Hendrick Felipe Scheifer
João Victor Briganti
Luiz Gustavo Takeda

Instalar o GNU/Linux e compilar o núcleo

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
Departa mento Acadêmico de Computação – DACOM
Bacharelado em Ciência da Computação – BCC

Campo Mourão
Outubro / 2024

Resumo

Este trabalho apresenta os procedimentos necessários para a instalação de uma distribuição GNU/Linux, focando na configuração e compilação do núcleo do sistema operacional. Utilizou-se o hipervisor VirtualBox (versão 7.1.2) para criar uma máquina virtual, na qual foi instalada a distribuição Debian (versão 12.7) e o núcleo Linux (versão 6.10.11). Durante o processo, foram executados diversos comandos básicos, possibilitando uma compreensão aprofundada da estrutura do sistema e a aplicação prática dos conceitos aprendidos em sala de aula. Os resultados evidenciam a importância da prática na consolidação do conhecimento teórico.

Palavras-chave: VirtualBox. Debian. Sistema Operacional.

Sumário

1 Introdução

Um sistema operacional é um dispositivo de *software* cuja função é tornar o computador mais simples e limpo, além de ser responsável pelo gerenciamento dos recursos da máquina (TANENBAUM, 2016).

A maioria dos computadores possui dois modos de operação: modo núcleo e modo usuário. O sistema operacional é um *software* que opera em modo núcleo, garantindo acesso total ao *hardware*, possibilitando a execução de qualquer instrução suportada pela máquina utilizada (TANENBAUM, 2016).

Este trabalho está estruturado da seguinte forma: após a introdução serão apresentados os objetivos do trabalho; em seguida, uma breve fundamentação essencial para o pleno entendimento do trabalho será apresentada; após isso, serão apresentados os materiais utilizados para a realização do trabalho, na seção 4; na seção 5, serão apresentadas e explicadas as etapas realizadas durante esta atividade; a seguir, na seção 6 e 7, os resultados serão discutidos e será apresentado as conclusões do trabalho, respectivamente.

2 Objetivos

Este trabalho visa instalar a distribuição GNU/Linux Debian em um ambiente virtual, compilar e instalar o núcleo Linux, e explicar detalhadamente cada um dos comandos utilizados durante todo o procedimento.

3 Fundamentação

A primeira versão do Linux foi desenvolvida por Linus Torvalds em 1991, como um sistema operacional para computadores que utilizavam o microprocessador, Intel 80386, que era um processador novo e o mais avançado da época. A parte mais interna de um sistema operacional é denominada núcleo, um *software* que fornece serviços básicos para todas as outras partes do sistema, gerencia o *hardware* e distribui recursos do sistema (LOVE, 2010).

Como o código-fonte do Linux é aberto, podemos configurá-lo antes de compilarmos, escolhendo apenas os *drivers* e conteúdos específicos que desejamos (LOVE, 2010). Uma distribuição GNU/Linux possui uma combinação específica de *softwares* aliados ao núcleo, o qual pode ser customizado conforme a necessidade. A distribuição utilizada para esta atividade é denominada GNU/Linux Debian.

4 Materiais

- Especificações do computador utilizado:
  - Modelo: Notebook Lenovo Thinkpad E14
  - CPU: AMD Ryzen 5-3500U
  - Memória Principal: 8GB RAM
  - Memória Secundária: SSD 256 NVME
  - Sistema Operacional: Fedora 40
- Hipervisor: VirtualBox 7.1.2
- Sistema Operacional utilizado no Hipervisor: GNU/Linux Debian 12.7
- Núcleo: Linux 6.10.11

5 Procedimentos e Resultados

Nesta seção, serão detalhados os procedimentos executados para a instalação da distribuição GNU/Linux Debian em um ambiente virtual, utilizando o hipervisor VirtualBox. Serão abordadas as configurações iniciais do hipervisor, a instalação do sistema operacional, verificações essenciais e os comandos utilizados para garantir o funcionamento correto do ambiente. Além disso, será descrito o processo de compilação do núcleo Linux, incluindo as etapas necessárias para a personalização e instalação da nova versão.

5.1 Configurações do Hipervisor

Os passos iniciais se deram por meio da instalação do Oracle VM VirtualBox e também da obtenção da imagem do Debian que foi usada para a instalação.

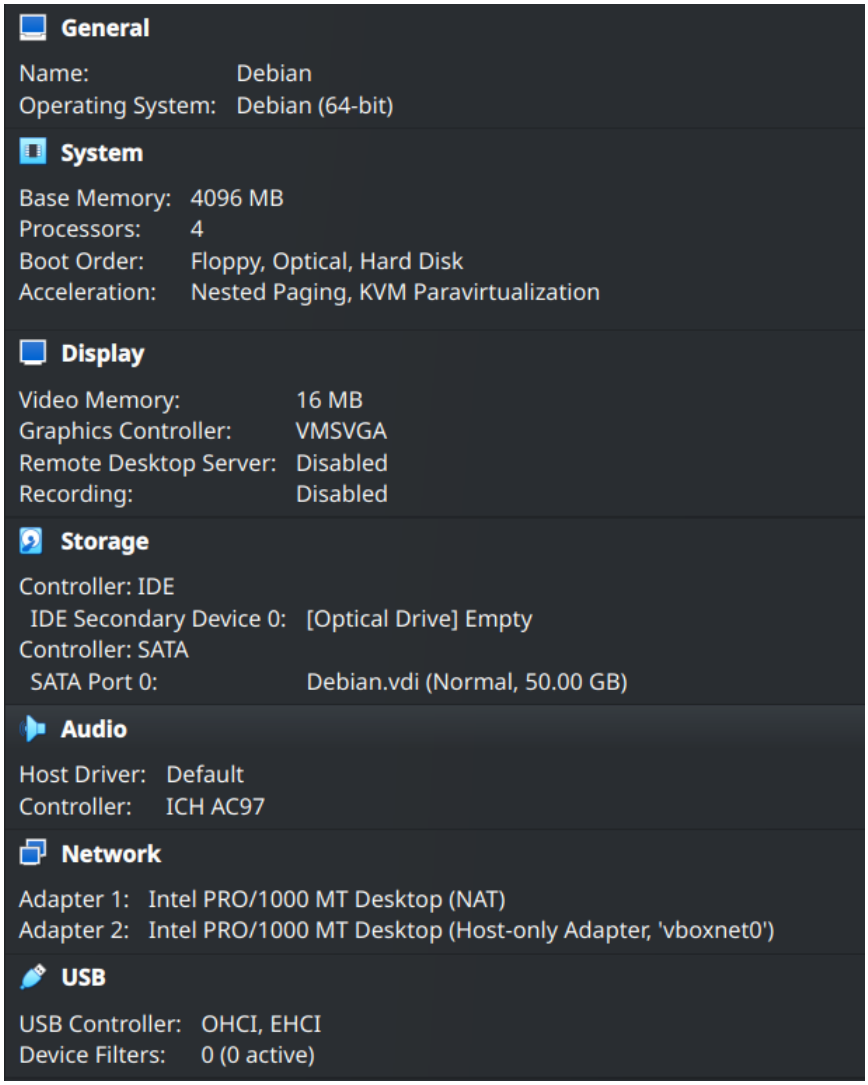


Figura 1 – Configurações do hipervisor.

