



**UNIVERSIDADE FEDERAL DO MARANHÃO  
BACHARELADO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA**

<b>Disciplina:</b> Sistemas Operacionais	<b>Profa.</b> Alana Oliveira
<b>Nome do aluno:</b>	Juan Pablo Furtado Mondego
<b>Sistema Operacional escolhido:</b>	Ubuntu 24.04.2 LTS
<b>Data de envio:</b>	23/12/25

**Estudo Analítico de um Sistema Operacional Real:  
UBUNTU 24.04.2 LTS**

## 1 Gerência de Processos

- Como o sistema cria e gerencia processos?

No Sistema Operacional Ubuntu o processo é uma instância de um programa (assim como eu muitos sistemas operacionais). Sempre que essa instância de um programa é criada ela carrega consigo um contexto de hardware, contexto de software e espaço de endereçamento que são carregados pelo sistema na memória e organizados pelo bloco de controle de processos (PCB).

Main	I/O	PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
2601	eryk	20	0	3768M	303M	135M	S	20.1	4.0	0:31.77	/usr/bin/gnom		
805	avahi	20	0	14496	10296	4024	S	14.9	0.1	4:31.33	avahi-daemon:		
6303	eryk	20	0	2795M	101M	83692	S	11.0	1.3	0:00.17	/usr/bin/gjs-		
6306	eryk	20	0	707M	84836	69860	S	7.1	1.1	0:00.11	/usr/bin/gnom		
6307	eryk	20	0	422M	28924	23036	R	7.1	0.4	0:00.11	/usr/bin/seah		
6016	eryk	20	0	9464	5864	3816	S	4.5	0.1	0:04.93	htop		
842	root	20	0	327M	21236	17140	S	1.3	0.3	0:13.55	/usr/sbin/Net		
2402	eryk	20	0	11048	6716	4668	S	1.3	0.1	0:00.57	/usr/bin/dbus		
2624	eryk	-21	0	3768M	303M	135M	S	1.3	4.0	0:03.86	/usr/bin/gnom		
6336	eryk	20	0	707M	84836	69860	R	1.3	1.1	0:00.02	/usr/bin/gnom		
2411	eryk	20	0	306M	10460	9436	S	0.6	0.1	0:00.04	/usr/bin/gnom		
2436	eryk	20	0	738M	7856	7088	S	0.6	0.1	0:00.03	/usr/libexec/		
2615	eryk	20	0	9480	5232	4720	S	0.6	0.1	0:00.02	/usr/bin/dbus		
2747	eryk	20	0	377M	12768	7540	S	0.6	4.0	0:00.96	/usr/bin/gnom		
2784	eryk	20	0	377M	12768	7540	S	0.6	0.2	0:01.41	/usr/bin/ibus		
3255	eryk	20	0	2663M	55292	41140	S	0.6	0.7	0:00.60	gjs /usr/share		
4250	eryk	20	0	692M	62728	49312	S	0.6	0.8	0:01.55	/usr/libexec/		
5667	eryk	20	0	1462M	127M	71160	S	0.6	1.7	0:02.27	/usr/bin/rhyt		
5669	eryk	20	0	1462M	127M	71160	S	0.6	1.7	0:04.07	/usr/bin/rhyt		
6129	eryk	20	0	1079M	99.2M	80264	S	0.6	1.3	0:00.29	/usr/bin/naut		
6302	eryk	20	0	512M	17736	15816	S	0.6	0.2	0:00.01	/usr/libexec/		

Figura 1 - Tabela PCB do linux pelo programa htop. Fonte: Autor.

Na figura 1 podemos identificar através do programa htop executado pelo terminal do GNOME do Ubuntu alguns programas sendo executados. É possível ver que cada um deles mantém informações importantes sobre a execução dos processos, com destaque para PID (Process identification ou Identificação do processo), PRI (Priority ou prioridade), CPU% e MEM% (Informações sobre o consumo da memória principal e da CPU) e TIME+ (tempo de execução).

Seguindo o passo a passo para a criação de um processo no sistema operacional Ubuntu assim como na maioria do sistemas baseados em linux temos a seguinte sequência de eventos:

1. Execução da chamada de sistema fork():

Assim que um programa é aberto o sistema executa a função fork que cria um clone exatamente igual do programa que está sendo executado, porém com um PID diferentes, pois é tratado como uma instância separada.

```
[11:26] ~ Bash
└─> ps -ef | grep bash
eryk     3991  2601  0 10:51 ?    00:00:00 /bin/bash /usr/bin/brave-browser-stable
eryk     4259  4250  0 10:52 pts/0  00:00:00 bash
eryk     7071  4259  0 11:26 pts/0  00:00:00 grep --color=auto bash
```

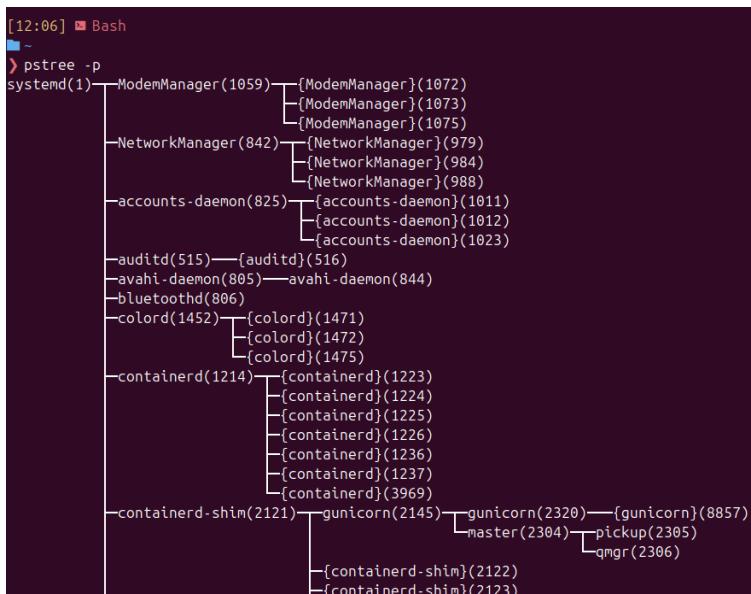
Figura 2 - Processos relacionados ao shell bash. Fonte: Autor.

