



**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
<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**

## Objetivo

Foi estabelecido neste relatório uma breve investigação sobre o Sistema Operacional conhecido como Ubuntu em sua versão 24.04.2 LTS (Long Term Support, versão voltada para um suporte prolongado de 5 anos). Trata-se de uma distribuição Linux, um sistema operacional de código aberto da família Debian que funciona sobre o Kernel Linux e é mantido pela Canonical Ltd, servindo tanto a servidores como a computadores pessoais.

A versão em específico usada para fazer os testes de funcionalidade é a conhecida como Noble e foi alocada tanto em uma máquina virtual via WSL 2 (Windows Subsistem Linux, subsistema do Windows para o Linux) nos computadores dos autores deste relatório como também foi utilizado um notebook com o sistema em questão instalado para testes e experimentações com o objetivo de verificar as particularidades em gerência de processos, gerência de memória, gerência de arquivos e segurança.

## 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	R	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
		2620	eryk	20	0	3768M	303M	135M	S	0.6	4.0	0:00.96	/usr/bin/gnom
		2747	eryk	20	0	377M	12768	7540	S	0.6	0.2	0:01.41	/usr/bin/ibus
		2784	eryk	20	0	377M	12768	7540	S	0.6	0.2	0:02.20	/usr/bin/ibus
		3255	eryk	20	0	2663M	55292	41140	S	0.6	0.7	0:00.60	gjs /usr/shar
		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. Execução da função exec():

Com a família de funções que é chamada pelo programa, o processo filho pode se tornar um outro processo independente do pai com configurações que serão processadas de uma forma diferente. Na prática o que acontece é que todas as informações previamente carregadas no processo pai, como dados, pilhas e heaps, são zeradas e o processo filho assume uma nova roupagem para assumir outra função, porém o PID permanece o mesmo definido depois do fork(). Isso garante que o processo pai continue executando independentemente do processo filho ou aguarde uma execução paralela.

### 3. 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.

```
[11:26] ✘ Bash
~
❯ ps -p 1 -o pid,comm,etime
  PID COMMAND          ELAPSED
      1 systemd        01:19:49
```

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.

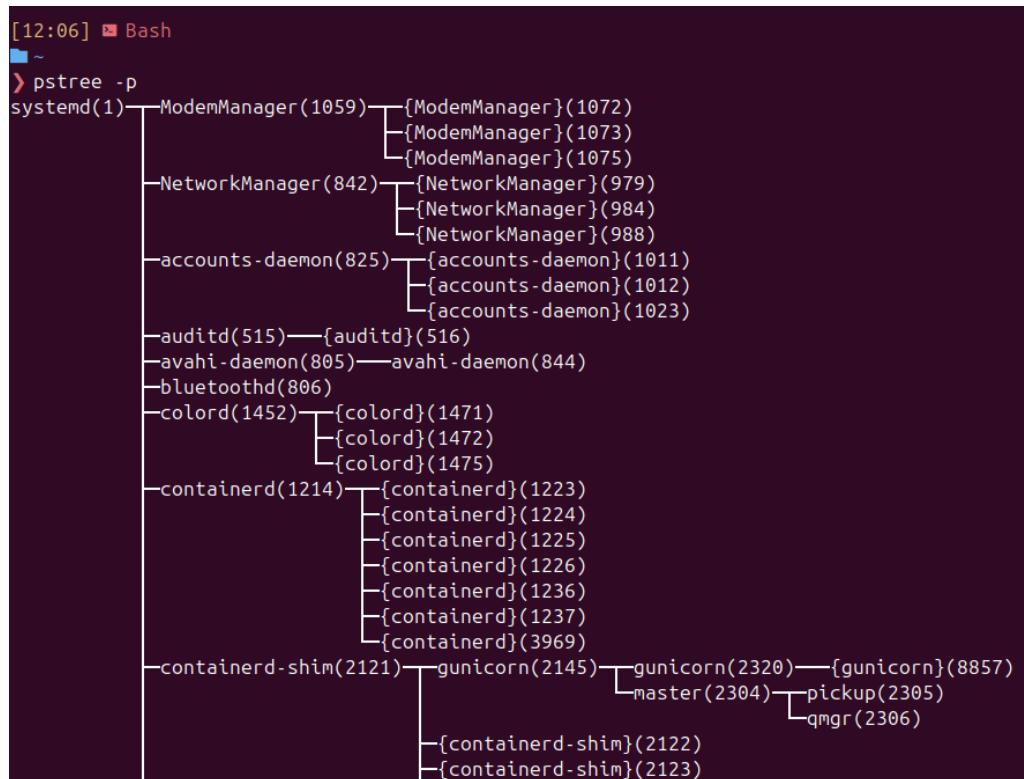


Figura 4 - Árvore de processos do linux. Fonte: Autor.

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).

Uma vez que esses processos são organizados, os processos são organizados para execução pelo escalonador que tem como método padrão o CFS (Completely Fair Scheduler - escalonador completamente justo). Esse método funciona com base em quantums, ou intervalos de tempo virtuais (vruntime) que são atribuídos a cada programa que podem variar entre 4 a 10 ms (milissegundos). Uma vez que esse tempo é finalizado o sistema remove o processos em execução e coloca outro que esteja com menos tempo de execução que ele, garantindo assim que todos os processos tenham uma divisão de tempo justa no uso da CPU (Computer Process Unit - unidade de processamento computacional) independente da prioridade. Dessa forma o sistema do Linux Ubuntu pode ser considerado multitarefa e preemptivo por tempo, sendo mais eficiente à medida que tem mais núcleos a disposição.

