

Breno Farias da Silva, RA: 2300516,
Heloísa Abrantes Roncato, RA: 2300524,
João Henrique Gouveia, RA: 2252759,
Matheus Santos de Andrade, RA: 2304570.

Laboratório 1 de SO Compilação do Kernel Linux

Relatório técnico de atividade prática solicitado pelo professor Rodrigo Campiolo na disciplina de Sistemas Operacionais do Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR

Departamento Acadêmico de Computação – DACOM

Bacharelado em Ciência da Computação – BCC

Campo Mourão

Março / 2021

Resumo

Este relatório apresentará o passo a passo seguido para a realização do Laboratório 1 de SO: Compilação do Kernel linux, utilizando imagens e descrições textuais, bem como a apresentação de problemas encontrados ao longo do processo e suas soluções.

Palavras-chave: Kernel. Compilação. Virtualização.

Lista de ilustrações

Figura 1 – Website VirtualBox.	5
Figura 2 – Interface VirtualBox	6
Figura 3 – Configuração inicial da máquina	6
Figura 4 – Alocação de memória	7
Figura 5 – Alocação de disco para a maquina	7
Figura 6 – Particionamento do disco para o SO	8
Figura 7 – Execução do ps aux	9
Figura 8 – Execução do df / df -h	9
Figura 9 – Execução dos comandos free -b e cat /proc/meminfo	10
Figura 10 – Execução dos comandos IP address show, IP route, cat /etc/resolv.conf e cat /etc/network/interfaces	10
Figura 11 – Execução do comando ping google.com	11
Figura 12 – Visualização dos repositórios e execução do apt update	11
Figura 13 – Visualização da versão do Kernel	12
Figura 14 – Terminal em modo SU	13
Figura 15 – Instalação dos pacotes básicos	13
Figura 16 – Download versão estável do Kernel linux	14
Figura 17 – Descompactação do kernel linux no diretório src	15
Figura 18 – Diretório da descompactação	15
Figura 19 – Instalação do pacote Flex	16
Figura 20 – Backup configurações atuais	16
Figura 21 – Salvando configurações atuais a partir do comando make menuConfig	16
Figura 22 – Execução do comando make -j4	17
Figura 23 – Execução do comando make modules_install	17
Figura 24 – Instalação do pacote do DWARF	18
Figura 25 – Execução do comando make install	18
Figura 26 – Visualização dos arquivos após compilação	19
Figura 27 – Visualização dos módulos compilados	19

Sumário

1	Introdução	5
2	Objetivos	5
3	Materiais	5
4	Procedimentos e Resultados	5
	4.1 Download e Setup da máquina virtual	5
	4.2 Download, compilação e instalação do Kernel e suas dependências .	13
5	Conclusões	19
6	Referências	20

1 Introdução

Utilizaremos um software para a virtualização de uma distribuição GNU Debian e passaremos por todo o passo a passo de instalação da máquina virtual, bem como a compilação e instalação de um kernel linux dentro dela.

2 Objetivos

Instalar um sistema operacional (Linux), compreender a estrutura básica do Linux e alguns comandos, bem como configurar e compilar o núcleo (kernel) Linux em sua última versão estável

3 Materiais

- Computador com ao menos 10GB de espaço livre.
- Software de virtualização como o Oracle VirtualBox.
- Distribuição estável recente que utilize o gerenciador de pacotes APT como o Debian 11.2.
- Kernel linux estável recente como o 5.16.14

4 Procedimentos e Resultados

4.1 Download e Setup da máquina virtual

Primeiramente, precisamos instalar um software de virtualização, nesse caso, utilizaremos o VirtualBox, que pode ser baixado através do link para diferentes distribuições e SO's.



Figura 1 – Website VirtualBox.

Precisaremos também baixar uma distribuição de SO para rodar na máquina virtual, nesse caso, utilizaremos o Debian. Após o fim do download, iniciaremos a configuração da máquina virtual. Basta abrir o software VirtualBox e adicionar uma nova máquina.

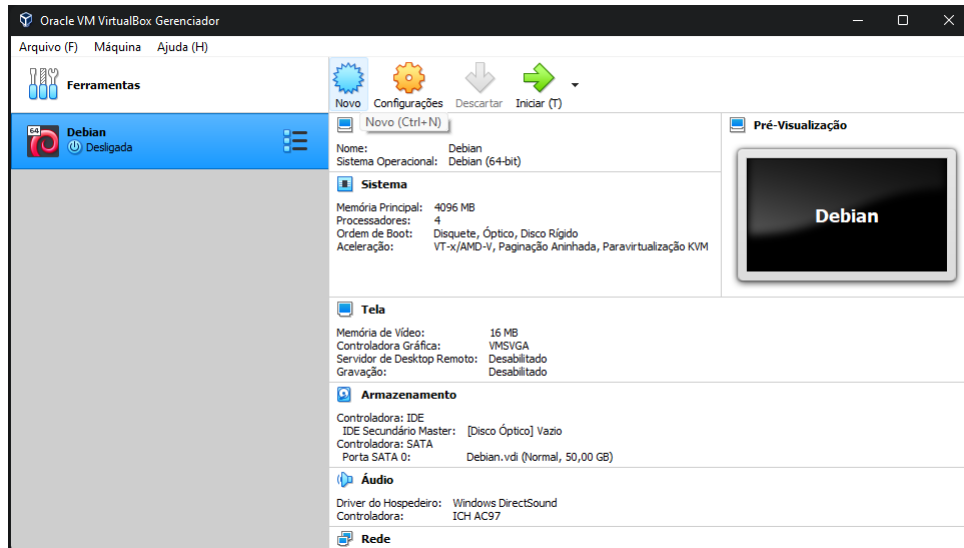


Figura 2 – Interface VirtualBox

Em seguida, selecionamos a categoria de SO (Linux), a distribuição (Debian) e a arquitetura (64 bits), bem como o local de armazenamento da máquina.

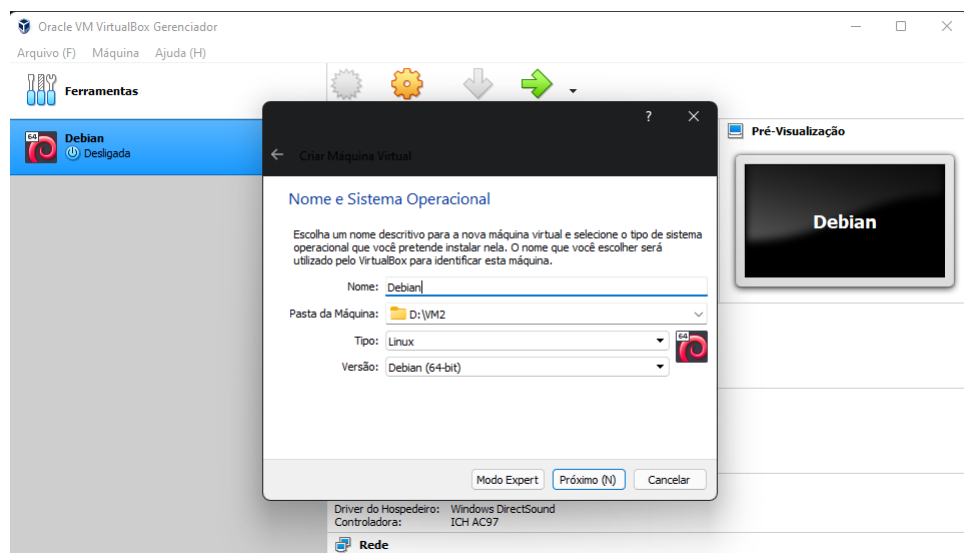


Figura 3 – Configuração inicial da máquina

Definimos a quantidade de memória RAM e o espaço de disco a serem alocados.

