

RELATÓRIO DO PROJETO
GERENCIAMENTO DE PROCESSOS

Disciplina: Sistemas Operacionais

Professor: Clóvis Ferraro

Grupo: 15

Sumário

1. Introdução
2. Metodologia
3. Comparação entre os Sistemas Operacionais
 - 3.1 Windows
 - 3.2 Android
 - 3.3 Linux
4. Análise Crítica
5. Conclusão
6. Autoavaliação
7. Referencias

1. Introdução

O gerenciamento de processos constitui um pilar fundamental para a estabilidade, desempenho e eficiência de qualquer Sistema Operacional existente, por estar ligado a funcionalidades muito íntimas e profundas dos dispositivos como a CPU e a memória. A importância desse gerenciamento torna-se ainda mais crítica em ambientes multitarefa, onde múltiplos processos competem por recursos limitados do sistema.

Esta etapa do projeto tem como objetivo implementar e analisar comandos de gerenciamento de processos em ambientes Windows, Linux e Android visando compreender o ciclo de vida dos processos, o controle de execução através de prioridades e a estrutura de gerenciamento de recursos em cada sistema operacional.

2. Metodologia

Windows: Para a utilização dessa análise devido à falta de sucesso da instalação de um sistema Windows em uma máquina virtual, foi utilizado um sistema de Windows 11 nativo do hardware, tomando as devidas seguranças e cautelas para evitar possíveis danos ao sistema principal.

Ferramentas

PowerShell

Gerenciador de tarefas

Comandos CMD

Lista de comandos utilizados:

- Get-Process(Mostra a lista de processos)
- Get-Process | Sort-Object CPU -Descending | Select-Object -First 10 (Mostra 10 processos que mais usam a cpu)
- Start-Process notepad / mspaint (iniciar aplicativos)
- Get-Process -Name "notepad", "mspaint" (Verificar os processos listados)
- Stop-Process -Name "mspaint" -Force / Stop-Process -Name "notepad" - Force (encerra os processos pelo nome)

Android: Os testes foram realizados utilizando o emulador **Pixel 4a – API 34** no Android Studio, acessado via comando **adb shell**. A partir desse terminal, executaram-se diversos comandos para listar, monitorar e manipular processos do sistema Android. Cada etapa foi registrada por meio de capturas de tela, representando visualmente os resultados obtidos. As figuras apresentadas a seguir correspondem a cada uma dessas etapas.

Linux: Para realizar os testes nessa etapa foi utilizado o sistema operacional Linux Lite 7.0 hospedado em uma máquina virtual do software “Oracle VirtualBox”

Detalhes Máquina Virtual	
RAM	4 GB
CPU	2
Memoria	25 GB
SO	Ubuntu (64-bit)

Comandos Utilizados

ps aux (Listar todos os processos)

top (Ver todos os processos em tempo real)

mousepad & (Abrir um editor mousepad já existente)

ps aux | grep mousepad (Ver o PID)

Kill (Encerrar o processo)

3.Comparação entre os Sistemas Operacionais

3.1 Windows

Etapas

Primeiro abri o PowerShell como administrador e listei todos os processos com o comando Get-Process.

PS C:\Windows\system32> Get-Process							
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
557	24	37648	40	1,63	3528	1	AccountsControlHost
139	9	2028	2492	1,03	8300	0	AggregatorHost
310	17	11412	4912	3,95	8668	1	AppActions
508	26	62068	32856	17,55	12680	1	ApplicationFrameHost
170	10	2100	1452	0,20	20616	1	AppShNotify
458	26	70684	65280	12,41	3864	0	audiogd
344	32	26472	60	1,41	12976	1	backgroundTaskHost
397	30	17164	8672	3,52	14728	1	backgroundTaskHost
802	37	23872	19324	25,25	16516	1	backgroundTaskHost
131	10	1780	1364	0,25	2604	0	BulletService
10227	15	3416	3492	14,11	3956	1	ColorEngine
124	11	1816	2540	0,05	2876	1	conhost
76	8	1096	900	0,00	13896	0	conhost
272	15	2816	6456	0,42	17128	1	conhost
208	14	2732	2624	0,14	23696	1	ConsultingMode
147	8	1412	1860	1,20	4580	0	ConsultingModeService
1287	131	69428	25332	26,61	11976	1	Copilot
734	33	37648	6228	6,58	4000	1	CrossDeviceResume
1020	34	3432	2704	129,86	392	1	csrss
884	31	2548	2268	11,59	932	0	csrss
691	25	30272	16064	159,42	11504	1	ctfmon
145	9	1548	320	0,11	7476	0	dasHost
262	12	3348	3016	1,28	4416	1	DAX3API
413	17	6460	3652	2,45	4620	0	DAX3API

Após isso fui visualizar quais são os 10 processos que mais estavam utilizando a CPU, com o comando Get-Process | Sort-Object CPU -Descending | Select-Object -First 10.

