UNIVERSIDADE DE VILA VELHA

Endrews Elias de Oliveira Prado

Jhuan da Matta Julião de Oliveira

Lucas Mattos Canela

Pedro Henrique de Oliveira Vieira Martins

Robert Valadão da Silva

Victor Griffo de Andrade Faria

Conceitos de Sistemas Operacionais

2° Bimestre

Vila Velha, ES

23 de novembro de 2025

Endrews Elias de Oliveira Prado
Jhuan da Matta Julião de Oliveira
Lucas Mattos Canela
Pedro Henrique de Oliveira Vieira Martins
Robert Valadão da Silva
Victor Griffo de Andrade Faria

Conceitos de Sistemas Operacionais
2° Bimestre

Universidade de Vila Velha (UVV)
Graduação em Ciência da Computação

Orientador: Jean-Remi

Vila Velha, ES
23 de novembro de 2025

"Grandes poderes trazem
grandes responsabilidades."
(Homem-Aranha)

RESUMO

Este trabalho apresenta uma visão sintética dos principais componentes e mecanismos de sistemas operacionais modernos, tomando o Linux como referência conceitual. Iniciamos pelo processo de boot, contrastando BIOS legado e UEFI, destacando o papel do firmware na inicialização do kernel e o uso de partições EFI. Em seguida abordamos armazenamento: diferenças estruturais entre HDD (trilhas, setores, cilindros) e SSD (páginas, blocos, NVMe) e implicações de desempenho. O particionamento é discutido via MBR e GPT, incluindo limitações, endereçamento e integridade. Examinamos a camada de sistemas de arquivos (inodes, blocos, metadados) e famílias clássicas e modernas (Ext, FAT, NTFS, BtrFS, ZFS) com ênfase em journaling e Copy-on-Write. Tratamos gestão de usuários e permissões (UID, GID, bits r/w/x), processo e segurança via sudo. No gerenciamento de processos descrevemos PID, PCB e estados (execução, pronto, bloqueado) e a troca de contexto. Finalizamos com técnicas de gerenciamento de memória primária: endereçamento virtual, paginação, segmentação e swapping, relacionando isolamento e eficiência. A síntese integra camadas de hardware e software para contextualizar decisões de projeto que impactam desempenho, segurança e confiabilidade.
Palavras-chave: sistemas operacionais; boot; UEFI; MBR; GPT; HDD; SSD;

sistemas de arquivos; processos; memória; permissões.

ABSTRACT

This report provides a concise overview of core components and mechanisms of modern operating systems with Linux as the guiding reference. It starts with the boot process, contrasting legacy BIOS and UEFI, outlining firmware responsibilities and the role of the EFI system partition. Storage is examined through structural differences between HDDs (tracks, sectors, cylinders) and SSDs (flash pages, blocks, NVMe) and their performance implications. Disk partitioning is discussed via MBR and GPT, emphasizing limitations, addressing and integrity. The filesystem layer (inodes, blocks, metadata) is reviewed across traditional and modern families (Ext, FAT, NTFS, BtrFS, ZFS) focusing on journaling and Copy-on-Write. User and permission management (UID, GID, r/w/x bits) and privilege elevation through sudo are presented. Process management covers PID, PCB, execution states (running, ready, blocked) and context switching. Finally, primary memory techniques—virtual addressing, paging, segmentation and swapping—are related to isolation and efficiency. The synthesis connects hardware and software layers to contextualize design choices affecting performance, security and reliability.
Keywords: operating systems; boot; UEFI; MBR; GPT; HDD; SSD; filesystems;

processes; memory; permissions.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

Parte I

Discos

Figura 1 – Tipos de discos de armazenamento.

1 PROCESSO DE BOOT

O processo de inicialização (*boot*) é a sequência de operações que o computador realiza ao ser ligado para carregar o sistema operacional. O termo deriva de "pull oneself up by one's bootstraps". Conceitos fundamentais de boot e interação com o kernel são discutidos em (TANENBAUM; BOS, 2015; SILBERSCHATZ; GALVIN; GAGNE, 2018; LOVE, 2010).

1.1 BIOS E ROM

O processo inicia-se na ROM (*Read-Only Memory*), onde reside o BIOS (*Basic Input/Output System*). O BIOS é gravado em um chip e é responsável pelo teste de hardware (POST) e pela inicialização básica. As configurações do BIOS (sequência de boot, data/hora) são armazenadas em uma memória CMOS alimentada por bateria. Historicamente, evoluímos de memórias PROM (programáveis uma vez), EPROM (apagáveis por UV), EEPROM (apagáveis eletricamente) até as modernas Flash ROM, que permitem atualizações de firmware (PATTERSON; HENNESSY, 2014). A padronização moderna de firmware segue a especificação (FORUM, 2023).

1.2 UEFI E LEGACY

Nos anos 90, a Intel iniciou a substituição do BIOS pelo EFI, que evoluiu para o padrão UEFI (*Unified Extensible Firmware Interface*) (FORUM, 2023).

- **Legacy BIOS:** Utiliza o MBR para boot. Carrega apenas o primeiro estágio do *bootloader* nos primeiros 446 bytes do disco.

- **UEFI:** Armazena na NVRAM o caminho para os carregadores. Utiliza uma partição específica (Partição EFI ou ESP), formatada geralmente em FAT32, onde residem os arquivos `.efi` dos sistemas operacionais.

1.3 GERENCIADORES DE BOOT

O carregador de inicialização (*bootloader*), como o GRUB ou LILO, é carregado pelo firmware. Sua função é carregar o Kernel do sistema operacional na memória RAM e transferir o controle para ele (LOVE, 2010). No Linux, o Kernel geralmente utiliza um sistema de arquivos temporário (*initramfs*) para montar a partição raiz.

2 MEMÓRIA SECUNDÁRIA

A memória secundária, ou memória de massa, é não-volátil e utilizada para armazenar grandes quantidades de dados fora do processador (PATTERSON; HENNESSY, 2014).

2.1 DISCOS RÍGIDOS (HDD)

Dominaram o mercado por décadas. São compostos por discos magnéticos (pratos) e cabeças de leitura/gravação mecânicas.

- **Geometria:** Organizados em Trilhas (círculos concêntricos), Setores (fatias das trilhas, geralmente 512 bytes) e Cilindros (conjunto de trilhas alinhadas verticalmente).
- **Endereçamento:** Antigamente usava-se CHS (Cylinder-Head-Sector). Modernamente utiliza-se LBA (*Logical Block Addressing*), que trata o disco como uma sequência linear de blocos.

2.2 UNIDADES DE ESTADO SÓLIDO (SSD)

Utilizam memória Flash (NAND ou NOR), baseada em transistores que armazenam carga elétrica (bits) sem partes móveis (ORGANIZATION, 2023).

- **Vantagens:** Alta velocidade de acesso, resistência a impactos e menor consumo de energia.
- **Estrutura:** Não possuem trilhas ou setores físicos, organizando-se em páginas e blocos. O protocolo NVMe (*Non-Volatile Memory Express*) foi criado para explorar a velocidade do barramento PCIe, superando as limitações do padrão SATA.

A redução drástica de latência em SSDs impacta também estratégias de paginação discutidas no Capítulo 7.

