

**RELATORIO DO PROJETO
GERENCIAMENTO DE MEMÓRIA**

Disciplina: Sistemas Operacionais

Professor: Clóvis Ferraro

Grupo:15

Sumário

1. Introdução
2. Metodologia
3. Comparação entre sistemas
 - 3.1 Windows
 - 3.2 Android
 - 3.3 Linux
4. Análise Crítica
5. Conclusão
6. Autoavaliação
7. Referências

1. Introdução

Esta etapa do relatório irá explorar o gerenciamento de memória e como ela é executada em cada um dos três principais sistemas operacionais: **Windows, Android e Linux**, com abordagens que exploram cada funcionalidade desses sistemas.

A gestão eficiente de memória constitui um dos pilares fundamentais para o desempenho e estabilidade para qualquer sistema operacional. Devido as aplicações demandarem quantidades crescentes de recursos computacionais, a capacidade do sistema em alocar, monitorar e otimizar o uso da memória torna-se muito importante para garantir uma experiência de usuário fluida e eficiente, fazendo com que todos os componentes e ferramentas funcionem de forma saudável.

2. Metodologia

- **Windows:**

Devido a impossibilidade da configuração de uma máquina virtual com o sistema operacional Windows, foi utilizado um sistema operacional já nativo do hardware do usuário, mais especificamente foi utilizado um sistema do Windows 11 e operações que não ofereçam riscos ao dispositivo de testes, garantindo um ambiente controlado.

Também foi utilizado as seguintes ferramentas:

- Task Manager (Gerenciamento de tarefas)
- Monitor de Recursos
- PowerShell

Também foi utilizado dois comandos:

- “**Get-Process | Sort-Object -Property WS -Descending | Select-Object -First 10 Name,**
@{Name="Memory(MB)";Expression={[math]::Round(\$_.WS/1MB,2)}},
@{Name="Private(MB)";Expression={[math]::Round(\$_.PrivateMemorySize/1MB,2)}}} Para a identificação de 10 processos que mais utilizam memória.
- “**Get-CimInstance -ClassName Win32_ComputerSystem | Select-Object TotalPhysicalMemory**” Para mostrar as estatísticas do sistema.

- **Android:**

Foram utilizados comandos e ferramentas de linha de comando (ADB) para monitoramento do consumo de memória e o comportamento dos processos no sistema Android.

Comandos utilizados: dumpsys meminfo, dumpsys meminfo [package], procrank, ps -A | grep lmkd, cat /proc/meminfo.

As capturas de tela a seguir documentam cada etapa do procedimento.

- **Linux:**

Configuração do ambiente:

- **Software:** Oracle VirtualBox 7.x
- **Sistema Convidado:** Linux Lite 6.x (baseado em Ubuntu 22.04 LTS)
- **Memória da VM:** 2 GB de RAM e 1 processador
- **Disco virtual:** 20 GB (VDI, dinamicamente alocado)

Instalação da ISO:

A ISO do Linux Lite foi baixada do site oficial:

<https://www.linuxliteos.com/download.php>

Instalação realizada normalmente com as configurações padrões

Execução dos testes:

- o Foram realizados testes práticos com o terminal para observar o consumo de memória em diferentes cenários:
 - Sistema ocioso (sem processos extras);
 - Execução simultânea de aplicativos (navegador, terminal e gerenciador de arquivos);
 - Forçamento de uso de swap.

Comandos utilizados

Durante os testes foram usados os seguintes comandos:

- Top ou htop
- Free -h
- vmstat
- swapon – show

3.Comparação entre sistemas

Windows

Os testes a seguir foram realizados com o objetivo de compreender e entender processos envolvendo a memória do dispositivo em interação com o Sistema operacional. Com os testes foi realizado:

- Identificação de vazamentos de memória
- Entendimento do comportamento da memória no Windows
- Entendimento do comportamento da memória no Windows

1. Preparação Inicial dos testes

Passo 1: Iniciar o Task Manager (Gerenciador de tarefas)

Pressionando as teclas: “**Ctrl**” + “**Shift**” + “**Esc**” o gerenciador de tarefas do dispositivo é iniciado.

Nome	Status	22% CPU	76% Memória	2% Disco	0% Rede
> Microsoft Edge (17)	●	0%	857,7 MB	0,1 MB/s	0,1 Mbps
> Notion (9)	●	2,0%	280,1 MB	0 MB/s	0 Mbps
> Microsoft 365 Copilot (7)	●	0%	166,2 MB	0 MB/s	0 Mbps
> Windows Explorer		5,9%	120,7 MB	0,1 MB/s	0 Mbps
> Antimalware Service Executable		2,0%	95,5 MB	0 MB/s	0 Mbps
> Gerenciador de Tarefas		5,9%	66,2 MB	0 MB/s	0 Mbps
Gerenciador de Janelas da Áre...		3,9%	55,6 MB	0,1 MB/s	0 Mbps
Secure System		0%	38,6 MB	0 MB/s	0 Mbps
> Pesquisar (7)		0%	36,6 MB	0 MB/s	0 Mbps
> McAfee Framework Host Servi...		0%	21,8 MB	0 MB/s	0 Mbps
> Windows Widgets (7)	●	0%	14,6 MB	0 MB/s	0 Mbps
> Host de Serviço: UtcSvc		0%	14,5 MB	0 MB/s	0 Mbps
> Host de Serviço: Serviço de Re...		0%	13,9 MB	0,1 MB/s	0 Mbps
> Host de Serviço: Serviço de Po...		0%	13,8 MB	0 MB/s	0 Mbps
Microsoft OneDrive		0%	11,7 MB	0 MB/s	0 Mbps

Passo 2: Configuração do Task Manager para análise

Para visualizarmos os dados foi selecionado colunas que nos mostraram informações do uso da memória no dispositivo.