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPO+	COMANDO
2601	eryk	20	0	3951948	329992	140236	S	8,3	4,2	4:33.42	gnome-shell
805	avahi	20	0	14640	10424	4024	S	3,0	0,1	21:31.50	avahi-daemon
5649	eryk	20	0	1500660	134648	71416	S	1,3	1,7	1:37.08	rhythmbox
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	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	0	I	0,3	0,0	0:00.74	kworker/1:0-i915-unordered
<b>11439</b>	<b>eryk</b>	<b>20</b>	<b>0</b>	<b>12128</b>	<b>6140</b>	<b>3964</b>	<b>R</b>	<b>0,3</b>	<b>0,1</b>	<b>0:00.20</b>	<b>top</b>
1	root	20	0	23464	14752	9632	S	0,0	0,2	0:04.29	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-rCU_gp
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-sync_wq
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-slub_flushwq
7	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-netns
9	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H-events_highpri
12	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/R-mm_percpu_wq
13	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_kthread
14	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_rude_kthread

Figura 5 - Ferramenta de visualização de processos via terminal top.

Fonte: Autor.

Na figura 5 podemos ver a porcentagem do uso da CPU de cada um dos processos e o tempo decorrido de uso, assim como a quantidade de processos totais, em execução, dormindo ou em espera, parado e zumbi (por curiosidade, os processos zumbi são processos filho que terminaram sua execução mas não tiveram seus resultados devidamente coletados pelo processo pai, fazendo com que esse processo já terminado permaneça aberto sem processamento e também sem ser encerrado).

Em determinados momentos o sistema utiliza um esquema de prioridade que também pode ser alterado pelo usuário. Esse nível é denominado Nice (nível de gentileza) e vai de -20 (alta prioridade) até +19 (baixa prioridade). Essa prioridade serve para dar mais velocidade a processos mais prioritários como as rotinas do sistema que são executadas em modo kernel pela raiz do sistema, como também pode ser observado na figura 5. E ainda, alguns processos são executados com política de tempo real (SCHED\_FIFO, SCHED\_RR) e tem prioridade máxima e não são preemptados por processos normais. A prioridade pode ser alterada pelos comandos “renice -n <nível> -p <PID>” (altera o nível de nice) e “chrt -p <PID>”.

Comando	O que mostra
<b>top</b>	Uso de CPU/memória por processo em tempo real
<b>htop</b>	Versão colorida, mostra threads também
<b>ps -eLf</b>	Mostra todas as threads
<b>cat /proc/PID/sched</b>	Mostra dados internos do agendamento
<b>chrt -p PID</b>	Política de escalonamento tempo real
<b>nice, renice</b>	Altera prioridade (gentileza)

Tabela 1 - comandos para visualizar e manipular processos. Fonte: Autor.

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

Podemos definir threads como uma linha de execução leve dentro de um processo. Isso acontece porquê dentro de um mesmo processo, várias threads compartilham o mesmo espaço de memória, as mesmas variáveis globais, os mesmo arquivos abertos, etc. o que permite tornar o processamento e a busca por informações mais rápida.

O sistema operacional Ubuntu suporta isso e tem um sistema bem robusto que trata cada thread de forma escalonável e independente, tendo cada uma delas o seu próprio PID (TID) e podendo ser agendada individualmente.

```
[15:15] ~ Bash
❯ ps -eLf
UID      PID    PPID    LWP    C NLWP STIME TTY          TIME CMD
root      1      0        1  0    1 10:46 ?          00:00:04 /sbin/init splash
root      2      0        2  0    1 10:46 ?          00:00:00 [kthreadd]
root      3      2        3  0    1 10:46 ?          00:00:00 [pool_workqueue_release]
root      4      2        4  0    1 10:46 ?          00:00:00 [kworker/R-rcu_gp]
root      5      2        5  0    1 10:46 ?          00:00:00 [kworker/R-sync_wq]
root      6      2        6  0    1 10:46 ?          00:00:00 [kworker/R-slub_flushwq]
root      7      2        7  0    1 10:46 ?          00:00:00 [kworker/R-netns]
root      9      2        9  0    1 10:46 ?          00:00:00 [kworker/0:0H-events_highpri]
root     12      2       12  0    1 10:46 ?          00:00:00 [kworker/R-mm_percpu_wq]
root     13      2       13  0    1 10:46 ?          00:00:00 [rcu_tasks_kthread]
root     14      2       14  0    1 10:46 ?          00:00:00 [rcu_tasks_rude_kthread]
root     15      2       15  0    1 10:46 ?          00:00:00 [rcu_tasks_trace_kthread]
root     16      2       16  0    1 10:46 ?          00:00:01 [ksoftirqd/0]
root     17      2       17  0    1 10:46 ?          00:00:17 [rcu_preempt]
root     18      2       18  0    1 10:46 ?          00:00:00 [rcu_exp_par_gp_kthread_worker/0]
root     19      2       19  0    1 10:46 ?          00:00:00 [rcu_exp_gp_kthread_worker]
root     20      2       20  0    1 10:46 ?          00:00:00 [migration/0]
root     21      2       21  0    1 10:46 ?          00:00:00 [idle_inject/0]
root     22      2       22  0    1 10:46 ?          00:00:00 [cpuhp/0]
root     23      2       23  0    1 10:46 ?          00:00:00 [cpuhp/1]
root     24      2       24  0    1 10:46 ?          00:00:00 [idle_inject/1]
root     25      2       25  0    1 10:46 ?          00:00:00 [migration/1]
root     26      2       26  0    1 10:46 ?          00:00:02 [ksoftirqd/1]
root     28      2       28  0    1 10:46 ?          00:00:00 [kworker/1:0H-events_highpri]
root     29      2       29  0    1 10:46 ?          00:00:00 [kdevtmpfs]
root     30      2       30  0    1 10:46 ?          00:00:00 [kworker/R-inet_frag_wq]
```

Figura 6 - Lista de processos com colunas LWP e NLWP que mostram as threads.

