implantar uma regra de iptables que bloqueia o scan clássico e que é amplamente utilizado, poucos são os Hackers que temporizam este scan.

**ATENÇÃO: ESSAS REGRAS ABAIXO RTAMBEM, SERVERM PARA DOS**

1. sudo iptables -I INPUT -p tcp -m state --state NEW -m recent --name ssh --set
2. sudo iptables -I INPUT -p tcp -m state --state NEW -m recent --name ssh --name ssh --update --seconds 1 --hitcount 1 -j DROP

Adicione as 2 regras conforme imagem abaixo.

```
GNU nano 3.2 /usr/local/sbin/gateway.sh

#!/bin/bash

iptables -t filter -F
iptables -t filter -X
iptables -t nat -F
iptables -t nat -X

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

sudo iptables -I INPUT -p tcp -m state --state NEW -m recent --name ssh --name ssh --set
sudo iptables -I INPUT -p tcp -m state --state NEW -m recent --name ssh --name ssh --update --seconds 1 --hitcount 1 -j DROP
```

Repare que com o nmap agora só foi descoberta a porta 80, já a porta 22 não aparece na listagem.

```
well@wpo:~$ sudo nmap -sS 192.168.0.11
Starting Nmap 7.80 (https://nmap.org) at 2021-10-11 21:27 -03
Nmap scan report for 192.168.0.11
Host is up (0.00087s latency).
Not shown: 999 filtered ports
PORT STATE SERVICE
80/tcp open http
MAC Address: 08:00:27:36:26:9B (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 4.65 seconds
well@wpo:~$
```

Poucos hackers temporizam o scan, este assunto está sendo abordado [neste link](#). No exemplo da figura abaixo foi realizado um ataque temporizado com parâmetro Paranoid, demorou mais de 900 segundos para mapear 4 portas, um computador pode possuir até 65.635 portas ativas. Um hacker astuto pode mapear aquelas portas que mais lhe convém para seu estilo de ataque.

```
well@wpo:~$ sudo nmap -T0 -p22,80 192.168.0.11
Starting Nmap 7.80 (https://nmap.org) at 2021-10-11 21:57 -03
Nmap scan report for 192.168.0.11
Host is up (0.00039s latency).

PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
MAC Address: 08:00:27:36:26:9B (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 900.37 seconds
well@wpo:~$
```

colocar mudança de http para https na infra, por iptables.

## 15.10 UFW (falta)

### Monitorando tráfego da rede interna

tcpdump é um programa de computador analisador de pacotes de rede de dados que é executado em uma interface de linha de comando. Ele permite que o usuário exiba o TCP/IP e outros pacotes sendo transmitidos ou recebidos em uma rede à qual o computador está conectado. Distribuído sob a licença BSD, tcpdump é um software livre.

Tcpdump funciona na maioria dos sistemas operacionais semelhantes ao Unix: Linux, Solaris, FreeBSD, DragonFly BSD, NetBSD, OpenBSD, OpenWrt, macOS, HP-UX 11i e AIX. Nesses sistemas, o tcpdump usa a biblioteca libpcap para capturar pacotes.

tcpdump imprime o conteúdo dos pacotes de rede. Ele pode ler pacotes de uma placa de interface de rede ou de um arquivo de pacote salvo criado anteriormente. tcpdump pode gravar pacotes na saída padrão ou em um arquivo.

Também é possível usar o tcpdump para o propósito específico de interceptar e exibir as comunicações de outro usuário ou computador. Um usuário com os privilégios necessários em um sistema que atua como um roteador ou gateway por meio do qual o tráfego não criptografado, como passagens de Telnet ou HTTP, pode usar tcpdump para visualizar IDs de login, senhas, URLs e conteúdo de sites sendo visualizados ou qualquer outra informação não criptografada.

O usuário pode, opcionalmente, aplicar um filtro baseado em BPF para limitar o número de pacotes vistos pelo tcpdump; isso torna a saída mais utilizável em redes com alto volume de tráfego.



