

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO ACRE
CAMPUS SENA MADUREIRA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE
SISTEMAS

ATIVIDADE DE PESQUISA EM GRUPO

DOSSIÊ TÉCNICO DO SISTEMA

ROTEIRO DO ESTUDANTE

Disciplina: Sistemas Operacionais

Professor: Renato Afonso

Sena Madureira – Acre

1. APRESENTAÇÃO DA ATIVIDADE

Cada grupo receberá um eixo da ementa da disciplina e produzirá um dossiê sobre ele, registrado na Ficha de Perícia. O dossiê tem duas metades que valem exatamente o mesmo: a pesquisa, que explica como o mecanismo deveria funcionar, e a perícia, que mostra o mecanismo funcionando de fato — com saídas de comandos capturadas por vocês, na máquina de vocês, com os números de vocês.

As duas metades precisam conversar. Um trabalho que explica bem a paginação mas não apresenta medição não vale; um trabalho cheio de capturas de tela sem explicação também não. O que se avalia é a ligação entre as duas coisas.

Por que o trabalho foi desenhado assim

Uma pesquisa sobre "o que é memória virtual" pode ser respondida em segundos por qualquer ferramenta de inteligência artificial, e a resposta será correta. Isso não é um problema: usem essas ferramentas para estudar, para tirar dúvidas, para entender um termo que travou a leitura.

Acontece que nenhuma dessas ferramentas tem acesso à máquina de vocês. Ela não sabe quantos módulos o kernel de vocês carregou, quanto tempo o disco de vocês levou para gravar 300 MB, nem qual é o número do inode do arquivo que vocês criaram. Esses números são a parte do trabalho que só existe se vocês executarem. E é neles que a avaliação se concentra.

2. COMO O TRABALHO SE ORGANIZA

A atividade inteira acontece em uma única sessão de cinco tempos de cinquenta minutos. Não há etapa em casa: pesquisa, medição, montagem e apresentação ocorrem dentro do bloco, com o professor em sala o tempo todo.

Essa escolha não é logística, é metodológica. Um trabalho levado para casa volta pronto, e não há como saber de onde veio. Um trabalho construído em sala, com o professor circulando entre as máquinas, é observável enquanto está sendo feito — e é isso que a atividade avalia.

Etapa 1

O grupo estuda os quatro pontos do seu eixo usando o material da disciplina, disponível no Linktree da turma, e a referência principal da bibliografia.

O objetivo desta etapa não é produzir texto bonito. É chegar na etapa seguinte sabendo o que os comandos vão mostrar.

Etapa 2 — Perícia

O grupo executa a sequência de comandos do seu eixo no próprio Windows do laboratório, captura as saídas e preenche a Ficha de Perícia — o formulário deste documento, que é o registro oficial do que foi medido. Em seguida ataca o complicador.

Etapa 3 — Montagem

O grupo faz um relatório em um documento.

A REGRA QUE SUSTENTA TUDO

Antes de cada bloco de comandos, executem no terminal: `echo "=== <Nome do grupo> — $(hostname) — $(date) ==="`. Essa linha precisa aparecer em todas as capturas de tela. Captura sem identificação e sem data e hora não é considerada evidência e o item correspondente é zerado.

3. OS EIXOS E SEUS ROTEIROS

ANTES DO PRIMEIRO COMANDO: ABRA O POWERSHELL, NÃO O PROMPT DE COMANDO

Os dois são pretos e parecidos, mas não são a mesma coisa. Se a linha começa com `C:\Windows\System32>` vocês estão no Prompt de Comando (`cmd`), e os comandos deste roteiro vão responder que não são reconhecidos. O PowerShell começa com `PS`, assim: `PS C:\Users\seunome>`. Para abrir: tecla Windows, digite `powershell` e dê Enter. Se já estiver no `cmd`, digite `powershell` ali mesmo e dê Enter. Confirmam o `PS` no início da linha antes de seguir. As poucas linhas do roteiro que rodam no `cmd` vêm com `"cmd /c"` na frente e funcionam nos dois.

EIXO 1 — CONCEITOS E ARQUITETURAS DE SISTEMAS OPERACIONAIS

Pergunta norteadora

Todo programa que roda nesta máquina passa parte do seu tempo executando código do próprio sistema operacional. Quanto tempo, e por quê?

O que pesquisar

- Modo núcleo e modo usuário: quais instruções são privilegiadas e o que acontece quando um programa comum tenta executá-las.
- Chamada de sistema: como um programa pede um serviço ao núcleo e o que ocorre na transição entre os dois modos.
- Arquitetura monolítica, micronúcleo e híbrida: que compromisso entre desempenho e isolamento cada uma assume, e onde o núcleo NT do Windows se encaixa.
- Driver de dispositivo: por que ele roda em modo núcleo e qual o risco que isso representa.