PS C:\Windows\system32> Get-Process Sort-Object CPU -Descending Select-Object -First 10							
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
1063	234	344128	199544	4.851,56	5388	0	MsMpEng
1505	132	575600	93412	4.662,31	16156	1	Discord
2722	56	243104	65752	4.355,55	1580	1	dwm
6690	0	60	2256	4.152,39	4	0	System
9723	200	437000	367800	1.386,14	8108	1	explorer
1298	66	135780	60804	809,20	6268	1	Notion
0	0	3672	108688	725,67	3000	0	Memory Compression
393	12	56904	34928	435,55	3220	0	svchost
287	16	5856	6728	427,66	1500	0	WUDFHost
1738	20	13208	17536	392,17	1380	0	svchost

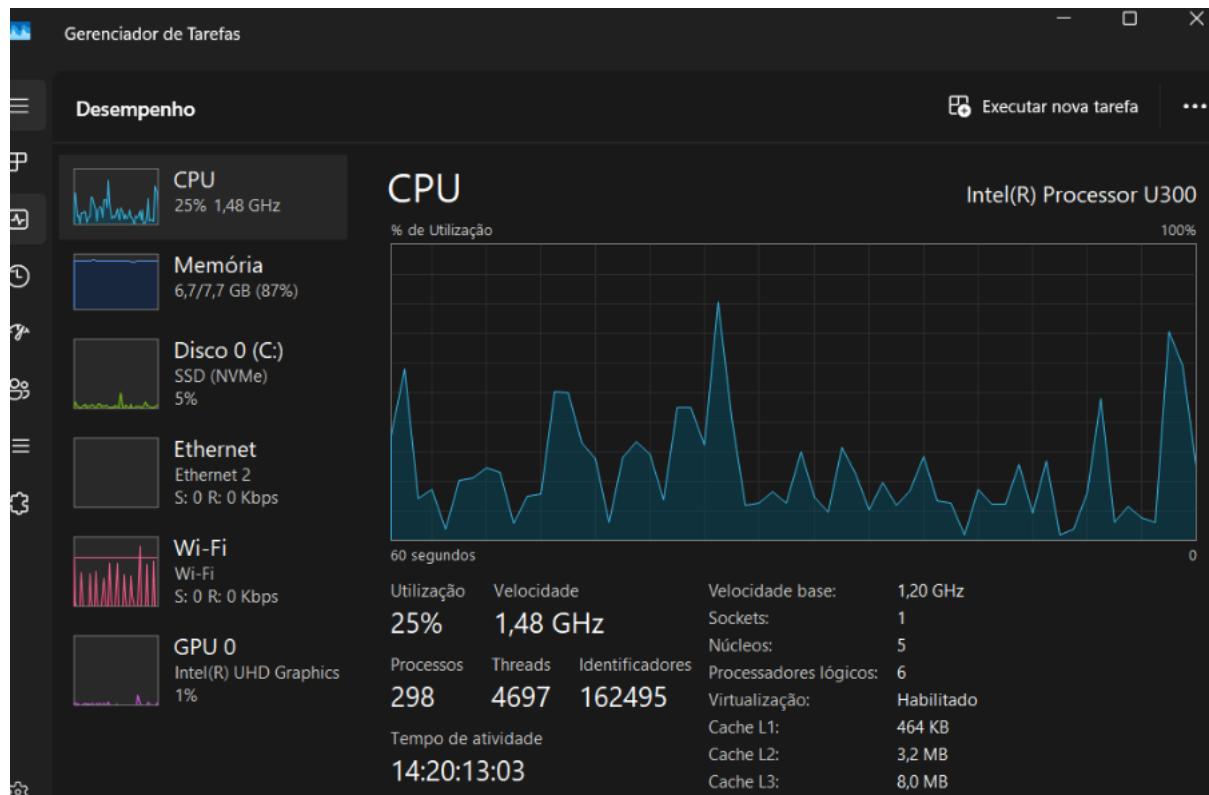
Gerenciador de Tarefas

No gerenciador de tarefas fui me habituar com sua interface e entender o que acontecia dentro dessa ferramenta. Lá é possível identificar diversos fatores como o armazenamento geral ou específico de processos, o desempenho da máquina e detalhes extras que ajudam a processar tarefas.

- **Processos**

Gerenciador de Tarefas		Digite um nome, editor ou PID para...				
Processos				Executar nova tarefa	Finalizar tarefa	G
	Nome	Status	58% CPU	87% Memória	3% Disco	9% Rede
Aplicativos (5)						
> Gerenciador de Tarefas			1,9%	92,0 MB	0 MB/s	0 Mbps
> Microsoft Edge (18)			3,8%	1.050,6 MB	0,1 MB/s	0,1 Mbps
> WhatsApp (2)			0%	106,4 MB	0,1 MB/s	0,1 Mbps
> Windows Explorer			0%	188,2 MB	0,1 MB/s	0 Mbps
> Windows PowerShell			0%	17,0 MB	0 MB/s	0 Mbps
Processos em segundo plano...						
> Antimalware Core Service			0%	3,2 MB	0 MB/s	0 Mbps
> Antimalware Service Executable			9,6%	126,2 MB	0,1 MB/s	0 Mbps
> Aplicativo de subsistema de s...			0%	0,8 MB	0 MB/s	0 Mbps
> AppActions			0%	2,1 MB	0 MB/s	0 Mbps
> Application Frame Host			1,9%	5,3 MB	0 MB/s	0 Mbps
> Assistência para Jogos			0%	0 MB	0 MB/s	0 Mbps
> Carregador CTF			0%	5,0 MB	0 MB/s	0 Mbps