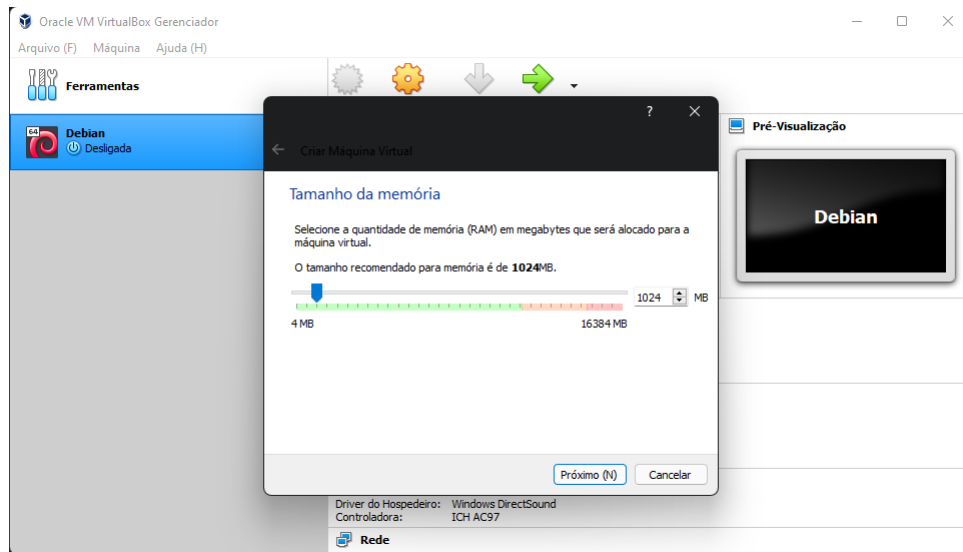


Figura 4 – Alocação de memória

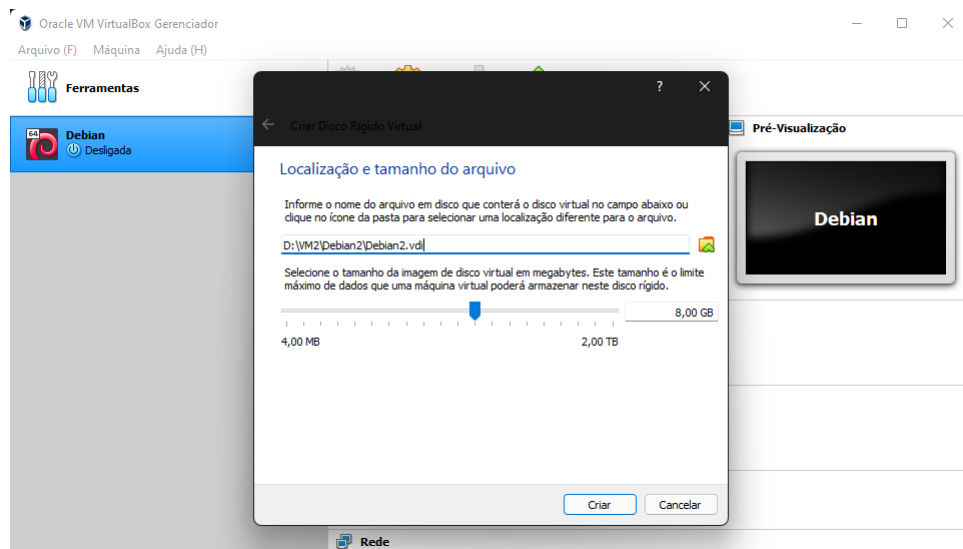


Figura 5 – Alocação de disco para a maquina

Após o término da configuração, basta selecionar a imagem do sistema baixado, nesse caso, Debian. (Cerca de 300MB). Ao iniciar a máquina virtual será necessário realizar a instalação do sistema operacional, sendo possível através de de uma GUI.

Preenche-se o nome do host e senha desejados, a alocação de partições apropriada. Nós criamos manualmente as partições root, home, var, usr no formato ext4, e por fim swap.

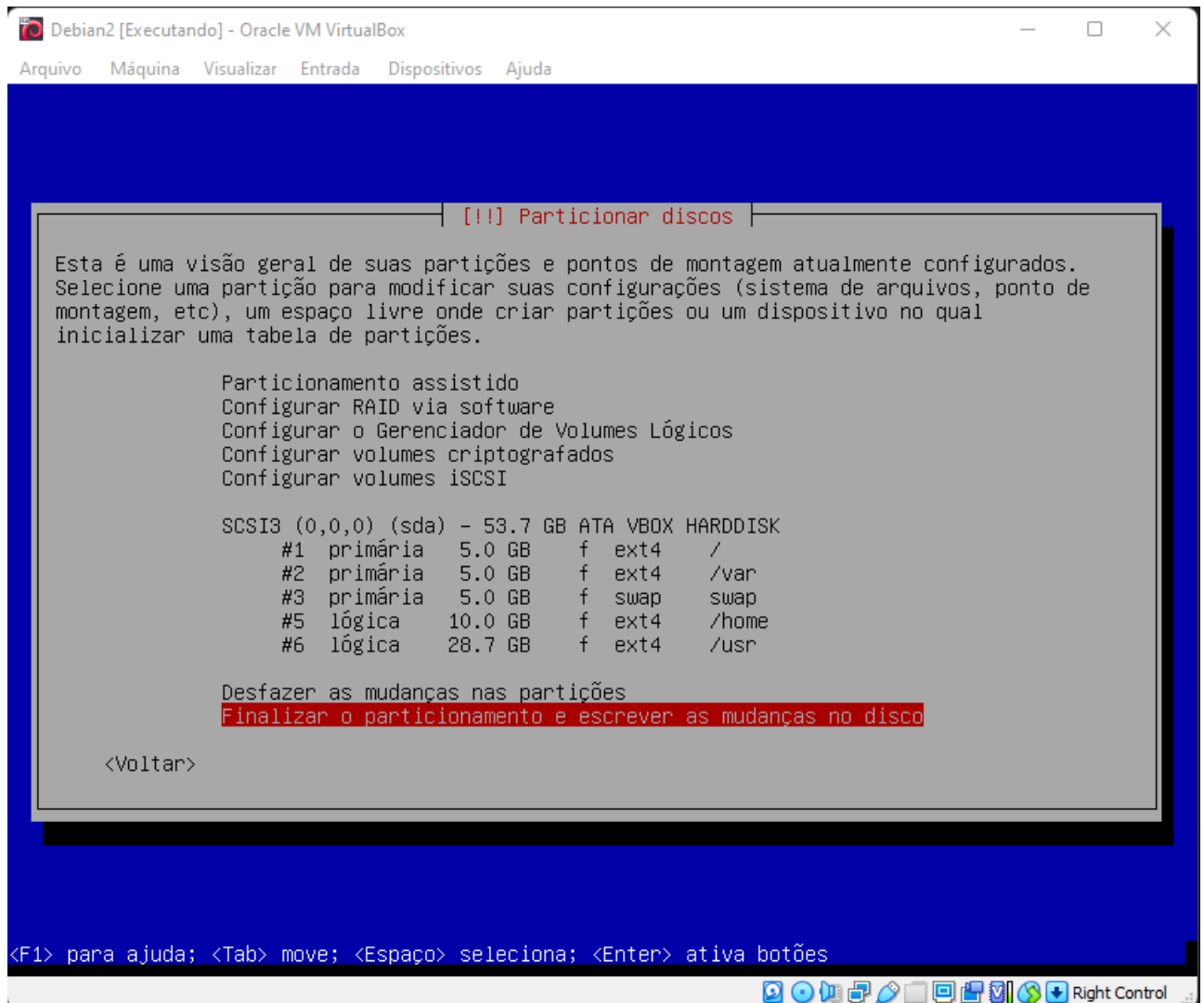


Figura 6 – Particionamento do disco para o SO

Selecionamos a opção `deb.debian.org` como gerenciador de pacotes, instalamos o GRUB na partição principal para tornar o SO inicializável e escolhemos a interface gráfica GNOME.