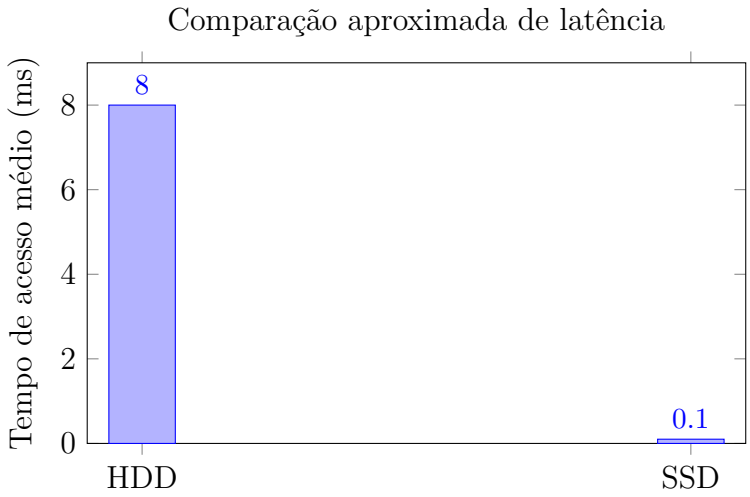


Figura 2 – Latência típica: HDD ~8 ms vs SSD ~0.1 ms.

3 PARTIÇÕES

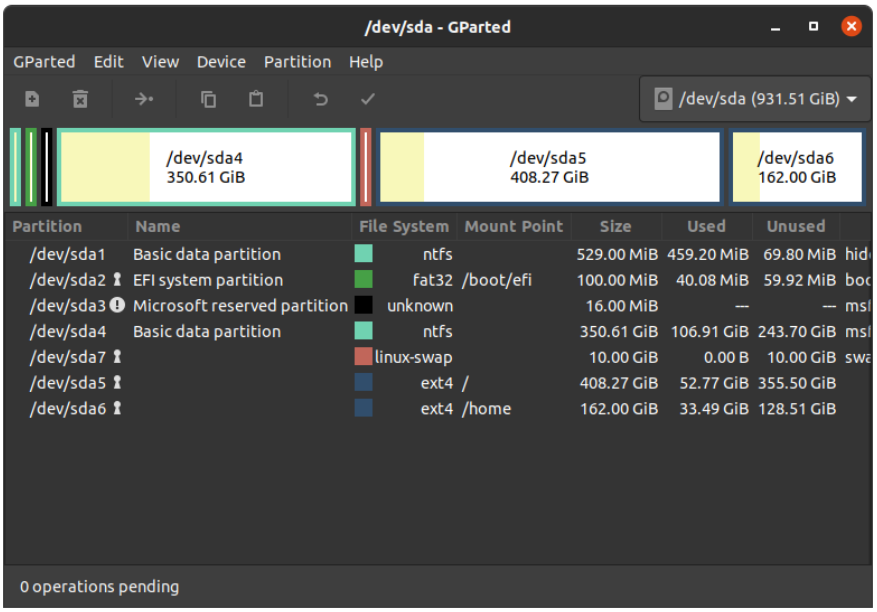


Figura 3 – Particionamento de disco rígido.

O particionamento divide o disco físico em unidades lógicas. Existem dois esquemas principais de tabelas de partição.

3.1 MBR (MASTER BOOT RECORD)

O MBR localiza-se no primeiro setor do disco (512 bytes) (CORPORATION, 2010).

- **Estrutura:** Contém a área de boot (446 bytes), a tabela de partições (64 bytes) e a assinatura (2 bytes).
- **Limitações:** Suporta apenas 4 partições primárias. Para contornar isso, criou-se a *Partição Estendida*, que pode conter múltiplas *Partições Lógicas*. Endereça no máximo 2TB de espaço devido ao limite de 32 bits.

3.2 GPT (GUID PARTITION TABLE)

Introduzido com o UEFI para superar as limitações do MBR (ver Seção 3.1) (CORPORATION, 2010; FORUM, 2023).

- **Características:** Utiliza endereçamento de 64 bits (suporta até 9.4 ZB). Permite um número praticamente ilimitado de partições (padrão de 128 no Windows).
- **Segurança:** Possui backup da tabela de partições no final do disco e CRC32 para verificação de integridade.

Figura 4 – Diferença de capacidade entre MBR e GPT (valores ilustrativos).

A maior capacidade e redundância da GPT cooperam com recursos avançados de sistemas de arquivos modernos (Capítulo 4).

3.3 GERENCIAMENTO

Ferramentas como `fdisk`, `gdisk` e `parted` são usadas para criar partições. O comando `lsblk` lista dispositivos de bloco. Para uso, as partições devem ser formatadas e montadas (`mount`) em um diretório do sistema. O arquivo `/etc/fstab` controla as montagens automáticas no boot.

4 SISTEMA DE ARQUIVOS

O Sistema de Arquivos (*Filesystem*) é a estrutura lógica usada para organizar e armazenar dados nas partições. No Linux, o VFS (*Virtual Filesystem*) abstrai os diferentes tipos de sistemas para o Kernel (LOVE, 2010; TANENBAUM; BOS, 2015).

4.1 CONCEITOS BÁSICOS

- **Área de Controle vs. Dados:** Metadados (permissões, datas, localização) ficam na área de controle; o conteúdo real fica na área de dados.
- **Inode (Index Node):** Estrutura fundamental no Linux que armazena os metadados de um arquivo. Cada arquivo é identificado por um número de inode único na partição.
- **Blocos:** Unidade mínima de alocação (geralmente 4KB).

4.2 FAMÍLIAS DE SISTEMAS DE ARQUIVOS

- **Ext (Ext2, Ext3, Ext4):** Padrão no Linux. O Ext3 introduziu o *journaling* (TWEEDIE, 2000). O Ext4 suporta volumes de até 1 EB.
- **FAT (FAT16, FAT32, exFAT):** Comuns em pendrives e compatibilidade com Windows. Não suportam permissões POSIX nativamente e sofrem com fragmentação.
- **NTFS:** Padrão do Windows, suporta ACLs, compressão e *journaling*.
- **Sistemas Modernos (CoW):** BtrFS (RODEH; BACIK; MASON, 2013) e ZFS (BONWICK; MOORE, 2007) utilizam *Copy-on-Write*, snapshots e alta integridade; LFS (ROSENBLUM; OUSTERHOUT, 1991) explora escrita sequencial.

extbfFS	**Journaling**	**Copy-on-Write**	**Snapshots**	**Referência**
Ext3/Ext4	Sim	Não	Limitado (Ext4 via tooling)	(TWEEDIE, 2000)
BtrFS	Sim	Sim	Sim	(RODEH; BACIK; MASON, 2013)
ZFS	Sim	Sim	Sim	(BONWICK; MOORE, 2007)
LFS	Não	Não	Não	(ROSENBLUM; OUSTERHOUT, 1991)
FAT/exFAT	Não	Não	Não	(Especificação)
NTFS	Sim	Não	Shadow copies (Windows)	(Microsoft Docs)

Tabela 1 – Resumo de características de sistemas de arquivos.

A escolha do sistema de arquivos impacta recuperação e integridade (ver journaling (TWEEDIE, 2000)).

4.3 LINKS

- **Hard Link:** Um novo nome para o mesmo inode. Só funciona na mesma partição e não se aplica a diretórios. Se o original for apagado, o dado persiste.
- **Soft Link (Simbólico):** Um arquivo especial que aponta para o caminho de outro arquivo. Pode cruzar partições. Se o alvo for apagado, o link quebra.