A Figura 1 ilustra as principais configurações utilizadas para a instalação do sistema operacional. Para a memória primária, foi selecionado um total de 4 GB e 4 processadores, enquanto a memória secundária foi alocada com 50 GB para uso da máquina virtual.

Além das configurações de processadores e memória, foram incluídas duas placas de rede. O adaptador 1 está configurado como Network Address Translation (NAT), permitindo acesso à Internet, enquanto o adaptador 2 está configurado para uso interno. Essa configuração foi escolhida pois o acesso à máquina será realizado por meio do Secure Shell Connection (SSH).

5.2 Instalação do Sistema Operacional

Os primeiros passos durante a instalação envolveram a seleção do idioma. Para facilitar a referência a manuais durante o processo, optou-se pelo inglês. Após essa con-

figuração, o Brasil foi escolhido como localização, permitindo que o sistema operacional definisse o *locale* a ser utilizado para formatações específicas relacionadas à região do usuário (KERRISK, 2010). Além disso, o relógio do sistema foi ajustado para o fuso horário de São Paulo, e o teclado foi configurado para a disposição adequada da região.

Após as configurações de região, foram criados um usuário comum e um usuário *root*. Na Figura 2 temos a descrição do *root*, administrador do sistema, possuindo privilégios amplos e acesso total a todos os recursos e configurações do sistema.

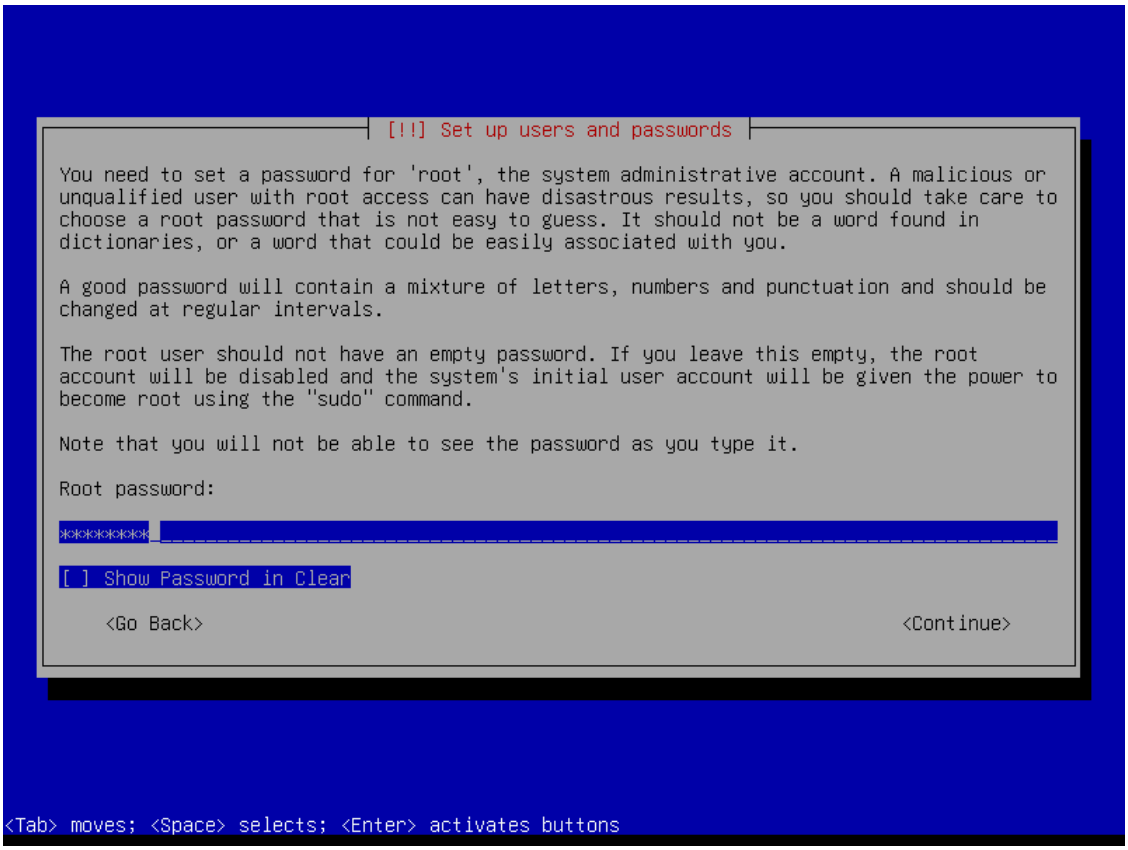


Figura 2 – Criação do usuário *root*.

Na Figura 3, estão apresentadas as configurações de particionamento e pontos de montagem do disco. O particionamento permite a divisão entre partições primárias e lógicas, sendo que cada uma delas é derivada do Master Boot Record (MBR). De maneira simplificada, as partições primárias são inicializáveis, enquanto as partições lógicas existem apenas como uma forma de contornar algumas limitações existentes na divisão das partições (NEGUS, 2012).

A memória secundária possui 50 GB de armazenamento, o que possibilitou que as partições fossem dispostas da seguinte maneira:

- /: É a raiz do sistema de arquivos, servindo como ponto de montagem para todos os outros dispositivos e sistemas de arquivos. Por ser o local onde o sistema operacional

inicia e onde estão armazenados arquivos críticos, foi alocado um tamanho de 3.6 GB para esta partição.

