RELATÓRIO DO PROJETO

GERENCIAMENTO DE PROCESSOS

Disciplina: Sistemas Operacionais

Professor: Clóvis Ferraro

Grupo: 15

Sumário

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 intimas 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	Ubunto (64-bit)

Comandos Utilizados
ps aux (Listar todos os processos)

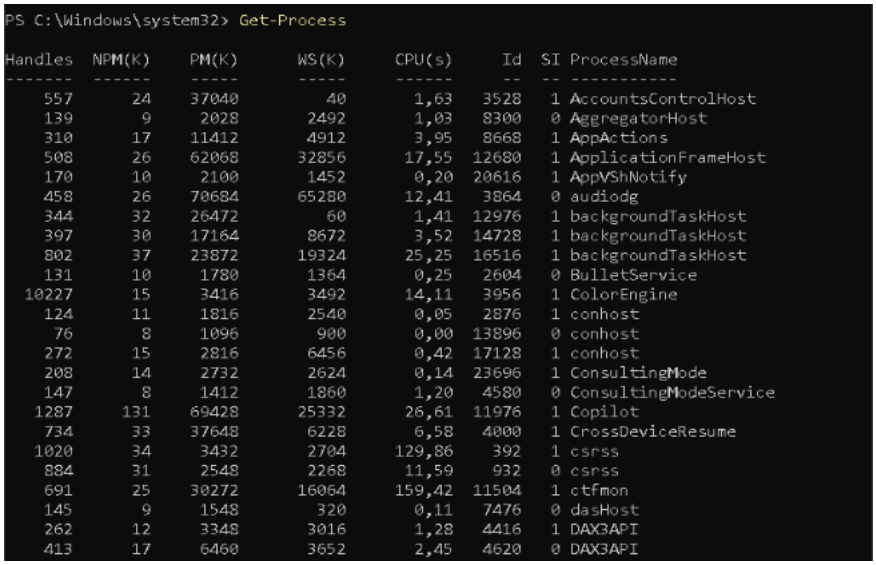
top (Ver todos os processos em tempo real)
mousepad & (Abrir um editor mousepad já existente)
ps aux | grep mousepad (Ver o PID)
Kill (Encerrar o processo)

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    37040         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 AppVShNotify
    458      26    70684      65280      12,41    3864   0 audiodg
    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.

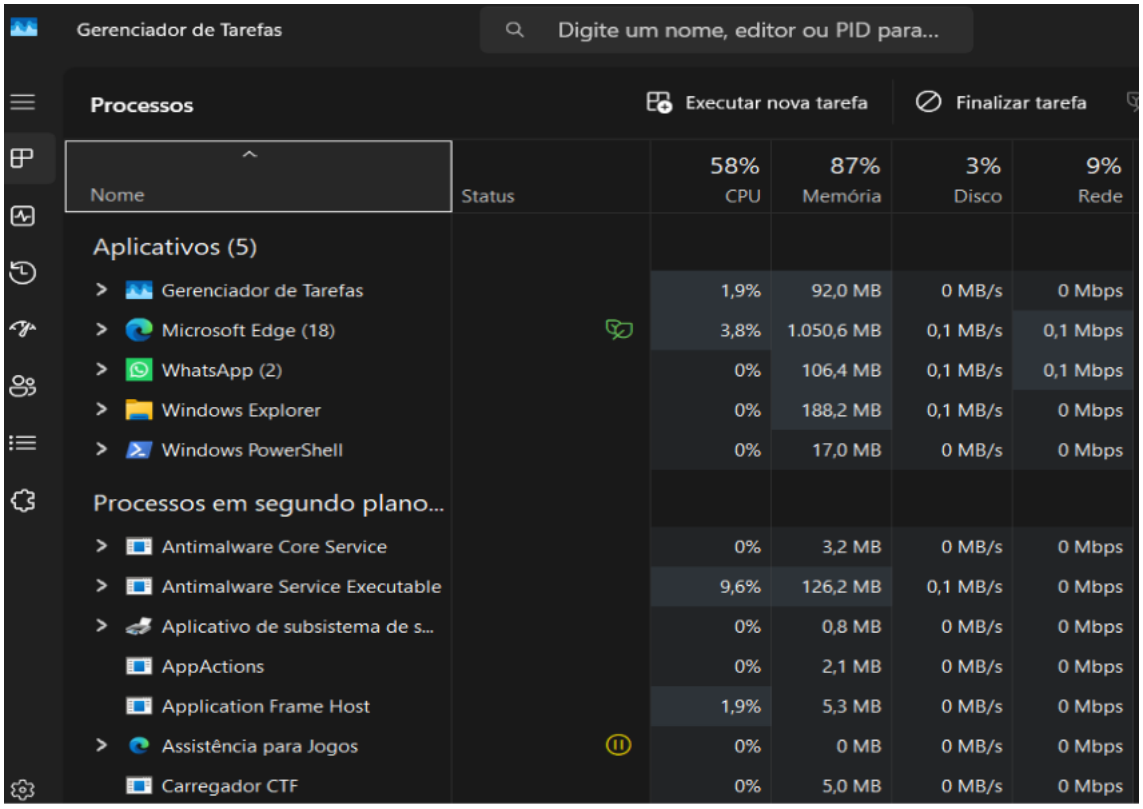
```
PS C:\Windows\system32> Get-Process | Sort-Object CPU -Descending | Select-Object -First 10

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
```