- Para selecionar dados específicos basta clicar no botão direto do mouse nos cabeçalhos das colunas e selecionar a informação de preferência.

Status	2% CPU	81% Memória	Tipo
●	2,0%	67,7 MB	✓ Status
	0%	1,5 MB	✓ Fornecedor
	0%	1,5 MB	✓ PID
	0%	0,7 MB	✓ Nome do processo
	0%	0,7 MB	✓ Linha de comando
●	0%	284,3 MB	✓ CPU
	0%	0,1 MB	✓ Memória
	0%	0 MB	✓ Disco ←
	0%	0 MB	✓ Rede
	0%	37,4 MB	GPU
	0%	107,3 MB	Mecanismo de GPU
●	0%	982,2 MB	✓ Uso de energia
	0%	2,4 MB	✓ Tendência de uso de energia
	0%	0,4 MB	Valores de recursos >
	0%	4,4 MB	0 MB/s 0 Mbps Muito baixo

Processos

Nome	Status	15% CPU	80% Memória	2% Disco	0% Rede	Uso de energia	Tendência de uso de energia
> Antimalware Service Executable		13,1%	90,4 MB	0 MB/s	0 Mbps	Alta	Baixa
> Gerenciador de Tarefas		2,2%	69,9 MB	0 MB/s	0 Mbps	Baixa	Muito baixo
> Host de Serviço: CaptureService		0%	1,7 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> wsappx		0%	4,0 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> SnippingTool		0%	34,4 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> Interrupções do sistema		0%	0 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> Gerenciador de Janelas da Área de Trabalho		0%	53,5 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> Microsoft Edge (17)	●	0%	1.004,6 MB	0,1 MB/s	0,1 Mbps	Muito baixo	Muito baixo
> System		0%	0,1 MB	0,1 MB/s	0 Mbps	Muito baixo	Muito baixo
> Windows Explorer		0%	120,2 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> Windows Driver Foundation		0%	2,5 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> Notion (9)	●	0%	286,7 MB	0,1 MB/s	0 Mbps	Muito baixo	Muito baixo
> Host de Serviço: Serviço de Ponto de Recuperação		0%	17,1 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> Samsung Update		0%	8,9 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> FocalTech RPC Server		0%	0,5 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> Host de Serviço: Gerenciador de Atualizações		0%	1,2 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> SamsungSystemSupportService		0%	1,0 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
> Microsoft Copilot		0%	7,8 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
< Microsoft Windows Search Filter		0%	1,5 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
< Microsoft Windows Search Filter		0%	1,4 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
< Microsoft Windows Search Processor		0%	1,9 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo

Passo 3: Análise do uso de memória

No Task Manager iremos selecionar a opção de nosso interesse de análise, neste caso a aba “memória”.



Nesta aba foi localizado e catalogado:

- **Gráfico de uso em tempo real**



- **Memoria em uso e disponível**

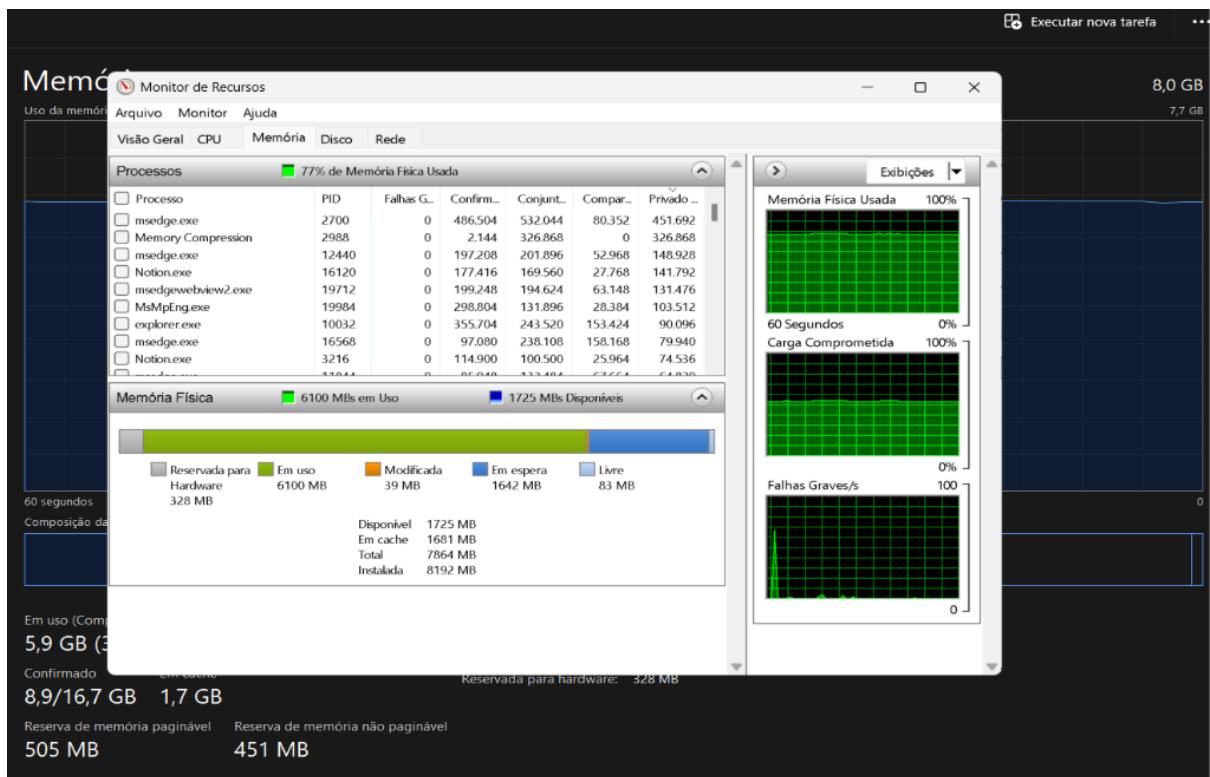
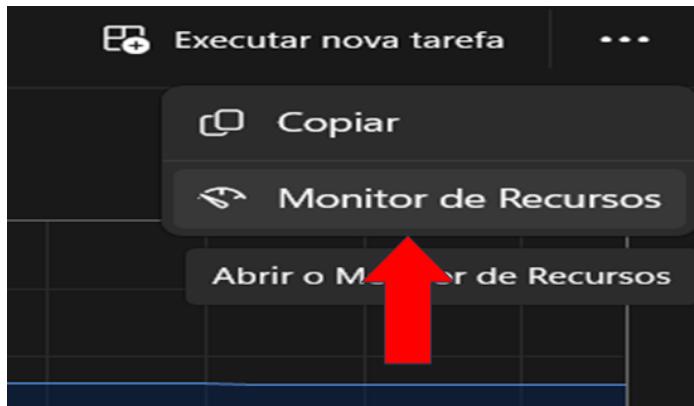
Em uso (Compactada)	Disponível
6,3 GB (410 MB)	1,4 GB