- */usr*: Este diretório é responsável por armazenar programas e dados de usuários. Por conter a maior quantidade de dados e ser amplamente utilizado durante a compilação, foi escolhido um tamanho de 32 GB para esta partição.
- */var*: Armazena dados variáveis, como logs e outros arquivos que podem mudar com frequência. Por não ser uma área crítica para o funcionamento do sistema, neste trabalho em específico, foi definido um tamanho de apenas 2.7 GB.
- *swap*: Esta partição é utilizada para armazenar páginas de memória que estavam na RAM, funcionando como uma extensão da memória virtual do sistema. Devido à quantidade de RAM já alocada durante a configuração do hipervisor, optou-se por um tamanho de 5.4 GB para a partição *swap*.
- */home*: Este diretório armazena os arquivos de configuração pessoal de cada usuário do sistema. Como é uma área frequentemente utilizada pelos usuários, foi definido um tamanho maior do que o das demais partições, totalizando 10 GB.

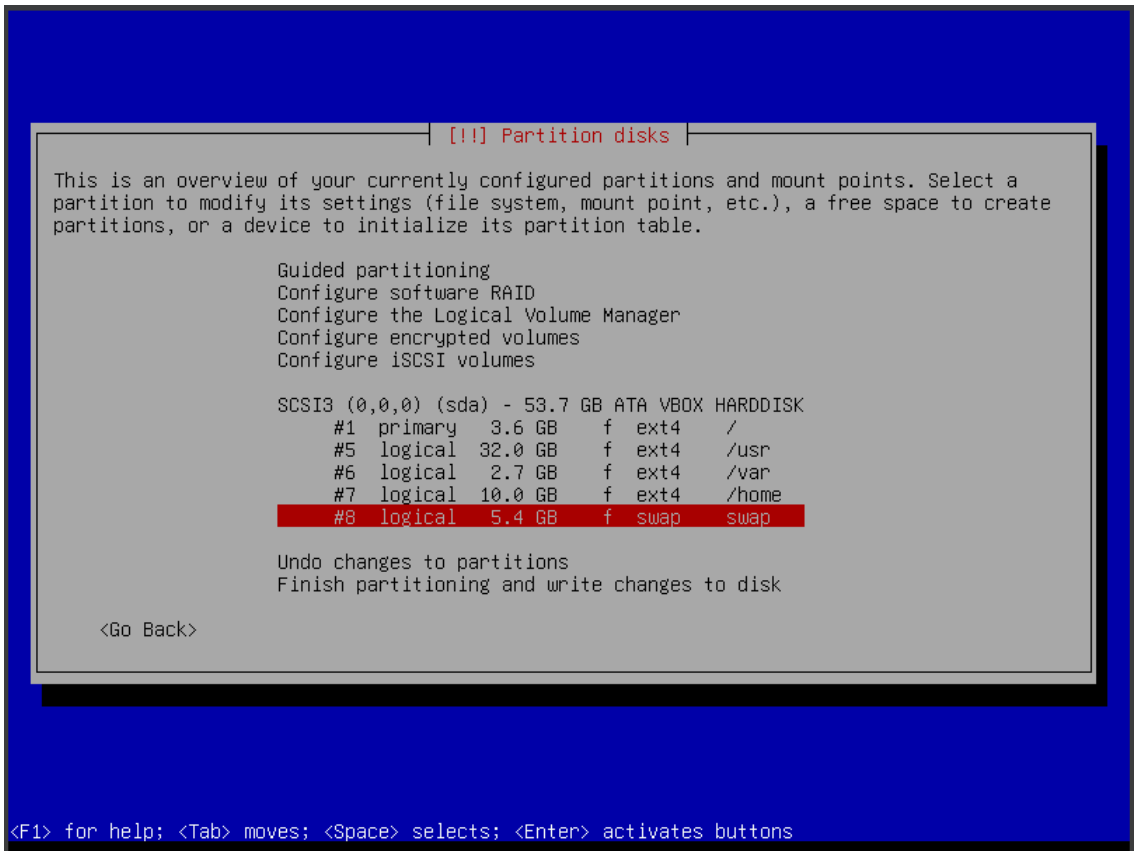


Figura 3 – Partições do sistema.

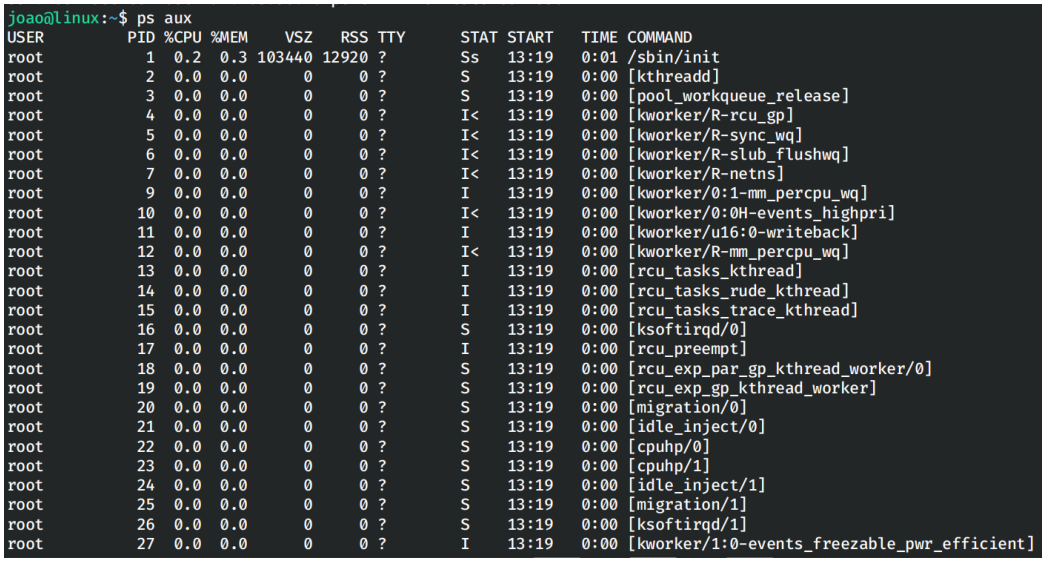
5.3 Verificações e Comandos

Após o processo de instalação, alguns comandos foram realizados para verificação e teste do sistema instalado.