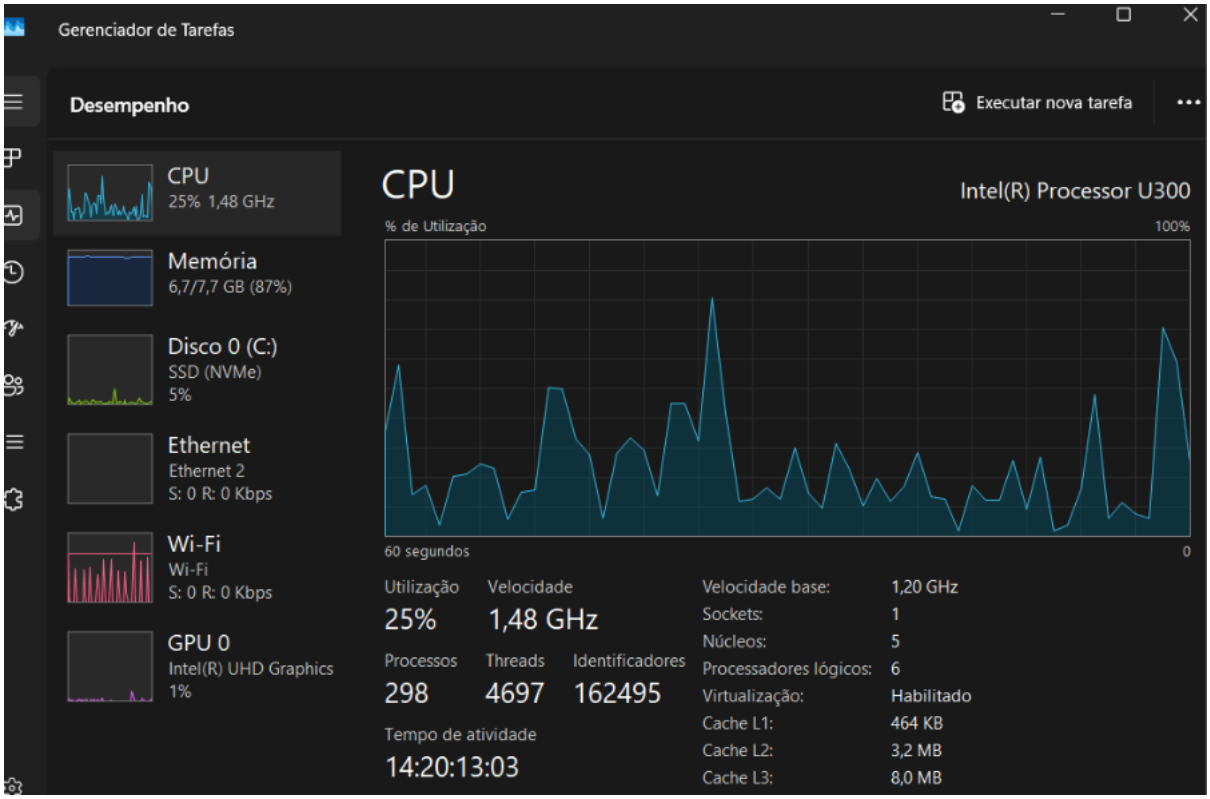
Gerenciador de Tarefas

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.