Na figura 2 por exemplo podemos ver uma listagem das instâncias clonadas do Shell bash e que também estão sendo utilizadas por outro programas, o que inclui o próprio terminal GNOME para a interação com o cliente. Todos eles foram clonados diretamente dos arquivos raiz do Shell, porém se comportam como processos independentes, tendo PIDs diferentes uns dos outros. Essa instância maior de onde todas as outras são clonadas é chamada de processo pai, enquanto as derivadas desta são nomeadas processos filhos.

## 2. Usa o Init ou systemd como “pai de todos”:

Como um destaque desta associação de heranças existentes no sistemas linux temos o processo inicial do sistema, o que é ativado na inicialização e dá o pontapé inicial para serviços e daemons do sistema, gerenciamento de processos do usuário e do sistema e controla sessões, logs, estados de parada, etc. Trata-se do processo que gerencia as rotinas do sistema e é também o nomeado com o PID 1. Este também é o responsável por “adotar” os processos filhos quando um processo pai quebra, para que esses processos continuem em execução.

Figura 3 - Processo systemd. Fonte: Autor.



Como exemplo podemos ver na figura acima pelo comando “ps -p 1 -o pid, comm, etime” no terminal GNOME do Ubuntu que o processo associado ao PID 1 é o próprio systemd. O tempo na informação “ELAPSED” é o tempo de execução do sistema desde a sua inicialização.

Figura 4 - Árvore de processos do linux. Fonte: Autor.

Depois que os processos já estão criados, eles são armazenados em uma estrutura de dados conhecida como árvore rubro-negra (uma estrutura de dados no estilo árvore que faz um auto balanceamento com base em dois tipos diferentes de nós que são denominados vermelho e preto). Essa estrutura foi escolhida para organizar os processos devido a sua eficiência tanto na busca quanto na inserção e remoção de itens, sendo superior a AVL (Adelson-Velsky e Landis - árvore binária de busca auto balanceada).

O Ubuntu gerencia a execução de processos através do escalonador CFS, que atribui intervalos de tempo virtuais entre 4 e 10ms para garantir uma distribuição justa e equilibrada dos recursos da CPU. Esse sistema multitarefa atua de forma preemptiva, interrompendo automaticamente processos que esgotam sua cota temporal para alternar para aqueles com menor tempo de uso acumulado. Essa dinâmica impede o monopólio do processador por uma única tarefa e otimiza a eficiência do sistema, especialmente em arquiteturas com múltiplos núcleos.

Figura 5 - Ferramenta de visualização de processos via terminal top. Fonte: Autor.

- **Há suporte a threads, multitarefa ou paralelismo?**

O gerenciamento de processos no Ubuntu é orquestrado pelo escalonador CFS (*Completely Fair Scheduler*), que utiliza intervalos de tempo virtuais (*quantums*) de 4 a

```
top - 14:59:42 up 4:12, 1 user, load average: 0,98, 0,94, 0,96
Tasks: 250 total, 1 em exec, 249 dormindo, 0 parado, 0 zumbi
%CPU(s): 5,9 us, 2,2 sy, 0,0 id, 91,0 ld, 0,7 wa, 0,0 hi, 0,2 si, 0,0 st
MB mem : 7665,4 total, 664,9 free, 3627,6 used, 4215,0 buff/cache
MB swap: 12288,0 total, 12287,7 free, 0,2 used, 4037,8 avail mem

 PID USUARIO PR NI VIRT RES SHR S %CPU %MEM TEMPO+ COMANDO
 2601 eryk 20 0 3951948 329992 140236 S 8,3 4,2 4:13:42 gnome-shell
 805 avahi 20 0 14640 10424 4024 S 3,0 0,1 21:31:50 avahi-daemon
 5649 eryk 20 0 1590660 134648 71416 S 1,3 1,7 1:37:08 rythmbox
 160 root -51 0 0 0 0 S 1,0 0,0 1:08:30 irq/133-ELAN0B00:00
 2386 eryk 20 0 405980 20864 15488 S 0,7 0,3 0:45:13 wireplumber
 4000 eryk 20 0 33,0g 509668 275000 S 0,7 6,5 4:45:81 brave
 4049 eryk 20 0 32,6g 190424 155184 S 0,7 2,4 1:28:70 brave
 5509 eryk 20 0 1410,2g 357968 155052 S 0,7 4,6 7:12:07 brave
10848 root 0 -20 0 0 0 D 0,7 0,0 0:01:20 kworker/u9:0+i915_flip
11058 root 20 0 0 0 I 0,7 0,0 0:01:16 kworker/0:1-events
2388 eryk 20 0 125516 29292 10352 S 0,3 0,4 0:26:75 pipewire-pulse
11137 root 20 0 0 0 I 0,3 0,0 0:00:74 kworker/1:0-i915-unordered
11439 eryk 20 0 12128 6140 3964 R 0,3 0,1 0:00:20 top
 1 root 20 0 23464 14752 9632 S 0,0 0,2 0:04:29 systemd
 2 root 20 0 0 0 S 0,0 0,0 0:00:00 kthreadd
 3 root 20 0 0 0 S 0,0 0,0 0:00:00 pool_workqueue_release
 4 root 0 -20 0 0 I 0,0 0,0 0:00:00 kworker/R-rcu_gp
 5 root 0 -20 0 0 I 0,0 0,0 0:00:00 kworker/R-sync_wq
 6 root 0 -20 0 0 I 0,0 0,0 0:00:00 kworker/R-stub_flushwq
 7 root 0 -20 0 0 I 0,0 0,0 0:00:00 kworker/R-netns
 9 root 0 -20 0 0 I 0,0 0,0 0:00:00 kworker/0:0H-events_highpri
12 root 0 -20 0 0 I 0,0 0,0 0:00:00 kworker/R-mm_percpu_wq
13 root 20 0 0 0 I 0,0 0,0 0:00:00 rcu_tasks_kthread
14 root 20 0 0 0 I 0,0 0,0 0:00:00 rcu_tasks_rude_kthread
```