Fonte: Autor.

Na figura 6 podemos ver uma lista de processos com o comando “ps -eLf” que tem uma coluna LWP (Light Weight Process - processo de peso leve) que armazena o TID (Thread Identification - identificação do fio)<sup>[10]</sup> e uma coluna NLWP que identifica quantas threads o processo tem.

O sistema também conta com multitarefa preemptiva, como já foi citado anteriormente. Isso permite que o sistema remova os processos da CPU quando o limite de tempo é atingido para dar lugar a outro processo, o que garante uma maior fluidez e a sensação de que o sistema está realizando várias tarefas ao mesmo tempo.

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.

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 núcleos na CPU instalada e está a dividir as cargas de

```
[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:      GenuineIntel
  Nome do modelo:          Intel(R) Celeron(R) 6305 @ 1.80GHz
  Família da CPU:          6
  Modelo:                  140
  Thread(s) per núcleo:   1
  Núcleo(s) por soquete:  2
  Soquete(s):              1

  0[|||||||||||||||||||||]  80.3%] Tasks: 153, 661 thr, 102 kthr; 1 running
  1[|||||||||||||||||||||]  86.0%] Load average: 0.92 1.07 0.93
Mem[|||||||||||||||||||||] [3.56G/7.49G] Uptime: 05:30:30
Swp[|||]
BogoMIPS:                3609,60
```

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.

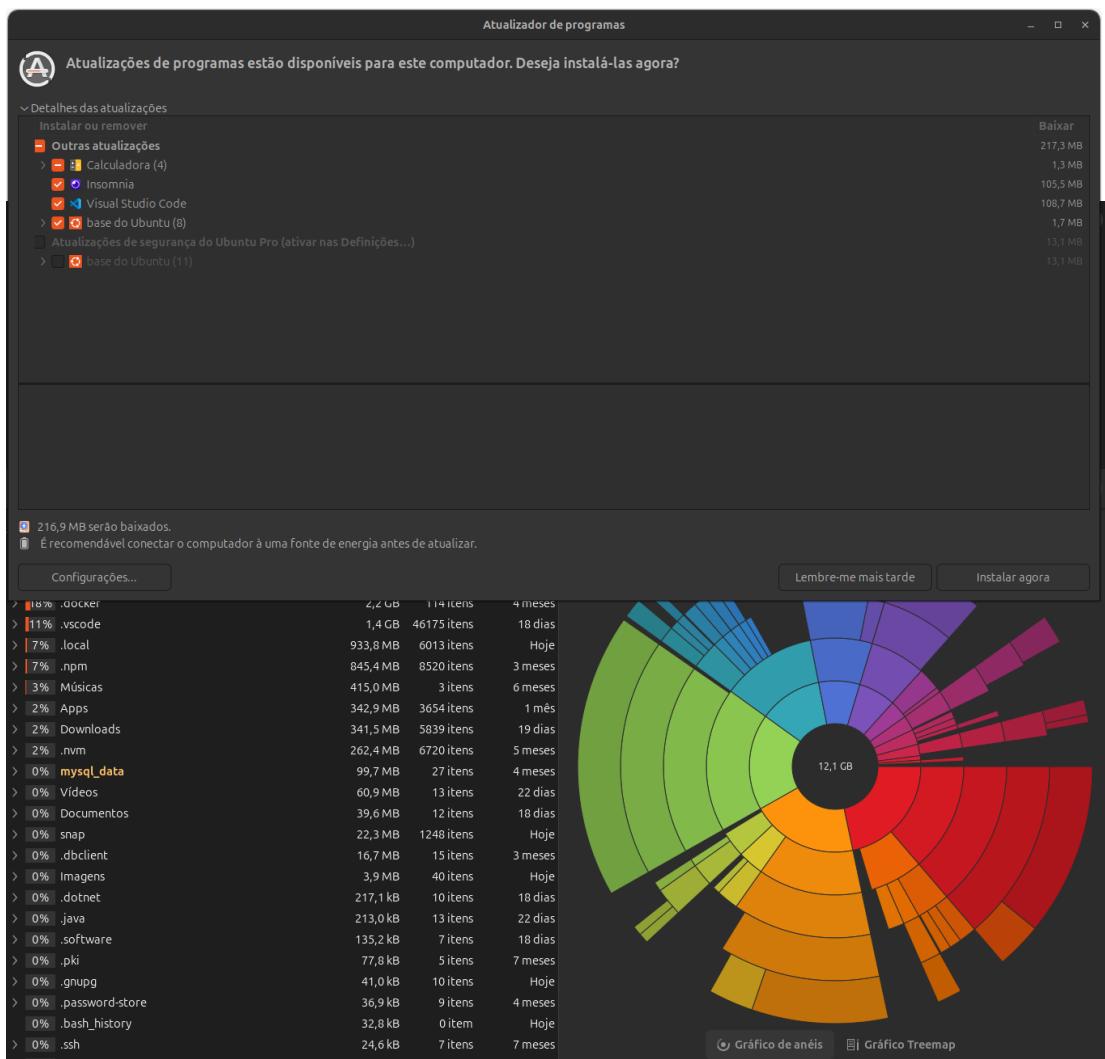


Figura 11 - Atualizador de programas do Ubuntu. Fonte: Autor.