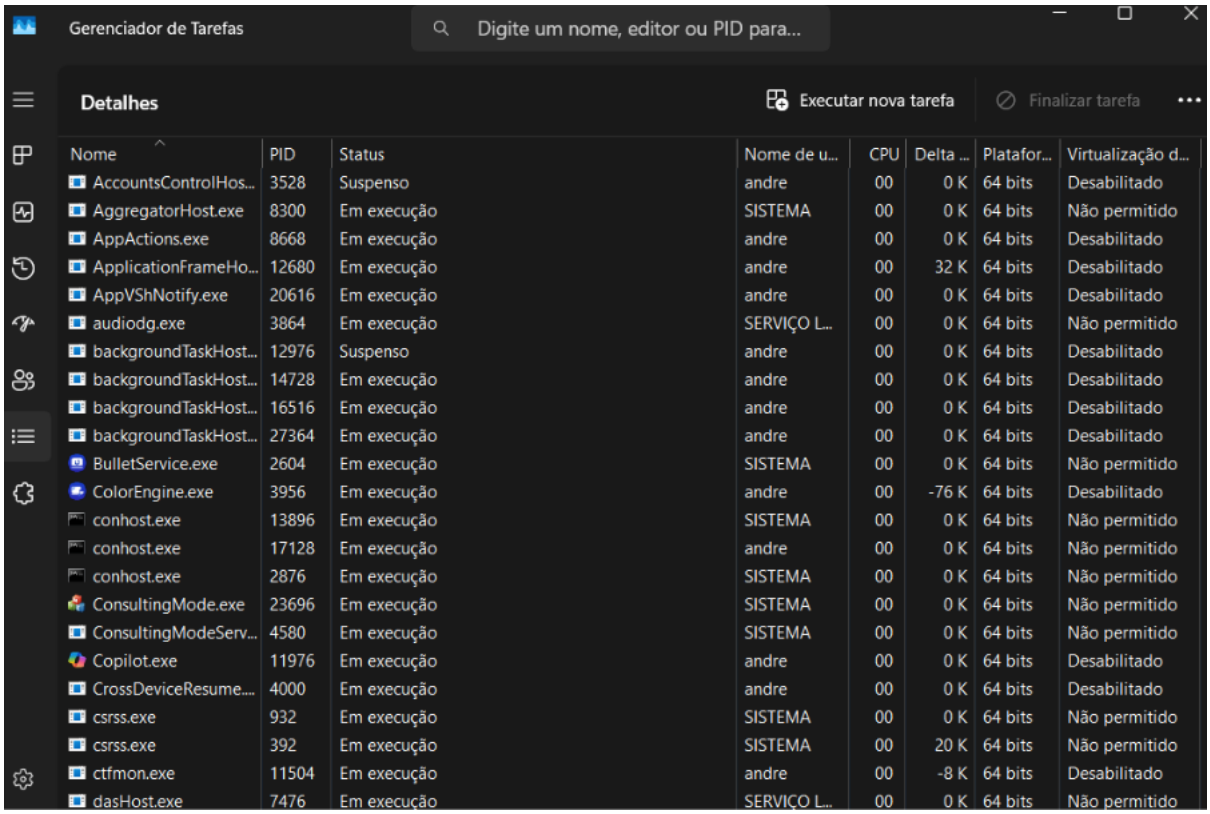
- **Processos**



- **Desempenho**



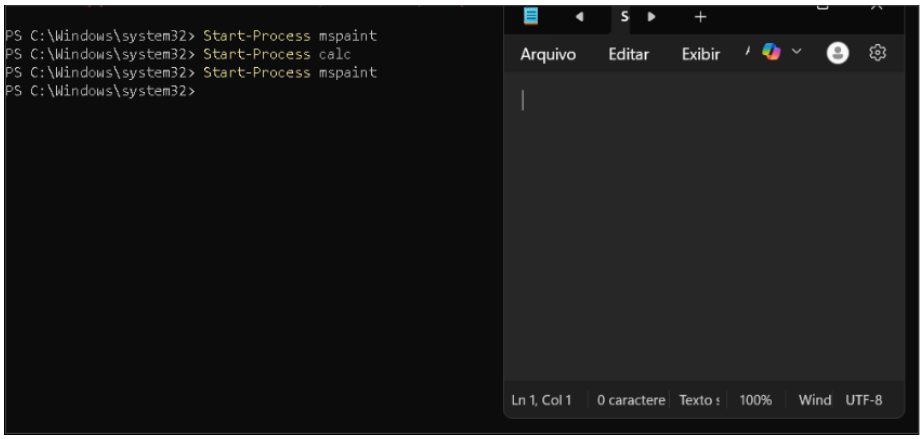
- **Detalhes**

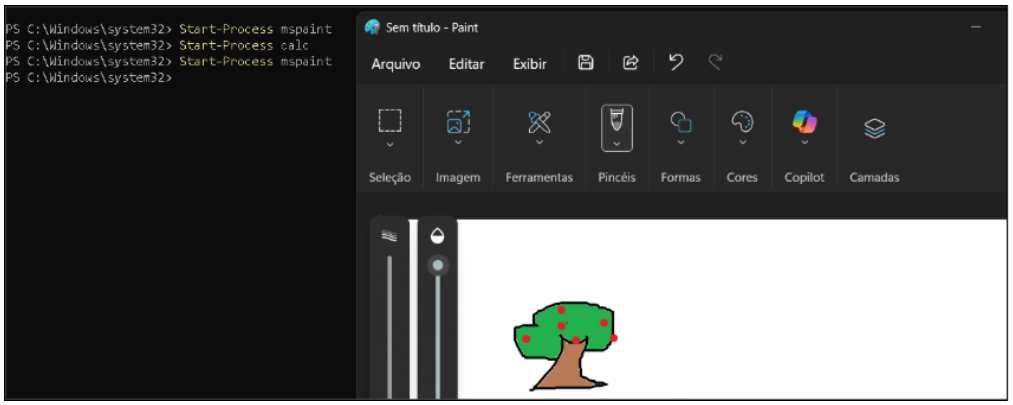


PowerShell

```
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".