10ms para distribuir o uso da CPU de forma justa e equilibrada. Como um sistema multitarefa preemptivo, o kernel interrompe automaticamente tarefas que esgotam sua cota temporal, priorizando aquelas com menor tempo de execução acumulado (*vruntime*) para garantir fluidez e eficiência em múltiplos núcleos. O monitoramento dessa dinâmica é realizado via terminal com o comando top, que permite visualizar o consumo de recursos e identificar estados críticos, como processos "zumbis" aguardando finalização. Para situações que exigem desempenho diferenciado, o sistema utiliza valores *Nice* (de

-20 a +19) e políticas de tempo real, assegurando que rotinas prioritárias tenham precedência absoluta sobre a execução padrão, ajustáveis manualmente por ferramentas como renice e chrt.

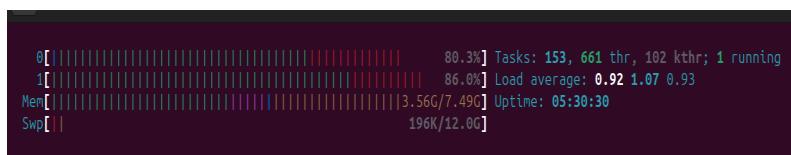
Além dessas ferramentas, ainda é possível escalar o desempenho do sistema ao adicionar mais núcleos na CPU. Isso garante a afirmação de que o sistema tem paralelismo em tempo real, uma vez que é capaz de administrar vários processos ou threads funcionando simultaneamente em núcleos separados. Ou seja, se o computador em questão tiver 2 ou mais núcleos, o sistema operacional ubuntu pode se aproveitar desses recursos para processar várias tarefas ao mesmo tempo.

Figura 7 - Informações da CPU do autor. Fonte: Autor.

```
[15:35] ~ Bash
lscpu
Arquitetura:          x86_64
Modo(s) operacional da CPU: 32-bit, 64-bit
Address sizes:        39 bits physical, 48 bits virtual
Ordem dos bytes:      Little Endian
CPU(s):
  Lista de CPU(s) on-line: 0,1
ID de fornecedor:
  Nome do modelo:        GenuineIntel
  Família da CPU:       Intel(R) Celeron(R) 6305 @ 1.80GHz
  Modelo:                140
  Thread(s) per núcleo:  1
  Núcleo(s) por soquete: 2
  Soquete(s):           1
  Step:                 1
  CPU(s) scaling MHz:   76%
  CPU MHz máx.:         1800,0000
  CPU MHz min.:         400,0000
  BogoMIPS:              3609,60
```

Figura 8 - Barras de utilização da CPU via terminal htop. Fonte: Autor.

Nas figuras 7 e 8 podemos observar que o sistema Ubuntu identificou a presença de mais de um núcleo na CPU instalada e está a dividir as cargas de processamento entre os dois núcleos.



- **Como o usuário ou o administrador pode visualizar, controlar ou encerrar processos?**

Usuários e administradores no Linux Ubuntu tem diversas formas de interagir com os processos do sistema. Até aqui já foram citadas diversas formas de interação e visualização dos processos como as ferramentas e comandos de terminal ps, top, htop, etc. ou até mesmo formas de alterar a prioridade e execução de programas. Além dessas temos também a própria interface gráfica

do sistema que se apresenta de forma bem completa na versão desktop, contendo até mesmo aplicativos de monitoramento, manutenção e manipulação do sistema de forma gráfica nas versões mais atuais como o Ubuntu 24.04.2 LTS (tais como Monitor do sistema, Atualizador de programas, etc.) que substituem as tradicionais ferramentas de linha de comando apresentadas até aqui.

Figura 9 - Monitor do Sistema Ubuntu. Fonte: Autor.

Figura 10 - Ferramenta de gerenciamento de armazenamento do sistema Ubuntu. Fonte: Autor.