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.

Por fim, o sistema ainda conta com algumas formas de encerrar os programas por meio de comandos no terminal do sistema. Segue abaixo mais uma tabela com comandos que podem ser úteis no gerenciamento de processos do sistema Ubuntu:

Comando	Função
<b>kill -STOP &lt;PID&gt;</b>	Pausa temporariamente a execução de um processo
<b>kill -CONT &lt;PID&gt;</b>	Retoma um processo pausado anteriormente
<b>pkill &lt;nome&gt;</b>	Encerra processos pelo nome
<b>xkill</b>	Modo gráfico: clique em uma janela para encerrá-la
<b>stress -c &lt;n&gt;</b>	Simula carga de CPU com n processos paralelos
<b>lscpu</b>	Exibe informações sobre núcleos e threads da CPU
<b>fg %&lt;número&gt;</b>	Retorna um processo do segundo plano para primeiro plano

Tabela 2 - Comandos para finalização de processos e outros. Fonte: Autor.

## 2 Gerência de Memória

- **O sistema usa memória virtual?**

O Ubuntu utiliza uma memória virtual nomeada como Swap que é parte do sistema de paginação por demanda do sistema operacional. Se trata de uma técnica adotada por diversos sistemas operacionais modernos, que reserva uma parte de tamanho fixo (personalizável via comando) do armazenamento secundário do sistema para armazenar partes das informações de um programa. Isso é necessário porque alguns programas podem ocupar espaço demais na memória principal caso fossem carregados completamente, porém, com essa

técnica seu conteúdo é fragmentado em páginas que são endereçadas e armazenadas no disco secundário e repassadas para a memória principal apenas quando solicitado. Isso permite que o sistema possa carregar mais programas na memória, dando mais capacidade de multitarefa e multiprogramação.

Porém como toda técnica, esta também tem suas limitações. O uso constante de memória secundária gera mais latência no sistema, pois a tecnologia que compõem esses dispositivos atingem uma velocidade inferior de leitura e gravação. Isso significa que todas as vezes que for necessário buscar por uma página faltante na memória principal gerará um atraso no processamento. Um outro problema presente nesse tipo de método é que o próprio sistema operacional precisa se encarregar de fazer o gerenciamento de páginas dos programas que estão presentes na memória principal e secundária, o que pode gerar uma sobrecarga no sistema. Por isso se faz necessário algoritmos inteligentes de troca de páginas, para evitar que haja um grande número do que é conhecido como page faults (situação de página faltante e o processo de transferência da memória secundária para principal, o que inclui a exclusão de páginas presentes na memória para liberar espaço, caso necessário).

```
[18:05] ■ Bash
█ ~
> free -h
              total        used        free      shared  buff/cache   available
Mem:       7.6Gi       668Mi       6.7Gi       3.8Mi       350Mi       6.9Gi
Swap:      2.0Gi          0B       2.0Gi
```

Figura 12 - Tabela com informações sobre o tamanho da memória principal e de Swap do sistema Ubuntu. Fonte: Autor.

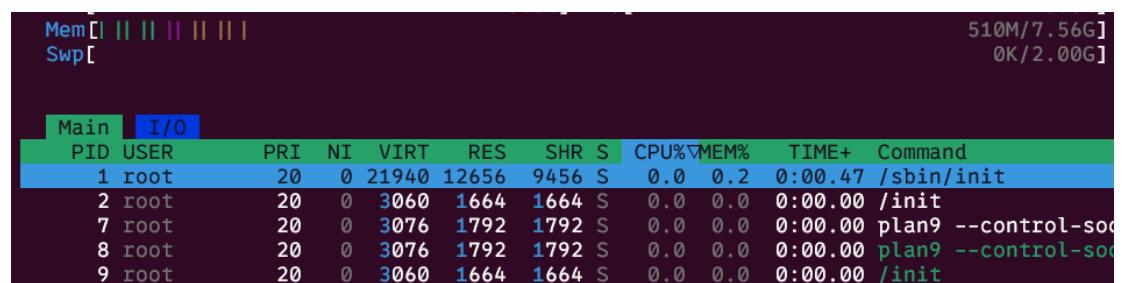


Figura 13 - Tabela com lista de processos do sistema Ubuntu via programa de linha de comando htop. Fonte: Autor.

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).

- **Há paginação ou segmentação?**

Aqui temos um gap: afinal, como podemos fazer para selecionar qual página será removida da memória principal? No Ubuntu isso é feito utilizando algoritmos inteligentes de substituição de páginas. Eles servem para tentar selecionar e remover páginas que não serão usadas com frequência, o que garante uma redução do problema de page faults excessivos, reduzindo a latência.