Ao finalizar, reiniciamos a máquina, e executamos o comando `ps aux`.


```

matheus@matheus:~$ ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.4  0.2 163968 10276 ?        Ss   09:34   0:01 /sbin/init
root           2  0.0  0.0      0   0 ?        S    09:34   0:00 [kthreadd]
root           3  0.0  0.0      0   0 ?        I<   09:34   0:00 [rcu_gp]
root           4  0.0  0.0      0   0 ?        I<   09:34   0:00 [rcu_par_gp]
root           5  0.0  0.0      0   0 ?        I    09:34   0:00 [kworker/8:0-
root           6  0.0  0.0      0   0 ?        I<   09:34   0:00 [kworker/8:0H
root           7  0.0  0.0      0   0 ?        I    09:34   0:00 [kworker/8:1-
root           8  0.0  0.0      0   0 ?        R    09:34   0:00 [kworker/u8:0
root           9  0.0  0.0      0   0 ?        I<   09:34   0:00 [mm_percpu_wq
root          10  0.0  0.0      0   0 ?        S    09:34   0:00 [rcu_tasks_ru
root          11  0.0  0.0      0   0 ?        S    09:34   0:00 [rcu_tasks_tr
root          12  0.0  0.0      0   0 ?        S    09:34   0:00 [ksoftirqd/6]
root          13  0.0  0.0      0   0 ?        I    09:34   0:00 [rcu_sched]
root          14  0.0  0.0      0   0 ?        S    09:34   0:00 [migration/0]
root          15  0.0  0.0      0   0 ?        S    09:34   0:00 [cpuhp/0]
root          16  0.0  0.0      0   0 ?        S    09:34   0:00 [cpuhp/1]
root          17  0.1  0.0      0   0 ?        S    09:34   0:00 [migration/1]
root          18  0.0  0.0      0   0 ?        I    09:34   0:00 [ksoftirqd/1]
root          19  0.0  0.0      0   0 ?        I<   09:34   0:00 [kworker/1:0-
root          20  0.0  0.0      0   0 ?        I<   09:34   0:00 [kworker/1:0H
root          21  0.0  0.0      0   0 ?        S    09:34   0:00 [cpuhp/2]
root          22  0.1  0.0      0   0 ?        S    09:34   0:00 [migration/2]
root          23  0.0  0.0      0   0 ?        S    09:34   0:00 [ksoftirqd/2]
root          24  0.0  0.0      0   0 ?        I    09:34   0:00 [kworker/2:0-
root          25  0.0  0.0      0   0 ?        I<   09:34   0:00 [kworker/2:0H
root          26  0.0  0.0      0   0 ?        S    09:34   0:00 [cpuhp/3]
root          27  0.1  0.0      0   0 ?        S    09:34   0:00 [migration/3]
root          28  0.0  0.0      0   0 ?        S    09:34   0:00 [ksoftirqd/3]
root          30  0.0  0.0      0   0 ?        I<   09:34   0:00 [kworker/3:0H
root          32  0.0  0.0      0   0 ?        I    09:34   0:00 [kworker/u8:1
root          33  0.0  0.0      0   0 ?        I    09:34   0:00 [kworker/u8:2
root          34  0.0  0.0      0   0 ?        S    09:34   0:00 [kdevtmpfs]
root          35  0.0  0.0      0   0 ?        I<   09:34   0:00 [netns]
root          36  0.0  0.0      0   0 ?        S    09:34   0:00 [kauditd]
root          37  0.0  0.0      0   0 ?        S    09:34   0:00 [khungtaskd]
root          38  0.0  0.0      0   0 ?        S    09:34   0:00 [oom_reaper]
root          39  0.0  0.0      0   0 ?        I<   09:34   0:00 [writeback]
root          40  0.0  0.0      0   0 ?        S    09:34   0:00 [kcompactd0]
root          41  0.0  0.0      0   0 ?        SN   09:34   0:00 [ksmd]
root          42  0.0  0.0      0   0 ?        SN   09:34   0:00 [khugepaged]
root          44  0.0  0.0      0   0 ?        I    09:34   0:00 [kworker/3:1-
root          61  0.0  0.0      0   0 ?        I<   09:34   0:00 [kintegrityd]
root          62  0.0  0.0      0   0 ?        I<   09:34   0:00 [blkcg]
root          63  0.0  0.0      0   0 ?        I<   09:34   0:00 [blkcg_punt_b
root          64  0.0  0.0      0   0 ?        I<   09:34   0:00 [edac-poller]
root          65  0.0  0.0      0   0 ?        I<   09:34   0:00 [devfreq_wq]
root          66  0.0  0.0      0   0 ?        I    09:34   0:00 [kworker/2:1-
root          67  0.0  0.0      0   0 ?        I<   09:34   0:00 [kworker/u8:1H
root          68  0.0  0.0      0   0 ?        S    09:34   0:00 [kswapd0]
root          69  0.0  0.0      0   0 ?        S    09:34   0:00 [kthreadd1]