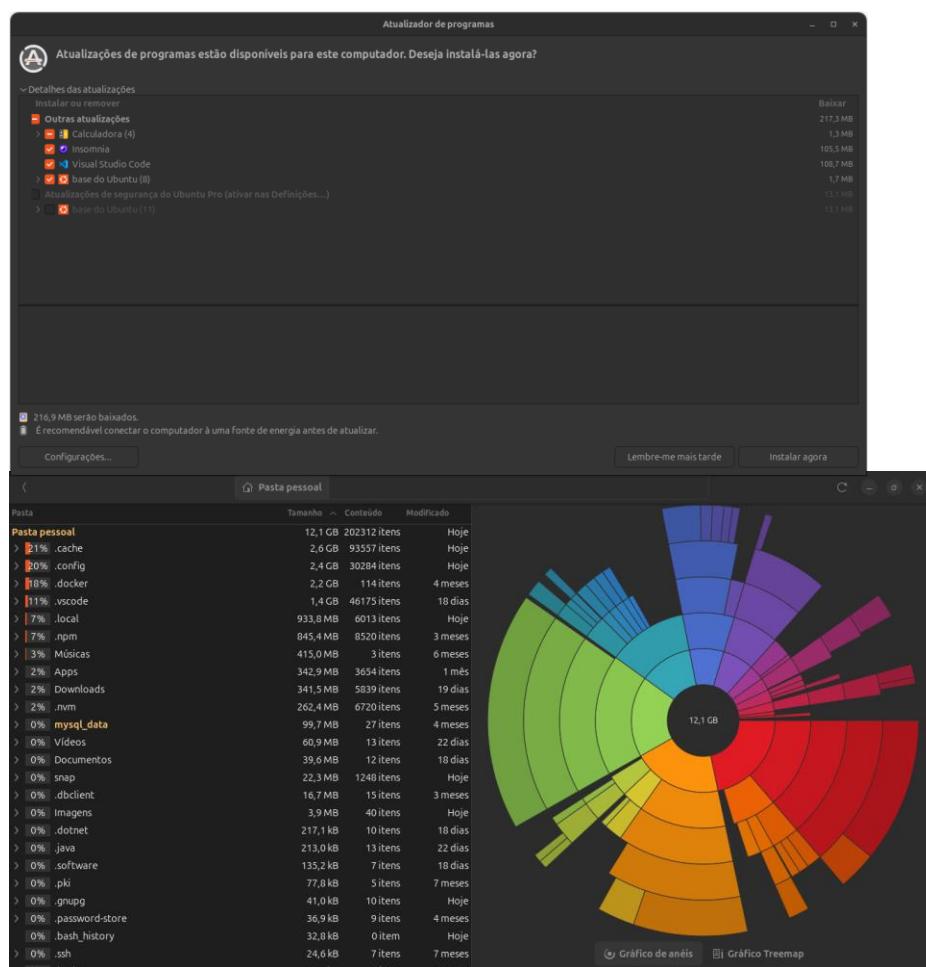


Figura 11 - Atualizador de programas do Ubuntu. Fonte: Autor.

O sistema ainda conta com uma diferenciação de usuários entre administradores e usuários comuns, sendo que os usuários comuns podem interagir apenas com os seus próprios programas, enquanto o administrador pode interagir com os programas de outros usuários e também sobre processos do próprio sistema operacional.

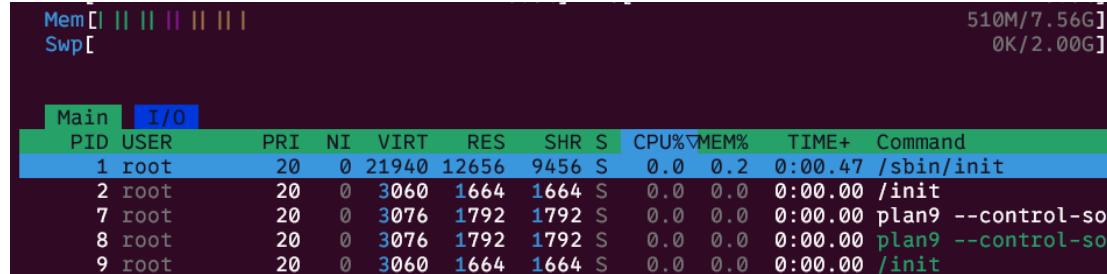
## 2 Gerência de Memória

- O sistema usa memória virtual?

O Ubuntu utiliza memória virtual através da técnica de Swap integrada à paginação por demanda, reservando uma área fixa no armazenamento secundário para alocar fragmentos de programas que não caberiam inteiramente na memória principal. Essa estratégia divide o conteúdo dos softwares em páginas que são carregadas para a RAM apenas quando solicitadas, o que amplia significativamente a capacidade de multitarefa e multiprogramação do sistema operacional. No entanto, o uso excessivo desse recurso pode introduzir latência devido à velocidade inferior de leitura e escrita dos discos em comparação à memória física, além de gerar sobrecarga de gerenciamento para o kernel.

Para mitigar esses gargalos e evitar o excesso de *page faults*, o sistema emprega algoritmos inteligentes de substituição que otimizam o trânsito de dados entre o disco e a memória ativa, garantindo estabilidade mesmo sob alta demanda.

e Swap do sistema Ubuntu. Fonte: Autor.



The screenshot shows the htop command output on a terminal window. At the top, there's a header with memory status: 'Mem [ 510M / 7.56G ]' and 'Swp [ 0K / 2.00G ]'. Below the header is a table with columns: Main, I/O, PID, USER, PRI, NI, VIRT, RES, SHR, S, CPU%, MEM%, TIME+, and Command. The table lists several processes:

Main	I/O	PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
		1	root	20	0	21940	12656	9456	S	0.0	0.2	0:00.47	/sbin/init
		2	root	20	0	3060	1664	1664	S	0.0	0.0	0:00.00	/init
		7	root	20	0	3076	1792	1792	S	0.0	0.0	0:00.00	plan9 --control-soc
		8	root	20	0	3076	1792	1792	S	0.0	0.0	0:00.00	plan9 --control-soc
		9	root	20	0	3060	1664	1664	S	0.0	0.0	0:00.00	/init

Figura 13 - Tabela com lista de processos do sistema Ubuntu via programa de linha de comando htop. Fonte: Autor.

Nas figuras 12 e 13 podemos observar algumas ferramentas de linha de comando via terminal GNOME no Ubuntu, como “free -h” e “htop”, que expõem dados de tamanho e uso das memórias principal e Swap. Na figura 12, mostra que o sistema tem aproximadamente 8 Gb de memória principal e 2 Gb de memória virtual ou Swap não utilizados no momento dos testes. Na figura 13 temos a ferramenta de visualização de processos htop que mostra o uso de cada processo em relação a memória principal (coluna RES) e na memória virtual Swap (coluna VIRT).

### 3 Gerência de Arquivos

- Qual é o sistema de arquivos utilizado (ex: NTFS, ext4, Btrfs...)?

No Ubuntu 24.04.2 LTS, o sistema de arquivos adotado por padrão é o ext4, uma evolução estável do ext3 que equilibra desempenho, confiabilidade e compatibilidade com versões anteriores. Essa escolha se justifica pelo amplo suporte oferecido pelas ferramentas de baixo nível (como mkfs.ext4 e e2fsck), pela maturidade do código e pela capacidade de atender à maioria dos cenários de uso quotidiano sem sacrificar a integridade dos dados.