- **Velocidade e slots da memória**

Velocidade:	4267 MT/s
Slots usados:	8 de 8

Passo 4: Monitor de Recursos

No canto superior direito iremos abrir o monitor de recursos e ir para os dados da “memória”



Na imagem podemos identificar o gráfico de memória física usado, a memória que está em uso ou livre e a memória que está reservada para o hardware.

2. Análise de memória e processos

Passo 1: Visualizar processos que mais consomem a memória

Voltando ao Task Manager fui identificar quais processos mais consomem a memória e quais são os que menos consomem.

- **Maior consumo**

Nome	Status		78% Memória
> Microsoft Edge (12)	●		683,6 MB
> Notion (9)	●		291,6 MB
> Antimalware Service Executable			99,8 MB
> Windows Explorer			90,0 MB
> Gerenciador de Tarefas			77,1 MB
> Microsoft 365 Copilot (7)	●		40,8 MB
> Iniciar			40,7 MB
> Gerenciador de Janelas da Áre...			39,1 MB
> Secure System			38,6 MB
> Pesquisar (7)			34,0 MB
> Monitor de Recursos e Desem...			32,6 MB

- Menor consumo

Nome	Status		80% Memória
> Configurações	(II)		0 MB
> Tela de Bloqueio padrão do ...	(II)		0 MB
■ Interrupções do sistema			0 MB
■ System			0,1 MB
■ SnippingTool	Suspenso		0,1 MB
■ Gerenciador de Sessão do Wi...			0,1 MB
■ Device Association Framework...			0,2 MB
■ Usermode Font Driver Host			0,2 MB
> Host de Serviço: Serviço de Ins...			0,3 MB
■ Samsung Consulting Mode Ge...			0,3 MB

Passo 2: Análise com o uso do PowerShell

Entrando no PowerShell como administrador executei os comandos para:

- **Identificar os 10 processos que utilizam mais memória**

```
"Get-Process | Sort-Object -Property WS -Descending | Select-Object -First 10
Name, @{Name="Memory(MB)";Expression={[math]::Round($_.WS/1MB,2)}},
@{Name="Private(MB)";Expression={[math]::Round($_.PrivateMemorySize/1MB,2)}}"
```

```

Administrator: Windows PowerShell
O Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos.

PS C:\Windows\system32> Get-Process | Sort-Object -Property WS -Des
                         ry(MB); Expression={[math]::Round($_.WS/1MB,2)}}, @{@Name="Private(MB
,2)}}

Name          Memory(MB)  Private(MB)
----          -----
msedge        697,54     664,52
explorer      278,04     353,39
msedgewebview2 242,34     196,65
MsMpEng       203,9      305,96
msedge        203,68     95,53
Taskmgr       178,84     138,31
Notion        167,31     176,48
msedge        164,38     223,82
WebViewHost   141,15     68,84
msedgewebview2 127,14     39,5

```

- **Estatísticas do sistema**

“Get-CimInstance -ClassName Win32_ComputerSystem | Select-Object TotalPhysicalMemory”

```

PS C:\Windows\system32> Get-CimInstance -ClassName Win32_ComputerSystem | Select-Object TotalPhysicalMemory

TotalPhysicalMemory
-----
8246001664

PS C:\Windows\system32>

```

3. Detecção de vazamento de memória

No Powershell fiz uma análise para identificar se algum processo estava tendo um possível vazamento de memória, em 3 minutos foi observado se ocorreriam alguma alteração:

- **18:28**

Nome	Status	Memória
> Microsoft Edge (12)	⚡	997,6 MB
> Notion (9)	⚡	292,3 MB
> Pesquisar (7)		185,6 MB

- **18:29**

Nome	Status	76% Memória
> Notion (9)	OK	295,2 MB
> Microsoft Edge (12)	OK	237,1 MB
> Pesquisar (7)		181,3 MB

- 18:30

Nome	Status	76% Memória
> Notion (9)	OK	296,5 MB
> Microsoft Edge (12)	OK	235,1 MB
> Pesquisar (7)		180,3 MB