5.3.1 ps aux

A Figura 4, mostra a saída do comando *ps* utilizado para a verificação dos processos em execução no sistema (NEGUS, 2012). A opção “aux”, é uma divisão de três parâmetros que alteram o comando da seguinte maneira:

- ’a’: Mostra os processo de todos os usuários.
- ’u’: Altera o formato da saída do programa.
- ’x’: Inclui processos de segundo plano, como *daemons*.



```
joao@linux:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.2  0.3 103440 12920 ?        Ss   13:19   0:01 /sbin/init
root           2  0.0  0.0      0     0 ?        S    13:19   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    13:19   0:00 [pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   13:19   0:00 [kworker/R-rcu_gp]
root           5  0.0  0.0      0     0 ?        I<   13:19   0:00 [kworker/R-sync_wq]
root           6  0.0  0.0      0     0 ?        I<   13:19   0:00 [kworker/R-slub_flushwq]
root           7  0.0  0.0      0     0 ?        I<   13:19   0:00 [kworker/R-netns]
root           9  0.0  0.0      0     0 ?        I    13:19   0:00 [kworker/0:1-mm_percpu_wq]
root          10  0.0  0.0      0     0 ?        I<   13:19   0:00 [kworker/0:0H-events_highpri]
root          11  0.0  0.0      0     0 ?        I    13:19   0:00 [kworker/u16:0-writeback]
root          12  0.0  0.0      0     0 ?        I<   13:19   0:00 [kworker/R-mm_percpu_wq]
root          13  0.0  0.0      0     0 ?        I    13:19   0:00 [rcu_tasks_kthread]
root          14  0.0  0.0      0     0 ?        I    13:19   0:00 [rcu_tasks_rude_kthread]
root          15  0.0  0.0      0     0 ?        I    13:19   0:00 [rcu_tasks_trace_kthread]
root          16  0.0  0.0      0     0 ?        S    13:19   0:00 [ksoftirqd/0]
root          17  0.0  0.0      0     0 ?        I    13:19   0:00 [rcu_preempt]
root          18  0.0  0.0      0     0 ?        S    13:19   0:00 [rcu_exp_par_gp_kthread_worker/0]
root          19  0.0  0.0      0     0 ?        S    13:19   0:00 [rcu_exp_gp_kthread_worker]
root          20  0.0  0.0      0     0 ?        S    13:19   0:00 [migration/0]
root          21  0.0  0.0      0     0 ?        S    13:19   0:00 [idle_inject/0]
root          22  0.0  0.0      0     0 ?        S    13:19   0:00 [cpuhp/0]
root          23  0.0  0.0      0     0 ?        S    13:19   0:00 [cpuhp/1]
root          24  0.0  0.0      0     0 ?        S    13:19   0:00 [idle_inject/1]
root          25  0.0  0.0      0     0 ?        S    13:19   0:00 [migration/1]
root          26  0.0  0.0      0     0 ?        S    13:19   0:00 [ksoftirqd/1]
root          27  0.0  0.0      0     0 ?        I    13:19   0:00 [kworker/1:0-events_freezable_pwr_efficient]
```

Figura 4 – Saída do comando *ps aux*.

5.3.2 df

A Figura 5 apresenta a saída do comando *df*, utilizado para verificar o espaço utilizado pelos sistemas de arquivos no sistema (NEGUS, 2012). A opção “-h” exibe a saída de maneira mais legível, facilitando a interpretação dos dados.

```
joao@linux:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            2.0G     0  2.0G   0% /dev
tmpfs           393M  604K  393M   1% /run
/dev/sda1       3.6G  229M  3.2G   7% /
/dev/sda5        32G  6.3G   24G  21% /usr
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/sda8        10G   60K  9.5G   1% /home
/dev/sda6       2.7G  308M  2.3G  12% /var
tmpfs           393M     0  393M   0% /run/user/1000
```

Figura 5 – Saída do comando *df -h*.

5.3.3 free

A Figura 6 apresenta a saída do comando *free*, utilizado para verificar a utilização da memória no sistema (SHOTTS, 2017). A opção "-b" exibe os valores de memória em bytes, proporcionando uma visão detalhada da quantidade exata de memória disponível e utilizada.

```
joao@linux:~$ free -b
               total        used        free      shared  buff/cache   available
Mem:      4117438464   326500352  3714129920      618496   292372480  3790938112
Swap:      699396096           0   699396096
```

Figura 6 – Saída do comando *free -b*.

5.3.4 cat /proc/meminfo

A Figura 7 apresenta a saída do comando *cat /proc/meminfo*. O comando *cat* é utilizado para exibir o conteúdo de arquivos de texto no terminal (SHOTTS, 2017). Neste caso, */proc/meminfo* é um arquivo do sistema que fornece dados em tempo real sobre a utilização da memória física e da *swap* (NGUYEN, 2004).

```
joao@linux:~$ cat /proc/meminfo
MemTotal:        4020936 kB
MemFree:         3628052 kB
MemAvailable:    3703128 kB
Buffers:           16812 kB
Cached:           254312 kB
SwapCached:            0 kB
Active:           143160 kB
Inactive:         152528 kB
Active(anon):        436 kB
Inactive(anon):    24732 kB
Active(file):     142724 kB
Inactive(file):   127796 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:        683004 kB
SwapFree:         683004 kB
Zswap:                 0 kB
Zswapped:              0 kB
Dirty:                 4 kB
Writeback:             0 kB
AnonPages:         24744 kB
Mapped:            26388 kB
Shmem:               604 kB
KReclaimable:      14472 kB
Slab:              36996 kB
SReclaimable:      14472 kB
SUnreclaim:        22524 kB
KernelStack:        1944 kB
PageTables:         1940 kB
SecPageTables:         0 kB
NFS_Unstable:          0 kB
```

Figura 7 – Saída do comando *cat /proc/meminfo*.

5.3.5 ip

A Figura 8 apresenta as saídas dos comandos *ip -c address show* e *ip -c route*. O comando *ip* é parte das ferramentas do sistema Linux utilizadas para exibir e configurar informações de rede (KERRISK, 2011). O parâmetro ”-c“ é utilizado para adicionar cores às saídas, facilitando a leitura das informações.

O comando *ip -c address show* fornece dados importantes, como o endereço Internet Protocol (IP), máscara de rede, *status* das interfaces e outras configurações relevantes.