Entretanto, durante o processo de instalação, o instalador gráfico do Ubuntu passa a oferecer também, como opção experimental, a criação de raiz em OpenZFS — incluindo suporte a criptografia de dados e snapshots — além da alternativa tradicional de partição LVM criptografada sobre ext4 . Usuários avançados podem ainda configurar manualmente Btrfs, XFS ou outros sistemas de arquivos, de acordo com necessidades específicas de snapshots, compressão ou escalabilidade.

# Exemplo de comando para exibir o tipo de sistema de arquivos de cada partição

```
ls@JuanPablo484806:~$ df -T -x tmpfs -x devtmpfs
Filesystem      Type      1K-blocks      Used Available Use% Mounted on
none            overlay        1924712       0   1924712    0% /usr/lib/modules/6.6.87.2-microsoft-standard-WSL2
drivers          9p        498406724 219991728 278414996  45% /usr/lib/wsl/drivers
/dev/sdd         ext4       1055762868 3237412 998821984  1% /
none            overlay        1924712       0   1924712    0% /usr/lib/wsl/lib
rootfs           rootfs       1919700     2664 1917036    1% /init
none            overlay        1924712       76   1924636    1% /mnt/wslg/versions.txt
none            overlay        1924712       76   1924636    1% /mnt/wslg/doc
C:\             9p        498406724 219991728 278414996  45% /mnt/c
```

Figura 14 – Saída de df -T destacando “ext4” como sistema de arquivos montado em /.  
Fonte: Autor.

Esse cenário demonstra que o Ubuntu mantém o ext4 como solução conservadora e testada em produção, garantindo baixo risco em ambientes corporativos e domésticos, ao mesmo tempo em que expande seu leque de opções para atender a exigências de segurança (criptografia) e gestão avançada de dados (ZFS).

Cada um desses diretórios pode conter sub-diretórios e arquivos conforme a função que desempenha, formando uma estrutura em árvore cujos nós intermediários são pastas e as folhas são arquivos ou links simbólicos.

# Exemplo: visualizar o primeiro nível da hierarquia de diretórios

```
Ls@JuanPablo484806:~$ tree -L 1 /
/
├── bin    -> usr/bin
├── bin.usr-is-merged
├── boot
├── dev
├── etc
├── home
├── init
├── lib   -> usr/lib
├── lib.usr-is-merged
├── lib64 -> usr/lib64
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── sbin.usr-is-merged
├── snap
├── srv
└── sys
└── tmp
└── usr
└── var
```

Figura 15 – Visão geral da árvore de diretórios de / no Ubuntu 24.04.2 LTS. Fonte: Autor.

Observou-se que o utilitário `tree` foi instalado via Snap, ficando disponível em `/snap/bin/tree`. Para invocá-lo, não é necessário especificar o caminho completo se `/snap/bin` já estiver incluído na variável de ambiente `$PATH`; caso contrário, basta executar diretamente o caminho absoluto. A saída gerada pelo comando (`listando apenas o primeiro nível da hierarquia de /`) foi exibida diretamente no terminal e utilizada para fins de análise e documentação da estrutura do sistema de arquivos.

Esse modelo hierárquico elimina a rigidez de múltiplos volumes acessados por letras distintas (como em Windows), permitindo montagens dinâmicas (por exemplo, dispositivos USB em `/media` ou sistemas de arquivos de rede em `/mnt`) sem alterar a lógica de caminho absoluto usada por aplicações e usuários.

- **Há suporte a permissões por usuário/grupo?**

O Ubuntu estrutura seu controle de acesso no robusto modelo POSIX, segmentando permissões de leitura, escrita e execução entre proprietário, grupo e outros, com configurações iniciais definidas pelo `umask`. Para superar as limitações desse esquema clássico, o sistema de arquivos ext4 suporta nativamente Listas de Controle de Acesso (ACLs), permitindo a atribuição granular de direitos a usuários ou grupos específicos sem alterar as regras gerais. Essa funcionalidade possibilita a definição de heranças de permissões em diretórios e ajustes finos de segurança, essenciais para ambientes multiusuário complexos. A gestão dessas permissões estendidas é realizada através de ferramentas como o `getfacl`, cuja instalação via pacote `acl` pode ser necessária para verificar as regras aplicadas além do padrão.

## # Exemplo: verificar permissões POSIX e existência de ACLs em /home

```
l@JianPablo:~$ sudo apt update && sudo apt install acl
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1391 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1684 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [225 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.5 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [9504 B]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [916 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [207 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [71.5 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [19.4 kB]
Get:14 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [2286 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en [311 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:17 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [523 kB]
Get:18 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:19 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:20 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.8 kB]
Get:21 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1596 kB]
Get:22 http://archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [306 kB]
Get:23 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [378 kB]
Get:24 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.4 kB]
Get:25 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2413 kB]
Get:26 http://archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [550 kB]
Get:27 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:28 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 c-n-f Metadata [516 B]
Get:29 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [30.3 kB]
Get:30 http://archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [6608 B]
Get:31 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:32 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Metadata [488 B]
Get:33 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [40.4 kB]
Get:34 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7288 B]
Get:35 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [368 B]
Get:36 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [29.5 kB]
```

Figura 16 – Saída de `ls -ld` mostrando “drwxr-xr-x” (permissões POSIX) e de `getfacl` com entradas estendidas de ACL. Fonte: Autor.

Dessa forma, o Ubuntu combina a simplicidade e eficiência do modelo POSIX com a flexibilidade das ACLs, assegurando controle granular de acesso em ambientes multiusuário sem sacrificar a compatibilidade com ferramentas e scripts legados.