```

Figura 7 – Execução do ps aux

Executando os comandos `df` e `df -h` podemos verificar os tamanhos das partições criadas, de forma que pode ser visualizado uma alocação de espaço exacerbado à partição `usr`, isso se deve ao fato de que iremos realizar a compilação do kernel na partição `usr`, enquanto as outras servirão para o funcionamento básico de nossa máquina.

```

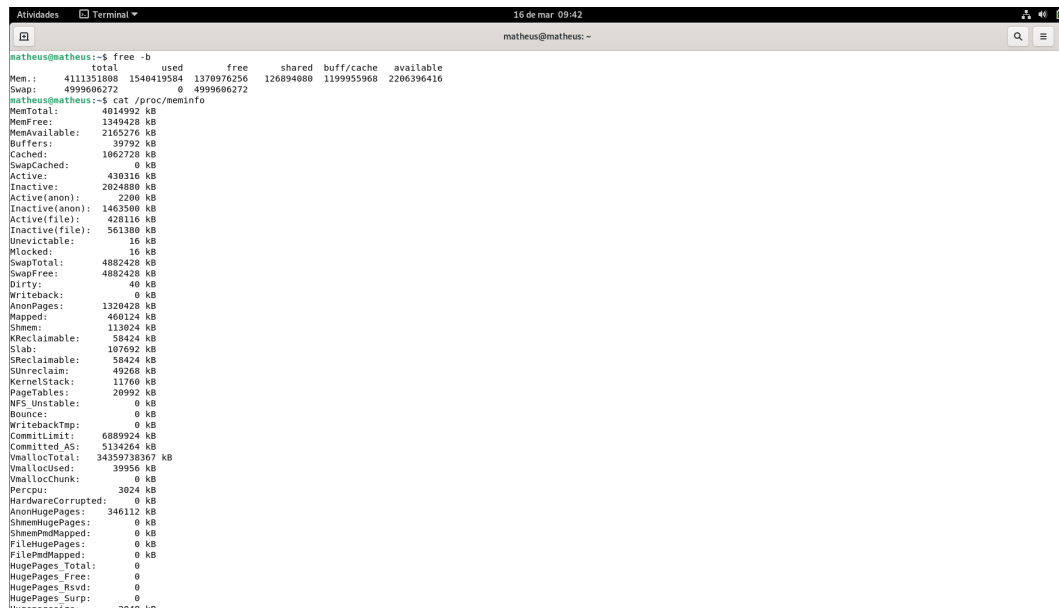
matheus@matheus:~$ df
Sist. Arq.  Blocos de 1K  Usado Disponível Uso% Montado em
udev          1984532      0  1984532    0% /dev
tmpfs         401500    1164   400336    1% /run
/dev/sda1    4720160  117724   4341964    3% /
/dev/sda6   27397236 4006660  21973516   16% /usr
tmpfs        2007496      0   2007496    0% /dev/shm
tmpfs         5120      4     5116    1% /run/lock
/dev/sda2   4721184  381740   4078940    9% /var
/dev/sda5   9509056 109120   8895312    2% /home
tmpfs        401496    9564   391932    3% /run/user/1000

matheus@matheus:~$ df -h
Sist. Arq.  Tam. Usado Disp. Uso% Montado em
udev        1,9G      0  1,9G    0% /dev
tmpfs       393M    1,2M  391M    1% /run
/dev/sda1    4,6G  115M  4,2G    3% /
/dev/sda6    27G   3,9G  21G   16% /usr
tmpfs        2,0G      0  2,0G    0% /dev/shm
tmpfs        5,0M   4,0K  5,0M    1% /run/lock
/dev/sda2    4,6G  373M  3,9G    9% /var
/dev/sda5    9,1G  107M  8,5G    2% /home
tmpfs        393M   9,4M  383M    3% /run/user/1000

```

Figura 8 – Execução do df / df -h

Podemos também verificar a memória alocada para nossa máquina utilizando os comandos `free -b` e `cat /proc/meminfo`.



```

matheus@matheus:~$ free -b
              total        used        free      shared  buff/cache   available
Mem:      4111351888 1540419584 1378976256 126894880 1199555968 2286396416
Swap:      4999606272          0 4999606272

matheus@matheus:~$ cat /proc/meminfo
MemTotal:        4014992 kB
MemFree:         1349428 kB
MemAvailable:    2165276 kB
Buffers:         39792 kB
Cached:          1862728 kB
SwapCached:      0 kB
Active:          430316 kB
Inactive:        2024880 kB
Active(anon):    2280 kB
Inactive(anon):  1463500 kB
Active(file):    420116 kB
Inactive(file):  561380 kB
Unevictable:     16 kB
Mlocked:         16 kB
SwapTotal:       4882428 kB
SwapFree:        4882428 kB
Dirty:           40 kB
Writeback:       0 kB
AnonPages:       1320428 kB
Mapped:          460124 kB
Shmem:           113024 kB
SReclaimable:    58424 kB
Slab:            107692 kB
SReclaimable:    56424 kB
SUnreclaim:      49268 kB
KernelStack:     11760 kB
PageTables:      20992 kB
NFS_Unstable:    0 kB
Bounce:          0 kB
WritebackTmp:    0 kB
CommitLimit:     6889924 kB
Committed_AS:    5134264 kB
VmallocTotal:    34359738367 kB
VmallocUsed:      39956 kB
VmallocChunk:    0 kB
Percpu:          3024 kB
HardwareCorrupted: 0 kB
AnonHugePages:   346112 kB
ShmemHugePages:  0 kB
ShmemPmdMapped:  0 kB
FileHugePages:   0 kB
FilePmdMapped:   0 kB
HugePages_Total: 0
HugePages_Free:  0
HugePages_Rsvd:  0
HugePages_Surp:  0
MemswTotal:      7648 kB

```

Figura 9 – Execução dos comandos `free -b` e `cat /proc/meminfo`

Para confirmar a conexão bem sucedida com a internet e suas respectivas configurações, basta executar os comandos:

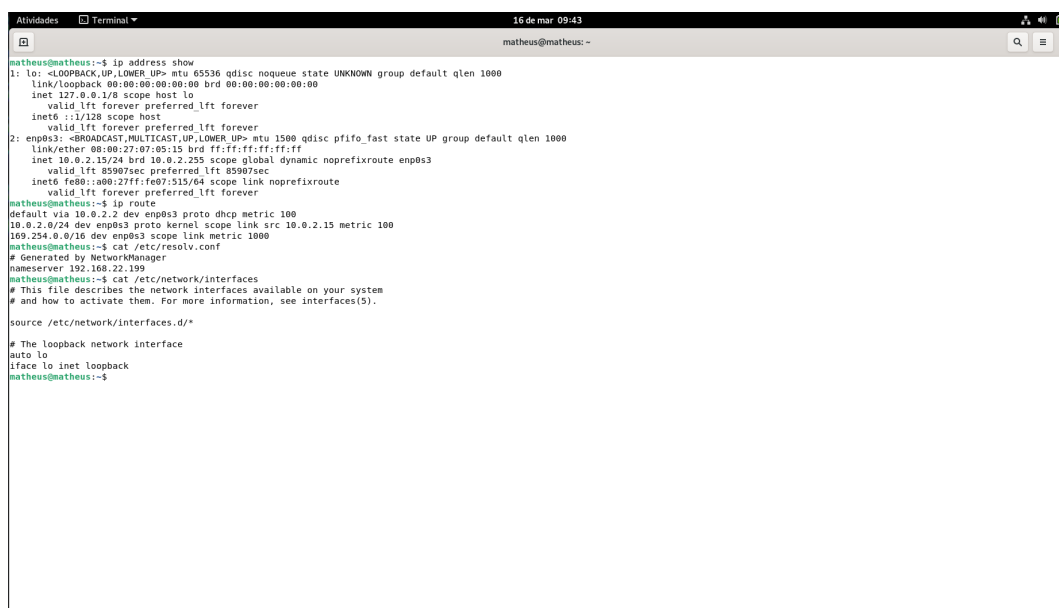
`ip address show` (exibe as interfaces de rede);

`ip route` (exibe a tabela de roteamento);

`cat /etc/resolv.conf` (configuração do dns);

`cat /etc/network/interfaces` (configuração das interfaces de rede);