O comando *ip -c route* exibe a tabela de roteamento.

```
joao@linux:~$ ip -c 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 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:cd:8d:52 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 82750sec preferred_lft 82750sec
    inet6 fd00::a00:27ff:fecd:8d52/64 scope global dynamic mngtmpaddr
       valid_lft 86379sec preferred_lft 14379sec
    inet6 fe80::a00:27ff:fecd:8d52/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:bf:86:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.10/24 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet 192.168.56.106/24 brd 192.168.56.255 scope global secondary dynamic noprefixroute enp0s8
       valid_lft 400sec preferred_lft 325sec
    inet6 fe80::71e6:74a9:ac93:4bb6/64 scope link
       valid_lft forever preferred_lft forever
joao@linux:~$ ip -c route
default via 10.0.2.2 dev enp0s3
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15
169.254.0.0/16 dev enp0s3 scope link metric 1000
192.168.56.0/24 dev enp0s8 proto dhcp scope link src 192.168.56.106 metric 1003
```

Figura 8 – Saídas dos comandos *ip -c address show* e *ip -c route*.

5.3.6 cat /etc/resolv.conf

A Figura 9 mostra a saída do comando *cat /etc/resolv.conf*, que exibe o conteúdo do arquivo responsável pela configuração dos servidores *Domain Name Server* (DNS) no sistema (NEGUS, 2012). Esse arquivo contém informações sobre quais servidores DNS o sistema deve consultar para resolver nomes de domínio em endereços IP.

```
joao@linux:~$ cat /etc/resolv.conf
domain local
search local.
nameserver 10.0.2.3
```

Figura 9 – Conteúdo do arquivo */etc/resolv.conf*.

5.3.7 cat /etc/network/interfaces

A Figura 10 apresenta a saída do comando *cat /etc/network/interfaces*, que exibe as configurações de rede do sistema (SILVA, 2020b). Este arquivo define as interfaces de rede disponíveis e suas configurações, como endereços IP, máscara de sub-rede e gateways.

```
joao@linux:~$ 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

# The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp
# This is an autoconfigured IPv6 interface
iface enp0s3 inet6 auto
```

Figura 10 – Conteúdo do arquivo */etc/network/interfaces*.

5.3.8 ping

A Figura 11 mostra a saída do comando *ping google.com*, utilizado para verificar a conectividade com o host especificado (SHOTTS, 2017). O comando envia pacotes ICMP Echo Request para o servidor do Google e mede o tempo de resposta, permitindo avaliar a latência da conexão.

```
joao@linux:~$ ping -c 4 google.com
PING google.com (172.217.30.46) 56(84) bytes of data.
64 bytes from gru10s10-in-f14.1e100.net (172.217.30.46): icmp_seq=1 ttl=255 time=80.7 ms
64 bytes from pngrua-ai-in-f14.1e100.net (172.217.30.46): icmp_seq=2 ttl=255 time=102 ms
64 bytes from gru10s10-in-f14.1e100.net (172.217.30.46): icmp_seq=3 ttl=255 time=124 ms
64 bytes from gru06s50-in-f14.1e100.net (172.217.30.46): icmp_seq=4 ttl=255 time=44.0 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 44.002/87.617/124.046/29.473 ms
```

Figura 11 – Saída do comando *ping google.com*.

5.3.9 Configuração de Repositórios

O arquivo *sources.list* é um componente essencial do gerenciador de pacotes *apt* no Debian (SILVA, 2020a). Neste arquivo é definido os repositórios de onde o sistema obtém os pacotes de software e suas respectivas atualizações. Cada linha do arquivo especifica uma fonte de pacotes, que pode ser um repositório oficial ou um repositório de terceiros.

O *security source list* é uma configuração específica que permite ao sistema acessar repositórios dedicados a atualizações de segurança. Esses repositórios contêm pacotes que corrigem vulnerabilidades e problemas de segurança, sendo fundamental para manter o sistema protegido contra ameaças.

Como apresentado na Figura 12 o Debian 12.7 utilizado neste trabalho, já está previamente configurado com as entradas necessárias para acessar os repositórios principais e de segurança.

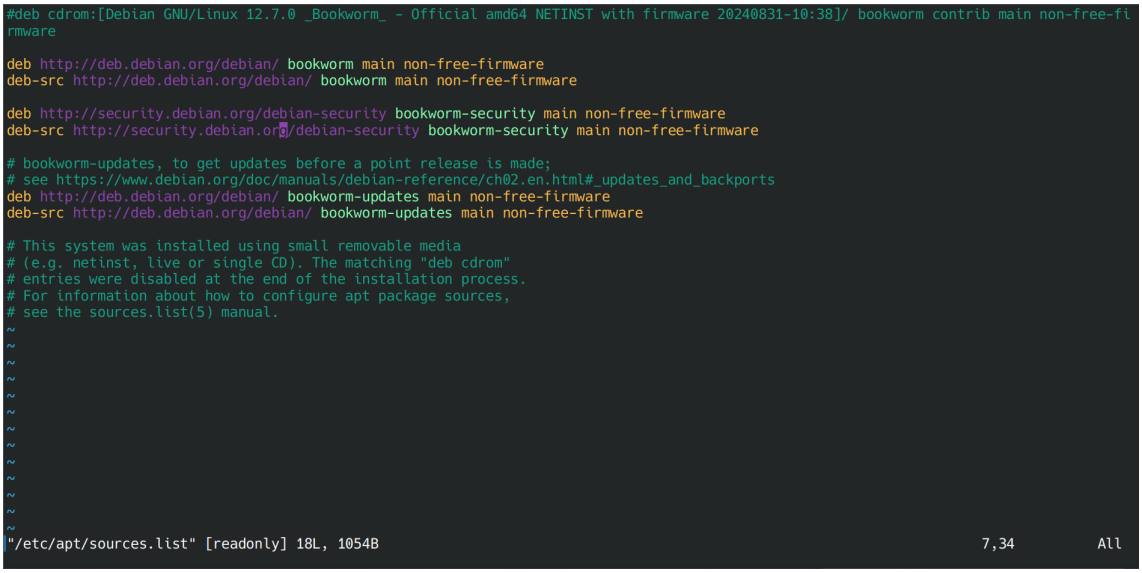


Figura 12 – Conteúdo do arquivo */etc/apt/sources.list*.

5.3.10 uname

A Figura 13 mostra a saída do comando *uname*, utilizado para exibir informações sobre o sistema operacional em execução (NEGUS, 2012). Este comando tem como saída detalhes sobre o sistema, como o nome do kernel, a versão do kernel, a arquitetura do processador e o nome do host. O parâmetro ”-a“ é usado para que a saída mostre todas as informações disponíveis.

```
joao@linux:~$ uname -a
Linux linux 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64 GNU/Linux
```

Figura 13 – Saída do comando *uname -a*.