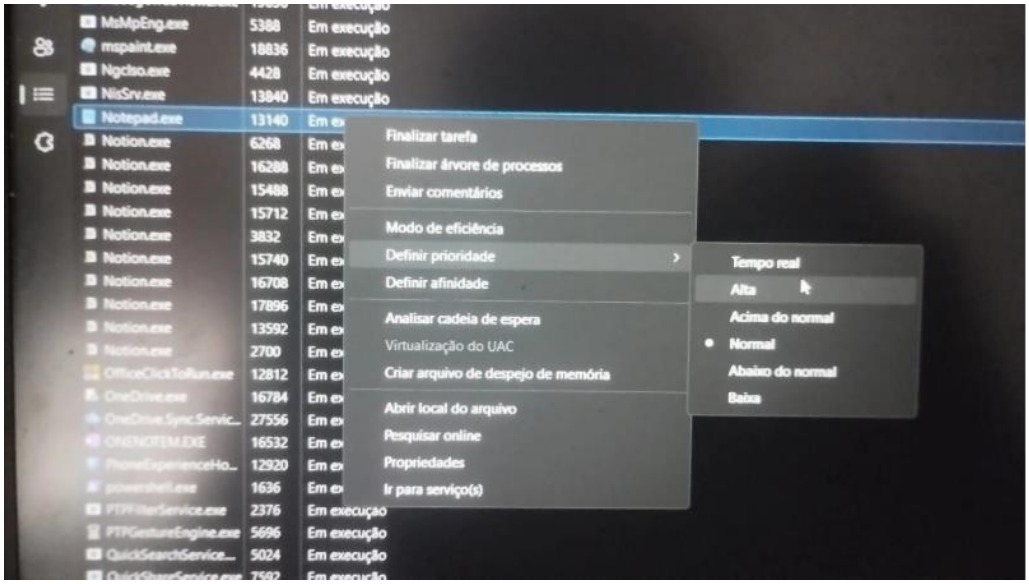
```
PS C:\Windows\system32> Get-Process -Name "notepad", "mspaint"

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).

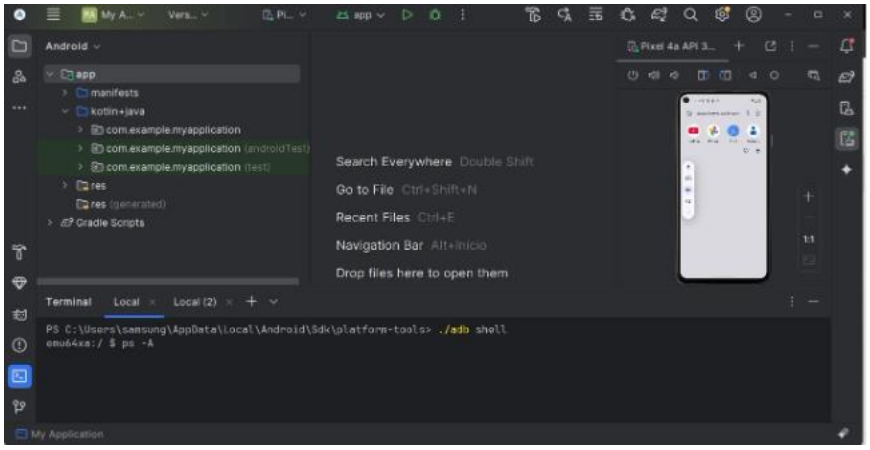


Figura 2 – Listagem de processos em execução (ps -A).