Parte II

Permissões

5 GRUPOS, USUÁRIOS, PODERES E PERMISSÕES

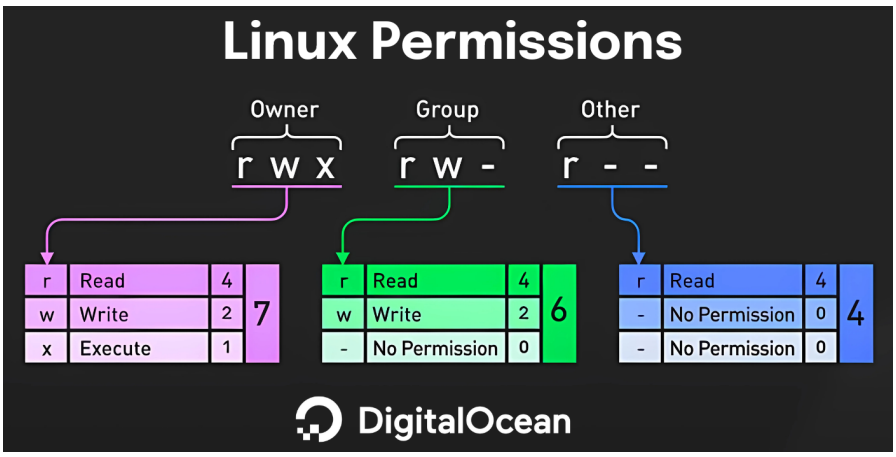


Figura 5 – Visão geral de usuários, grupos e permissões no Linux.

O Linux é um sistema multiusuário que controla o acesso através de UIDs (User IDs) e GIDs (Group IDs) (LOVE, 2010; TANENBAUM; BOS, 2015).

5.1 GERENCIAMENTO DE USUÁRIOS E GRUPOS

- **Arquivos de Configuração:**
  - `/etc/passwd`: Mapeia usuários para UIDs, shells e diretórios home.
  - `/etc/shadow`: Armazena as senhas criptografadas (hashes) de forma segura.
  - `/etc/group`: Define os grupos e seus membros.
- **Comandos:** `adduser`, `deluser`, `passwd`, `addgroup`.

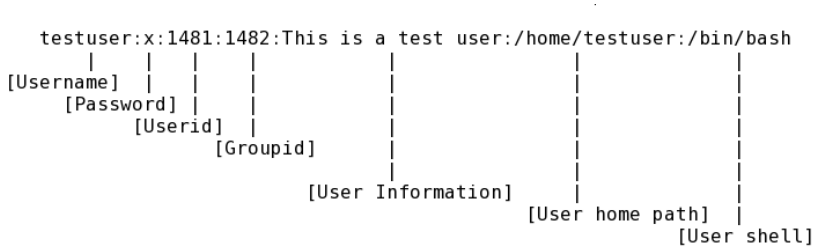


Figura 6 – Propriedades do usuário no Linux.

5.2 PERMISSÕES DE ARQUIVOS (MODO OCTAL E SIMBÓLICO)

Cada arquivo possui permissões para Dono (u), Grupo (g) e Outros (o) (SILBERSCHATZ; GALVIN; GAGNE, 2018). A estrutura de metadados de cada arquivo (inode) é detalhada no Capítulo 4.

- **Tipos:** Leitura (**r**=4), Escrita (**w**=2), Execução (**x**=1).
- **Comandos:**
  - `chmod`: Altera as permissões (ex: `chmod 755 arquivo`).[1]
  - `chown`: Altera o dono e o grupo (ex: `chown user:group arquivo`).[2]

5.3 PRIVILÉGIOS ESPECIAIS

O usuário `root` (UID 0) tem acesso total. O comando `sudo` (*SuperUser Do*) permite que usuários comuns executem comandos com privilégios elevados, configurados via `/etc/sudoers` (STALLINGS, 2018). Essa elevação é frequentemente necessária para operações de montagem de partições (ver Capítulo 3).

[1] Manual: <https://man7.org/linux/man-pages/man1/chmod.1.html>
[2] Manual: <https://man7.org/linux/man-pages/man1/chown.1.html>

6 GERENCIAMENTO DE PROCESSOS

Um processo é uma instância de um programa em execução. O gerenciamento de processos é central para o sistema operacional, permitindo multitarefa através de *time sharing* (TANENBAUM; BOS, 2015; SILBERSCHATZ; GALVIN; GAGNE, 2018). Processos essenciais são disparados já na fase de inicialização (Capítulo 1).

6.1 IDENTIFICAÇÃO E ESTRUTURA

- **PID:** Todo processo possui um identificador único (*Process ID*).
- **PCB (Process Control Block):** Estrutura de dados no Kernel que armazena o contexto do processo (registradores, estado, prioridade, contadores).
- **Contexto:** Hardware (registradores da CPU), Software (quotas, privilégios) e Endereçamento (memória alocada).

6.2 ESTADOS DO PROCESSO

Um processo transita entre estados:

1. **Execução (Running):** Está usando a CPU.
2. **Pronto (Ready):** Aguardando vez na CPU (escalonamento).
3. **Espera (Wait/Blocked):** Aguardando um evento (ex: E/S).

A troca entre processos é chamada de *Mudança de Contexto* (LOVE, 2010).

6.3 MONITORAMENTO

O sistema de arquivos virtual `/proc` expõe informações do Kernel sobre processos. Comandos como `top`, `htop`, `ps` e `uptime` (para ver média de carga/load

Estado	**Descrição resumida**	**Referência**
Running	Em execução na CPU	(TANENBAUM; BOS, 2015)
Ready	Apto, aguardando CPU	(SILBERSCHATZ; GALVIN; GAGNE, 2018)
Blocked	Esperando evento E/S	(STALLINGS, 2018)
Zombie	Finalizado, aguardando coleta	(LOVE, 2010)
Stopped	Suspenso por sinal	(LOVE, 2010)

Tabela 2 – Estados de processo e descrição.

average) são usados para monitoramento (LOVE, 2010). A interação com `/proc` complementa a análise de desempenho de discos (Capítulo 2).[1]

[1] Documentação: <https://man7.org/linux/man-pages/man5/proc.5.html>

7 GERENCIAMENTO DE MEMÓRIA PRIMÁRIA

O gerenciador de memória controla o acesso da CPU à memória principal (RAM), alocando e desalocando espaços para processos.

7.1 TIPOS DE MEMÓRIA

- **RAM (Random Access Memory):** Memória volátil de acesso rápido. Divide-se em DRAM (Dinâmica, precisa de refresh, usada na memória principal) e SRAM (Estática, mais rápida, usada em Cache L1/L2).

- **Cache:** Armazena dados usados frequentemente para reduzir o tempo de acesso à DRAM.

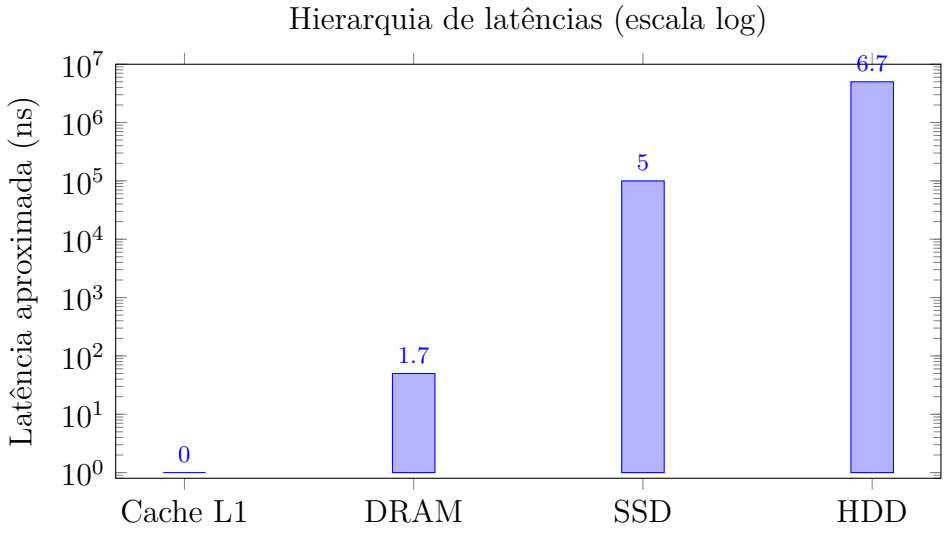


Figura 7 – Ordem de grandeza de latências de memória e armazenamento.