- **Desempenho**



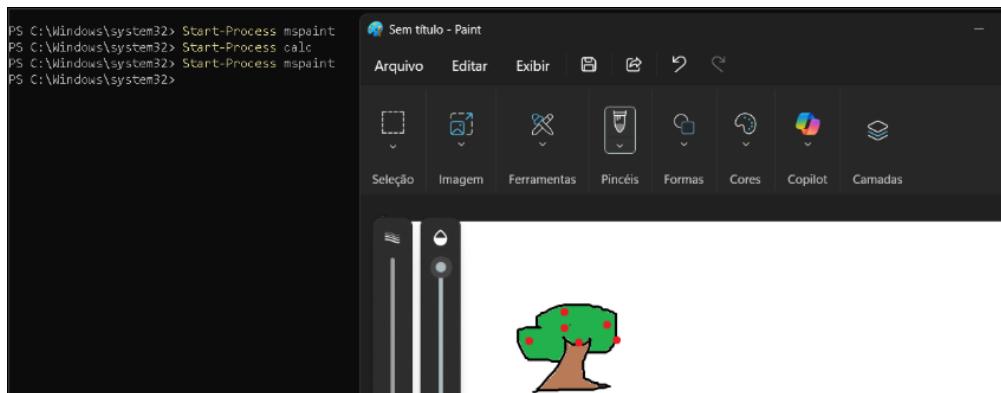
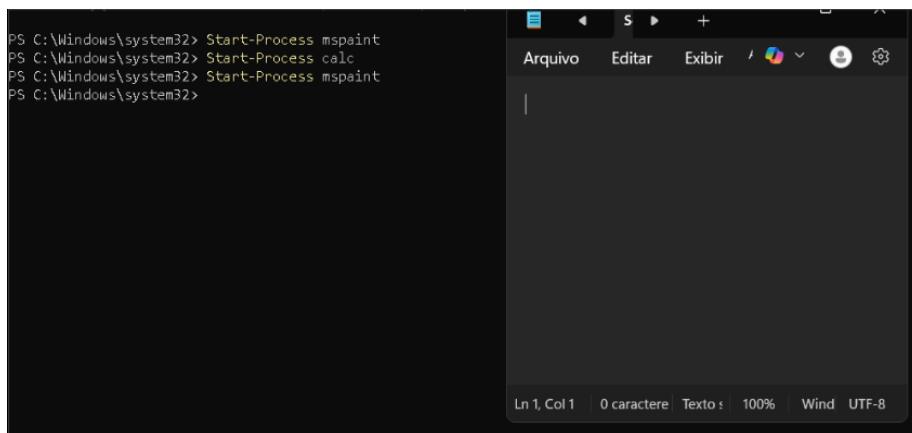
- **Detalhes**

Detalhes				Executar nova tarefa		Finalizar tarefa		...
	Nome	PID	Status	Nome de u...	CPU	Delta ...	Platafor...	Virtualizaç... d...
⊕	AccountsControlHos...	3528	Suspenso	andre	00	0 K	64 bits	Desabilitado
⊕	AggregatorHost.exe	8300	Em execução	SISTEMA	00	0 K	64 bits	Não permitido
⊕	AppActions.exe	8668	Em execução	andre	00	0 K	64 bits	Desabilitado
⌚	ApplicationFrameHo...	12680	Em execução	andre	00	32 K	64 bits	Desabilitado
⌚	AppVShNotify.exe	20616	Em execução	andre	00	0 K	64 bits	Desabilitado
⚡	audiogd.exe	3864	Em execução	SERVIÇO L...	00	0 K	64 bits	Não permitido
⚡	backgroundTaskHost...	12976	Suspenso	andre	00	0 K	64 bits	Desabilitado
⚡	backgroundTaskHost...	14728	Em execução	andre	00	0 K	64 bits	Desabilitado
⚡	backgroundTaskHost...	16516	Em execução	andre	00	0 K	64 bits	Desabilitado
⚡	backgroundTaskHost...	27364	Em execução	andre	00	0 K	64 bits	Desabilitado
⚡	BulletService.exe	2604	Em execução	SISTEMA	00	0 K	64 bits	Não permitido
⌚	ColorEngine.exe	3956	Em execução	andre	00	-76 K	64 bits	Desabilitado
⌚	conhost.exe	13896	Em execução	SISTEMA	00	0 K	64 bits	Não permitido
⌚	conhost.exe	17128	Em execução	andre	00	0 K	64 bits	Não permitido
⌚	conhost.exe	2876	Em execução	SISTEMA	00	0 K	64 bits	Não permitido
🖼️	ConsultingMode.exe	23696	Em execução	SISTEMA	00	0 K	64 bits	Não permitido
🖼️	ConsultingModeServ...	4580	Em execução	SISTEMA	00	0 K	64 bits	Não permitido
🖼️	Copilot.exe	11976	Em execução	andre	00	0 K	64 bits	Desabilitado
🖼️	CrossDeviceResume...	4000	Em execução	andre	00	0 K	64 bits	Desabilitado
🖼️	csrss.exe	932	Em execução	SISTEMA	00	0 K	64 bits	Não permitido
🖼️	csrss.exe	392	Em execução	SISTEMA	00	20 K	64 bits	Não permitido
⚙️	ctfmon.exe	11504	Em execução	andre	00	-8 K	64 bits	Desabilitado
⚙️	dasHost.exe	7476	Em execução	SERVIÇO L...	00	0 K	64 bits	Não permitido

PowerShell

```
PS C:\Windows\system32> Start-Process notepad  
PS C:\Windows\system32> Start-Process mspaint  
PS C:\Windows\system32> ■
```

Para testar o gerenciamento de processos foi usado os comandos Start-Process notepad/mspaint , para executar os aplicativos do “Paint” e “Bloco de anotações”.