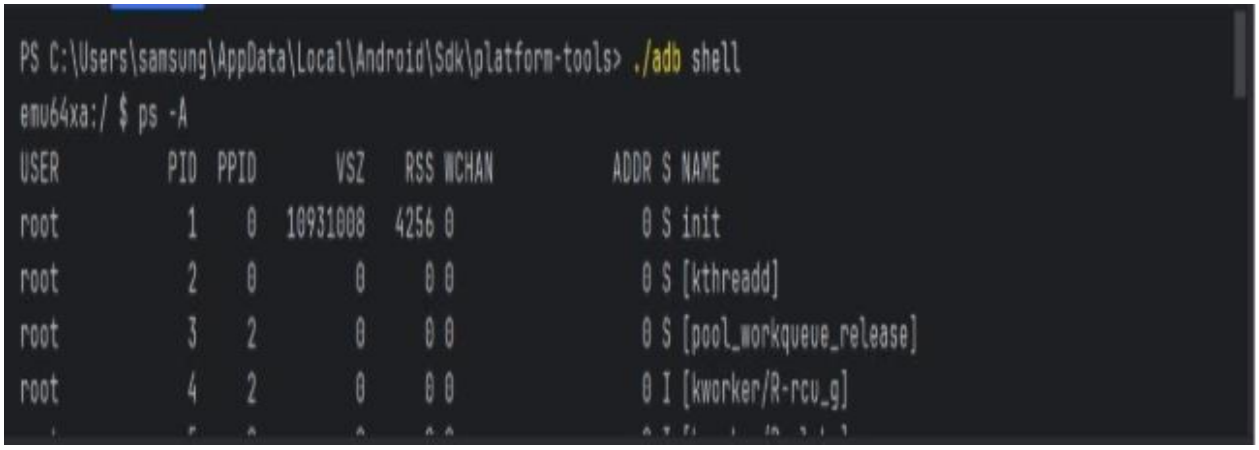


Figura 3 – Monitoramento de processos em tempo real (top).

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

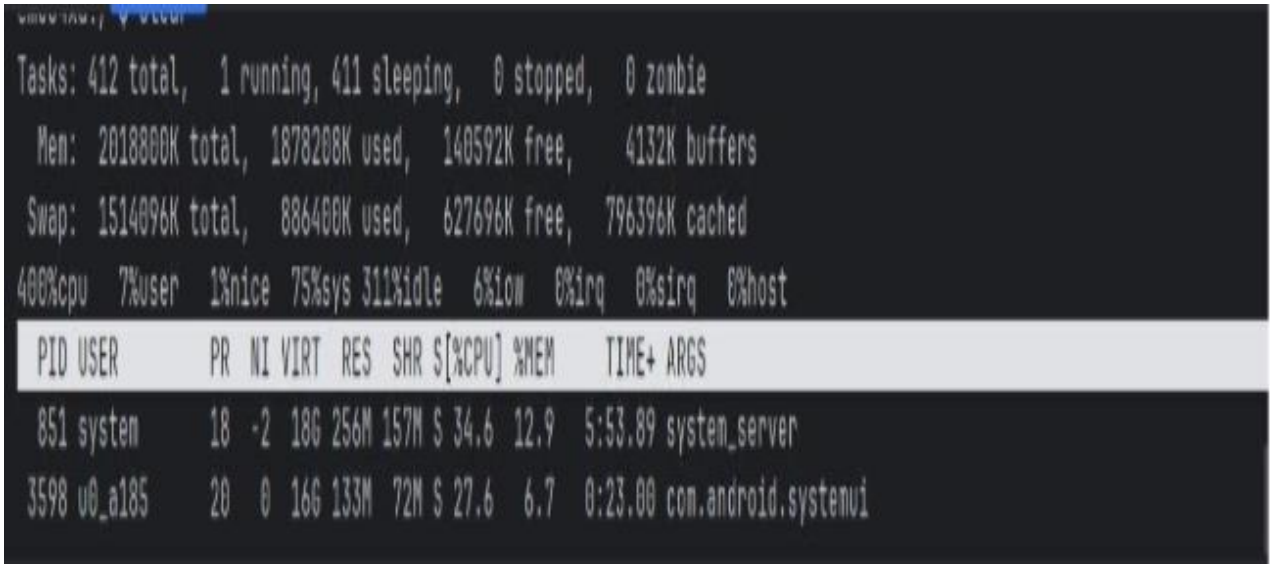


Figura 5 – Visualização detalhada do processo system_server (ps -A | grep system_server).