Monitorando interface de REDE com tcpdump, [acessando o link](#).

A instalação é feita pelo apt com o comando abaixo no **GATEWAY**.

### 1. sudo apt install **tcpdump** -y

Para utilizar é simples, digite o comando tcpdump informando a interface de rede alvo.

### 1. sudo tcpdump -i enp0s3

Veja um exemplo do output do comando.

```
21:47:31.016234 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [P.], seq 1303144:1304604, ack 66451, win 65535, length 1460
21:47:31.017479 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [P.], seq 1304604:1306064, ack 66451, win 65535, length 1460
21:47:31.017497 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [!], ack 1306064, win 65535, length 0
21:47:31.034136 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [P.], seq 66451:66987, ack 1306064, win 65535, length 536
21:47:31.034297 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [!], ack 66987, win 65535, length 0
21:47:31.039770 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [P.], seq 66987:67523, ack 1306064, win 65535, length 536
21:47:31.089938 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [!], ack 67523, win 65535, length 0
21:47:31.118949 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [P.], seq 1306064:1307524, ack 67523, win 65535, length 1460
21:47:31.119007 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [!], ack 1307524, win 65535, length 0
21:47:31.120070 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [P.], seq 1307524:1808984, ack 67523, win 65535, length 1460
21:47:31.120090 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [!], ack 1808984, win 65535, length 0
21:47:31.121323 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [P.], seq 1308984:1310444, ack 67523, win 65535, length 1460
21:47:31.121342 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [!], ack 1310444, win 65535, length 0
21:47:31.122502 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [P.], seq 1310444:1311904, ack 67523, win 65535, length 1460
21:47:31.122521 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [!], ack 1311904, win 65535, length 0
21:47:31.123721 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [P.], seq 1311904:1313364, ack 67523, win 65535, length 1460
21:47:31.123739 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [!], ack 1313364, win 65535, length 0
21:47:31.124978 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [P.], seq 1313364:1314824, ack 67523, win 65535, length 1460
21:47:31.124997 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [!], ack 1314824, win 65535, length 0
21:47:31.126218 IP 192.99.34.48.https > 10.0.2.15.43766: Flags [P.], seq 1314824:1316284, ack 67523, win 65535, length 1460
21:47:31.126238 IP 10.0.2.15.43766 > 192.99.34.48.https: Flags [!], ack 1316284, win 65535, length 0
```

<https://arashmilani.com/post?id=53>

export EASYRSA\_VARS\_FILE=/usr/share/easy-rsa/vars

iptables -A FORWARD -i tun+ -j ACCEPT

```
iptables -A FORWARD -i tun+ -o enp0s8 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i enp0s8 -o tun+ -m state --state RELATED,ESTABLISHED -j ACCEPT
```

<https://www.cyberciti.biz/faq/howto-linux-configuring-default-route-with-ipcommand/>

## 4.15 Linux Professional Institute Certification

### QUESTION 63

Which of the following shell redirections will write standard output and standard error output to a file named filename?

- A. 2>&1 >filename
- B. >filename 2>&1
- C. 1>&2>filename
- D. >>filename
- E. 1&2>filename

### QUESTION 64

In the vi editor, which of the following commands will copy the current line into the vi buffer?

- A. c
- B. cc
- C. 1c
- D. yy
- E. 1y

### QUESTION 65

Which of the following sequences in the vi editor saves the opened document and exits the editor? (Choose TWO correct answers.)

- A. esc ZZ
- B. ctrl :w!
- C. esc zz
- D. esc :wq!
- E. ctrl XX

**QUESTION 69**

Which of the following commands will print the last 10 lines of a text file to the standard output?

- A. cat -n 10 filename
- B. dump -n 10 filename
- C. head -n 10 filename
- D. tail -n 10 filename

**QUESTION 72**

What happens after issuing the command vi without any additional parameters?