### • Quais operações básicas são disponíveis?

Assim como em qualquer distribuição Linux, as operações básicas de gerenciamento de arquivos podem ser agrupadas segundo o ciclo CRUD – criação, leitura, atualização e remoção – e são realizadas predominantemente por meio de utilitários de linha de comando ou, opcionalmente, pela interface gráfica Nautilus (Explorador de Arquivos).

Para a criação de arquivos e diretórios, usam-se touch e mkdir, respectivamente; a leitura é feita com cat, less ou ls; a atualização (movimentação/renomeação) com mv; e a remoção com rm (ou rmdir para diretórios vazios). Além disso, o comando cp duplica arquivos e diretórios, enquanto find e locate auxiliam na localização de conteúdo em todo o sistema. Todas essas ferramentas dispõem de opções de linha de comando (por exemplo, -r para recursividade, -i para confirmação interativa e -v para modo verbose) que atendem a cenários desde *scripts automatizados* até *uso interativo cuidadoso*.

## # Exemplos de operações básicas

```

ls@JuanPablo484806:~$ # 1. Preparar o ambiente (criar arquivos que faltam)
mkdir -p backup entrega
touch documento.pdf relatório.docx arquivo_antigo.tmp

# 2. Executar as operações da imagem
touch novo_arquivo.txt
mkdir -p projeto_exemplo
cp documento.pdf backup/documento.pdf
mv relatório.docx entrega/relatorio.docx
rm -i arquivo_antigo.tmp # Responda 'y' se pedir confirmação
ls -lah
rm: remove regular empty file 'arquivo_antigo.tmp'?
total 72K
drwxr-x--- 13 ls  ls  4.0K Dec 24 11:52 .
drwxr-xr-x  3 root root 4.0K Nov 24 18:19 ..
-rw-r--r--  1 ls  ls  841 Nov 26 17:04 .bash_history
-rw-r--r--  1 ls  ls  220 Nov 24 18:19 .bash_logout
-rw-r--r--  1 ls  ls  3.7K Nov 24 18:19 .bashrc
drwxr-x---  4 ls  ls  4.0K Nov 26 15:05 cache
drwxr-xr-x  3 ls  ls  4.0K Nov 26 15:05 dotnet
drwxr-xr-x  3 ls  ls  4.0K Nov 26 15:59 ipython
drwxr-xr-x  2 ls  ls  4.0K Nov 26 15:01 landscape
-rw-r--r--  1 ls  ls  0 Dec 24 11:26 .motd_shown
-rw-r--r--  1 ls  ls  807 Nov 24 18:19 .profile
-rw-r--r--  1 ls  ls  0 Nov 24 18:51 .sudo_as_admin_successful
drwxr-xr-x  4 ls  ls  4.0K Nov 26 15:03 vscode-remote-containers
drwxr-xr-x  5 ls  ls  4.0K Nov 26 15:05 vscode-server
-rw-r--r--  1 ls  ls  268 Nov 26 15:05 .wget-hsts
drwxr-xr-x  3 ls  ls  4.0K Nov 24 18:28 Documentos
-rw-r--r--  1 ls  ls  0 Dec 24 11:52 arquivo_antigo.tmp
drwxr-xr-x  2 ls  ls  4.0K Dec 24 11:52 backup
-rw-r--r--  1 ls  ls  0 Dec 24 11:52 documento.pdf
drwxr-xr-x  2 ls  ls  4.0K Dec 24 11:52 entrega
-rw-r--r--  1 ls  ls  0 Dec 24 11:52 novo_arquivo.txt
drwxr-xr-x  3 ls  ls  4.0K Nov 26 15:55 pipeline_dados
drwxr-xr-x  2 ls  ls  4.0K Dec 24 11:51 projeto_exemplo

```

*Figura 18 – Execução de operações básicas de criação, cópia, movimentação e listagem de arquivos no Ubuntu 24.04.2 LTS. Fonte: Autor.*

O estudo prático de manipulação de arquivos iniciou-se com operações básicas, onde tentativas de movimentação e exclusão falharam pela ausência prévia dos alvos, enquanto a criação de novos recursos via touch e mkdir obteve êxito imediato e confirmado. Para o gerenciamento de acesso, o sistema evidenciou sua robustez ao combinar ferramentas de terminal, como chmod, chown e ACLs para controle granular, com interfaces gráficas intuitivas no Nautilus. A validação técnica desses conceitos exigiu a preparação do ambiente através da criação do arquivo script.sh, que foi submetido a regras rígidas de segurança com o comando chmod 750. Essa configuração específica garantiu privilégios totais ao proprietário e apenas leitura e execução ao grupo, bloqueando efetivamente o acesso de terceiros e demonstrando, na prática, a aplicação das políticas de segurança e propriedade do sistema de arquivos no Ubuntu.

```

ls@JuanPablo484806:~$ echo '#!/bin/bash' > script.sh
echo 'echo Olá, mundo!' >> script.sh
chmod 750 script.sh
ls -l script.sh
-rwxr-x--- 1 ls  ls 30 Dec 24 11:56 script.sh
ls@JuanPablo484806:~$ ./script.sh
Olá, mundo!

```

*Figura 19.1 – Saída do comando chmod, chown e getfacl após criação do ambiente de teste. Fonte: Autor.*

*Em seguida, foi criado o usuário matheus e o grupo estagiarios, vinculando o usuário recém-criado ao grupo. Essa etapa permitiu demonstrar o uso do comando chown, utilizado para alterar o dono e o grupo associados ao arquivo:*

```
ls@JuanPablo484806:~$ sudo addgroup estagiarios
sudo adduser --disabled-password --gecos "" matheus
sudo usermod -aG estagiarios matheus
sudo chown matheus:estagiarios script.sh
ls -l script.sh
info: Selecting GID from range 1000 to 59999 ...
info: Adding group `estagiarios' (GID 1001) ...
info: Adding user `matheus' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `matheus' (1002) ...
info: Adding new user `matheus' (1002) with group `matheus (1002)' ...
info: Creating home directory `/home/matheus' ...
info: Copying files from `/etc/skel' ...
info: Adding new user `matheus' to supplemental / extra groups `users' ...
info: Adding user `matheus' to group `users' ...
-rwxr-x--- 1 matheus estagiarios 30 Dec 24 11:56 script.sh
```

Figura 19.2 – Saída do comando chmod, chown e getfacl após criação do ambiente de teste. Fonte: Autor.