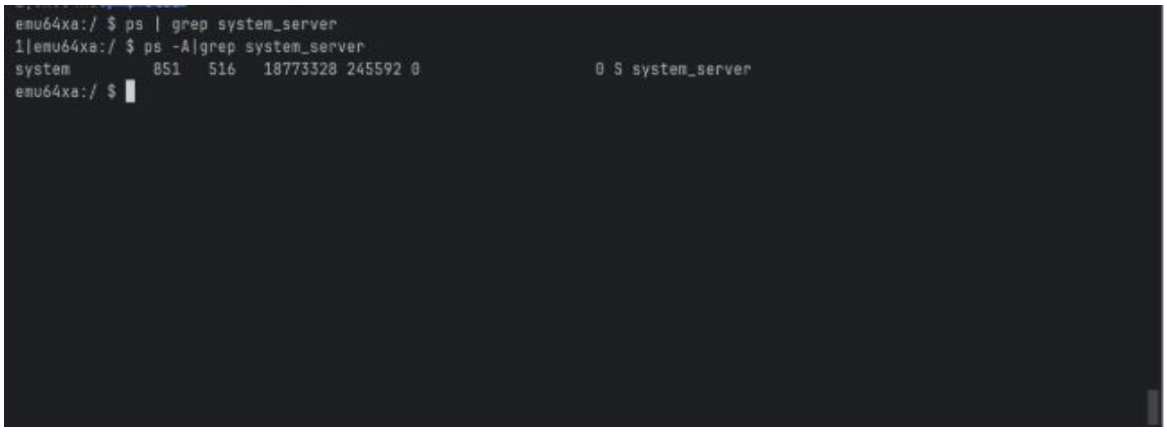


Figura 6 – Tentativa de encerrar processo do sistema (Operation not permitted).

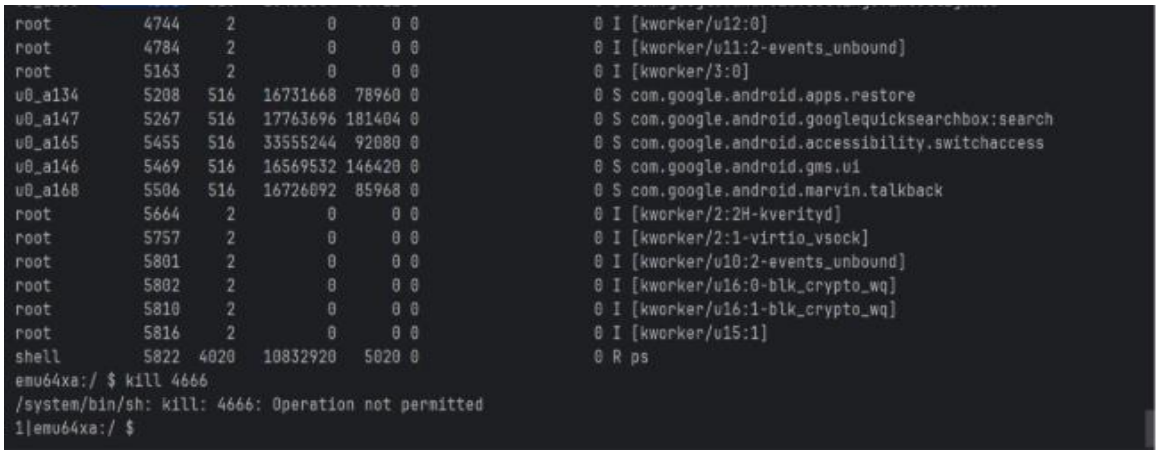


Figura 7 – Criação de um processo de teste (sleep 100 &).

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

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

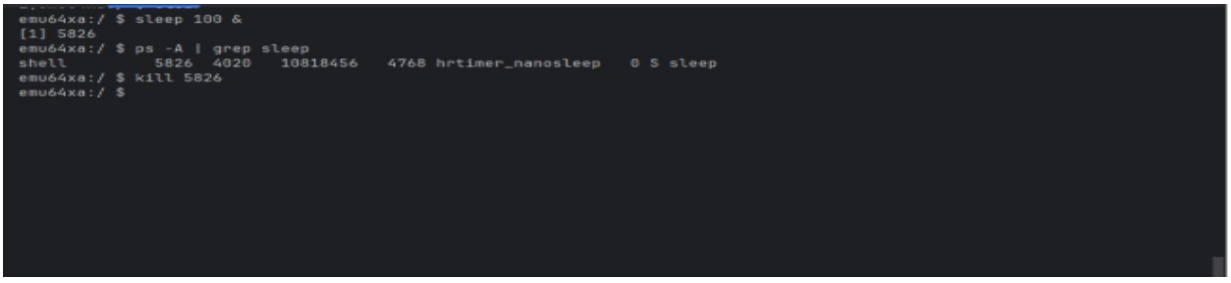


Figura 10 – Criação de processo com prioridade ajustada (nice -n 10 sleep 99 &).