`ping www.google.com.br` (conectividade).



```

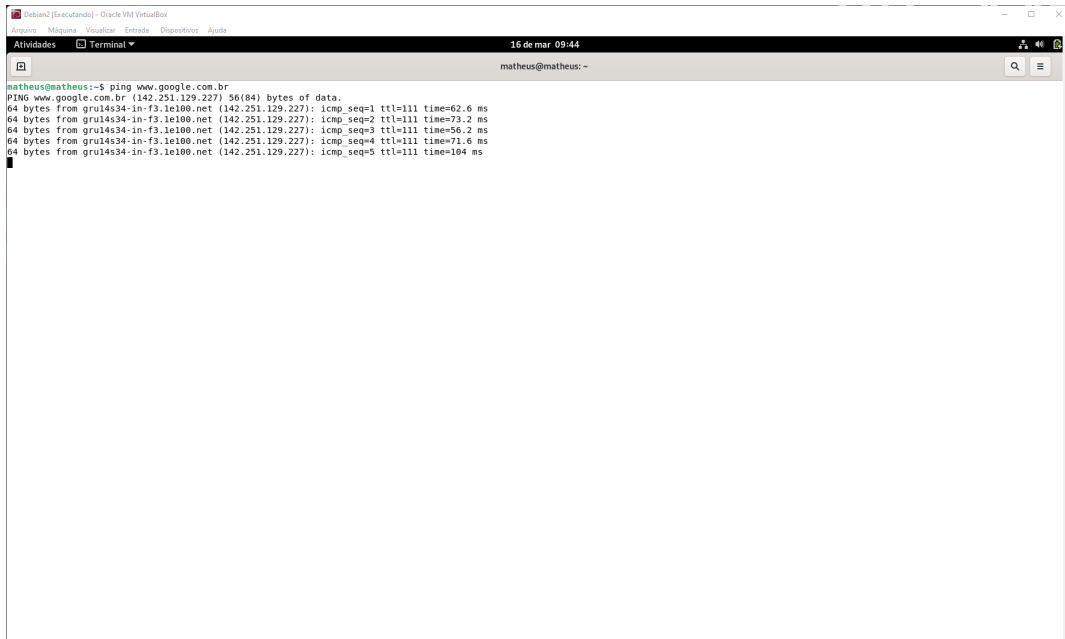
matheus@matheus:~$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid lft forever preferred lft forever
    inet6 ::1/128 scope host
        valid lft forever preferred lft forever
2: enp8s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:07:05:15 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp8s3
        valid lft 85907sec preferred lft 85907sec
    inet6 fe80::a00:27ff:fe07:515/64 scope link noprefixroute
        valid lft forever preferred lft forever
matheus@matheus:~$ ip route
default via 10.0.2.2 dev enp8s3 proto dhcp metric 100
10.0.2.0/24 dev enp8s3 proto kernel scope link src 10.0.2.15 metric 100
169.254.0.0/16 dev enp8s3 scope link metric 1000
matheus@matheus:~$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 192.168.22.199
matheus@matheus:~$ cat /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
matheus@matheus:~$

```

Figura 10 – Execução dos comandos `IP address show`, `IP route`, `cat /etc/resolv.conf` e `cat /etc/network/interfaces`



```
matheus@matheus:~$ ping www.google.com.br
PING www.google.com.br (142.251.129.227): 56(84) bytes of data:
64 bytes from gru14s34-in-f3.1e100.net (142.251.129.227): icmp_seq=1 ttl=111 time=62.6 ms
64 bytes from gru14s34-in-f3.1e100.net (142.251.129.227): icmp_seq=2 ttl=111 time=73.2 ms
64 bytes from gru14s34-in-f3.1e100.net (142.251.129.227): icmp_seq=3 ttl=111 time=56.2 ms
64 bytes from gru14s34-in-f3.1e100.net (142.251.129.227): icmp_seq=4 ttl=111 time=71.6 ms
64 bytes from gru14s34-in-f3.1e100.net (142.251.129.227): icmp_seq=5 ttl=111 time=104 ms
```

Figura 11 – Execução do comando ping google.com

O próximo passo é verificar e adicionar os repositórios de gerenciamento de pacotes de atualização e segurança, bem como garantir que o sistema está atualizado.

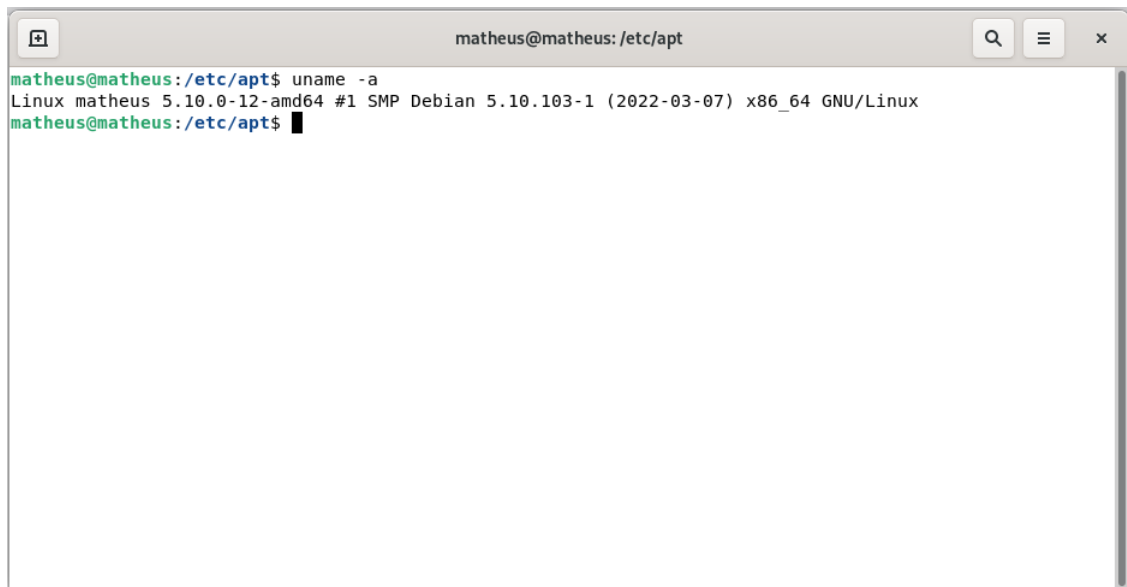
```
cd /etc/apt
nano sources.list
apt-get update
```



```
matheus@matheus:~$ cd /etc/apt
matheus@matheus:/etc/apt$ nano sources.list
matheus@matheus:/etc/apt$ apt-get update
Lendo listas de pacotes... Pronto
E: Não foi possível abrir arquivo de trava /var/lib/apt/lists/lock - open (13: Permissão negada)
E: Impossível criar acesso exclusivo ao directório /var/lib/apt/lists/
W: Problema ao remover o link do ficheiro /var/cache/apt/pkgcache.bin - RemoveCaches (13: Permissão negada)
W: Problema ao remover o link do ficheiro /var/cache/apt/srcpkgcache.bin - RemoveCaches (13: Permissão negada)
matheus@matheus:/etc/apt$
```

Figura 12 – Visualização dos repositórios e execução do apt update

Podemos também verificar a versão do kernel atual utilizando o comando `uname -a`.