Por fim, criou-se o diretório /pasta\_compartilhada, no qual foi aplicada uma Access Control List (ACL) permitindo que o usuário matheus tivesse acesso de leitura e escrita, independentemente das permissões padrão POSIX. A exibição dessa ACL foi feita com getfacl:

```
ls@JuanPablo484806:~$ sudo mkdir /pasta_compartilhada
sudo setfacl -m u:matheus:rw /pasta_compartilhada
getfacl /pasta_compartilhada
getfacl: Removing leading '/' from absolute path names
# file: pasta_compartilhada
# owner: root
# group: root
user::rwx
user:matheus:rw-
group::r-x
mask::rwx
other::r-x
```

Figura 19.3 – Saída do comando chmod, chown e getfacl após criação do ambiente de teste. Fonte: Autor.

## Proteção e Segurança

- Como o sistema trata a autenticação de usuários?7

A autenticação de usuários é gerida por meio do framework Pluggable Authentication Modules (PAM), que abstrai o processo de verificação de credenciais das aplicações e consolida diversas políticas de segurança em pilhas modulares. Quando um usuário tenta efetuar login (seja via terminal, interface gráfica (GDM) ou serviço remoto (SSH)) o sistema executa a seguinte sequência expositiva:

# Exemplo: trecho de /etc/pam.d/common-auth

```
ls@JuanPablo484806:~$ cat /etc/pam.d/common-auth
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.). The default is to use the
# traditional Unix authentication mechanisms.
#
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules. See
# pam-auth-update(8) for details.
#
# here are the per-package modules (the "Primary" block)
auth    [success=1 default=ignore]      pam_unix.so nullok
# here's the fallback if no module succeeds
auth    requisite                  pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
auth    required                   pam_permit.so
# and here are more per-package modules (the "Additional" block)
auth    optional                  pam_cap.so
# end of pam-auth-update config
```

Figura 20 – Configuração típica de autenticação PAM em Ubuntu 24.04.2 LTS. Fonte: Autor.

Adicionalmente, o Ubuntu 24.04 introduz o Authd, um daemon que permite integrar diretamente provedores de identidade baseados em OIDC (Microsoft Entra ID, Okta, Google), abrindo caminho para autenticação federada em desktops e servidores. Essa evolução demonstra o compromisso da plataforma em oferecer tanto a robustez do modelo local via PAM e arquivos shadow quanto a flexibilidade de ambientes corporativos que demandam identidade em nuvem.

**Criptografia de disco completo (LUKS / dm-crypt):** Durante o processo de instalação do sistema, o instalador do Ubuntu oferece ao usuário a opção de criptografar integralmente o disco rígido. Essa criptografia é implementada por meio do subsistema

*dm-crypt* em conjunto com o formato LUKS2 (Linux Unified Key Setup), que suporta múltiplas chaves de desbloqueio e algoritmos avançados de derivação de chave. Essa abordagem assegura que todas as partições — inclusive swap e /root — permaneçam inacessíveis sem autenticação prévia. A única exceção é a partição /boot, mantida em texto claro para permitir que o GRUB initialize o sistema.

Após a inicialização, o usuário insere uma senha de desbloqueio, momento em que o kernel monta a partição criptografada, assegurando proteção contra acesso físico não autorizado, especialmente em dispositivos móveis ou notebooks.

```
ls@JuanPablo484806:~$ lsblk -o NAME,FSTYPE,SIZE,TYPE,MOUNTPOINT | grep crypt
cryptsetup status /dev/mapper/ubuntu--vg-root
Command 'cryptsetup' not found, but can be installed with:
sudo apt install cryptsetup-bin
```

Figura 21 – Verificar status de criptografia LUKS nas partições em Ubuntu 24.04.2 LTS. Fonte: Autor.

A análise do ambiente de teste indicou o uso de LVM sem a criptografia de disco completo (LUKS) habilitada, embora esse recurso de segurança esteja disponível nativamente no instalador do Ubuntu. Para cenários que exigem proteção granular sem o custo de desempenho da criptografia total, o sistema suporta o fscrypt integrado ao sistema de arquivos ext4, permitindo blindar apenas diretórios específicos de forma transparente. Essa solução equilibra segurança e performance, pois cifra apenas os dados sensíveis do usuário e pode ser integrada ao módulo de autenticação PAM. Dessa forma, o desbloqueio dos arquivos ocorre automaticamente durante o início da sessão, garantindo uma experiência de uso fluida e protegida.

#### **Existe separação clara entre usuários comuns e administradores?.**

# Verificar membros do grupo sudo

```
ls@JuanPablo484806:~$ getent group sudo
sudo:x:27:ls
```

Figura 22 – Usuários com permissão para sudo em Ubuntu 24.04.2 LTS. Fonte: Autor.

A política de sudo é definida em /etc/sudoers (editável com visudo para evitar erros de sintaxe). Por exemplo, a linha abaixo concede a todos os membros do grupo sudo o direito de executar qualquer comando como root:

```
%sudo  ALL=(ALL:ALL) ALL
```

- Há antivírus, firewall, SELinux, AppArmor, UAC etc.?

No Ubuntu 24.04.2 LTS, as camadas de segurança associadas a antivírus e firewall complementam o controle de acesso descrito anteriormente:

**Firewall (UFW / iptables):** O Ubuntu disponibiliza por padrão o UFW (“Uncomplicated Firewall”), uma interface simplificada para o iptables, embora venha desativado após a instalação. Seu modelo de funcionamento define, geralmente, regras padrão de negação de conexões de entrada e liberação de saída, podendo ser ajustado via CLI ou pela GUI Gufw.

