Gerenciamento de Memória

Conceitos Gerais

1. Introdução

- Gerencia de memória é o componente de um sistema operacional que determina o local da memória onde deverá ser colocado o código de um novo processo.

- Esse código geralmente foi lido de um arquivo previamente armazenado em um dispositivo de entrada e saída (HD, por exemplo).

- Os gerenciadores de memória podem ser divididos em duas classes: os que alternam os processos entre a memória principal e o disco durante a execução (swapping) e os que não alternam. Nosso foco será no swapping, visto que é a técnica utilizada atualmente.

- Já o gerenciamento de E/S cuida das portas responsáveis pela entrada e saída de informações de componentes como HD, SSD, portas USB, etc.

2. Gerenciamento de Memória

- Antes de definir o conceito de gerenciamento de memória, é importante conhecer o termo **memory leak** (vazamento de memória). Esse problema pode ocorrer em duas situações:

1. Blocos de memória estão alocados e disponíveis para serem utilizados pelo programa, mas não são acessíveis porque a aplicação não tem nenhum ponteiro apontando para essa área de memória. Ou seja, tais blocos de memória não podem ser utilizados nem pelo programa nem por outros processos (ficam alocados e sem acesso!);

2. Blocos de memória possuem dados que poderiam ser liberados por estarem inacessíveis e que, por algum “esquecimento”, ainda são referenciados no código. Ou seja, mesmo sem estarem sendo usados, não podem ser liberados.

2. Gerenciamento de Memória

- O **gerenciador de memória** do **sistema operacional (SO)** é o componente responsável por controlar e organizar o uso da **memória principal (RAM)**. Ele garante que os processos (programas em execução) tenham acesso à memória de forma eficiente, segura e sem conflitos.

- As principais funções do gerenciador de memória do SO são:

1. Alocação de memória

- Quando um processo é iniciado, o SO reserva uma parte da memória para ele.
- Também pode alocar mais memória sob demanda (ex: quando o programa precisa de mais espaço).

2. Desalocação de memória

- Quando o processo termina, o SO libera a memória usada, tornando-a disponível para outros processos.

2. Gerenciamento de Memória

3. Gerenciamento de memória virtual

- Usa o disco rígido como uma extensão da RAM (memória virtual), permitindo que mais processos rodem do que a memória física suportaria.

4. Isolamento de processos

- Garante que um processo não acesse a memória de outro (segurança e estabilidade).

5. Compartilhamento de memória

- Permite que processos compartilhem certas regiões de memória, se necessário (por exemplo, bibliotecas compartilhadas).

6. Gerenciamento de cache e buffers

- Otimiza o acesso à memória utilizando técnicas como cache de dados e buffers, aumentando o desempenho.

3. Swapping

- Há momentos em que não há memória principal suficiente para todos os processos ativos, de forma que os processos excedentes devem ser mantidos em disco e trazidos para a memória dinamicamente.

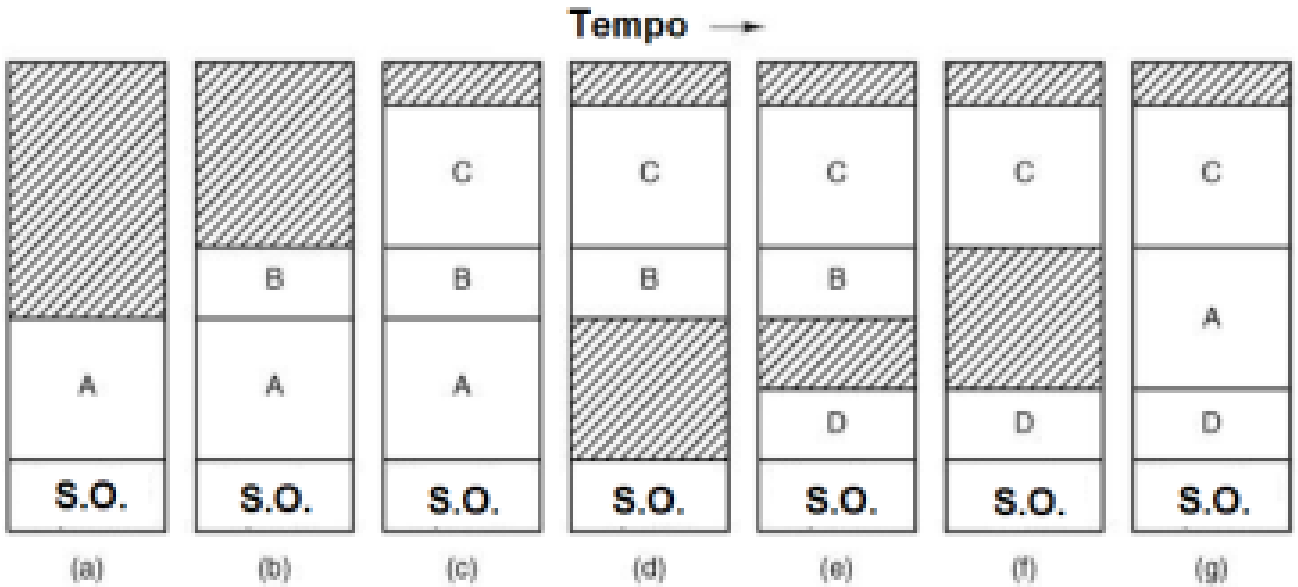
- O sistema operacional **move processos inteiros** da memória principal para o disco (geralmente em uma área chamada *swap space*) e, quando for necessário, traz de volta para a RAM.