O algoritmo utilizado para essa seleção é uma mistura dos modelos LRU (Least Recently Used - Menos recentemente usado), Clock (Relógio), Multi-Level Aging (Envelhecimento em vários níveis) e ainda nos modelos mais recentes como a versão 24.04.2 do Ubuntu, conta com métodos de Algorithm-based memory reclaiming (Recuperação de memória baseada em algoritmo) que servem para entender se o sistema está engasgando por falta de CPU (a unidade de processamento computacional) ou de RAM (memória principal), e ainda, quanto de RAM grupos de processos podem usar. Esse algoritmo é parte do kernel Linux, e está sempre ativado para detectar quando a memória principal está quase cheia e iniciar a organização das páginas pela técnica do Swap.

Esse algoritmo pode ser chamado de Clock Puro/LRU de duas listas onde os processos são organizados em duas listas com “páginas ativas” (usadas mais recentemente) e “páginas inativas” (não utilizadas a um tempo), o que se aproxima do algoritmo de LRU. Esses processos podem transitar entre essas listas à medida que são usados ou não pela CPU, o que se aproxima do algoritmo de Aging. E por fim, quando esses processos permanecem na lista de “processos inativos” eles correm o risco de serem selecionados pelo algoritmo de Clock quando for necessário remover uma página da memória principal.

A segmentação nos sistemas Linux modernos existem apenas em nível de hardware (chip especializado em paginação MMU - memory management unit - unidade de gerenciamento de memória) porém o kernel do sistema entende o espaço de memória como uma sequência linear de páginas. Uma segmentação simbólica ainda existe a nível de segurança, para dividir o espaço da memória

principal entre os processos da raiz do sistema e do usuário. Porém essa segmentação é completamente modular.

- **Como ocorre o gerenciamento entre processos ativos e memória disponível?**

Podemos fazer uma analogia entre a memória principal do sistema operacional Ubuntu e um pallet que pode ser preenchido com cartelas de ovos, que representam os programas/processos. Inicialmente vai se colocando cada cartela completa com ovos até encher todos os espaços disponíveis para as cartelas. Em um determinado momento o sistema detecta que o pallet está enchendo e inicia um segundo pallet que, para efeito de representação, está no segundo andar do prédio. Esse segundo pallet podemos nomear como o Swap do sistema no armazenamento secundário. Uma vez detectado essa alta demanda da memória principal, o sistema começa a dividir esses processos em fragmentos, aqui representados pelos ovos, que são movidos para cartelas paralelas no pallet do swap, e cada cartela de processo no pallet da memória principal tem uma espaço reservado no pallet de Swap. Dessa forma o sistema sabe exatamente onde colocar cada ovo de cada processo e os leva andar acima ou andar abaixo apenas quando for solicitado. Porém, se mais cartelas forem solicitando lugar no pallet, e já não houver mais espaço, o sistema se vê obrigado a chutar algumas cartelas de ovos, dessa forma matando vários processos e abrindo espaço no pallet. Isso é necessário pois se não houver espaço na memória principal para o funcionamento do sistema ou de alguns processos principais, toda a casa dos ovos pode simplesmente colapsar, estragando assim todos os ovos.

Essa analogia demonstra como é feito o gerenciamento de memória pelo kernel Linux quando há uma demanda crescente de processamento. O gerenciamento entre os processos ativos e a memória disponível é feito por meio do uso de memória virtual isolada para cada processo, combinada com as técnicas de paginação e observação do uso de memória pelo kernel. Quando a memória RAM começa a atingir seu limite, o sistema ativa o processo kswapd para mover páginas menos utilizadas para a área de swap. Caso a pressão de memória continue crítica, o kernel pode acionar o OOM Killer, que encerra processos para evitar travamentos. Além disso, o Ubuntu permite configurar limites de memória por processo ou grupo de processos utilizando os cgroups, garantindo maior

controle sobre o uso de recursos. O sistema mantém o uso da memória equilibrado mesmo em situações de alta concorrência entre processos.

- **Há ferramentas para monitoramento (ex: top, free, etc.)?**

Por fim, podemos garantir que o sistema Ubuntu conta com diversas ferramentas para o monitoramento e gerenciamento de memória, como já foi mostrado anteriormente. Segue abaixo uma tabela com algumas ferramentas de linha de comando que podem ser inseridas no terminal GNOME para interação e visualização dos processos de gerenciamento da memória:

Comando	Descrição
<b>free -h</b>	Mostra o uso atual da memória RAM e da área de swap, com valores legíveis (em MB/GB).
<b>htop</b>	Interface interativa que mostra o uso de CPU, RAM, swap e detalhes de cada processo.
<b>top</b>	Similar ao htop, exibe o uso da CPU e da memória em tempo real, mas de forma mais simples.
<b>vmstat</b>	Mostra estatísticas de memória virtual, processos, E/S e uso da CPU.
<b>cat /proc/meminfo</b>	Exibe detalhes técnicos da memória do sistema, incluindo buffers, cache, swap, etc.
<b>cat /proc/&lt;PID&gt;/status</b>	Mostra o consumo de memória de um processo específico, incluindo VmSize, VmRSS e VmSwap.
<b>cat /proc/vmstat</b>	grep -E 'pg'
<b>cat /proc/&lt;PID&gt;/oom_score</b>	Exibe a pontuação de um processo para o OOM Killer (quanto mais alto, maior a chance de ser morto).

Tabela 3 - Ferramentas de linha de comando para o monitoramento do gerenciamento de memória no Ubuntu. Fonte: Autor.

## 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       1655762868  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).

- Como os arquivos são organizados?