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

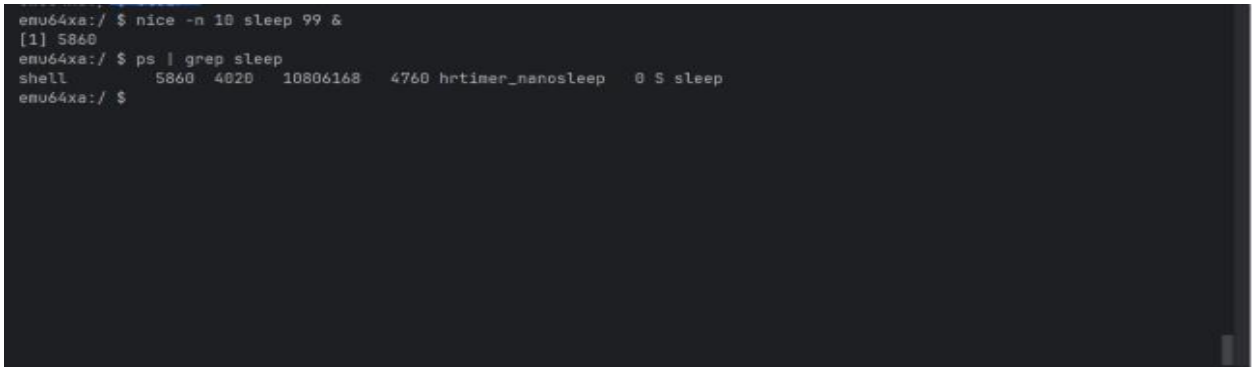
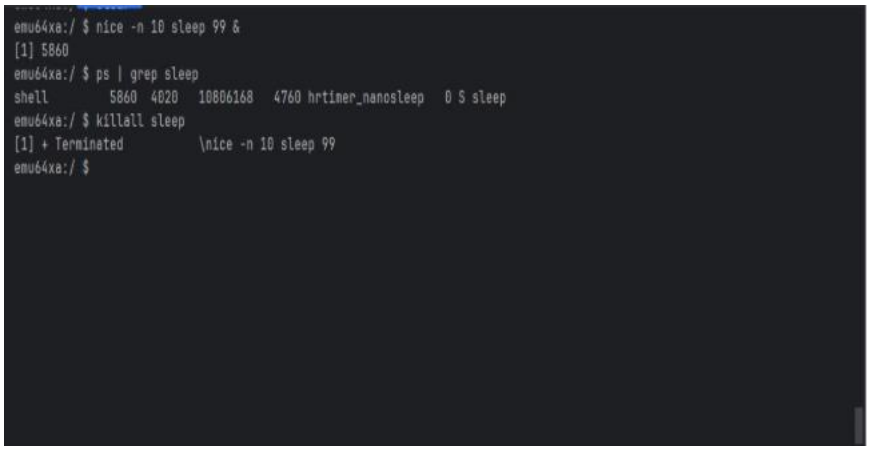
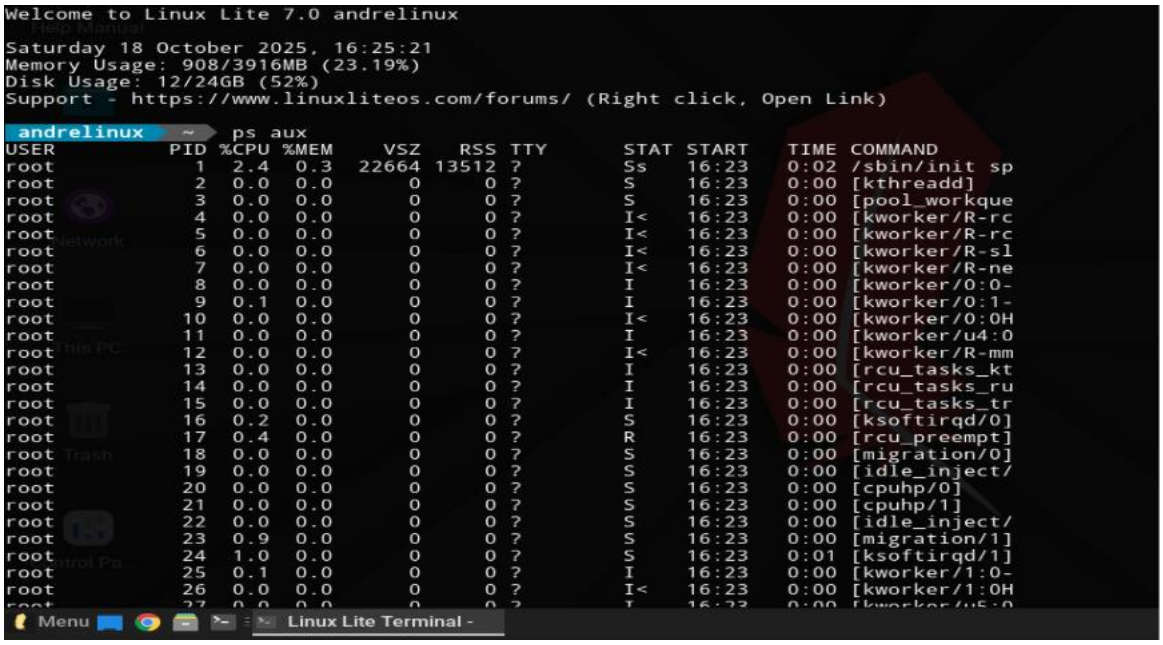