A terminal window titled 'matheus@matheus: /etc/apt' with search, menu, and close buttons. The prompt is 'matheus@matheus: /etc/apt\$'. The command 'uname -a' has been entered and executed, resulting in the output: 'Linux matheus 5.10.0-12-amd64 #1 SMP Debian 5.10.103-1 (2022-03-07) x86_64 GNU/Linux'. The prompt is now 'matheus@matheus: /etc/apt\$' with a cursor.

```
matheus@matheus: /etc/apt$ uname -a
Linux matheus 5.10.0-12-amd64 #1 SMP Debian 5.10.103-1 (2022-03-07) x86_64 GNU/Linux
matheus@matheus: /etc/apt$
```

Figura 13 – Visualização da versão do Kernel

4.2 Download, compilação e instalação do Kernel e suas dependências

Inicialmente, vamos abrir o terminal em modo super usuário, usando o comando `su -`. A partir disso, não será sempre necessário adicionar o comando `sudo` para executar cada comando, todavia, para fins didáticos, deixamos os comandos com `sudo` antes.

```
matheus@matheus:/etc/apt$ sudo su
Presumimos que você recebeu as instruções de sempre do administrador
de sistema local. Basicamente, resume-se a estas três coisas:

#1) Respeite a privacidade dos outros.
#2) Pense antes de digitar.
#3) Com grandes poderes vêm grandes responsabilidades.

[sudo] senha para matheus:
matheus não está no arquivo sudoers. Este incidente será relatado.
matheus@matheus:/etc/apt$ su -
Senha:
root@matheus:~# cd /root
root@matheus:~#
```

Figura 14 – Terminal em modo SU

Vamos instalar os pacotes básicos necessários para a instalação do kernel linux usando o comando:

```
sudo apt-get install build-essential libncurses5-dev xz-utils libssl-dev bc bison 1
```

```
root@matheus:~# apt-get install build-essential libncurses5-dev xz-utils libssl-dev bc bison libelf-dev libncurses5-dev
Lendo listas de pacotes... Pronto
Construindo árvore de dependências... Pronto
Lendo informação de estado... Pronto
xz-utils is already the newest version (5.2.5-2).
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu dpkg-dev fakeroot g++ g++-10 gcc gcc-10
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libatomic1
  libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0
  libfakeroot libgcc-10-dev libitm1 liblsan0 libncurses-dev libnsl-dev libsigsegv2 libstdc++-10-dev
  libtirpc-dev libtsan0 libubsan1 linux-libc-dev m4 make manpages-dev patch zlib1g-dev
Pacotes sugeridos:
  binutils-doc bison-doc debian-keyring g++-multilib g++-10-multilib gcc-10-doc gcc-multilib
  autoconf automake libtool flex gdb gcc-doc gcc-10-multilib gcc-10-locales glibc-doc ncurses-doc
  libssl-doc libstdc++-10-doc m4-doc make-doc ed diffutils-doc
Os NOVOS pacotes a seguir serão instalados:
  bc binutils binutils-common binutils-x86-64-linux-gnu bison build-essential dpkg-dev fakeroot g++
  g++-10 gcc gcc-10 libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libasan6 libatomic1 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev
  libctf-nobfd0 libctf0 libelf-dev libfakeroot libgcc-10-dev libitm1 liblsan0 libncurses-dev
  libncurses5-dev libnsl-dev libsigsegv2 libssl-dev libstdc++-10-dev libtirpc-dev libtsan0 libubsan1
  linux-libc-dev m4 make manpages-dev patch zlib1g-dev
0 pacotes atualizados, 45 pacotes novos instalados, 0 a serem removidos e 0 não atualizados.
É preciso baixar 54,8 MB de arquivos.
Depois desta operação, 212 MB adicionais de espaço em disco serão usados.
Você quer continuar? [S/n] s
Obter:1 http://security.debian.org/debian-security bullseye-security/main amd64 linux-libc-dev amd64 5.10.103-1 [1.466 kB]
Obter:2 http://deb.debian.org/debian bullseye/main amd64 bc amd64 1.07.1-2+b2 [109 kB]
Obter:3 http://deb.debian.org/debian bullseye/main amd64 binutils-common amd64 2.35.2-2 [2.220 kB]
Obter:4 http://security.debian.org/debian-security bullseye-security/main amd64 libssl-dev amd64 1.1.1k-1+deb11u2 [1.810 kB]
Obter:5 http://deb.debian.org/debian bullseye/main amd64 libbinutils amd64 2.35.2-2 [570 kB]
Obter:6 http://deb.debian.org/debian bullseye/main amd64 libctf-nobfd0 amd64 2.35.2-2 [110 kB]
Obter:7 http://deb.debian.org/debian bullseye/main amd64 libctf0 amd64 2.35.2-2 [53,2 kB]
Obter:8 http://deb.debian.org/debian bullseye/main amd64 binutils-x86-64-linux-gnu amd64 2.35.2-2 [1.809 kB]
Obter:9 http://deb.debian.org/debian bullseye/main amd64 binutils amd64 2.35.2-2 [61,2 kB]
Obter:10 http://deb.debian.org/debian bullseye/main amd64 libsigsegv2 amd64 2.13-1 [34,8 kB]
Obter:11 http://deb.debian.org/debian bullseye/main amd64 m4 amd64 1.4.18-5 [204 kB]
```

Figura 15 – Instalação dos pacotes básicos

Logo após, fizemos o download da versão estável listada do kernel linux usando o seguinte comando:

```
wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.16.14.tar.xz
```

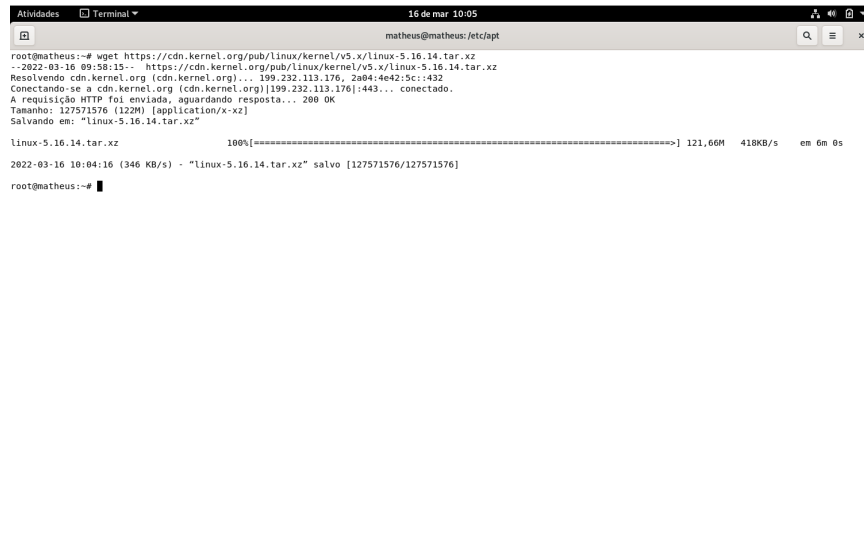


Figura 16 – Download versão estável do Kernel linux

A descompactação do arquivo será feita no diretório source (“src”), o qual está contido no seguinte path:

```
cd /usr/src/
```

Sendo assim, foi executado o seguinte comando:

```
cd /usr/src/linux-5.16.14
```