Valores ilustrativos baseados em ordens de grandeza.[1] Ver Capítulo 2 para detalhes físicos de SSD e HDD (PATTERSON; HENNESSY, 2014).

7.2 TÉCNICAS DE GERENCIAMENTO

- **Endereçamento Virtual:** Os processos usam endereços lógicos que são traduzidos para endereços físicos pela MMU (Memory Management Unit). Isso isola a memória de cada processo (TANENBAUM; BOS, 2015; SILBERSCHATZ; GALVIN; GAGNE, 2018).
- **Paginação:** Divide a memória em blocos de tamanho fixo chamados páginas (geralmente 4KB). Permite carregar partes do processo sob demanda e elimina fragmentação externa (STALLINGS, 2018).
- **Segmentação:** Divide a memória em segmentos de tamanhos variáveis baseados na lógica do programa (código, dados, pilha) (SILBERSCHATZ; GALVIN; GAGNE, 2018).
- **Swapping:** Quando a RAM está cheia, o sistema move processos inativos para uma área no disco (Swap), liberando memória principal. O acesso ao disco é significativamente mais lento que à RAM (PATTERSON; HENNESSY, 2014).

Permissão	Valor	Descrição	Referência
r	4	Leitura	(SILBERSCHATZ; GALVIN; GAGNE, 2018)
w	2	Escrita	(STALLINGS, 2018)
x	1	Execução	(LOVE, 2010)
rwx	7	Leitura + Escrita + Execução	(TANENBAUM; BOS, 2015)
rw-	6	Leitura + Escrita	(POSIX)
r-x	5	Leitura + Execução	(POSIX)
—	0	Sem acesso	(POSIX)

Tabela 3 – Mapa de permissões simbólicas e octais.

[1] Referência de latências: <https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html>

Parte III

Exercícios

8 ATIVIDADES - CAPÍTULO 6

8.1 ATIVIDADE 6.1

ENUNCIADO

Pesquise as opções do comando `ps` e crie um comando para listar apenas os identificadores e nomes dos processos que estão no estado **R** (running). Verifique também o comando `top`, uma versão interativa do comando `ps`, atualizando a listagem de processos a cada n segundos e ordenando-os por uso de CPU e memória. Consulte os estados de processo descritos em 6.2 e relacione diferenças de carga com operações de disco (Capítulo 2).

8.2 ATIVIDADE 6.2

ENUNCIADO

A partir do Shell, execute um programa em background (por exemplo, o navegador web). Observe a árvore hierárquica de processos e visualize as dependências destes processos. Observe ainda seu PID e PPID e compare aos do `bash`. Verifique também o comando `jobs`, que mostra a situação de todos os processos em background naquele Shell.

8.3 ATIVIDADE 6.3

ENUNCIADO

Utilize o daemon `cron` para realizar um backup daqui a cinco minutos dos arquivos `.log` da pasta `/var/log`. O arquivo de backup deve ser compactado e colocado dentro de sua pasta.

Utilize `crontab -e` para editar o arquivo e o comando `tar -zcvf <destino> <origem>` para fazer a compactação dos arquivos `.log`.

8.4 ATIVIDADE 6.4

ENUNCIADO

Crie um script que mostra o usuário atual, o diretório atual e o Shell em uso. Lembre que em `/etc/passwd` encontram-se listados os usuários e seus shells. Ver também gerenciamento de usuários na Seção 5.1.

Use o comando `cut` com a sintaxe:

```
1 cut -d DELIMITADOR -f NUMERO\_DO\_CAMPO
```

O comando `cut` corta somente o(s) campo(s) de número `NUMERO_DO_CAMPO` e utiliza o separador `DELIMITADOR` para delimitar cada campo. No caso do arquivo `passwd`, o delimitador é ":".

8.5 ATIVIDADE 6.8

ENUNCIADO

Crie um script que verifica o número de parâmetros recebidos na linha de comando. Verifique se existe apenas um parâmetro. Use o comando `test` para validar e `exit` para sair se o número for incorreto. Lembre-se de que o número de parâmetros está na variável `$#`. Relacione com o contexto de processo (Seção 6.1).

8.6 ATIVIDADE 6.9

ENUNCIADO

Estenda a atividade anterior para verificar se o parâmetro é um diretório. Exemplo de uso: `./testedir /home`. Use o teste `[[ -d $1 ]]` para verificar. A

variável `$?` indica status: zero sucesso; diferente de zero falha. Considere permissões de acesso (Seção 5.2).

8.7 ATIVIDADE 6.10

ENUNCIADO

Adapte a atividade anterior para receber dois parâmetros: um diretório e uma letra. Se o primeiro for diretório válido, liste todos os arquivos que começam com a letra informada (ex.: `ls $2*`). Impacto de listar muitos arquivos pode variar conforme latências de armazenamento (Figura 2).

9 ATIVIDADES - CAPÍTULO 7

9.1 ATIVIDADE 7.6

ENUNCIADO

Crie um arquivo chamado `agenda` contendo nomes e telefones (`nome sobrenome telefone`). Crie um script que recebe como parâmetro um nome e imprime as linhas do arquivo `agenda` que o contenham. Verifique se foi passado parâmetro.

```
1         #!/bin/bash
2         ??????????????????
3
4         then
5             ??????????????????
6             ??????????????????
7             ??????????????????
8
9             exit 1
10        else
11            ??????????????????
12            ??????????????????
13            then
14                echo 'Recuperando informacoes'
15                grep ?????????? agenda
16            fi
17        fi
18
```

9.2 ATIVIDADE 7.9

ENUNCIADO

1. Crie um script que recebe duas palavras como parâmetro e verifica se a primeira está contida na segunda.

2. Crie um script que recebe o nome de um arquivo texto como parâmetro, verifica se o usuário possui permissão de leitura deste arquivo e, caso tenha, exiba as seguintes informações sobre o arquivo: número de caracteres, número de palavras e número de linhas do arquivo (`utilize o comando wc`).

9.3 ATIVIDADE 7.11

ENUNCIADO

Crie um script que apresenta um menu:

1. Exibir status das partições (`df -h`)
2. Exibir usuários logados (`who`)
3. Exibir data/hora (`date`)
4. Sair

Leia a opção e execute o comando correspondente; valide entradas inválidas.