Figura 12 – Encerramento do processo com killall sleep.

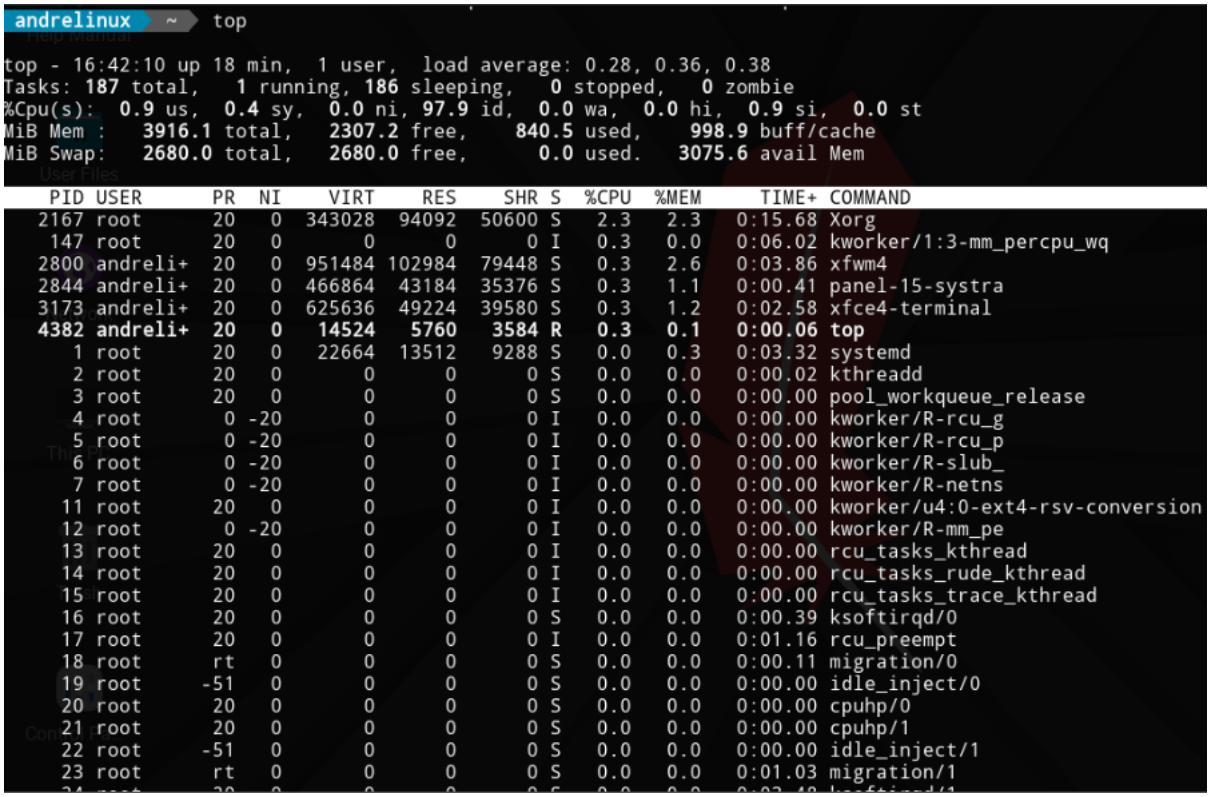


3.3 Linux

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.



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



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”.

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

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 especificas 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 interfases 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=Cj0KCQjw9czHBhCyARIsAFZlN8TOpQum75VFDAClOR2GdSqu2nlg4Plo-BXwfu-537vX_fqov_jP77gaAmjQEALw_wcB

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