Com o âmbito de fazer a descompactação do arquivo baixado pelo comando do `wget`.

```
tar xvf linux-5.16.14.tar.xz -C /usr/src
```

```

linux-5.16.14/tools/virtio/vringh_test.c
linux-5.16.14/tools/virtio/xen/
linux-5.16.14/tools/virtio/xen/xen.h
linux-5.16.14/tools/vm/
linux-5.16.14/tools/vm/.gitignore
linux-5.16.14/tools/vm/Makefile
linux-5.16.14/tools/vm/page-types.c
linux-5.16.14/tools/vm/page_owner_sort.c
linux-5.16.14/tools/vm/slabinf-gnuplot.sh
linux-5.16.14/tools/vm/slabinf.c
linux-5.16.14/tools/vmi/
linux-5.16.14/tools/vmi/Makefile
linux-5.16.14/tools/vmi/dell-smbios-example.c
linux-5.16.14/usr/
linux-5.16.14/usr/.gitignore
linux-5.16.14/usr/Kconfig
linux-5.16.14/usr/Makefile
linux-5.16.14/usr/default_cpio_list
linux-5.16.14/usr/gen_init_cpio.c
linux-5.16.14/usr/gen_initramfs.sh
linux-5.16.14/usr/include/
linux-5.16.14/usr/include/.gitignore
linux-5.16.14/usr/include/Makefile
linux-5.16.14/usr/initramfs_data.S
linux-5.16.14/virt/
linux-5.16.14/virt/Makefile
linux-5.16.14/virt/kvm/
linux-5.16.14/virt/kvm/Kconfig
linux-5.16.14/virt/kvm/async_pf.c
linux-5.16.14/virt/kvm/async_pf.h
linux-5.16.14/virt/kvm/binary_stats.c
linux-5.16.14/virt/kvm/coalesced_mmio.c
linux-5.16.14/virt/kvm/coalesced_mmio.h
linux-5.16.14/virt/kvm/dirty_ring.c
linux-5.16.14/virt/kvm/eventfd.c
linux-5.16.14/virt/kvm/irqchip.c
linux-5.16.14/virt/kvm/kvm_main.c
linux-5.16.14/virt/kvm/mmu_lock.h
linux-5.16.14/virt/kvm/vfio.c
linux-5.16.14/virt/kvm/vfio.h
linux-5.16.14/virt/lib/
linux-5.16.14/virt/lib/Kconfig
linux-5.16.14/virt/lib/Makefile
linux-5.16.14/virt/lib/irqbypass.c
root@matheus:~# █

```

Figura 17 – Descompactação do kernel linux no diretório src

Após isso, usando o seguinte comando foi possível ir até o diretório da descompactação:

```
cd /usr /src/linux-5.16.14
```

```

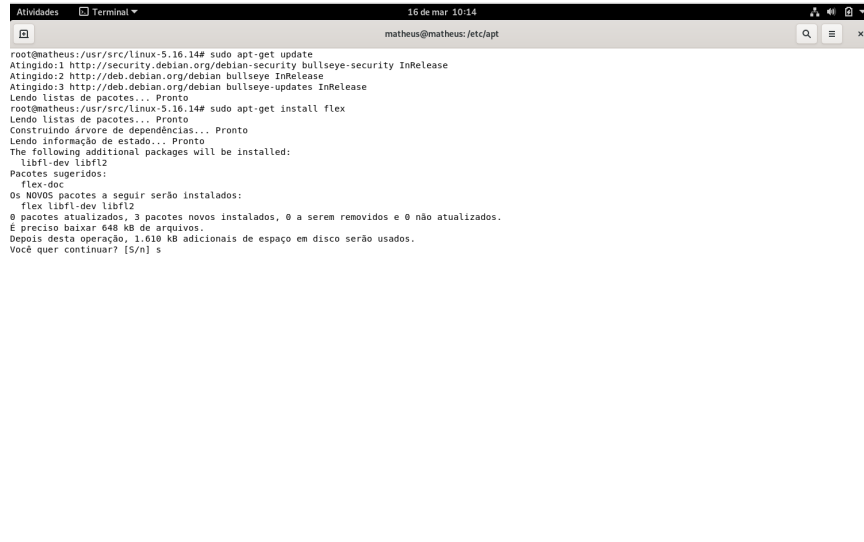
root@matheus:~# cd /usr/src/linux-5.16.14
root@matheus:/usr/src/linux-5.16.14# █

```

Figura 18 – Diretório da descompactação

A seguir, será feita uma cópia do arquivo de configuração atual, todavia haverá um problema com dependência dos pacotes básicos. O erro se dá para falta do pacote flex, o qual está relacionado com um analisador léxico do linux. A palavra flex significa fast lexical analyzer generator.

```
sudo apt-get install flex
```



```

root@matheus:/usr/src/linux-5.16.14# sudo apt-get update
Atingido:1 http://security.debian.org/debian-security bullseye-security InRelease
Atingido:3 http://deb.debian.org/debian bullseye-updates InRelease
Lendo listas de pacotes... Pronto
root@matheus:/usr/src/linux-5.16.14# sudo apt-get install flex
Lendo listas de pacotes... Pronto
Construindo árvore de dependências... Pronto
Lendo informação de estado... Pronto
The following additional packages will be installed:
  libfl-dev libfl2
Pacotes sugeridos:
  flex-doc
Os NOVOS pacotes a seguir serão instalados:
  flex libfl-dev libfl2
0 pacotes atualizados, 3 pacotes novos instalados, 0 a serem removidos e 0 não atualizados.
É preciso baixar 648 kB de arquivos.
Depois desta operação, 1.610 kB adicionais de espaço em disco serão usados.
Você quer continuar? [S/n] s

```

Figura 19 – Instalação do pacote Flex

Por precaução, fizemos uma cópia da configuração atual usando o seguinte comando:

```
cp -v /boot/config-$(uname -r) .config
```

```

root@matheus:/usr/src/linux-5.16.14# cp -v /boot/config-$(uname -r) .config
'/boot/config-5.10.0-12-amd64' -> '.config'
root@matheus:/usr/src/linux-5.16.14#

```

Figura 20 – Backup configurações atuais

Logo após, foi necessário executar o comando:

make menuconfig

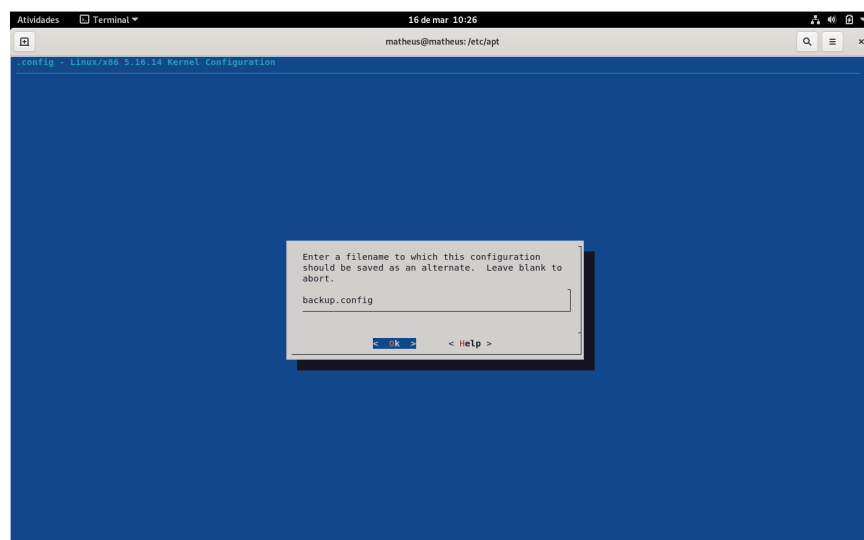


Figura 21 – Salvando configurações atuais a partir do comando make menuConfig

Por fim, será feita a instalação e compilação do kernel linux e seus módulos. Para realizar o procedimento mencionado, será executado o comando:

```
make -j4
```

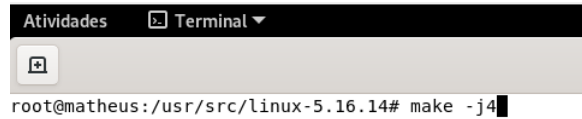


Figura 22 – Execução do comando make -j4

O comando mencionado foi usado com a finalidade de construir programas executáveis e bibliotecas a partir do código-fonte. Com o intuito de compilar também os módulos do kernel do linux, foi necessário executar o comando:

```
make modules_install
```