10 EXERCÍCIOS RESOLVIDOS

1 Roteiro de Atividades 6

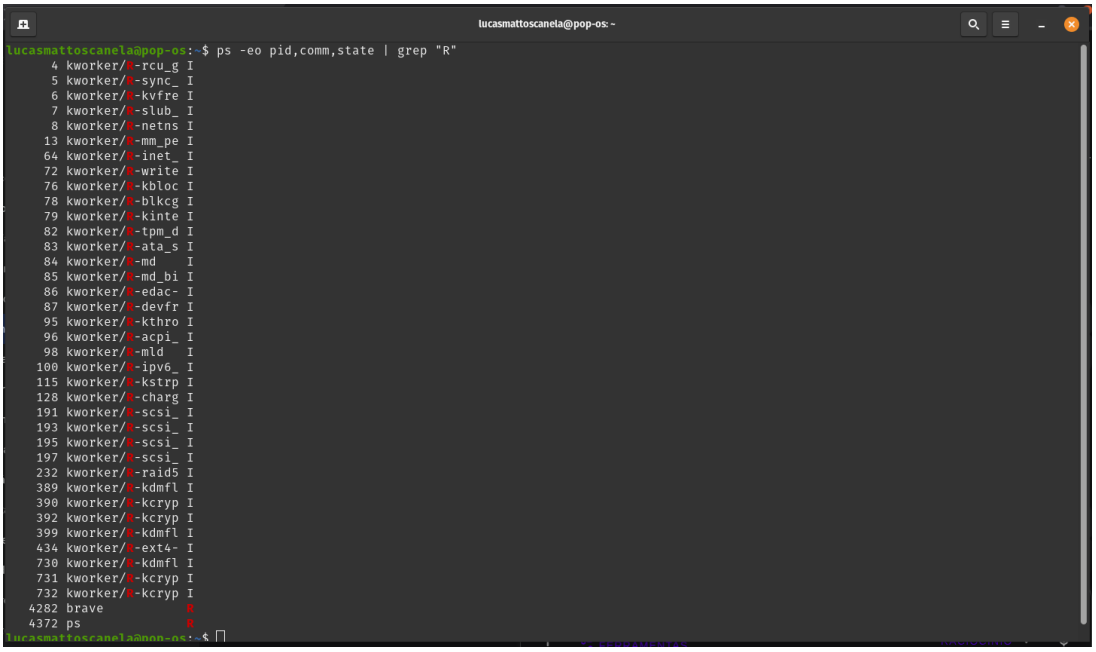
1.1 Atividade 6.1 - Processos em Estado Específico

Comando para listar apenas processos no estado Running (R).

Comando:

```
ps -eo pid,comm,state | grep "R"
```

Figura 1 – Execução do comando ps e grep



Fonte: O autor (2025)

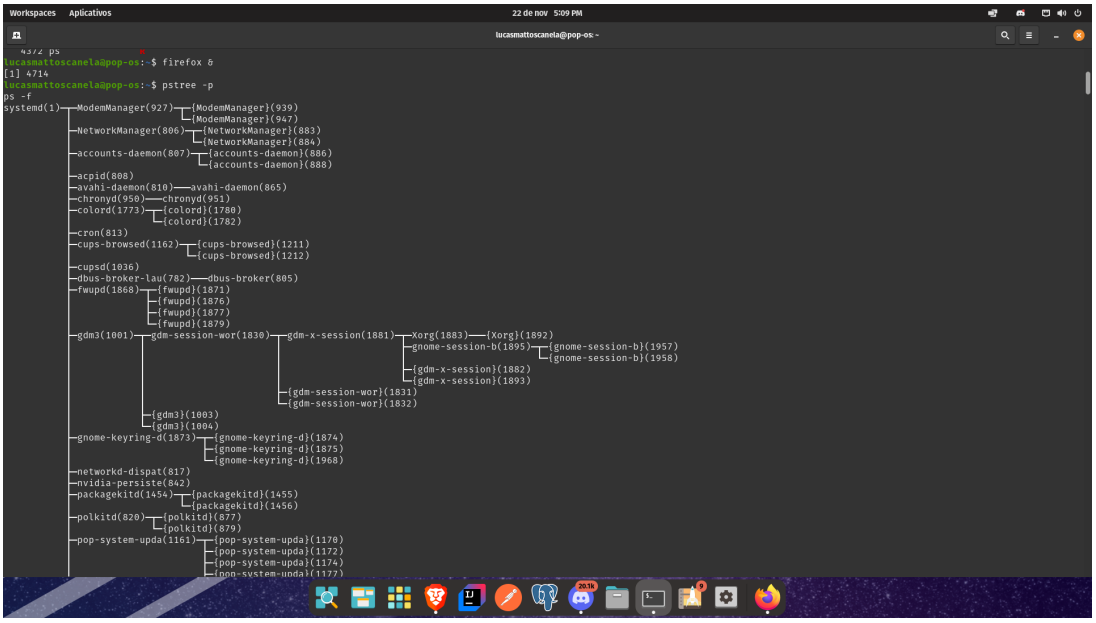
1.2 Atividade 6.2 - Processos em Background

Execução do navegador em background e análise da árvore de processos.

Comandos:

```
firefox &
pstree -p
jobs
```

Figura 2 – Visualização do firefox & e de pstree -p



Fonte: O autor (2025)

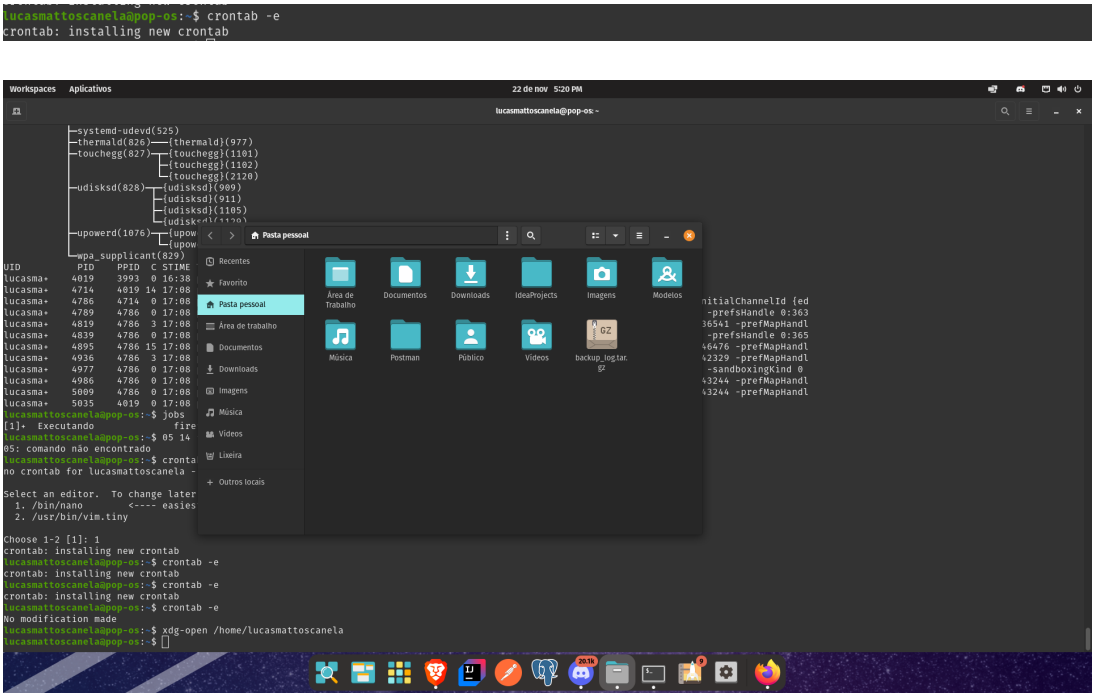
1.3 Atividade 6.3 - Daemon Cron (Backup)

Configuração do Crontab para backup agendado da pasta `/var/log`.

Linha do Crontab:

```
15 17 * * * tar -zcvf /home/lucasmattoscanela/backup_log.tar.gz /var/log/*.log
```

Figura 3 – Edição e verificação do Crontab



Fonte: O autor (2025)

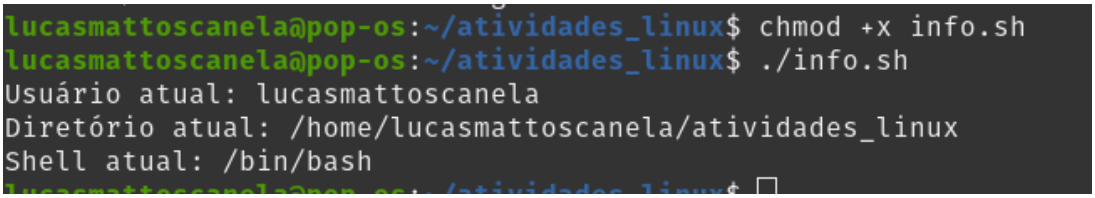
1.4 Atividade 6.4 - Script de Informações do Usuário

Script que exibe usuário, diretório atual e shell padrão.