Hora	Processo 1	MB	Processo 2	MB	Processo 3	MB
18:28	Microsoft Edge	997	Notion	292	Pesquisar	185
18:29	Microsoft Edge	237	Notion	295	Pesquisar	181
18:30	Microsoft Edge	235	Notion	296	Pesquisar	180

Realizando esse monitoramento com o gerenciador de tarefas foi concluído

que:

- Nenhum processo apresentou crescimento significativo de memória.
- O Microsoft Edge demonstrou um funcionamento normal reduzindo seu uso de memória após não o utilizar por um tempo.
- Não foi detectado um vazamento de memória

Android

Análise do Uso de Memória via Interface Gráfica

O primeiro passo foi acessar as Opções do Desenvolvedor → Memória, onde é possível visualizar o uso médio de RAM e a distribuição por aplicativos.

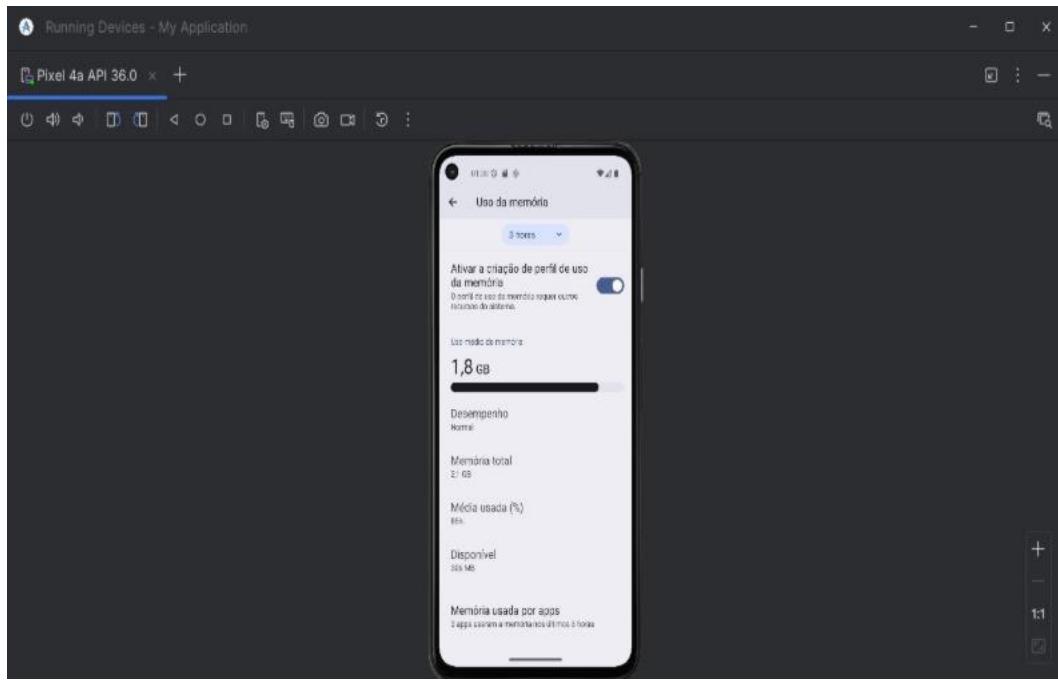


Figura 1 - Uso total de memória RAM e desempenho geral do sistema Android.