- A. vi starts and loads the last file used and moves the cursor to the position where vi was when it last exited.
- B. vi starts and requires the user to explicitly either create a new or load an existing file.
- C. vi exits with an error message as it cannot be invoked without a file name to operate on.
- D. vi starts in command mode and opens a new empty file.
- E. vi starts and opens a new file which is filled with the content of the vi buffer if the buffer contains text.

**QUESTION 75**

Which of the following commands determines the type of a file by using a definition database file which contains information about all common file types?

- A. magic
- B. type
- C. file
- D. pmagic
- E. hash

**QUESTION 76****SIMULATION**

Which command is used in a Linux environment to create a new directory? (Specify ONLY the command without any path or parameters.)

**QUESTION 77**

Which of the following commands prints all files and directories within the /tmp directory or its subdirectories which are also owned by the user root? (Choose TWO correct answers.)

- A. find /tmp -uid root -print
- B. find -path /tmp -uid root
- C. find /tmp -user root -print
- D. find /tmp -user root
- E. find -path /tmp -user root print

**QUESTION 82**

After successfully creating a hard link called bar to the ordinary file foo, foo is deleted from the filesystem. Which of the following describes the resulting situation?

- A. foo and bar would both be removed.
- B. foo would be removed while bar would remain accessible.
- C. foo would be removed. bar would still exist but would be unusable.
- D. Both foo and bar would remain accessible.
- E. The user is prompted whether bar should be removed, too.

**QUESTION 83**

After moving data to a new filesystem, how can the former path of the data be kept intact in order to avoid reconfiguration of existing applications? (Choose TWO correct answers.)

- A. By creating an ACL redirection from the old to the new path of the data.
- B. By creating a hard link from the old to the new path of the data.
- C. By creating a symbolic link from the old to the new path of the data.
- D. By running the command touch on the old path.
- E. By mounting the new filesystem on the original path of the data.

**QUESTION 84**

Which of the following commands changes the ownership of file.txt to the user dan and the group staff?

- A. chown dan/staff file.txt
- B. chown dan:staff file.txt
- C. chown -u dan -g staff file.txt
- D. chown dan -g staff file.txt

**QUESTION 85**

Which of the following commands makes /bin/foo executable by everyone but writable only by its owner?

- A. chmod u=rwx,go=rx /bin/foo
- B. chmod o+rwx,a+rx /bin/foo
- C. chmod 577 /bin/foo
- D. chmod 775 /bin/foo

**QUESTION 88**

Which of the following settings for umask ensures that new files have the default permissions -rw-r---- ?

- A. 0017
- B. 0640
- C. 0038
- D. 0027

**QUESTION 96) Which of the following file permissions belong to a symbolic link?**

- A. -rwxrwxrwx
- B. +rwxrwxrwx

- C. lrwxrwxrwx
- D. srwxrwxrwx

QUESTION 97) Creating a hard link to an ordinary file returns an error. What could be the reason for this?

- A. The source file is hidden.
- B. The source file is read-only.
- C. The source file is a shell script.
- D. The source file is already a hard link.
- E. The source and the target are on different filesystems.

## 5.16 Linux Professional Institute Certification

### QUESTION 70

Which of the following commands prints a list of usernames (first column) and their primary group (fourth column) from the /etc/passwd file?

- A. fmt -f 1,4 /etc/passwd
- B. split -c 1,4 /etc/passwd
- C. cut -d : -f 1,4 /etc/passwd
- D. paste -f 1,4 /etc/passwd

## 6.16 Linux Professional Institute Certification

### QUESTION 87

What does the command mount -a do?

- A. It ensures that all file systems listed with the option noauto in /etc/fstab are mounted.
- B. It shows all mounted file systems that have been automatically mounted.
- C. It opens an editor with root privileges and loads /etc/fstab for editing.
- D. It ensures that all file systems listed with the option auto in /etc/fstab are mounted.
- E. It ensures that all file systems listed in /etc/fstab are mounted regardless of their options.

### QUESTION 89

Which of the following is the device file name for the second partition on the only SCSI drive?