A organização dos arquivos segue rigorosamente o Filesystem Hierarchy Standard (FHS), um conjunto de diretrizes mantido pela Linux Foundation que

estabelece convenções para a disposição de diretórios e conteúdos em sistemas Unix-like. Todas as pastas e arquivos residem em uma única árvore hierárquica iniciada na raiz /, independentemente de estarem efetivamente armazenados em dispositivos físicos distintos. Essa abordagem facilita a interoperabilidade de aplicações, scripts e ferramentas de administração, conferindo uniformidade à documentação e previsibilidade à localização de recursos.

Em nível superior, destacam-se diretórios como:

<b>/bin e /sbin:</b>	Comandos essenciais para todos os usuários e para operações em modo de usuário único
<b>/etc</b>	Arquivos de configuração do sistema, com lógica de “tudo que não é binário” que permanece estático;
<b>/usr</b>	Hierarquia secundária, destinada a utilitários e aplicações de múltiplos usuários, geralmente em modo somente-leitura;
<b>/var</b>	Dados variáveis (logs, caches, spools), cujo conteúdo muda continuamente em tempo de execução;
<b>/home</b>	Diretórios pessoais dos usuários, onde residem documentos e configurações individuais;
<b>/tmp</b>	Área para arquivos temporários, frequentemente limpa a cada reinicialização.

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 controle de acesso a arquivos baseia-se primariamente no modelo POSIX de permissões, que distingue três classes de usuários – proprietário (user), grupo (group) e outros (others) – e três tipos de permissão para cada classe: leitura (r), escrita (w) e execução (x). Esse esquema minimalista, porém robusto, atribui a cada arquivo e diretório um conjunto de nove bits que determinam quem pode ver, modificar ou executar o recurso. Ao criar um novo arquivo, suas permissões iniciais são definidas pelo *umask* do processo criador, ajustável conforme política de segurança ou conveniência dos usuários (por exemplo, `umask 022` resulta em `rwxr-xr-x`).

Além do modelo básico, o ext4 – sistema de arquivos padrão no Ubuntu – já vem configurado para oferecer *suporte a ACLs (Access Control Lists)* por usuário e grupo, permitindo especificar direitos além do trio “user/group/others” clássico. Com as ACLs, um administrador pode conceder, a um usuário ou grupo específico, permissão diferenciada (por exemplo, dar acesso de escrita a um membro sem alterar as permissões gerais de “others”), bem como definir regras padrão de herança em diretórios para novos objetos criados em seu interior .

Para verificar se havia permissões estendidas (ACLs) aplicadas ao diretório /home, foi utilizado o comando getfacl. Como o utilitário não está presente por padrão em algumas instalações, foi necessário instalar o pacote acl via APT:

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

```

ls@JuanPablo484806:~$ sudo apt update && sudo apt install acl
Hit: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://archive.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 [1506 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 [6048 B]
Get:31 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:32 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f 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]

```

```

...
Get:40 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:41 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Fetched 13.6 MB in 5s (2627 kB/s)
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 NEW packages will be installed:
  acl
0 upgraded, 1 newly installed, 0 to remove and 59 not upgraded.
Need to get 39.4 kB of archives.
After this operation, 197 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 acl amd64 2.3.2-1build1.1 [39.4 kB]
Fetched 39.4 kB in 1s (44.7 kB/s)
Selecting previously unselected package acl.
(Reading database ... 47774 files and directories currently installed.)
Preparing to unpack .../acl_2.3.2-1build1.1_amd64.deb ...
Unpacking acl (2.3.2-1build1.1) ...
Setting up acl (2.3.2-1build1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
ls@JuanPablo484806:~$ getfacl /home
getfacl: Removing leading '/' from absolute path names
# file: home
# owner: root
# group: root
user::rwx
group::r-x
other::r-x

```

Figura 16 e 17 – 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 verboso) 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-------  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
drwx-----  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.*

Foram utilizados comandos básicos de criação, listagem e tentativa de movimentação e remoção de arquivos no diretório pessoal. Como os arquivos documento.pdf, relatório.docx e arquivo\_antigo.tmp ainda não existiam no momento da execução, o terminal retornou mensagens de erro informando sua ausência.

Já os comandos touch novo\_arquivo.txt e mkdir projeto\_exemplo foram bem-sucedidos, como confirmado na saída do ls -lah.

Para o controle de permissões e atribuição de propriedade, utilizam-se os comandos chmod (para alterar modos POSIX), chown (para modificar dono) e chgrp (para grupo), além das ACLs via getfacl/setfacl, que permitem gerenciar

direitos granulares por usuário ou grupo. Graficamente, o Nautilus disponibiliza, através do menu de contexto → “Propriedades” → aba “Permissões”, formulários intuitivos para ajustar esses mesmos atributos sem recorrer ao terminal.