5.3.11 Comandos Gerais

Existem comandos fundamentais para a interação com o sistema operacional e a realização de diversas operações (SILVA, 2020a). Abaixo, apresentamos uma lista de comandos comumente usados:

- `clear`: Restaura o terminal a um estado inicial, “limpando” todas as saídas anteriores e comandos executados.
- `ls -l`: Exibe uma lista detalhada dos arquivos e diretórios no diretório especificado. Caso nenhum diretório seja indicado, a listagem será feita do diretório atual.
- `cd`: Altera o diretório de trabalho atual para o diretório especificado no parâmetro, permitindo a navegação pelo sistema de arquivos.
- `cat`: Concatena e exibe o conteúdo de um ou mais arquivos na saída padrão.

- `rm`: Remove permanentemente o arquivo especificado do sistema.
- `nano`: Editor de texto em linha de comando que permite a criação e modificação de arquivos de texto diretamente no terminal.
- `cp`: Copia um arquivo de origem para o destino especificado.
- `grep`: Busca e exibe as linhas de um arquivo que correspondem a um padrão especificado pelo usuário.
- `head`: Exibe as primeiras linhas de um arquivo, por padrão somente as 10 primeiras linhas são exibidas.
- `tail`: Mostra as últimas linhas de um arquivo. Similar ao comando `head`, também exibe 10 linhas por padrão.
- `mv`: Move ou renomeia arquivos.

A Figura 14 e a Figura 15 ilustram o uso desses comandos.

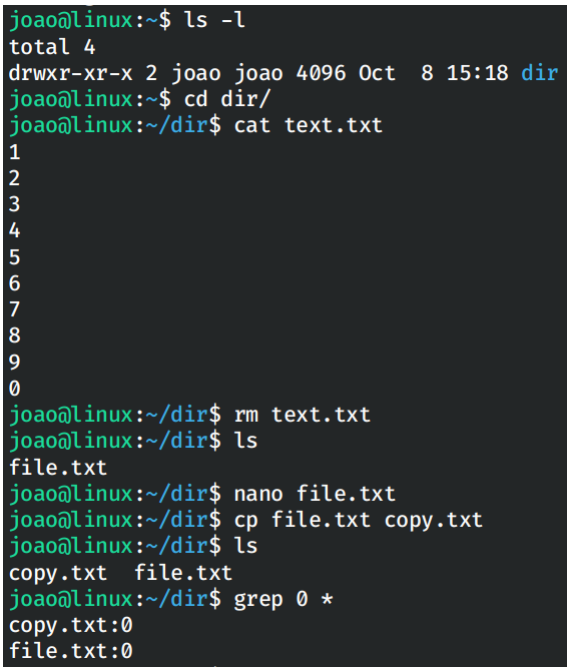


Figura 14 – Exemplo de comandos gerais do Linux em um terminal.

Figura 15 – Exemplo de comandos gerais do Linux em um terminal.

5.4 Compilação do Núcleo

Para o processo de compilação do núcleo Linux, é essencial instalar uma série de programas e bibliotecas necessárias. O comando utilizado para essa instalação é o *apt install build-essentials bc dwarves bison flex gnupg libncurses-dev libelf-dev libssl-dev wget*. Este comando instala um conjunto de pacotes no sistema, cada um com funções específicas que contribuem para a compilação e configuração do núcleo.

- `build-essentials`: Um conjunto de pacotes essenciais para construção de pacotes no Debian.
- `bc`: Calculadora de linha de comando.
- `dwarves`: Conjunto de ferramentas depuração para arquivos no formato Executable and Linkable Format(ELF).
- `bison`: Gerador de analisador sintático de uso geral.
- `flex`: Gerador de analisadores léxicos.
- `gnupg`: Ferramenta para encriptação de dados e geração de assinaturas digitais.
- `libncurses-dev`: Biblioteca de manipulação de caracteres em terminais.
- `libelf-dev`: Biblioteca para leitura e escrita de arquivos ELF.

- `libssl-dev`: Parte do projeto OpenSSL para implementação do protocolo Secure Socket Layer(SSL) e Transport Layer Security (TLS).
- `wget`: Utilitário para obter arquivos usando Hypertext Transfer Protocol (HTTP) ou File Transfer Protocol (FTP).

Durante a realização deste trabalho, a versão mais recente do núcleo Linux disponível era a 6.10.11. A obtenção dessa versão foi realizada por meio do comando *wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.10.11.tar.xz*. O código-fonte é fornecido em um arquivo comprimido, requerendo sua extração. Isso pode ser feito com o comando *tar -xf linux-6.10.11.tar.xz -C /usr/src/linux-6.10.11/*. A Figura 16 ilustra este processo.

```
root@linux:~/kernel# wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.10.11.tar.xz
--2024-09-22 15:33:30--  https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.10.11.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.93.176, 2a04:4e42:16::432
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.93.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 145185812 (138M) [application/x-xz]
Saving to: ‘linux-6.10.11.tar.xz’

linux-6.10.11.tar.xz              100%[=============================================================>] 138.46M  8.75MB/s    in 17s

2024-09-22 15:33:47 (8.15 MB/s) - ‘linux-6.10.11.tar.xz’ saved [145185812/145185812]

root@linux:~/kernel# ls
linux-6.10.11.tar.xz
root@linux:~/kernel# tar -xf linux-6.10.11.tar.xz -C /usr/src/
root@linux:~/kernel# ls -l /usr/src
total 4
drwxrwxr-x 26 root root 4096 Sep 18 14:25 linux-6.10.11
```

Figura 16 – Obtenção do núcleo e extração do código-fonte.

A compilação do núcleo Linux é baseada em um arquivo de configuração, que no diretório de compilação é denominado por *.config*. No Debian, esse arquivo pode ser encontrado no diretório */boot*, onde estão armazenados diversos parâmetros que serão utilizados durante o processo de compilação. Antes de iniciar a compilação, é necessário obter este arquivo de configuração e copiá-lo para o diretório onde a compilação será realizada, que neste trabalho é o diretório */usr/src/linux-6.10.11/*. Para isso, foi utilizado o comando *cp /boot/config-$(uname -r) .config*.

Após essa configuração inicial, o comando *make localmodconfig* é executado. Esse comando ajusta o arquivo de configuração de modo que apenas os módulos já carregados no sistema sejam considerados durante a compilação.