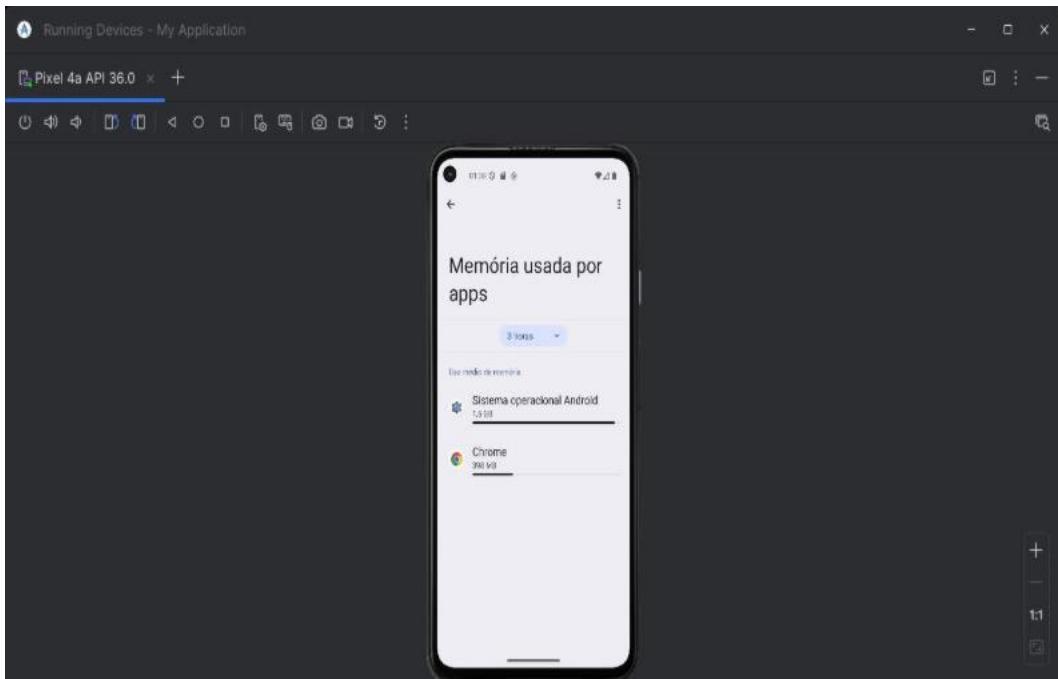
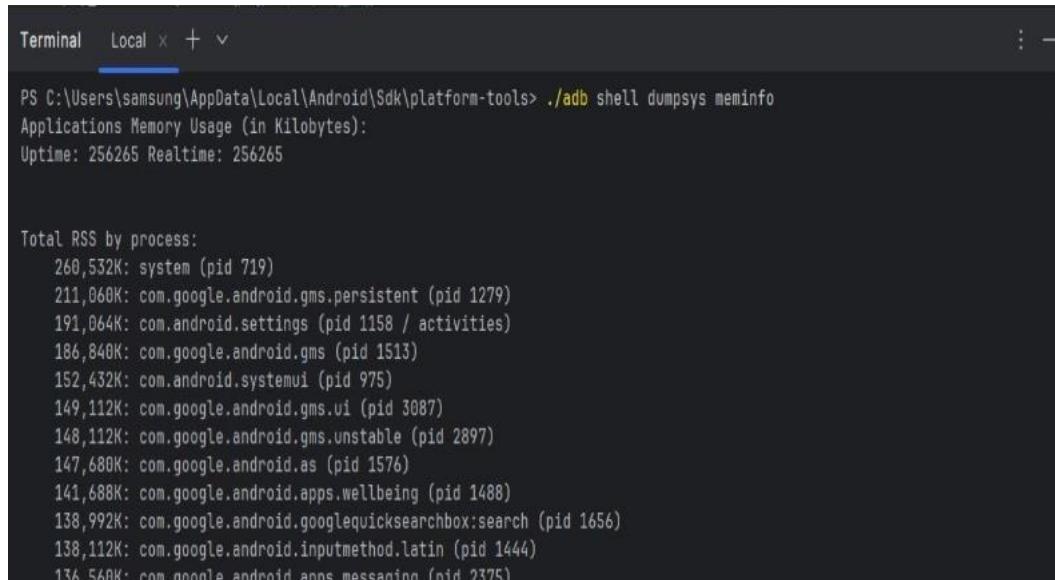


Figura 2 - Memória utilizada por aplicativos ativos e em segundo plano.

Comando dumpsys meminfo

O comando abaixo foi executado para obter informações detalhadas do uso de memória no sistema: adb shell dumpsys meminfo

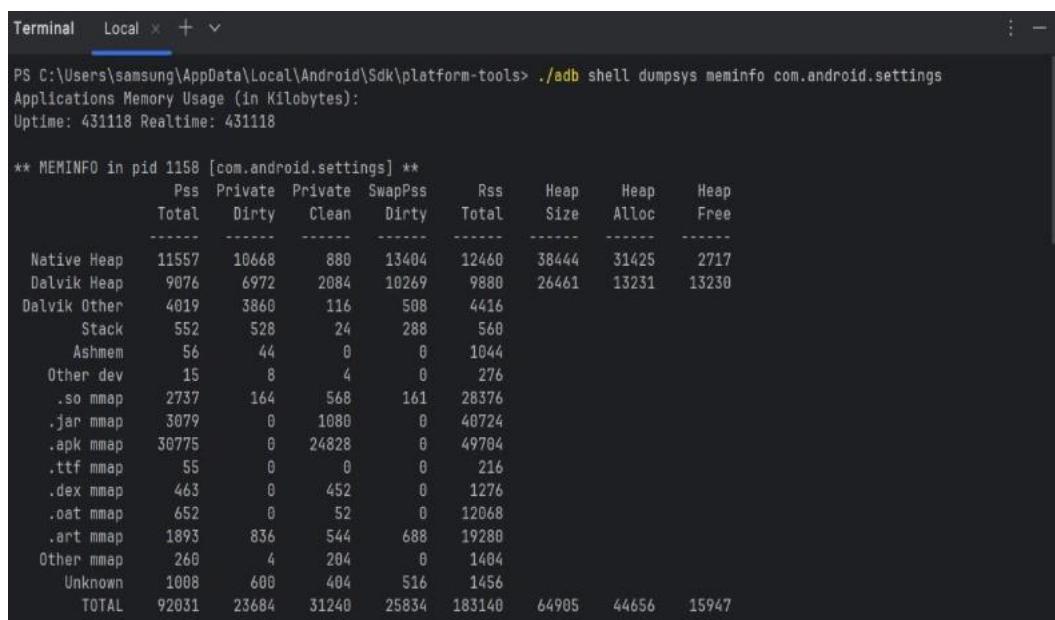


```
PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools> ./adb shell dumpsys meminfo
Applications Memory Usage (in Kilobytes):
Uptime: 256265 Realtime: 256265

Total RSS by process:
260,532K: system (pid 719)
211,060K: com.google.android.gms.persistent (pid 1279)
191,064K: com.android.settings (pid 1158 / activities)
186,840K: com.google.android.gms (pid 1513)
152,432K: com.android.systemui (pid 975)
149,112K: com.google.android.gms.ui (pid 3087)
148,112K: com.google.android.gms.unstable (pid 2897)
147,680K: com.google.android.es (pid 1576)
141,688K: com.google.android.apps.wellbeing (pid 1488)
138,992K: com.google.android.googlequicksearchbox:search (pid 1656)
138,112K: com.google.android.inputmethod.latin (pid 1444)
136,540K: com.google.android.apps.messaging (pid 2375)
```

Figura 3 - Resultado do comando dumpsys meminfo mostrando consumo de memória por processo.