Para garantir a execução bem-sucedida dos comandos de gerenciamento de permissões, foi necessário preparar previamente o ambiente com os elementos mencionados nos exemplos. Primeiramente, criou-se o arquivo script.sh contendo um simples script de shell, ao qual foram atribuídas permissões específicas com o comando chmod 750, garantindo acesso total ao proprietário, leitura e execução ao grupo, e nenhum acesso aos demais usuários:

```
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.*

Dessa forma, o Ubuntu oferece um leque completo de operações de gerenciamento de arquivos, abrangendo desde interações rápidas em terminal até controles mais elaborados via GUI, garantindo flexibilidade e produtividade em ambientes acadêmicos e corporativos.

## Proteção e Segurança

- Como o sistema trata a autenticação de usuários??

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:

**Solicitação de credenciais:** O programa de login (por exemplo, login, gdm-password, ou o daemon sshd) apresenta um prompt para nome de usuário e senha, que são então repassados ao PAM via chamada à biblioteca libpam.

**Módulo pam\_unix e verificação local:** O módulo pam\_unix.so acessa os registros de usuário em /etc/passwd e o hash de senha em /etc/shadow, aplicando o método de criptografia configurado (por padrão, SHA-512 definido em /etc/login.defs). Se as credenciais conferirem, o módulo retorna sucesso; caso contrário, registra falha em log e devolve erro ao programa de login.

**Controles adicionais de conta:** Após autenticação do par usuário/senha, o PAM avalia regras de controle de conta (via pam\_unix ou módulos como pam\_tally2 para bloqueio após tentativas fracassadas), checando status de expiração de senha, validade de shell, restrições de horário ou origem, antes de liberar o acesso ao shell ou sessão gráfica .

**Sessão e ambiente:** Uma vez autenticado, o PAM carrega o ambiente do usuário (pam\_env.so) e pode desbloquear o chaveiro GNOME (pam\_gnome\_keyring.so), além de executar scripts de inicialização de sessão definidos em /etc/security e /etc/profile.d .

A autenticação no Ubuntu é definida em arquivos de configuração localizados no diretório /etc/pam.d/. A figura abaixo exibe o conteúdo do arquivo common-auth, responsável pela etapa de autenticação em diversas aplicações do sistema. Nele, é possível observar o uso do módulo pam\_unix.so, que realiza a verificação de senha contra os arquivos /etc/passwd e /etc/shadow.

```
# 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.

- **Há mecanismos de criptografia ou controle de acesso?**

O Ubuntu 24.04.2 LTS adota mecanismos robustos de criptografia e controle de acesso para proteger os dados do sistema e garantir o isolamento entre usuários. Essas medidas operam tanto em nível de disco quanto de diretório, além de se integrarem a políticas de permissões refinadas.

**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.*

*No ambiente de teste utilizado, o volume /dev/mapper/ubuntu--vg-root está inativo para o comando cryptsetup e não aparece como crypt no lsblk. Isso indica que o sistema foi instalado com LVM (Logical Volume Manager), mas sem criptografia LUKS habilitada. A funcionalidade, no entanto, permanece disponível para quem selecionar a opção “Encrypt the new Ubuntu installation for security” no instalador.*

**Criptografia em nível de diretório (fscrypt):** Para cenários em que se deseja proteger apenas diretórios específicos (como /home/usuário/Private), o Ubuntu oferece suporte ao fscrypt, um mecanismo de criptografia por diretório integrado ao sistema de arquivos ext4 (e também ao f2fs). O fscrypt opera de forma transparente, criptografando os arquivos do diretório sem afetar o restante do sistema, o que o torna ideal para proteger dados pessoais sem comprometer a performance global.

O processo consiste na inicialização do sistema com suporte ao fscrypt, criação de diretórios protegidos e definição de políticas de desbloqueio, que podem ser integradas à autenticação PAM para desbloqueio automático na sessão do usuário.

# Exemplo: Comandos úteis:

```
sudo apt install fscrypt  
sudo fscrypt setup /  
sudo fscrypt encrypt /home/matheus/Private \  
--label=Private --user=matheus
```

**Nota:** Durante os testes, o comando `fscrypt encrypt` retornou um erro indicando a ausência do arquivo de configuração `/etc/fscrypt.conf`, confirmando que o `fscrypt` exige uma configuração inicial com `fscrypt setup /`. Como a criptografia em diretórios é um recurso opcional e avançado, o ambiente de testes não foi configurado com essa proteção ativa. Ainda assim, o suporte nativo à ferramenta confirma a disponibilidade do mecanismo no Ubuntu 24.04.2 LTS.

**Permissões POSIX e ACLs:** Tanto a criptografia em disco quanto por diretório é complementada por mecanismos tradicionais de controle de acesso, como as permissões POSIX (leitura, escrita e execução por usuário, grupo e outros) e as ACLs (Access Control Lists), que permitem atribuições mais granulares. Essas técnicas garantem que apenas os usuários autorizados possam visualizar ou modificar determinados arquivos e pastas.

- **Existe separação clara entre usuários comuns e administradores?**

No Ubuntu 24.04.2 LTS, a distinção entre usuários comuns e administradores baseia-se no controle de pertença a grupos e nas regras definidas em `/etc/sudoers`, de forma a garantir segregação de funções e minimizar os riscos de uso inadvertido de privilégios elevados.

O usuário root, ou superusuário, existe por padrão, mas sua conta costuma vir bloqueada (sem senha) e inacessível por login direto, forçando os administradores a atuarem sempre via sudo, o que gera registro (logging) de cada operação realizada com privilégios elevados.

Os usuários comuns são criados sem privilégios de administração e pertencem, por padrão, a grupos restritos (como `users` ou a um grupo com o mesmo nome do usuário). Já o grupo `sudo` (sucessor do antigo `admin` em Ubuntu) agrupa aqueles que podem executar comandos como root, bastando pré-fixá-los com sudo e autenticar-se com sua própria senha de usuário

```
# 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
```

Além disso, o Ubuntu pode empregar arquivos de configuração adicionais sob /etc/sudoers.d/, permitindo atribuições refinadas (por usuário ou grupo) sem alterar o arquivo principal e facilitando auditoria e controle de versões.