Para o gerenciamento de firewall no Ubuntu 24.04.2 LTS, é utilizado o UFW (Uncomplicated Firewall), uma interface simplificada para o sistema iptables. No entanto, esse componente não vem ativado nem instalado por padrão em algumas versões. Ao tentar executar o comando ufw inicialmente, o sistema retornou a mensagem command not found, indicando a ausência do utilitário.

```
ls@JuanPablo484806:~$ sudo ufw enable
sudo ufw status verbose
sudo: ufw: command not found
sudo: ufw: command not found
```

Figura 23A – Tentativa inicial de comando UFW. Fonte: Autor.

```
ls@JuanPablo484806:~$ sudo apt update && sudo apt install ufw
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
59 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  iptables libipip4tc2 libipip6tc2 libnetfilter-contrack3 libnfnnetlink0 libnftables1 libnftnl11
Suggested packages:
```

Figura 23B – Instalação do pacote por meio do comando. Fonte: Autor.

```
ls@JuanPablo484806:~$ sudo ufw enable
sudo ufw status verbose
Firewall is active and enabled on system startup
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
```

Figura 23C – Ativação do firewall. Fonte: Autor.

**Antivírus (ClamAV, rkhunter, chkrootkit):** Diferentemente de ambientes Windows, o Ubuntu não inclui antivírus ativo por padrão; contudo, o pacote ClamAV está disponível nos repositórios oficiais e pode ser instalado para escaneamentos pontuais ou

em modo daemon. Ferramentas adicionais como rkhunter e chkrootkit O Linux Security Module (LSM) utilizado como controle de acesso mandatório também são recomendadas para detecção de rootkits.

**SELinux:** Embora suportado, o Ubuntu opta por não habilitar o SELinux, mantendo o estado “disabled” conforme configuração em /etc/selinux/config. Para verificar:

```
ls@JuanPablo484806:~$ aa-status
sestatus
apparmor module is loaded.
apparmor filesystem is not mounted.
Command 'sestatus' not found, but can be installed with:
sudo apt install policycoreutils
```

Figura 24 – Saída do comando aa-status e tentativa de verificação do SELinux com sestatus. Fonte: Autor.

O Ubuntu 24.04.2 adota o AppArmor como solução padrão de segurança (MAC), cujos módulos foram detectados ativos durante a análise. A ausência nativa do SELinux e de suas ferramentas confirma a escolha arquitetural da distribuição, que prioriza exclusivamente o AppArmor para o isolamento e controle de aplicações



## Comparativo com o que foi estudado

- **Quais pontos do sistema analisado se aproximam dos modelos teóricos estudados?**

É possível observar que os modelos estudados servem como um embasamento teórico para o desenvolvimento de sistemas operacionais modernos, mas não são seguidos à risca. Isso pode significar que apenas os modelos mais clássicos não são suficientes para se obter um melhor desempenho, mas quando utilizados de forma combinada e somado a conhecimentos mais atuais, resultam em resultados mais eficientes e seguros.

No caso do sistema operacional Ubuntu e outros sistemas baseados no kernel Linux mantém muitos dos conceitos estudados quando falamos de gerenciamento de armazenamento e segurança, mas mistura métodos e incrementa outros no segmento de gerenciamento de processos e de memória. Isso garante um desempenho mais confiável para padrões modernos que mantém uma demanda crescente de processamento de dados.

- **Há alguma particularidade que chamou sua atenção?**

Uma das coisas que mais chama atenção no sistema operacional mantido e melhorado pela Canonical Ltd. é o seu método de gerenciamento de memória é extremamente refinado e atualizado com as tendências mais recentes. O algoritmo LRU de duas listas / Clock puro com Aging se mostra muito eficiente mesmo em alta demanda pela memória principal, o que o torna uma sistema operacional capaz de se tornar multi-tarefas mesmo em máquinas com baixo desempenho. Seu sistema de gerenciamento de processos também ajuda muito nesse quesito, uma vez que é capaz de lidar com uma justa divisão de recursos para cada processos em execução, que garante que o sistema possa lidar com diversas tarefas mesmo com uma CPU que pode variar de um simples dual-core com 2 threads até um mais eficiente com 6 cores e 12 threads ou mais. Por isso é um sistema muito elogiado pelo seu excelente desempenho, seja na tarefa de dar uma sobrevida para um computador pessoal de hardware defasado, até fazer funcionar um servidor com a mais recente tecnologia.

- **Você percebeu alguma diferença importante entre sistemas (caso conheça mais de um)?**

Ao comparar o Ubuntu 24.04.2 LTS com outras plataformas, observa-se que as implementações partem de princípios teóricos similares, mas aplicam-nos de forma distinta. No Windows, cada volume recebe uma letra (C:, D: etc.), enquanto no Ubuntu todos os dispositivos integram-se a uma única árvore de diretórios sob `/`. Esse modelo simplifica a montagem de pendrives, discos de rede e outras unidades, sem alterar os caminhos de arquivos.

Em relação a sistemas de arquivos, o NTFS (Windows) oferece journaling e ACLs, mas depende de drivers externos no Linux. Já o ext4 (Ubuntu) fornece journaling nativo e suporte a ACLs estendidas, garantindo desempenho e confiabilidade sem dependências adicionais.

Para controle de acesso mandatório, o SELinux (Fedora, Red Hat) opera por “negação padrão” e exige configurações detalhadas. O AppArmor (Ubuntu) utiliza perfis por aplicação com sintaxe mais simples e é ativado por padrão. Embora menos granular que o SELinux, o AppArmor facilita a adoção em cenários acadêmicos e corporativos.

Essas diferenças indicam que sistemas operacionais modernos combinam conceitos clássicos com ajustes práticos, equilibrando segurança, desempenho e facilidade de uso.

## Referências