Perícia obrigatória

Os seis passos contam uma história em três partes. Os passos 1 a 3 mostram QUEM roda em modo núcleo: não apenas o kernel, mas cada driver carregado, todos com poder total sobre a máquina. Os passos 4 e 5 medem QUANTO TEMPO um programa comum passa dentro dessa zona privilegiada. O passo 6 mostra O QUE mantém o sistema ocupado mesmo quando ninguém está usando o computador.

Depois de alguns comandos, a janela do PowerShell fica cheia de saídas antigas. Digitem **cls** e dêem Enter: a tela esvazia e o cursor volta ao topo. Nada é apagado do sistema — só a exibição. Os arquivos, os processos e os resultados continuam todos lá.

Executem na ordem, capturando a saída de cada etapa.

LEMBREM

Antes de cada bloco de comandos, executem no terminal:

```
echo "=== <Nome do grupo> — $(hostname) — $(date) ===".
```

COMANDO	O QUE OBSERVAR E REGISTRAR
<code>gcim Win32_OperatingSystem Select Caption, Version, OSArchitecture</code>	Anotem nome, versão e arquitetura. Todos os números seguintes valem para ESTE sistema: sem esta linha, a medição não tem dono.
<code>Get-CimInstance Win32_SystemDriver Where-Object State -eq "Running" Measure-Object Select-Object -ExpandProperty Count</code>	Devolve um número: a quantidade de drivers de sistema cujo estado é Running. A consulta não inclui necessariamente componentes de driver que funcionam em modo usuário; portanto, registrem exatamente o que foi contado.
<code>Get-CimInstance Win32_SystemDriver Where-Object State -eq "Running" Select-Object -First 12 Name, DisplayName, State, StartMode</code>	Mostra doze drivers de sistema em execução. Escolham um, anotem o nome exato e pesquise qual dispositivo ou serviço ele .
<code>Get-Process explorer Format-List Name, UserProcessorTime, PrivilegedProcessorTime, TotalProcessorTime</code>	Aqui "explorer" é o NOME do processo consultado: o Explorador de Arquivos (explorer.exe), que desenha a área de trabalho, a barra de tarefas e as janelas de pasta. Get-Process sem nome traz todos os processos; com o nome, traz só aquele. Escolhemos este porque está sempre rodando e mexe muito em arquivos e tela. Format-List exibe um campo por linha. São três cronômetros do MESMO processo: tempo gasto no código dele, tempo gasto dentro do núcleo e a soma dos dois. Vêm no formato hora:minuto:segundo (por exemplo 00:00:12.34), então convertam para segundos antes de comparar.
<code>Get-CimInstance Win32_SystemDriver -Filter "Name='Ntfs'" Format-List Name, DisplayName, Description, ServiceType, State, StartMode, PathName</code>	Consultar seus detalhes, substituindo Ntfs pelo nome escolhido do processo escolhido. Como identificar o modo Observe ServiceType: • Kernel Driver: driver que funciona em modo núcleo. • File System Driver: driver de sistema de arquivos que funciona em modo núcleo. • Own Process ou Share Process: indica um serviço executado como processo,

	não um driver comum carregado no núcleo.
Get-Process Where CPU Sort CPU -Desc Select -First 8 Name, UserProcessorTime, PrivilegedProcessorTime	Os mesmos dois cronômetros para os oito processos mais ativos. O filtro "Where CPU" descarta os processos de outros usuários, cujos tempos o sistema não deixa ler.
(Get-Service Where Status -eq Running).Count	Devolve UM número: quantos serviços estão em execução agora. São eles que produzem atividade em segundo plano e, com ela, boa parte das transições para o modo núcleo.

COMPLICADOR

Escolham dois processos: um que mexe muito em arquivos e janelas (o explorer serve) e um que basicamente calcula. Para cada um, convertam PrivilegedProcessorTime e TotalProcessorTime para segundos, dividam um pelo outro e obtenham o percentual de tempo em modo núcleo. Os dois percentuais serão bem diferentes. Expliquem a diferença a partir do que cada processo pede ao sistema — e digam por que um programa que só faz conta quase não precisa do núcleo.

$$\text{Percentual em modo núcleo} = \frac{\text{PrivilegedProcessorTime}}{\text{TotalProcessorTime}} \times 100$$

ENTREGA ESPECÍFICA DESTE EIXO

Tabela comparativa das três arquiteturas com uma coluna de vantagem e uma de desvantagem, mais os percentuais de tempo em modo núcleo dos dois processos escolhidos.

Arquitetura	Como funciona	Vantagem	Desvantagem	Exemplo
Monolítica				
Micronúcleo				
Híbrida				

Processo escolhido	Modo usuário (s)	Modo núcleo (s)	Tempo total (s)	Percentual em modo núcleo (%)
Processo 1				
Processo 1				

EIXO 2 — PROCESSOS

Pergunta norteadora

O que o sistema operacional faz com um processo que, neste instante, não está fazendo absolutamente nada?