Com essa separação clara usuários comuns executam aplicações e tarefas diárias sem risco de afetar todo o sistema, administradores (membros de sudo) elevam privilégios pontualmente, com autenticação e logging e a conta root permanece disponível para casos extremos (ex.: recuperação de sistema), mas seu uso direto é desestimulado.

Esse modelo modular garante segurança por delegação, pois minimiza o número de sessões com poderes totais e mantém histórico de atividades sensíveis para auditoria.

- **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 libip4tc2 libip6tc2 libnetfilter-conntrack3 libnftnetlink0 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 também são recomendadas para detecção de rootkits.

O Linux Security Module (LSM) utilizado como controle de acesso mandatório (MAC) no Ubuntu é o AppArmor, enquanto o SELinux permanece desabilitado por padrão:

**AppArmor:** Instalado e carregado automaticamente, o AppArmor restringe programas a perfis predefinidos. O comando aa-status revela quais perfis estão em modo “enforce” ou “complain”.

**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 LTS utiliza o AppArmor como sua principal solução de controle de acesso mandatório (MAC), restringindo o comportamento de aplicações com base em perfis de segurança. Ao executar o comando aa-status, o sistema indicou que o módulo do AppArmor está carregado, mas que o sistema de arquivos correspondente não está montado, sugerindo que o serviço pode não estar ativo por completo no momento da análise.

Em relação ao SELinux, ao tentar executar o comando sestatus, o terminal informou que o utilitário não está disponível e sugeriu a instalação do pacote policycoreutils. Essa resposta confirma que o SELinux não está habilitado por padrão no Ubuntu, o que é esperado, dado que essa distribuição prioriza o uso do AppArmor para controles de segurança em espaço de usuário.



## 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

*Indique aqui os sites, manuais, comandos e fontes consultadas para desenvolver sua análise.*

1. Manual Pages (man pages) do Linux, acessível no terminal com comandos como:

man free

man

vmstat

man ps

man kill

man proc

man exec

man nice

man top

man htop

man systemd

Ou

online:

<https://man7.org/linux/man-pages/>

2. Ubuntu Official Documentation

<https://help.ubuntu.com/lts/ubuntu-help/index.html>

3. Kernel.org – Linux Kernel Scheduler Documentation

<https://www.kernel.org/doc/html/latest/scheduler/index.html>

4. Red Hat Developer – Entendendo o agendamento de processos no Linux

<https://developers.redhat.com>

5. The Geek Stuff – Comandos úteis para gerenciar processos no Linux

<https://www.thegeekstuff.com>

6. Stack Overflow e StackExchange Unix/Linux

<https://unix.stackexchange.com>

7. Wikipedia – Process Management in Unix/Linux  
[https://en.wikipedia.org/wiki/Process\\_management\\_\(computing\)](https://en.wikipedia.org/wiki/Process_management_(computing))
8. Linux Memory Management - LWN.net  
<https://lwn.net/Kernel/Index/#Memory-management>
9. Understanding Linux Virtual Memory Management – IBM Developer  
<https://developer.ibm.com/articles/l-linux-memory/>
10. htop Official GitHub  
<https://github.com/htop-dev/htop>
11. systemd Resource Control  
<https://www.freedesktop.org/software/systemd/man/systemd.resource-control.html>
12. *Arquitetura de Sistemas Operacionais* – Francis B. Machado & Luiz Paulo Maia
13. Canonical Ltd. Ubuntu 24.04.2 LTS Desktop Guide. Disponível em:  
<https://help.ubuntu.com/lts/installation-guide/> . Acesso em: Jul. 2025.
14. Linux Foundation. Filesystem Hierarchy Standard (FHS) Revision 3.2. Disponível em: [https://refspecs.linuxfoundation.org/FHS\\_3.0/fhs-3.0.pdf](https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf) . Acesso em: Jul. 2025.
15. The ext4 FAQ. Disponível em:  
[https://ext4.wiki.kernel.org/index.php/Ext4\\_Howto](https://ext4.wiki.kernel.org/index.php/Ext4_Howto) . Acesso em: Jul. 2025.
16. Access Control Lists HOWTO. Linux Documentation Project. Disponível em:  
<https://tldp.org/HOWTO/ACL-HOWTO/> . Acesso em: Jul. 2025.
17. PAM-AUTH-UPDATE(8). Manual page do Ubuntu. Disponível em:  
<http://manpages.ubuntu.com/manpages/focal/en/man8/pam-auth-update.8.html> . Acesso em: Jul. 2025.

18. `cryptsetup(8)`. Manual page do Ubuntu (LUKS/dm-crypt). Disponível em: <http://manpages.ubuntu.com/manpages/focal/en/man8/cryptsetup.8.html>. Acesso em: Jul. 2025.
19. `fscrypt(8)`. Manual page do Ubuntu. Disponível em: <http://manpages.ubuntu.com/manpages/focal/en/man8/fscrypt.8.html>. Acesso em: Jul. 2025.
20. Uncomplicated Firewall (UFW) – Ubuntu Community Help Wiki. Disponível em: <https://help.ubuntu.com/community/UFW>. Acesso em: Jul. 2025.
21. AppArmor – Ubuntu Community Help Wiki. Disponível em: <https://help.ubuntu.com/community/AppArmor>. Acesso em: Jul. 2025.
22. policycoreutils (sestatus) – Ubuntu Packages. Disponível em: <https://packages.ubuntu.com/jammy/policycoreutils>. Acesso em: Jul. 2025.