Também foi feita a análise de dois aplicativos distintos (Configurações e Chrome) para observar o uso de PSS e RSS individualmente.



```
PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools> ./adb shell dumpsys meminfo com.android.settings
Applications Memory Usage (in Kilobytes):
Uptime: 431118 Realtime: 431118

** MEMINFO in pid 1158 [com.android.settings] **
      Pss   Private  Private  SwapPss     Rss    Heap    Heap    Heap
      Total     Dirty     Clean     Dirty     Total     Size    Alloc    Free
-----+-----+-----+-----+-----+-----+-----+-----+-----+
Native Heap  11557    10668      880   13404   12460   38444   31425   2717
Dalvik Heap   9076     6972    2084   10269    9880   26461   13231  13230
Dalvik Other   4019     3860      116     508    4416
  Stack      552      528      24     288      560
  Ashmem       56       44       0       0    1044
  Other dev      15        8       4       0     276
  .so mmap    2737      164      568     161   28376
  .jar mmap   3079        0     1080        0   40724
  .apk mmap  30775        0    24828        0   49704
  .ttf mmap      55        0       0       0     216
  .dex mmap    463        0      452        0   1276
  .oat mmap    652        0      52        0   12068
  .art mmap   1893      836      544     688   19280
Other mmap    260        4      204        0    1404
Unknown      1008      600      404      516    1456
TOTAL     92031   23684   31240   25834  183140   64905   44656   15947
```

Figura 4 - Uso de memória do aplicativo Configurações (com.android.settings).

```
Terminal Local × + ▾

PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools> ./adb shell dumpsys meminfo com.android.settings
Applications Memory Usage (in Kilobytes):
Uptime: 537274 Realtime: 537274

** MEMINFO in pid 3560 [com.android.settings] **
      Pss  Private  Private SwapPss   Rss   Heap   Heap
      Total    Dirty    Clean    Dirty   Total   Size Alloc   Free
-----+-----+-----+-----+-----+-----+-----+-----+
Native Heap     1096     948     140    918   2092   12240   5392   2130
Dalvik Heap     461     420     24    144   1448   2399   1800   599
Dalvik Other     361     244     108    151    984
  Stack       124     108     16    120    132
  Ashmem       10      0      0      0    576
Other dev        4      0      4      0    192
.so mmap      925     16    420     78   18924
.jar mmap      594      0      0      0   26608
.apk mmap     5878      0    5876      0   6092
.ttf mmap      612      0    612      0    612
.dex mmap      910      0    900      0   1672
.oat mmap      163      0      0      0   8672
.art mmap     922     352    268    367   18340
Other mmap      11      4      0      0    924
Unknown       110     48     56    276    644
TOTAL      14227   2140   8424   2046   87912   14639   7192   2729
```

Figura 5 - Uso de memória do aplicativo Chrome (com.android.chrome).

Comando procrank

O comando procrank lista os processos e suas métricas de memória (VSS, RSS, PSS, USS).

adb shell procrank

```
Terminal Local × + ▾

PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools> ./adb shell procrank
PID      Vss      Rss      Pss      Uss      Swap      PSwap      USwap      cmdline
4058  1088264K  5040K  2086K  2052K      0K      0K      OK  procrank
241   10803296K  2524K  165K      4K    724K      3K      OK  /system/bin/sh
-----+-----+-----+-----+-----+-----+-----+-----+
                  2251K    2056K    724K      3K      0K  TOTAL

ZRAM: 0K physical used for 676492K in swap (1514092K total swap)
RAM: 2018796K total, 104680K free, 4684K buffers, 837712K cached, 7836K shmem, 210048K slab
PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools>
```

Figura 6 - Resultado do procrank mostrando PSS e RSS de processos ativos.

Processo LMKD

O Android moderno utiliza o Low Memory Killer Daemon (LMDK) em vez do antigo módulo lowmemorykiller. O comando abaixo mostra o processo em execução responsável por gerenciar a liberação de memória. `adb shell ps -A | grep lmkd`

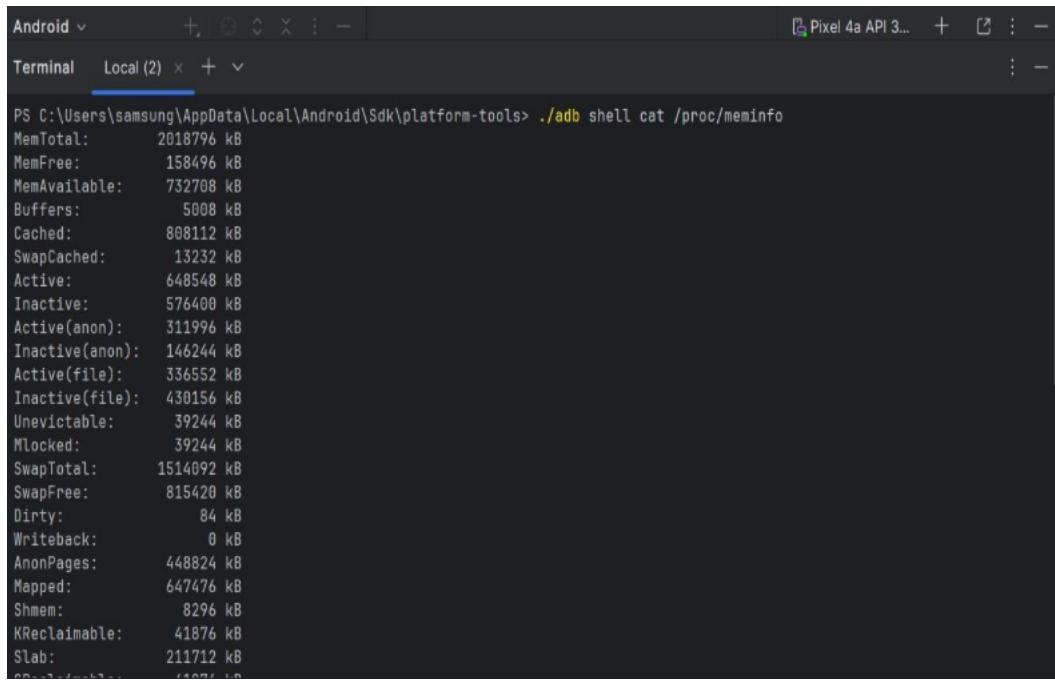
```
Android ▾ Pixel 4a API 3... + ⌂ : -  
Terminal Local x + ×  
emu64xa:/ $ ps -A | grep lmkd  
lmkd      238      1  10810912  4212 0          0 S lmkd  
emu64xa:/ $
```