O que pesquisar

- Estados de um processo: em execução, pronto e bloqueado — e o que provoca cada transição entre eles.
- Bloco de Controle de Processo: quais informações o núcleo precisa guardar para conseguir parar um processo e retomá-lo depois.
- Troca de contexto: o que é salvo, o que é restaurado e por que essa operação tem custo.
- Escalonamento preemptivo, quantum, prioridade e afinidade de processador: como o núcleo decide quem executa e onde.

Perícia obrigatória

O escalonador só se revela quando há escassez: se cada processo tiver um núcleo à disposição, todos rodam à vontade e não sobra decisão nenhuma para observar. Por isso o experimento FABRICA a escassez. Vocês vão criar dois processos idênticos, igualmente famintos por processador, e obrigá-los a caber no mesmo núcleo. A partir daí, tudo o que acontecer é decisão do sistema operacional. Depois vocês mudam a prioridade de um deles e medem de novo: a diferença entre as duas medições é o escalonador funcionando.

Executem na ordem, capturando a saída de cada etapa.

COMANDO	O QUE OBSERVAR E REGISTRAR
<code>\$env:NUMBER_OF_PROCESSORS</code>	Quantos núcleos esta máquina tem. Chamem esse número de N e anotem: sem ele, os percentuais do passo 6 não fazem sentido. É o passo que mais se esquece.
<code>Get-Process Sort CPU -Desc Select -First 07 Id, Name, CPU, Threads, PriorityClass</code>	A fotografia do sistema ANTES de vocês mexerem. Id identifica o processo (o nome pode repetir, o Id não). CPU é o total acumulado de segundos de processador, não o uso atual. Threads mostra que quase todo processo tem várias linhas de execução: o escalonador na verdade escalona threads. PriorityClass é o valor que vocês vão alterar adiante; anotem que quase todas são Normal.
<code>\$p1 = Start-Process powershell -ArgumentList '-Command','while(1){}' -PassThru \$p2 = Start-Process powershell -ArgumentList '-Command','while(1){}' -PassThru</code>	Duas linhas, uma de cada vez. Abrem duas janelas rodando um laço infinito que não faz nada útil, e é esse o ponto: um programa que só calcula NUNCA fica bloqueado esperando disco ou rede, está sempre pronto para executar. Qualquer pausa que ele tiver foi o sistema que impôs. O -PassThru guarda cada processo criado nas variáveis \$p1 e \$p2.
<code>\$p1.Id \$p2.Id</code>	Os dois números que identificam os competidores. Anotem na ficha: vocês vão precisar deles no Gerenciador de Tarefas.

<pre>\$p1.ProcessorAffinity = 1 \$p2.ProcessorAffinity = 1</pre>	Afinidade é a lista de núcleos em que o processo TEM PERMISSÃO de rodar, escrita como mapa de bits: 1 é o núcleo 0, 2 é o núcleo 1, 3 seria os dois juntos. Este passo é o que torna o experimento possível. Sem ele, o Windows poria cada laço em um núcleo diferente, ambos rodariam a 100% e não haveria disputa alguma para observar.
Gerenciador de Tarefas, aba Detalhes	Localizem os dois powershell.exe pelos Id anotados e registrem a coluna CPU de cada um. Se a coluna não aparecer, botão direito no cabeçalho para ativá-la. Leiam a nota abaixo da tabela ANTES de interpretar esses números.
<pre>(Get-Process -Id \$p1.Id).PriorityClass = "Idle"</pre>	Rebaixa UM dos dois à prioridade mínima. Idle não quer dizer parado: quer dizer "só execute quando não houver nada melhor a fazer". O processo continua pronto e faminto; o escalonador é que passa a preteri-lo.
<pre>Get-Process -Id \$p2.Id Select-Object Id, Name, PriorityClass Get-Process -Id \$p1.Id Select-Object Id, Name, PriorityClass</pre>	Esses dois comandos servem para consultar informações básicas dos processos P1 e P2 que você criou.
Gerenciador de Tarefas, aba Detalhes	Mesmos Id, mesma conversão. O processo em prioridade Normal deve tomar quase todo o núcleo 0 e o rebaixado deve cair para perto de zero. A diferença entre esta captura e a anterior é a evidência central do eixo.
<pre>Stop-Process -Id \$p1.Id, \$p2.Id</pre>	Encerra apenas os dois competidores. NÃO usem Stop-Process -Name powershell: isso fecharia também a janela de vocês, levando junto o histórico de comandos que será entregue.

3. REGRAS DE EXECUÇÃO

Durante a sessão

- Os comandos do roteiro apenas leem informações do sistema ou criam arquivos na pasta temporária do próprio usuário. Nenhum deles altera configuração da máquina. Não executem comandos fora do roteiro sem falar com o professor.

Sobre o uso de inteligência artificial

- É permitido e incentivado usar ferramentas de IA para estudar os conceitos, esclarecer termos e revisar a redação dos slides.
- É vedado apresentar como medição da máquina qualquer número que não tenha sido efetivamente capturado. Saída de comando inventada ou gerada por IA não será aceita.