E para verificar se os testes ocorreram bem utilizei o comando Get-Process -Name "notepad", "mspaint".

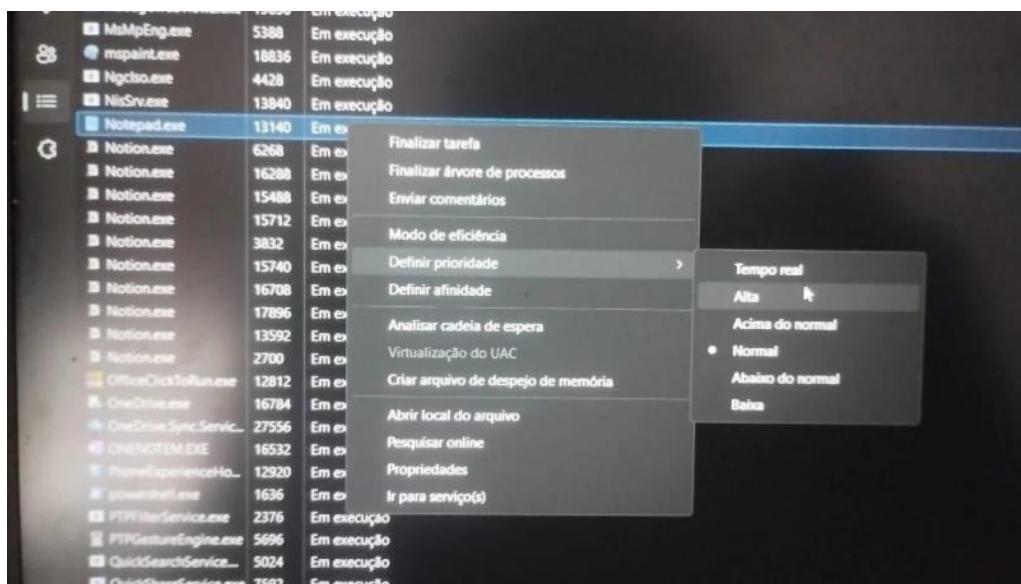
```
PS C:\Windows\system32> Get-Process -Name "notepad", "mspaint"
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
1254	70	172916	164160	25,80	28044	1	mspaint
1297	46	112084	115772	5,47	20084	1	Notepad

```
PS C:\Windows\system32>
```

Gerenciamento de tarefas

Após executar o início do processo dos aplicativos segui novamente ao gerenciamento de tarefas.



Segui para a aba de “detalhes” em busca do aplicativo do “bloco de notas” na intenção de alterar suas propriedades, para demonstrar que é possível ter essa liberdade pelo Windows.

Encerramento de processos

```
PS C:\Windows\system32> Stop-Process -Name "notepad" -Force
PS C:\Windows\system32> Stop-Process -Name "mspaint" -Force
PS C:\Windows\system32> ■
```

Após todos os testes encerrei os dois processos iniciados com os comandos Stop-Process – Name “mspaint”/”notepad” - Force.

3.2 Android

Figura 1 – Acesso ao shell do Android (adb shell).

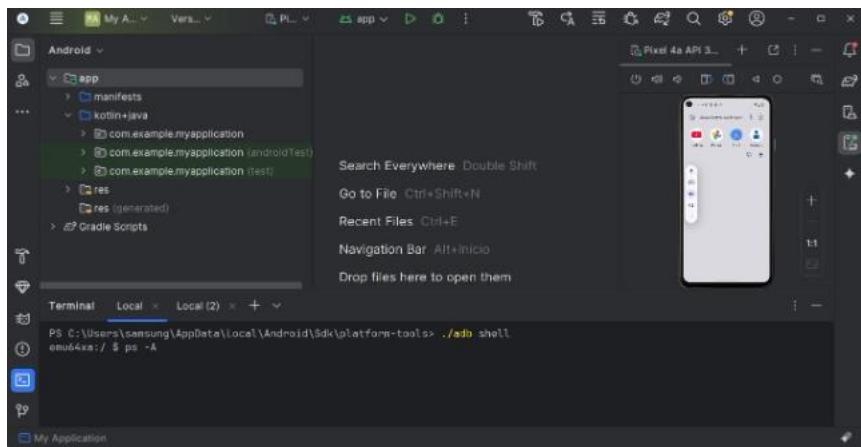


Figura 2 – Listagem de processos em execução (ps -A).