Script (info.sh):

```
#!/bin/bash
echo "Usuário atual: $USER"
echo "Diretório atual: $(pwd)"
echo "Shell atual: $(grep "^$USER:" /etc/passwd | cut -d: -f7)"
```

Figura 4 – Execução do script info.sh



Fonte: O autor (2025)

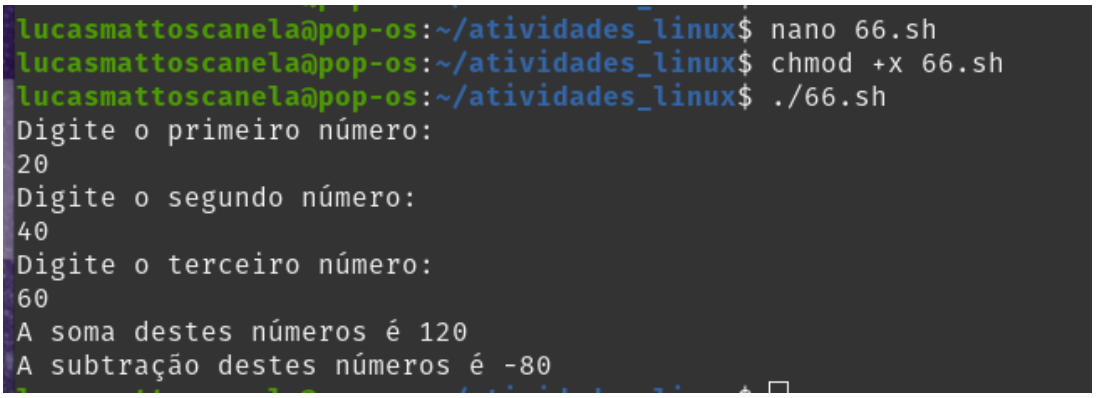
1.5 Atividade 6.6 - Leitura de Variáveis e Operações

Script que lê três números e exibe soma e subtração.

Script (calc.sh):

```
#!/bin/bash
echo "Digite o primeiro número:"
read n1
echo "Digite o segundo número:"
read n2
echo "Digite o terceiro número:"
read n3

soma=$(($n1 + $n2 + $n3))
sub=$(($n1 - $n2 - $n3))

echo "A soma destes números é $soma"
echo "A subtração destes números é $sub"
```

Figura 5 – Execução do script calc.sh



Fonte: O autor (2025)

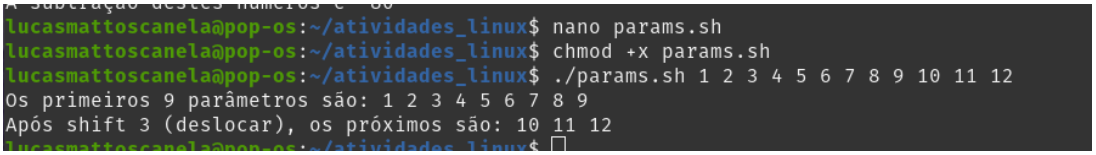
1.6 Atividade 6.7 - Parâmetros e Shift

Script que recebe 12 parâmetros e utiliza `shift` para deslocamento.

Script (params.sh):

```
#!/bin/bash
echo "Os primeiros 9 parâmetros são: $1 $2 $3 $4 $5 $6 $7 $8 $9"
shift 3
echo "Após shift 3, os próximos são: $7 $8 $9"
```

Figura 6 – Execução do script params.sh



Fonte: O autor (2025)

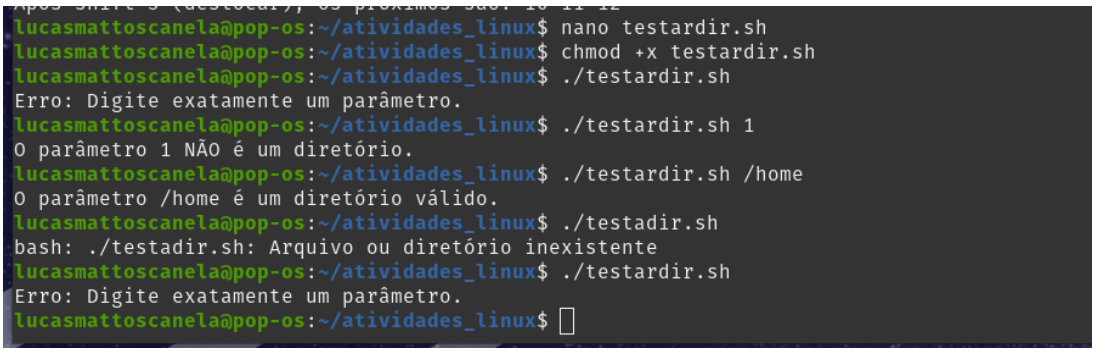
1.7 Atividades 6.8 e 6.9 - Testes de Parâmetros e Diretórios

Script que verifica número de parâmetros e existência de diretório.

Script (testadir.sh):

```
#!/bin/bash
if [ $# -ne 1 ]; then
    echo "Erro: Digite exatamente um parâmetro."
    exit 1
fi

if [[ -d "$1" ]]; then
    echo "O parâmetro $1 é um diretório válido."
else
    # Acentos como "Ã" agora funcionam graças ao bloco literate
    echo "O parâmetro $1 NÃO é um diretório."
fi
```

Figura 7 – Execução do script testadir.sh



Fonte: O autor (2025)

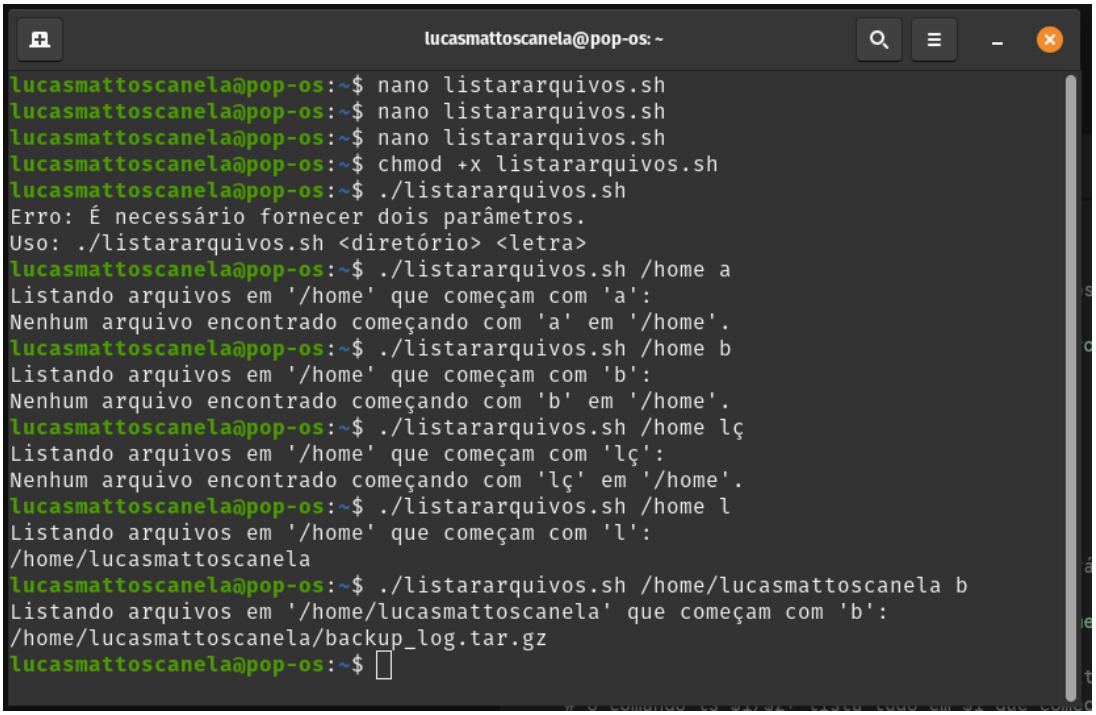
1.8 Atividade 6.10 - Listando arquivos passados por parâmetro