Figura 7 - Processo lmkd ativo no sistema, responsável pelo gerenciamento agressivo de memória.

Estrutura de Memória Global (/proc/meminfo)

Por fim, o comando abaixo fornece um panorama geral da memória física e virtual do dispositivo:

adb shell cat /proc/meminfo



A screenshot of a terminal window titled "Pixel 4a API 3...". The window has tabs for "Terminal" and "Local (2)". The "Terminal" tab is active, showing the command `./adb shell cat /proc/meminfo` and its output. The output lists various memory metrics in kB:

```
PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools> ./adb shell cat /proc/meminfo
MemTotal:       2018796 kB
MemFree:        158496 kB
MemAvailable:   732708 kB
Buffers:         5008 kB
Cached:          808112 kB
SwapCached:     13232 kB
Active:          648548 kB
Inactive:        576400 kB
Active(anon):   311996 kB
Inactive(anon): 146244 kB
Active(file):   336552 kB
Inactive(file): 430156 kB
Unevictable:    39244 kB
Mlocked:        39244 kB
SwapTotal:      1514092 kB
SwapFree:       815420 kB
Dirty:           84 kB
Writeback:       0 kB
AnonPages:      448824 kB
Mapped:          647476 kB
Shmem:           8296 kB
KReclaimable:   41876 kB
Slab:            211712 kB
```

Figura 8 - Estatísticas globais de memória do sistema Android (RAM, cache, swap, etc.).

Linux

Top ou htop : monitora os processos ativos e consumo de recursos em tempo real.

```

top - 20:22:06 up 31 min, 1 user, load average: 0.04, 0.03, 0.08
Tasks: 165 total, 1 running, 164 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.4 us, 0.3 sy, 0.0 ni, 97.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 8137.5 total, 6433.4 free, 815.2 used, 1139.5 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used, 7322.2 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2169 root 20 0 349932 101068 50672 S 1.3 1.2 0:05.24 Xorg
3900 linuxli+ 20 0 14504 5888 3712 R 0.7 0.1 0:00.14 top
57 root 20 0 0 0 0 I 0.3 0.0 0:00.14 kworker/u2:5-flush-8:0
3838 root 20 0 0 0 0 I 0.3 0.0 0:00.21 kworker/0:0-events
3858 linuxli+ 20 0 625816 51260 41264 S 0.3 0.6 0:01.22 xfce4-terminal
1 root 20 0 22744 13660 9436 S 0.0 0.2 0:03.55 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_g
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-rcu_p
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-slub_
7 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-netns
10 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_pe
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
15 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace_kthread
16 root 20 0 0 0 0 S 0.0 0.0 0:00.31 ksoftirqd/0
17 root 20 0 0 0 0 I 0.0 0.0 0:00.41 rcu_preempt
18 root rt 0 0 0 0 S 0.0 0.0 0:00.01 migration/0
19 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0
20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kdevtmpfs
22 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-inet_
24 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kauditd
25 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khungtaskd
26 root 20 0 0 0 0 S 0.0 0.0 0:00.00 oom_reaper
27 root 20 0 0 0 0 I 0.0 0.0 0:00.31 kworker/u2:2-flush-8:0

```

Linux Lite Terminal - PT 8:22:06 pm

Free -h esse é meu uso de memória RAM e swap em formato legível.

```

linuxlite ~ free -h
              total        used        free      shared  buff/cache   available
Mem:       7.9Gi      865Mi      6.1Gi      5.2Mi      1.3Gi      7.1Gi
Swap:      2.0Gi          0B      2.0Gi

linuxlite ~

```

Network

This PC

Vmstat mostra estatísticas sobre a memória e processos e CPU

```
linuxlite ~ ➤ vmstat
procs -----memory----- swap-- io---- system-- cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
gu
0 0 0 6587596 38532 1128644 0 0 381 386 1448 3 12 2 86 0
0 0
```

Swapon –show mostra as partições e arquivos de swap ativos

RESULTADOS E OBSERVAÇÕES

- O Linux Lite demonstrou **ótimo gerenciamento automático de memória**, priorizando o uso da RAM antes da swap.
- O **uso da swap** permaneceu baixo na maior parte do tempo, sendo ativado apenas sob carga elevada.

```
linuxlite ~ ➤ swapon --show
NAME TYPE SIZE USED PRIO
/swapfile file 2G 0B -2
linuxlite ~ ➤
```

O comando free -h mostrou que, mesmo com vários aplicativos abertos, o sistema manteve parte da RAM livre por meio de **caching eficiente**

- O top revelou que processos leves como o ambiente XFCE consomem pouca memória, ideal para máquinas virtuais e computadores antigos. • O sistema ajustou dinamicamente o cache e buffers, evitando lentidão perceptível.