```
PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools> ./adb shell
emu64xa:/ $ ps -A
USER      PID  PPID      VSZ      RSS WCHAN          ADDR S NAME
root      1  0  10931008  4256 0           0 S init
root      2  0      0  0 0           0 S [kthreadd]
root      3  2      0  0 0           0 S [pool_workqueue_release]
root      4  2      0  0 0           0 I [kworker/R-rcu_g]
```

Figura 3 – Monitoramento de processos em tempo real (top).

```
emu64xa:/ $ top
```

Figura 4 – Filtragem de processo específico (ps | grep system_server).

```

Tasks: 412 total, 1 running, 411 sleeping, 0 stopped, 0 zombie
Mem: 2018800K total, 1878208K used, 140592K free, 4132K buffers
Swap: 1514096K total, 886400K used, 627696K free, 796396K cached
400%cpu 7%user 1%nice 75%sys 311%idle 6%io 0%irq 0%irq 0%host

 PID USER      PR NI VIRT  RES  SHR S[%CPU] %MEM    TIME+  ARGS
 851 system    18 -2 186 256M 157M S 34.6 12.9  5:53.89 system_server
 3598 u0_a185   20  0 166 133M 72M S 27.6  6.7  0:23.00 com.android.systemui

```

Figura 5 – Visualização detalhada do processo system_server (ps -A | grep system_server).

```

emu64xa:/ $ ps | grep system_server
1|emu64xa:/ $ ps -A|grep system_server
system      851  516  18773328 245592 0          0 S system_server
emu64xa:/ $ 

```

Figura 6 – Tentativa de encerrar processo do sistema (Operation not permitted).

```

root      4744  2      0      0 0          0 I [kworker/u12:0]
root      4784  2      0      0 0          0 I [kworker/u11:2-events_unbound]
root      5163  2      0      0 0          0 I [kworker/3:0]
u0_a134   5208  516  16731668 78960 0          0 S com.google.android.apps.restore
u0_a147   5267  516  17763696 181404 0          0 S com.google.android.googlequicksearchbox:search
u0_a165   5455  516  33555244 92080 0          0 S com.google.android.accessibility.switchaccess
u0_a146   5469  516  16569532 146420 0          0 S com.google.android.gms.ui
u0_a168   5506  516  16726892 85968 0          0 S com.google.android.marvin.talkback
root      5664  2      0      0 0          0 I [kworker/2:2H-kverityd]
root      5757  2      0      0 0          0 I [kworker/2:1-virtio_vsock]
root      5801  2      0      0 0          0 I [kworker/u10:2-events_unbound]
root      5802  2      0      0 0          0 I [kworker/u16:0-blk_crypto_wq]
root      5810  2      0      0 0          0 I [kworker/u16:1-blk_crypto_wq]
root      5816  2      0      0 0          0 I [kworker/u15:1]
shell     5822  4020  10832920 5820 0          0 R ps
emu64xa:/ $ kill 4666
/system/bin/sh: kill: 4666: Operation not permitted
1|emu64xa:/ $ 

```

Figura 7 – Criação de um processo de teste (sleep 100 &).

```
emu64xa:/ $ sleep 100 &
[1] 5826
emu64xa:/ $
```

Figura 8 – Verificação do processo criado (ps -A | grep sleep).

```
emu64xa:/ $ sleep 100 &
[1] 5826
emu64xa:/ $ ps -A | grep sleep
shell 5826 4020 10818456 4768 hrtimer_nanosleep 0 S sleep
emu64xa:/ $
```

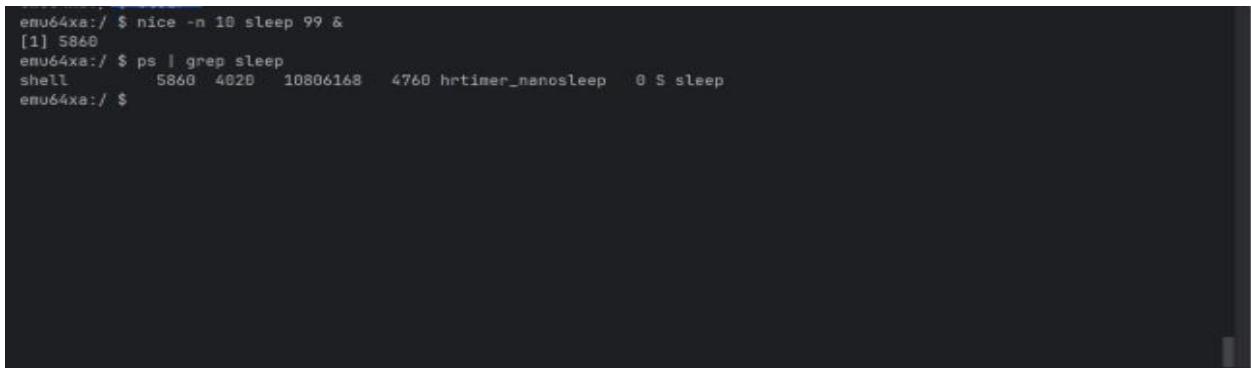
Figura 9 – Encerramento do processo de teste (kill PID).