Script que recebe um diretório e uma letra como parâmetros, verificando se o primeiro parâmetro é um diretório válido e listando os arquivos que iniciam com a letra especificada.

Script (atividade6_10.sh):

```
#!/bin/bash

# Verifica se foram passados exatamente 2 parâmetros
if [ $# -ne 2 ]; then
    echo "Erro: É necessário fornecer dois parâmetros."
    echo "Uso: $0 <diretório> <letra>"
    exit 1
fi

diretorio=$1
letra=$2

# Verifica se o primeiro parâmetro é um diretório válido (-d)
if [[ -d "$diretorio" ]]; then
    echo "Listando arquivos em '$diretorio' que começam com '$letra':"

    # Lista os arquivos que começam com a letra fornecida
    ls -d "$diretorio/$letra"* 2>/dev/null

    # Verifica se o comando ls encontrou algo
    if [ $? -ne 0 ]; then
        echo "Nenhum arquivo encontrado com esse padrão."
    fi
else
    echo "Erro: O parâmetro '$diretorio' NÃO é um diretório válido."
fi
```

Figura 8 – Execução do script da Atividade 6.10



Fonte: O autor (2025)

2 Roteiro de Atividades 7

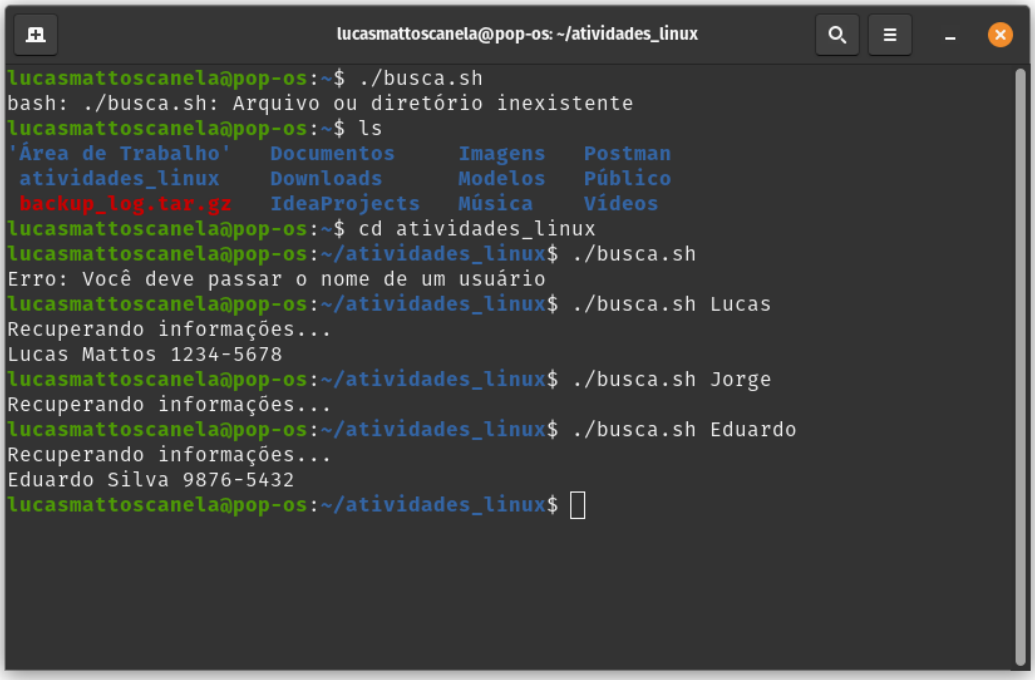
2.1 Atividade 7.6 - Uso de Grep na Agenda

Script que busca um nome em um arquivo de agenda.

Script (busca.sh):

```
#!/bin/bash
if [ -z "$1" ]; then
    echo "Erro: Você deve passar o nome de um usuário"
    exit 1
else
    echo "Recuperando informações..."
    grep "$1" agenda
fi
```

Figura 9 – Busca na agenda



Fonte: O autor (2025)

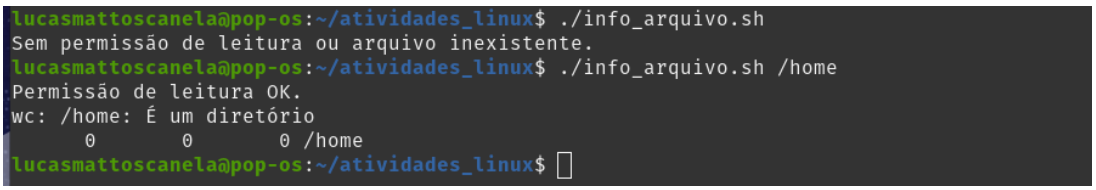
2.2 Atividade 7.9 - Expressões Regulares

Script que verifica permissão de leitura e conta linhas/palavras do arquivo.

Script (info_arquivo.sh):

```
#!/bin/bash
arquivo=$1
if [ -r "$arquivo" ]; then
    echo "Permissão de leitura OK."
    wc "$arquivo"
else
    echo "Sem permissão de leitura ou arquivo inexistente."
fi
```

Figura 10 – Verificação de arquivo



Fonte: O autor (2025)