4. Análise Crítica

André (Windows): Após todos os testes é possível ressaltar diversos pontos. O primeiro é a forma dinâmica e acessível que o Windows disponibiliza essas

ferramentas e visualização de dados, refletindo diretamente na filosofia desse sistema, de ser mais amigável com seus usuários.

Outro ponto a ser ressaltado é como o próprio sistema sempre realoca a memória e recursos automaticamente, sendo possível identificar isso sem dificuldades através dos gráficos disponibilizados constantemente, se entrelaçando com a filosofia de design do sistema novamente. O sistema que foi analisado apresenta respostas e funcionamentos saudáveis, não demonstrando vazamentos de memórias, mostrou boa liberação de recursos e uma boa performance.

Com os testes realizados é notável um maior avanço na capacidade de gerenciar esse recurso de memória, pois foi possível desenvolver:

- Capacidade de detectar vazamentos de memória
- Habilidade para analisar consumo de processos
- Competências para usar as ferramentas do Windows como Powershell

Com todos esses conhecimentos o gerenciamento de memória se torna algo palpável e de certa forma “controlável”, possibilitando interversões e manipulação direta minha.

Eduardo (Android): A análise permitiu compreender o funcionamento interno do gerenciamento de memória do Android. O sistema utiliza métricas como PSS (Proportional Set Size) para determinar o consumo real de cada processo, além de empregar o LMKD para liberar memória sob pressão. O Android é mais agressivo na deslocação em comparação a sistemas desktop, priorizando responsividade e estabilidade.

O estudo evidencia a eficiência do Android ao equilibrar desempenho e consumo, demonstrando a importância de compreender o uso de ferramentas como dumpsys e procrank para análise técnica de dispositivos móveis.

Vinicius (Linux): O Linux Lite se mostrou rápido e estável dentro da máquina virtual, usando bem a memória mesmo com pouco recurso. Isso mostra que ele é uma boa opção para testes e para quem precisa de um sistema leve. O estudo também ajudou a entender melhor como o Linux usa a memória e a swap.

O gerenciamento de memória do Linux Lite é eficiente para um sistema de baixo consumo. Ele faz bom uso do **gerenciamento dinâmico de cache** e do **balanceamento entre RAM e swap**, garantindo estabilidade mesmo em ambientes limitados. No entanto:

- Em configurações com pouca RAM (<1GB), o desempenho pode cair quando o uso de swap é intenso.
- O VirtualBox pode limitar a performance caso o host também esteja com pouca memória disponível.
- Assim, é importante dimensionar corretamente os recursos da VM para garantir uma análise fiel ao comportamento real do sistema.

5. Conclusão

Com todos os testes realizados é possível concluir que: P gerenciamento de memória é um dos “órgãos vitais” de um sistema operacional e de um dispositivo tecnológico no geral. Devido a importância da memória para o funcionamento dos dispositivos, essa etapa de gerenciamento é extremamente importante e necessária, pois ela possibilita que o sistema garanta o bom funcionamento dos nossos dispositivos.

Cada sistema operacional refletiu sua filosofia de design na maneira como o usuário pode interagir com as etapas do gerenciamento de memória. Windows apresentou melhor flexibilidade e a possibilidade dos seus usuários interessados nessa etapa de memória visualizar com clareza como os processos ocorrem, possibilitando também o controle dessas informações.

O Android se mostrou com mais resistência nessas etapas, priorizando responsividade e estabilidade, continuando mais compacto e sem tanta visualização gráfica.

Linux apresentou uma interface analítica, porém com um gerenciamento de memória eficiente e leve, sendo uma boa escolha para quem deseja testes rápidos ou usar um sistema não tão pesado.

6. Autoavaliação

André (Windows): Uma dificuldade me ocorreu no momento que iniciei os testes para identificar algum vazamento de memória, com pesquisas na internet notei que era recomendado identificar essa etapa pelo Powershell, porém não consegui encontrar uma forma de fazer isso por essa ferramenta.

Então decidi usar o gerenciamento de tarefas e o método de espera, para identificar anomalias na memória de processos específicos, o que possibilitou um resultado mais simples, porém parecido.

Eduardo (Android): Durante o experimento, a principal dificuldade foi compreender as métricas PSS e RSS, e interpretar os dados apresentados pelos comandos ADB. Entretanto, o aprendizado foi significativo, especialmente ao observar o comportamento dinâmico da memória e o papel fundamental do LMKD no sistema Android.

Vinícius (Linux): algumas dificuldades foram encontradas, como ajustar a memória da máquina virtual para evitar travamentos e lentidão. Também foi preciso instalar alguns comandos manualmente, como o **htop**, para fazer os testes. Mesmo com esses desafios, foi possível concluir o relatório com bons resultados.

7. Referências

Aula gerenciamento de memória:

https://youtu.be/JzQE3NfW7fg?si=jizdiPe42rUzc_HY

Monitor de Recursos ou Gerenciador de tarefas:

<https://youtube.com/shorts/7OkOhrBKgyI?si=H17EXjyPIN7Lqety>

Gerenciamento de memória dos sistemas:

<https://blog.grancursosonline.com.br/sistemas-operacionais-gerencia-de-memoria/>

Página Oficial do Linux Lite: www.linuxliteos.com

<https://www.site24x7.com/pt/learn/linux/optimize-memory.html>