```
emu64xa:/ $ sleep 100 &
[1] 5826
emu64xa:/ $ ps -A | grep sleep
shell 5826 4020 10818456 4768 hrtimer_nanosleep 0 S sleep
emu64xa:/ $ kill 5826
emu64xa:/ $
```

Figura 10 – Criação de processo com prioridade ajustada (nice -n 10 sleep 99 &).

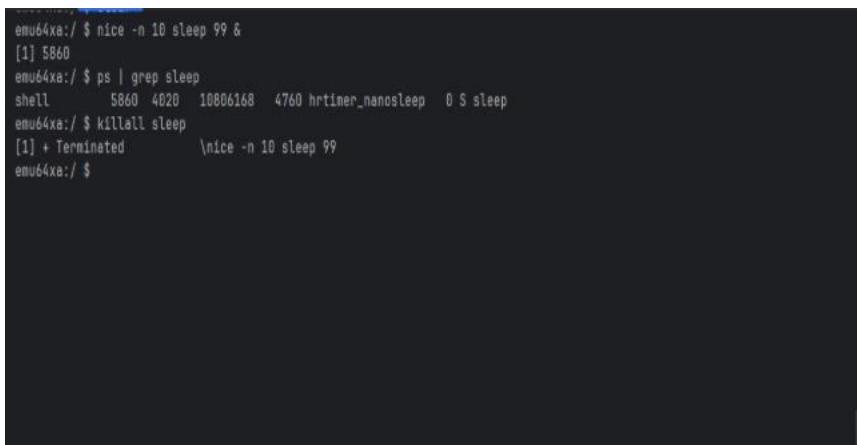
```
emu64xa:/ $ nice -n 10 sleep 99 &
```

Figura 11 – Verificação do processo criado (ps | grep sleep).



```
emu64xa:/ $ nice -n 10 sleep 99 &
[1] 5860
emu64xa:/ $ ps | grep sleep
shell      5860  4020  10806168  4760 hrtimer_nanosleep  0 S sleep
emu64xa:/ $
```

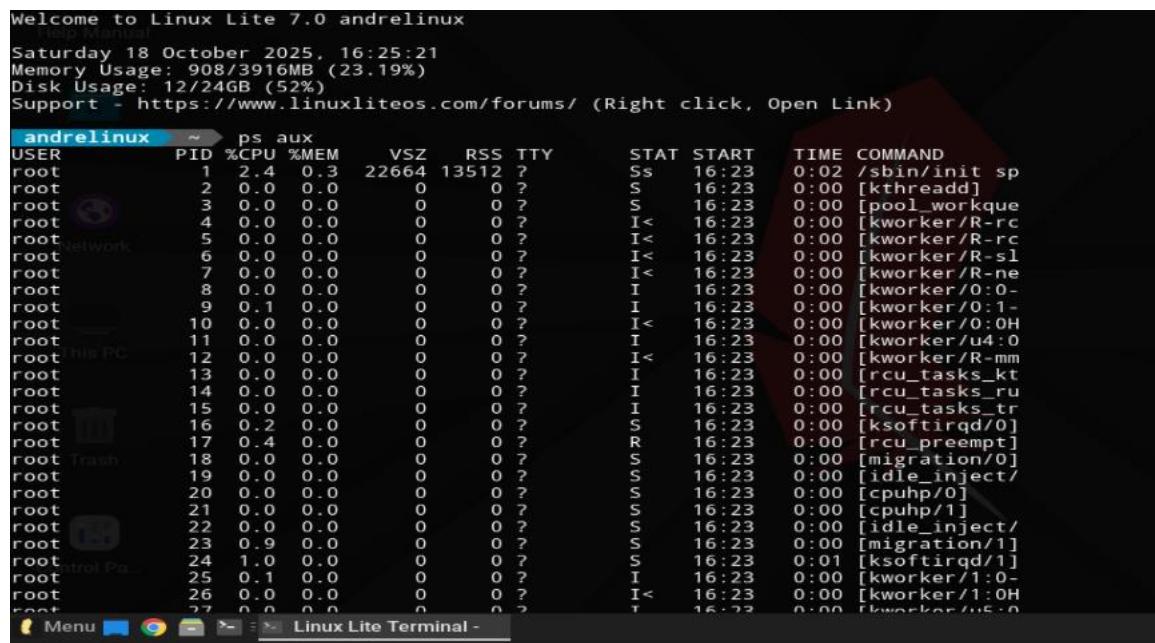
Figura 12 – Encerramento do processo com killall sleep.



```
emu64xa:/ $ nice -n 10 sleep 99 &
[1] 5860
emu64xa:/ $ ps | grep sleep
shell      5860  4020  10806168  4760 hrtimer_nanosleep  0 S sleep
emu64xa:/ $ killall sleep
[1]+  Terminated          nice -n 10 sleep 99
emu64xa:/ $
```

3.3 Linux

Para iniciar ativei a máquina virtual e entrei no terminal de comando do Linux Lite e usei o comando “ps aux” para listas os processos ativos da máquina e do sistema.