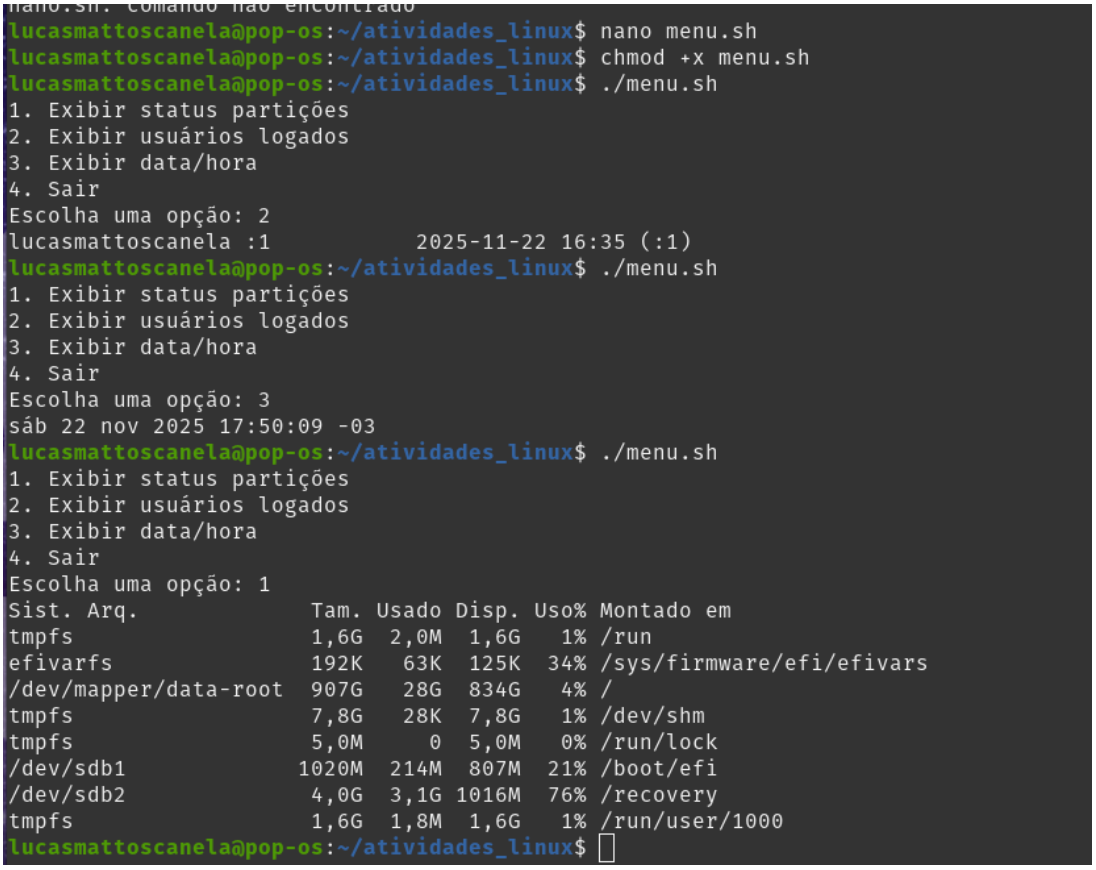
2.3 Atividade 7.11 - Menu de Opções

Menu interativo para exibir status, usuários ou data.

Script (menu.sh):

```
#!/bin/bash
echo "1. Exibir status partições"
echo "2. Exibir usuários logados"
echo "3. Exibir data/hora"
echo "4. Sair"
read -p "Escolha uma opção: " opcao

case $opcao in
    1) df -h ;;
    2) who ;;
    3) date ;;
    4) exit 0 ;;
    *) echo "Opção inválida" ;;
esac
```

Figura 11 – Execução do menu



Fonte: O autor (2025)

REFERÊNCIAS

BONWICK, J.; MOORE, B. *ZFS: The Last Word in File Systems*. [S.l.], 2007. White paper. Citado nas páginas 23 e 24.

CORPORATION, I. *GUID Partition Table (GPT) Disk Layout*. 2010. Technical documentation. Disponível em especificações EFI/UEFI. Citado nas páginas 21 e 22.

FORUM, U. *Unified Extensible Firmware Interface (UEFI) Specification*. [S.l.], 2023. Citado nas páginas 17 e 22.

LOVE, R. *Linux Kernel Development*. 3. ed. [S.l.]: Addison-Wesley Professional, 2010. Citado nas páginas 17, 18, 23, 27, 29, 30 e 32.

ORGANIZATION, N. E. *NVM Express Base Specification*. [S.l.], 2023. Citado na página 19.

PATTERSON, D. A.; HENNESSY, J. L. *Computer Organization and Design: The Hardware/Software Interface*. 5. ed. [S.l.]: Morgan Kaufmann, 2014. Citado nas páginas 17, 19 e 32.

RODEH, O.; BACIK, J.; MASON, C. Btrfs: The linux b-tree file system. *ACM Transactions on Storage*, v. 9, n. 3, p. 1–32, 2013. Citado nas páginas 23 e 24.

ROSENBLUM, M.; OUSTERHOUT, J. K. The design and implementation of a log-structured file system. In: *Proceedings of the 13th ACM Symposium on Operating Systems Principles (SOSP)*. [S.l.: s.n.], 1991. p. 1–15. Citado nas páginas 23 e 24.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Operating System Concepts*. 10. ed. [S.l.]: Wiley, 2018. Citado nas páginas 17, 28, 29, 30 e 32.

STALLINGS, W. *Operating Systems: Internals and Design Principles*. 9. ed. [S.l.]: Pearson, 2018. Citado nas páginas 28, 30 e 32.

TANENBAUM, A. S.; BOS, H. *Modern Operating Systems*. 4. ed. [S.l.]: Pearson, 2015. Citado nas páginas 17, 23, 27, 29, 30 e 32.

TWEEDIE, S. C. Ext3, journaling file system. In: *Proceedings of the Linux 2000 Expo*. London, UK: [s.n.], 2000. Citado nas páginas 23 e 24.

GLOSSÁRIO

abnTeX2 suíte para LaTeX que atende os requisitos dasnormas da ABNT para elaboração de documentos técnicos e científicos brasileiros. 41, *veja*LaTeX

BIOS firmware legado em ROM que realiza POST e inicia o boot. 41, *veja*boot

bloco unidade mínima de alocação em sistemas de arquivos. 41, *veja*inode

boot processo de inicialização que prepara o kernel para execução.

bootloader programa (ex: GRUB) que carrega e transfere controle ao kernel. 41, *veja*boot

cache memória rápida que armazena dados frequentes reduzindo latência. 41, *veja*SRAM

CHS Cylinder-Head-Sector: antigo esquema físico de endereçamento. 41, *veja*LBA

Copy-on-Write adiar cópia até modificação garantindo integridade. 41, *veja*inode

DRAM memória dinâmica principal que exige refresh. 41, *veja*SRAM

ESP EFI System Partition: partição FAT usada por carregadores UEFI. 41, *veja*UEFI

GID identificador de grupo agregando usuários. 41, *veja*UID

GPT GUID Partition Table: esquema com endereçamento 64 bits e redundância. 41, *veja*MBR

hard link nome adicional para o mesmo inode; mantém dados se o outro nome é removido.

HDD disco magnético com trilhas, setores e cilindros. 41, *veja*SSD

initramfs sistema de arquivos temporário usado na fase inicial do kernel. 41, *veja*boot

inode estrutura de metadados de arquivo (permissões, apontadores).

journaling registro de transações para recuperação consistente. 41, *veja*inode

LaTeX ferramenta de computador para autoria dedocumentos criada por D. E. Knuth.

LBA Logical Block Addressing: endereçamento linear de blocos. 41, *veja*CHS

link simbólico arquivo que referencia caminho alvo; quebra se alvo removido.

MBR Master Boot Record: setor inicial com código de boot e tabela simples de partições. 41, *veja*GPT

memória virtual abstração de endereços isolados por processo. 41, *veja*MMU

MMU unidade que traduz endereços virtuais em físicos. 41, *veja*paginação

NVMe protocolo otimizado para acesso a SSD via PCIe. 41, *veja*SSD

paginação divisão da memória em páginas fixas. 41, *veja*segmentação

PCB Process Control Block: contexto do processo. 41, *veja*PID

PID identificador único de processo. 41, *veja*PCB

root usuário administrativo com UID 0. 41, *veja*sudo

segmentação memória organizada em segmentos lógicos variáveis. 41, *veja*paginação

SRAM memória estática rápida usada em cache. 41, *veja*DRAM

SSD unidade de estado sólido baseada em memória flash. 41, *veja*NVMe

sudo mecanismo para executar comandos com privilégios elevados. 41, *veja*root

swapping movimentação de páginas/processos inativos para disco.

troca de contexto alternância da CPU entre processos salvando/restaurando estado. 41, *veja*PID

UEFI padrão moderno de firmware com NVRAM e partição EFI. 41, *veja*boot

UID identificador único de usuário. 41, *veja*GID