Para as configurações de módulo o comando *make menuconfig* é usado, com ele uma interface é apresentada para o usuário de modo que o mesmo possa escolher algumas opções que serão ou não compiladas com o núcleo. Na Figura 17 temos o uso deste menu para a habilitação do módulo para Universal Serial Bus (USB) *mass storage*, o que basicamente habilita o suporte para *pen drive*, alguns tipos de dispositivos de disco rígido via USB e afins. Já a Figura 18 utiliza o menu para habilitar o NT File System (NTFS) um sistema de arquivos utilizado no Windows.

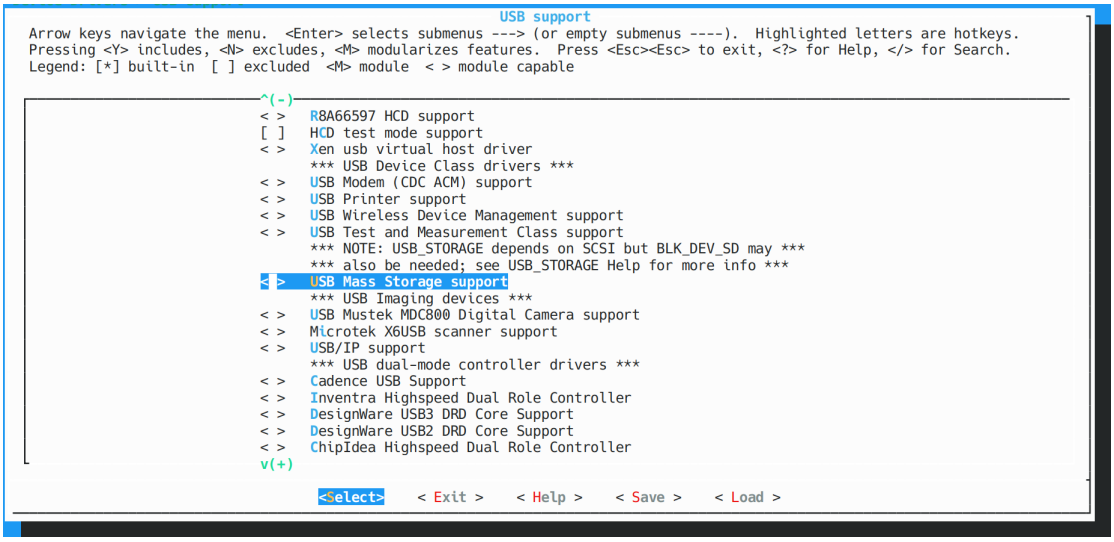


Figura 17 – Habilitando módulo para USB *mass storage*.

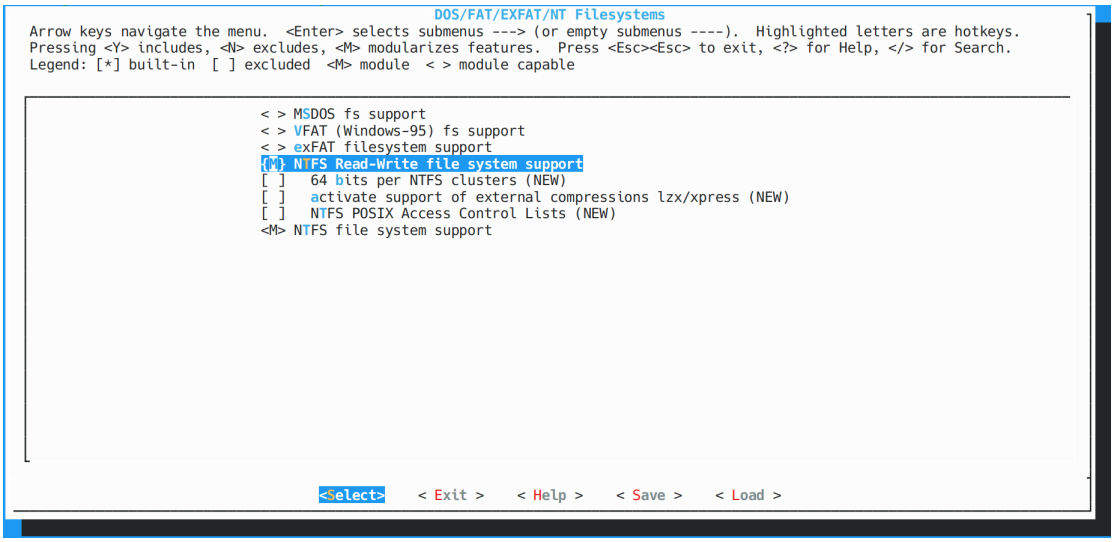


Figura 18 – Habilitando módulo para sistema de arquivos NTFS.

O uso deste menu não se limita apenas à habilitação de módulos importantes, mas também permite a desabilitação de módulos que não serão utilizados. A Figura 19 ilustra a configuração necessária para desabilitar *drivers* de dispositivos específicos para computadores Macintosh.

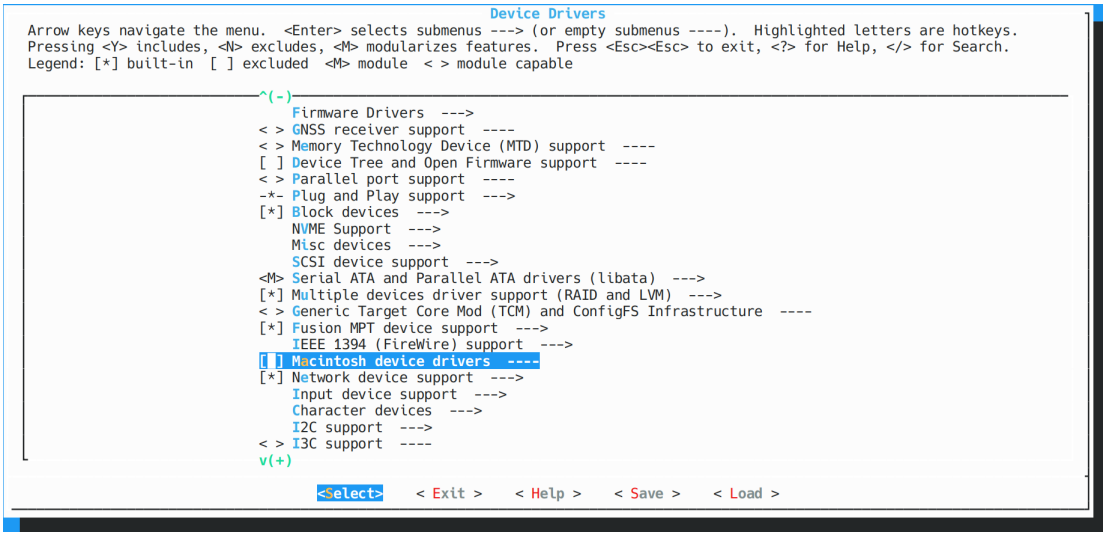


Figura 19 – Desabilitando módulos de *driver* para computadores Macintosh.