```
Welcome to Linux Lite 7.0 andrelinux
Help Manual
Saturday 18 October 2025, 16:25:21
Memory Usage: 908/3916MB (23.19%)
Disk Usage: 12/24GB (52%)
Support - https://www.linuxliteos.com/forums/ (Right click, Open Link)

andrelinux ~ ➔ ps aux
USER     PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1  2.4  0.3  22664 13512 ?        Ss  16:23   0:02 /sbin/init sp
root      2  0.0  0.0      0   0 ?        S  16:23   0:00 [kthreadd]
root      3  0.0  0.0      0   0 ?        S  16:23   0:00 [pool_workque
root      4  0.0  0.0      0   0 ?        I< 16:23   0:00 [kworker/R-rc
root      5  0.0  0.0      0   0 ?        I< 16:23   0:00 [kworker/R-rc
root      6  0.0  0.0      0   0 ?        I< 16:23   0:00 [kworker/R-sl
root      7  0.0  0.0      0   0 ?        I< 16:23   0:00 [kworker/R-ne
root      8  0.0  0.0      0   0 ?        I  16:23   0:00 [kworker/O-0-
root      9  0.1  0.0      0   0 ?        I  16:23   0:00 [kworker/O:1-
root     10  0.0  0.0      0   0 ?        I< 16:23   0:00 [kworker/O:0H
root     11  0.0  0.0      0   0 ?        I  16:23   0:00 [kworker/u4:0
root     12  0.0  0.0      0   0 ?        I< 16:23   0:00 [kworker/R-mm
root     13  0.0  0.0      0   0 ?        I  16:23   0:00 [rcu_tasks_kt
root     14  0.0  0.0      0   0 ?        I  16:23   0:00 [rcu_tasks_ru
root     15  0.0  0.0      0   0 ?        I  16:23   0:00 [rcu_tasks_tr
root     16  0.2  0.0      0   0 ?        S  16:23   0:00 [ksoftirqd/0]
root     17  0.4  0.0      0   0 ?        R  16:23   0:00 [rcu_preempt]
root     18  0.0  0.0      0   0 ?        S  16:23   0:00 [migration/0]
root     19  0.0  0.0      0   0 ?        S  16:23   0:00 [idle_inject/
root     20  0.0  0.0      0   0 ?        S  16:23   0:00 [cpuhp/0]
root     21  0.0  0.0      0   0 ?        S  16:23   0:00 [cpuhp/1]
root     22  0.0  0.0      0   0 ?        S  16:23   0:00 [idle_inject/
root     23  0.9  0.0      0   0 ?        S  16:23   0:00 [migration/1]
root     24  1.0  0.0      0   0 ?        S  16:23   0:01 [ksoftirqd/1]
root     25  0.1  0.0      0   0 ?        I  16:23   0:00 [kworker/1:0-
root     26  0.0  0.0      0   0 ?        I< 16:23   0:00 [kworker/1:0H
root     27  0.0  0.0      0   0 ?        T  16:23   0:00 [kworker/1:0]
```

Depois disso utilizei um comando de funcionalidade similar, porém este somente mostrou os processos ativos e em tempo real do sistema (comando utilizado: “top”)

```
andrelinux ~ top
top - 16:42:10 up 18 min, 1 user, load average: 0.28, 0.36, 0.38
Tasks: 187 total, 1 running, 186 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.9 us, 0.4 sy, 0.0 ni, 97.9 id, 0.0 wa, 0.0 hi, 0.9 si, 0.0 st
MiB Mem : 3916.1 total, 2307.2 free, 840.5 used, 998.9 buff/cache
MiB Swap: 2680.0 total, 2680.0 free, 0.0 used. 3075.6 avail Mem
User Files
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2167 root 20 0 343028 94092 50600 S 2.3 2.3 0:15.68 Xorg
147 root 20 0 0 0 I 0.3 0.0 0:06.02 kworker/1:3-mm_percpu_wq
2800 andreli+ 20 0 951484 102984 79448 S 0.3 2.6 0:03.86 xfwm4
2844 andreli+ 20 0 466864 43184 35376 S 0.3 1.1 0:00.41 panel-15-systra
3173 andreli+ 20 0 625636 49224 39580 S 0.3 1.2 0:02.58 xfce4-terminal
4382 andreli+ 20 0 14524 5760 3584 R 0.3 0.1 0:00.06 top
1 root 20 0 22664 13512 9288 S 0.0 0.3 0:03.32 systemd
2 root 20 0 0 0 S 0.0 0.0 0:00.02 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_g
5 root 0 -20 0 0 I 0.0 0.0 0:00.00 kworker/R-rcu_p
6 root 0 -20 0 0 I 0.0 0.0 0:00.00 kworker/R-slub_
7 root 0 -20 0 0 I 0.0 0.0 0:00.00 kworker/R-netns
11 root 20 0 0 0 I 0.0 0.0 0:00.00 kworker/u4:0-ext4-rsv-conversion
12 root 0 -20 0 0 I 0.0 0.0 0:00.00 kworker/R-mm_pe
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
15 root 20 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace_kthread
16 root 20 0 0 0 S 0.0 0.0 0:00.39 ksoftirqd/0
17 root 20 0 0 0 I 0.0 0.0 0:01.16 rcu_preempt
18 root rt 0 0 0 S 0.0 0.0 0:00.11 migration/0
19 root -51 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0
20 root 20 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
21 root 20 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1
22 root -51 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1
23 root rt 0 0 0 S 0.0 0.0 0:01.03 migration/1
24 root 20 0 0 0 S 0.0 0.0 0:02.42 ksoftirqd/1

```

Com isso feito parti para os testes, comecei abrindo um editor que já existe no sistema com o comando “mousepad &”, logo em seguida fui buscar o PID (o identificador de processo) desse editor, para assim ver o ciclo completo do processo de: “abrir -> vizualizar o PID -> encerrar”.