- A. /dev/hda1
- B. /dev/sda2
- C. /dev/sd0a2
- D. /dev/sd1p2

### QUESTION 90

In order to display all currently mounted filesystems, which of the following commands could be used? (Choose TWO correct answers.)

- A. cat /proc/self/mounts
- B. free
- C. mount
- D. lsmounts
- E. cat /proc/filesystems

#### QUESTION 92

Which of the following commands changes the number of days before the ext3 filesystem on /dev/sda1 has to run through a full filesystem check while booting?

- A. tune2fs -d 200 /dev/sda1
- B. tune2fs -c 200 /dev/sda1
- C. tune2fs -i 200 /dev/sda1
- D. tune2fs -n 200 /dev/sda1
- E. tune2fs --days 200 /dev/sda1

#### QUESTION 93

Which type of filesystem is created by mkfs when it is executed with the block device name only and without any additional parameters?

- A. ext2
- B. ext3
- C. ext4
- D. XFS
- E. VFAT

#### QUESTION 94

How many fields are in a syntactically correct line of /etc/fstab?

- A. 3
- B. 4
- C. 5
- D. 6
- E. 7

QUESTION 98) Which of the following commands creates an ext3 filesystem on /dev/sdb1? (Choose TWO correct answers.)

- A. /sbin/mke2fs -j /dev/sdb1
- B. /sbin/mkfs -t ext3 /dev/sdb1
- C. /sbin/mkfs -c ext3 /dev/sdb1
- D. /sbin/mke3fs -j /dev/sdb1

QUESTION 100) Which utility would be used to change how often a filesystem check is performed on an ext2 filesystem without losing any data stored on that filesystem?

- A. mod2fs

- B. fsck
- C. tune2fs
- D. mke2fs
- E. fixe2fs

QUESTION 101) Which of the following Linux filesystems preallocates a fixed number of inodes at the filesystem's make/creation time and does NOT generate them as needed? (Choose TWO correct answers.)

- A. ext3
- B. JFS
- C. ext2
- D. XFS
- E. procfs

QUESTION 102) What is the purpose of the Filesystem Hierarchy Standard?

- A. It is a security model used to ensure files are organized according to their permissions and accessibility.
- B. It provides unified tools to create, maintain and manage multiple filesystems in a common way.
- C. It defines a common internal structure of inodes for all compliant filesystems.
- D. It is a distribution neutral description of locations of files and directories.

## 7.16 Linux Professional Institute Certification

QUESTION 61

Regarding the command:

nice -5 /usr/bin/prog

Which of the following statements is correct?

- A. /usr/bin/prog is executed with a nice level of -5.
- B. /usr/bin/prog is executed with a nice level of 5.
- C. /usr/bin/prog is executed with a priority of -5.
- D. /usr/bin/prog is executed with a priority of 5.

QUESTION 62

Which shell command is used to continue background execution of a suspended command?

- A. &
- B. bg
- C. cont
- D. exec
- E. :&

QUESTION 66

When starting a program with the nice command without any additional parameters, which nice level is set for the resulting process?

- A. -10
- B. 0
- C. 10
- D. 20

#### QUESTION 67

Which of the following commands will reduce all consecutive spaces down to a single space?

- A. tr '\s' '' < a.txt > b.txt
- B. tr -c '' < a.txt > b.txt
- C. tr -d '' < a.txt > b.txt
- D. tr -r '' '\n' < a.txt > b.txt
- E. tr -s '' < a.txt > b.txt

#### QUESTION 68

Which character, added to the end of a command, runs that command in the background as a child process of the current shell?

- A. !
- B. +
- C. &
- D. %
- E. #

#### QUESTION 71

Which of the following signals is sent to a process when the key combination CTRL+C is pressed on the keyboard?

- A. SIGTERM
- B. SIGINT
- C. SIGSTOP
- D. SIGKILL

#### QUESTION 78

When running the command

```
sed -e "s/a/b/" /tmp/file >/tmp/file
```

While /tmp/file contains data, why is /tmp/file empty afterwards?