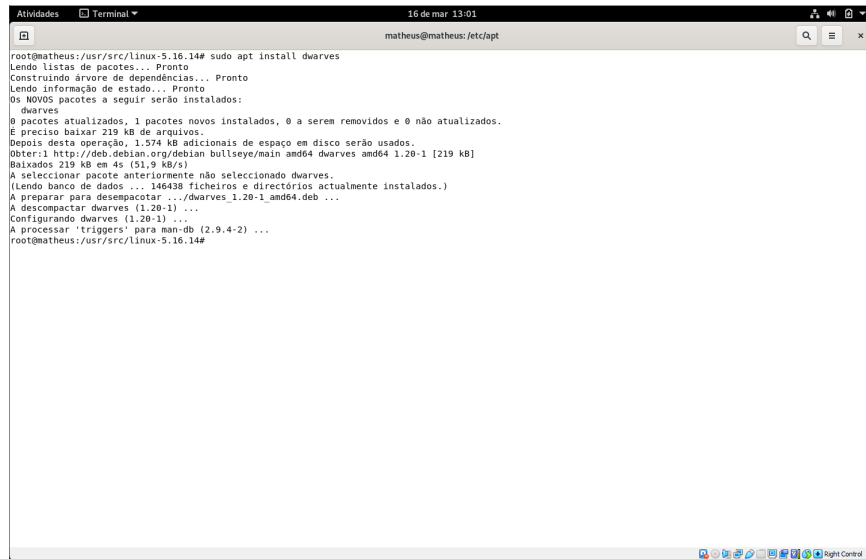
Figura 23 – Execução do comando make modules_install

De modo a finalizar a compilação do kernel, será executado o comando:

```
make install
```

Todavia, haverá novamente um erro com dependências de pacote, gerando a necessidade de instalar o pacote do DWARF.

```
sudo apt install dwarves
```



```

root@matheus:/usr/src/linux-5.16.14# sudo apt install dwarves
Lendo listas de pacotes... Pronto
Construindo árvore de dependências... Pronto
Lendo informação de estado... Pronto
Os NOVOS pacotes a seguir serão instalados:
  dwarves
0 pacotes atualizados, 1 pacotes novos instalados, 0 a serem removidos e 0 não atualizados.
É preciso baixar 219 kB de arquivos.
Depois desta operação, 1.574 kB adicionais de espaço em disco serão usados.
Obter:1 http://deb.debian.org/debian bullseye/main amd64 dwarves amd64 1.20-1 [219 kB]
Baixados 219 kB em 4s (51,9 kB/s)
A seleccionar pacote anteriormente não seleccionado dwarves.
(Lendo banco de dados ... 146438 ficheiros e directórios actualmente instalados.)
A preparar para desempacotar .../dwarves_1.20-1_amd64.deb ...
A descompactar dwarves (1.20-1) ...
Configurando dwarves (1.20-1) ...
A processar 'triggers' para nan-db (2.9.4-2) ...
root@matheus:/usr/src/linux-5.16.14#

```

Figura 24 – Instalação do pacote do DWARF

Agora que o pacote do DWARF está instalado na máquina virtual, será possível de fato fazer a compilação final do kernel linux. O pacote instalado “dwarves” pertence ao DWARF, o qual é um tipo de formato de arquivo de debug.

```

root@matheus:/usr/src/linux-5.16.14# make install
arch/x86/Makefile:142: CONFIG_X86_X32 enabled but no binutils support
sh ./arch/x86/boot/install.sh 5.16.14 \
    arch/x86/boot/bzImage System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 5.16.14 /boot/vmlinuz-5.16.14
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.16.14 /boot/vmlinuz-5.16.14
update-initramfs: Generating /boot/initrd.img-5.16.14
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.16.14 /boot/vmlinuz-5.16.14
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.16.14 /boot/vmlinuz-5.16.14
Generating grub configuration file ...
Found background image: /usr/share/images/desktop-base/desktop-grub.png
Found linux image: /boot/vmlinuz-5.16.14
Found initrd image: /boot/initrd.img-5.16.14
Found linux image: /boot/vmlinuz-5.10.0-12-amd64
Found initrd image: /boot/initrd.img-5.10.0-12-amd64
Found linux image: /boot/vmlinuz-5.10.0-10-amd64
Found initrd image: /boot/initrd.img-5.10.0-10-amd64
done
root@matheus:/usr/src/linux-5.16.14# █

```

Figura 25 – Execução do comando make install

5 Conclusões

O kernel foi compilado com sucesso. Podemos verificar o tamanho dos arquivos gerados na compilação a partir do fonte navegando até à pasta de /boot e executando o comando `ls -lh`.

```
matheus@matheus:~$ cd /boot
matheus@matheus:/boot$ ls -lh
total 148M
-rw-r--r-- 1 root root 231K dez  8 13:21 config-5.10.0-10-amd64
-rw-r--r-- 1 root root 231K mar  7 18:06 config-5.10.0-12-amd64
-rw-r--r-- 1 root root 143K mar 16 14:58 config-5.16.14
drwxr-xr-x 5 root root 4,0K mar 16 14:58 grub
-rw-r--r-- 1 root root 40M mar 15 19:14 initrd.img-5.10.0-10-amd64
-rw-r--r-- 1 root root 40M mar 15 19:13 initrd.img-5.10.0-12-amd64
-rw-r--r-- 1 root root 43M mar 16 14:58 initrd.img-5.16.14
-rw-r--r-- 1 root root  83 dez  8 13:21 System.map-5.10.0-10-amd64
-rw-r--r-- 1 root root  83 mar  7 18:06 System.map-5.10.0-12-amd64
-rw-r--r-- 1 root root 4,3M mar 16 14:58 System.map-5.16.14
-rw-r--r-- 1 root root 6,6M dez  8 13:21 vmlinuz-5.10.0-10-amd64
-rw-r--r-- 1 root root 6,6M mar  7 18:06 vmlinuz-5.10.0-12-amd64
-rw-r--r-- 1 root root 6,6M mar 16 14:58 vmlinuz-5.16.14
matheus@matheus:/boot$
```

Figura 26 – Visualização dos arquivos após compilação

E também o tamanho com os módulos compilados, indo para a pasta /lib/modules e executando o comando `du -sh 5.16.14`

```
matheus@matheus:/boot$ cd /lib/modules
matheus@matheus:/lib/modules$ du -sh 5.16.14
140M    5.16.14
matheus@matheus:/lib/modules$
```

Figura 27 – Visualização dos módulos compilados

6 Referências

WALLEN, Jack. How to Compile a Linux Kernel - Linux.com. Linux.com. Disponível em: <<https://www.linux.com/topic/desktop/how-compile-linux-kernel-0/>>. Acesso em: 18 Mar. 2022.

BuildADebianKernelPackage - Debian Wiki. Debian.org. Disponível em: <<https://wiki.debian.org/BuildADebianKernelPackage>>. Acesso em: 18 Mar. 2022.

SRINIVASAN, Vishhvak. Compiling a Custom Linux Kernel on a Virtual Machine. Medium. Disponível em: <<https://medium.com/@vishhvak/compiling-a-custom-linux-kernel-on-a-virtual-machine-12be9d32189b>>. Acesso em: 18 Mar. 2022.