```
andrelinux ~ 1 mousepad PID
[1] 4434
andrelinux ~ ps aux | grep mousepad
andreli+ 4434 1.7 1.3 699684 54648 pts/0 S+ 16:48 0:00 mousepad
andreli+ 4455 0.0 0.0 9148 2176 pts/0 S+ 16:48 0:00 grep mousepad
```

Visto a linha do processo encerrei os mesmos com o comando “kill” seguido do código do PID (4434).

```
andrelinux ~ 1 kill 4434
andrelinux ~
```

4. Análise Crítica

- **Windows (André):** Com o fim dos testes é possível tirar uma boa conclusão e ideia de como esse gerenciamento de processos funciona dentro do sistema, constantemente durante a análise através do gerenciador de tarefas (na aba “processos” e “desempenho”) é possível notar mudanças de desempenho e memória de ferramentas e aplicativos, justamente pelos processos que estão ocorrendo como o uso de algum aplicativo ou comandos executados que conversam com o sistema.
- **Android (Eduardo):** Os experimentos realizados permitiram uma compreensão aprofundada sobre o gerenciamento de processos no Android. Através dos comandos **ps**, **top**, **grep**, **nice** e **kill**, foi possível observar, na prática, a forma como o sistema identifica, lista, prioriza e encerra processos. O Android, embora baseado no kernel Linux, impõe restrições de segurança que impedem o encerramento de processos essenciais, garantindo estabilidade e segurança. As dificuldades enfrentadas durante a execução — como permissões limitadas e respostas negativas de encerramento — foram fundamentais para compreender a diferença entre o gerenciamento de processos em um sistema desktop e em um ambiente móvel. Em síntese, o estudo demonstrou o ciclo completo de vida de um processo no Android, desde sua criação, listagem e monitoramento até o encerramento controlado, evidenciando a eficiência e robustez do sistema em gerenciar seus recursos de maneira segura e otimizada.
- **Linux (Vinicius):** O gerenciamento de processos no Linux demonstra uma filosofia de design centrada de forma tática e flexibilidade, mas que exige maior conhecimento técnico do usuário. Mesmo com comandos e etapas simples é possível notar que o Linux ofereceu saídas mais específicas e detalhadas.

5. Conclusão

Através da implementação prática dos comandos de gerenciamento de processos nos ambientes dos sistemas do Windows, Android e Linux, foi possível consolidar o entendimento sobre os conceitos fundamentais do gerenciamento de processos,

bem como suas diferenças essenciais. Por meio das interfaces de cada sistema conseguimos obter as informações desejadas, nos aprofundando ainda mais nas etapas que o sistema operacional executa e passa, além de utilizarmos comandos mais complexos e com saídas que exigem maior interpretação por parte dos usuários.

Os objetivos propostos nesta etapa do relatório foram plenamente alcançados. Foi possível implementar e analisar com sucesso os comandos de gerenciamento de processos em todos os ambientes, monitorando processos ativos, consumo de recursos, identificando PIDs e manipulando prioridades de execução.

Como análise da abordagem de design dos sistemas ainda é notório a facilidade do Windows de demonstrar informações aos usuários, comparado com o Linux e o Android que apresentam saídas similares com o Linux sendo predominantemente mais técnico e o Android mais simplório e menos convidativo.

6. Autoavaliação

Dificuldades encontradas

Durante a execução dos experimentos, algumas dificuldades foram observadas, principalmente relacionadas às permissões de usuário e à limitação do shell do Android. Em diversas tentativas de encerramento de processos críticos, como o **PID 1 (init)**, o comando retornou *Operation not permitted*. Esse comportamento é esperado, já que o Android protege seus processos essenciais para evitar falhas graves no sistema. Outra dificuldade foi a ausência de resultados ao executar o comando **ps | grep system_server** em algumas versões do Android, devido às mudanças na nomenclatura dos processos e nas restrições impostas ao usuário shell. Além disso, o comando **kill** apresentou erros de sintaxe quando os símbolos '<' e '>' não foram substituídos corretamente pelo número real do processo (PID). Esses desafios reforçam a importância de compreender os privilégios de execução e o funcionamento interno do Android em relação ao Linux tradicional.

7. Referências

PID: <https://youtu.be/EbWKIkF2L1I?si=QVkc2g4GwUgJEsi1>

KILL Linux: https://youtu.be/xwcD5xrR5nE?si=YtxxG_63BQ1SM5Ch

Comandos do Linux: https://www.devmedia.com.br/comandos-importantes-linux/23893?gad_source=1&gad_campaignid=23152113331&gclid=Cj0KCQjw9czHBhCyARIsAFZIN8TOpQum75VFDACIOR2GdSqu2nlg4Pl0-BXwfu-537vX_fqov_jP77gaAmjQEALw_wcB

Gerenciamento Android:

<https://developer.android.com/guide/components/processes-and-threads?hl=pt-br>