Após as configurações no núcleo a compilação pode ser realizada. Na compilação do núcleo o comando *make -j2 LOCALVERSION=-utfpr* foi utilizado, onde *-j2* especifica a quantidade de núcleos utilizados durante a compilação e *LOCALVERSION=-utfpr* é um parâmetro opcional que especifica um nome que será concatenado ao nome do núcleo.

Após as configurações no núcleo, o processo de compilação pode ser iniciado. Para isso, foi utilizado o comando *make -j2 LOCALVERSION=-utfpr*. Nesse comando, "-j2" especifica o número de núcleos a serem utilizados durante a compilação, otimizando o tempo de execução. O parâmetro opcional *LOCALVERSION=-utfpr* adiciona um sufixo ao nome do núcleo, facilitando a identificação da versão compilada.

Uma vez concluída a compilação, o comando *make modules_install* deve ser utilizado para instalar os módulos do núcleo compilado. Em seguida, o comando *make install* instala o núcleo em si. Vale ressaltar que o novo núcleo só será utilizado após a reinicialização da máquina. A Figura 20 mostra a saída do comando *uname -a* após a reinicialização do comando.

```
joao@linux:~$ uname -a
Linux linux 6.10.11-utfpr #2 SMP PREEMPT_DYNAMIC Sun Sep 22 19:44:05 -03 2024 x86_64 GNU/Linux
```

Figura 20 – Saída do comando *uname -a* após a compilação do núcleo.

A Figura 21 mostra a saída do comando *du -sh /lib/modules/6.10.11-utfpr/* que retorna o espaço em disco ocupado pelos módulos do núcleo, e também a saída do comando *ls -lh /boot* que têm como saída os arquivos essenciais para o funcionamento do núcleo, nele encontramos duas versões do núcleo sobre o nome *vmlinuz*, as configurações de compilação do mesmo sobre o nome *config* o *initrd* usado para iniciar o sistema operacional e por fim *System.map* que armazena alguns símbolos do núcleo para facilitar a depuração (VASUDEVAN, 2003).

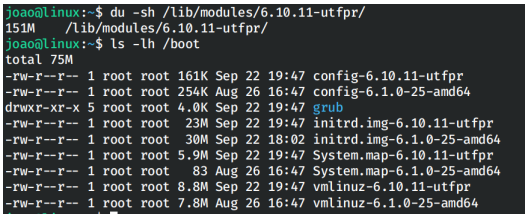


Figura 21 – Obtenção do núcleo e extração do código-fonte.

6 Discussão dos Resultados

Os resultados obtidos foram satisfatórios, alcançando o objetivo de instalar o GNU/Linux Debian e compilar o núcleo do Linux com sucesso. No entanto, algumas dificuldades foram encontradas, especialmente em relação ao dimensionamento das partições, o que exigiu ajustes manuais e revisão das etapas de particionamento. Além disso, houve certa confusão com a disposição e clareza das informações sobre o processo de compilação do núcleo, o que tornou a navegação pelos guias e documentação mais desafiadora do que o esperado. Apesar desses obstáculos, foi possível remover módulos desnecessários, como os *drivers* de dispositivo para Macintosh, otimizando o núcleo para o hardware utilizado. O mapeamento detalhado dos passos de instalação e compilação proporcionou uma compreensão prática e aprofundada da estrutura e configuração do sistema.

7 Conclusões

O objetivo principal de instalar o GNU/Linux Debian, compreender a estrutura básica do Linux, explorar alguns comandos essenciais, e configurar e compilar o núcleo foi atingido com êxito. A grande vantagem desse processo foi a capacidade de personalizar o núcleo, removendo partes que não seriam utilizadas, como módulos irrelevantes, resultando em economia de recursos do sistema. No entanto, para aqueles que buscam uma solução mais prática e *plug and play*, esse processo não é recomendado, pois demanda conhecimentos avançados e pode ser desnecessariamente complexo para usuários comuns. Esse tipo de atividade é mais adequado para profissionais e entusiastas que desejam um controle maior sobre seu sistema operacional.

8 Referências

KERRISK, M. *The Linux Programming Interface: A Linux and UNIX System Programming Handbook*. San Francisco: No Starch Press, 2010. ISBN 978-1-59327-220-3. Citado na página 7.

KERRISK, M. *ip(8) - Linux manual page — man7.org*. 2011. https://www.man7.org/linux/man-pages/man8/ip.8.html. [Acessado 08-10-2024]. Citado na página 11.

LOVE, R. *Linux Kernel Development*. [S.l.]: Addison-Wesley Educational Publishers Inc, 2010. Citado na página 4.

NEGUS, C. *Linux bible*. [S.l.]: John Wiley & Sons, 2012. v. 772. Citado 4 vezes nas páginas 7, 9, 12 e 14.

NGUYEN, B. */proc — tldp.org*. 2004. https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html. [Acessado 08-10-2024]. Citado na página 10.

SHOTTS, W. *The Linux command line*. [S.l.]: LinuxCommand. org, 2017. Citado 2 vezes nas páginas 10 e 13.

SILVA, G. M. da. *Guia Foca Linux, Iniciante-Intermediário*. 2020. https://www.guiafoca.org/guiaonline/inicianteintermediario/. [Acessado 13-10-2024]. Citado 2 vezes nas páginas 13 e 14.

SILVA, G. M. da. *Guia Foca Linux, Intermediário*. 2020. https://www.guiafoca.org/guiaonline/intermediario/. [Acessado 13-10-2024]. Citado na página 12.

TANENBAUM, H. B. A. S. *Sistemas OPeracioanis Modernos*. [S.l.]: Pearson, 2016. Citado na página 4.

VASUDEVAN, A. *The Linux Kernel HOWTO — faqs.org*. 2003. http://www.faqs.org/docs/Linux-HOWTO/Kernel-HOWTO.html#kernel_files_info. [Acessado 09-10-2024]. Citado na página 19.