- A. The file order is incorrect. The destination file must be mentioned before the command to ensure redirection.
- B. The command sed did not match anything in that file therefore the output is empty.
- C. When the shell establishes the redirection it overwrites the target file before the redirected command starts and opens it for reading.
- D. Redirection for shell commands do not work using the > character. It only works using the | character instead.

**QUESTION 79**

When given the following command line. echo "foo bar" | tee bar | cat

Which of the following output is created?

- A. cat
- B. foo bar
- C. tee bar
- D. bar
- E. foo

**QUESTION 80**

Which of the following commands can be used to determine how long the system has been running? (Choose TWO correct answers.)

- A. uptime
- B. up
- C. top
- D. uname -u
- E. time up

## 7.16 Linux Professional Institute Certification

**QUESTION 61**

Regarding the command:

nice -5 /usr/bin/prog

Which of the following statements is correct?

- A. /usr/bin/prog is executed with a nice level of -5.
- B. /usr/bin/prog is executed with a nice level of 5.
- C. /usr/bin/prog is executed with a priority of -5.
- D. /usr/bin/prog is executed with a priority of 5.

**QUESTION 62**

Which shell command is used to continue background execution of a suspended command?

- A. &
- B. bg
- C. cont
- D. exec
- E. :&

**QUESTION 66**

When starting a program with the nice command without any additional parameters, which nice level is set for the resulting process?

- A. -10
- B. 0
- C. 10
- D. 20

#### QUESTION 67

Which of the following commands will reduce all consecutive spaces down to a single space?

- A. tr '\s' '' < a.txt > b.txt
- B. tr -c '' < a.txt > b.txt
- C. tr -d '' < a.txt > b.txt
- D. tr -r '' '\n' < a.txt > b.txt
- E. tr -s '' < a.txt > b.txt

#### QUESTION 68

Which character, added to the end of a command, runs that command in the background as a child process of the current shell?

- A. !
- B. +
- C. &
- D. %
- E. #

#### QUESTION 71

Which of the following signals is sent to a process when the key combination CTRL+C is pressed on the keyboard?

- A. SIGTERM
- B. SIGINT
- C. SIGSTOP
- D. SIGKILL

#### QUESTION 78

When running the command

```
sed -e "s/a/b/" /tmp/file >/tmp/file
```

While /tmp/file contains data, why is /tmp/file empty afterwards?

- A. The file order is incorrect. The destination file must be mentioned before the command to ensure redirection.
- B. The command sed did not match anything in that file therefore the output is empty.
- C. When the shell establishes the redirection it overwrites the target file before the redirected command starts and opens it for reading.
- D. Redirection for shell commands do not work using the > character. It only works using the | character instead.

#### QUESTION 79

When given the following command line. echo "foo bar" | tee bar | cat  
Which of the following output is created?

- A. cat
- B. foo bar
- C. tee bar
- D. bar
- E. foo

#### QUESTION 80

Which of the following commands can be used to determine how long the system has been running? (Choose TWO correct answers.)

- A. uptime
- B. up
- C. top
- D. uname -u
- E. time up

## 10.7 Linux Professional Institute Certification

#### QUESTION 60

From a Bash shell, which of the following commands directly executes the instruction from the file /usr/local/bin/runme.sh without starting a subshell? (Please select TWO answers.)

- A. source /usr/local/bin/runme.sh
- B. . /usr/local/bin/runme.sh
- C. /bin/bash /usr/local/bin/runme.sh
- D. /usr/local/bin/runme.sh
- E. run /usr/local/bin/runme.sh

#### QUESTION 73

Which of the following command sets the Bash variable named TEST with the content FOO?

- A. set TEST="FOO"
- B. TEST = "FOO"
- C. var TEST="FOO"
- D. TEST="FOO"

#### QUESTION 74

Which variable defines the directories in which a Bash shell searches for executable commands?

- A. BASHEXEC
- B. BASHRC
- C. PATH
- D. EXECPATH

## E. PATHRC