- Exemplo: se há 3 processos na memória e chega um quarto que não cabe, o SO pode escolher um processo pouco usado, “expulsá-lo” inteiro para o disco (swap out) e liberar espaço. Quando esse processo precisar ser executado novamente, ele será carregado de volta para a RAM (swap in).

- **Características**:
- Trabalha com o **processo inteiro** (não com partes dele), é mais simples de implementar.
- Pode causar muita movimentação de dados entre RAM ↔ disco, tornando o sistema lento.
- Usado em sistemas antigos

3. Swapping

- A imagem mostra o uso da RAM ao longo do tempo. No instante 3, não há espaço para carregar o processo D, de forma que foi feito um swap do processo A no disco para alocação do processo D.



4. Memória Virtual

- Já a técnica da **memória virtual** funciona dividindo os processos em **páginas** (blocos pequenos de memória, tipicamente 4 KB). O SO mantém apenas as páginas mais usadas na RAM, enquanto as páginas menos usadas ficam armazenadas no disco.

• Quando um processo acessa uma página que não está na RAM, ocorre um **page fault** (falha de página). O SO então traz apenas aquela página do disco para a RAM.

- **Características**:
  - O processo **não precisa caber inteiro na RAM**.
  - Permite que os programas pareçam ter mais memória do que realmente existe.
  - Usa técnicas como **paginação** ou **segmentação**.
  - Mais eficiente que swapping, porque só traz para a memória as partes realmente necessárias.
  - Base para sistemas modernos (Linux, Windows, macOS, etc.).

4. Memória Virtual

- **Paginação**: consiste em dividir a memória em blocos de **mesmo tamanho fixo**.
  - Na RAM, esses blocos são chamados de **quadros de página** (*frames*).
  - No processo, esses blocos são chamados de **páginas**.

- Quando o processo é executado, suas páginas são carregadas em quadros disponíveis da RAM, não necessariamente de forma contígua.

- **Vantagens**: Elimina fragmentação externa (sempre cabe uma página inteira em um quadro) e o gerenciamento é mais simples.

- **Desvantagem**: Pode haver **fragmentação interna** (última página do processo pode não usar todo o quadro).

4. Memória Virtual

- **Segmentação**: Consiste em dividir a memória em **partes lógicas** de tamanhos diferentes, de acordo com a estrutura do programa (cada estrutura fica em um segmento).

- Cada segmento pode ter tamanho distinto, e o sistema precisa guardar onde cada um começa e seu tamanho.

- **Exemplo prático**:
  Pense em uma caixa de ferramentas (memória). Dentro dela, você organiza segmentos: uma gaveta para chaves de fenda (código), outra para martelos (dados), outra para pregos (pilha). Cada segmento tem um espaço diferente, mas organizado de acordo com sua função.

- Pode sofre **fragmentação externa** (pode sobrar buracos de memória entre segmentos), sendo mais complexo de gerenciar.

5. Gerenciamento de E/S

- Uma das principais funções do S.O. é controlar todos os dispositivos de E/S. Para isso, é necessário utilizar comandos, interrupções e tratamento de erros.

- Para que a comunicação entre o SO e o dispositivo ocorra, é necessário que um driver seja instalado.

- De uma forma geral os dispositivos de E/S são divididos em duas categorias: dispositivos de bloco e dispositivos de caractere. O primeiro armazena informações em blocos de tamanho fixo (ex.: 512 bytes, 32 KB, entre outros), como os discos.

- O dispositivo de caractere envia ou aceita um fluxo de caracteres, sem nenhuma estrutura de bloco. Ele não tem endereços (como ocorre com os blocos) e não possui operações de busca. Alguns exemplos são as impressoras, interfaces de rede e mouses.

5. Gerenciamento de E/S

- O SO geralmente usa 3 técnicas para gerenciamento dos dispositivos de E/S, são elas:

1. **E/S mapeada em memória**: Cada dispositivo possui registradores (memória) utilizados para se comunicar com o processador. O sistema operacional escreve nesses registradores, solicitando para o dispositivo enviar dados, aceitar dados, ligar-se ou desligar-se, etc.

2. **Interrupções**: Quando uma operação do dispositivo de E/S termina, ele envia uma interrupção ao processador, que começa a executar a rotina de tratamento da interrupção. Tal código informa ao S.O. que a E/S está concluída, então o S.O. pode verificar os bits de status para saber se tudo ocorreu bem (se não ocorreu, pode tentar novamente).

3. **Acesso direto à memória (DMA)**: Permite que certos dispositivos de hardware (como discos rígidos, placas de som, placas de rede, etc.) acessem a memória RAM diretamente, sem precisar passar pelo processador (CPU) o tempo todo. Geralmente o DMA fica na placa mãe.

6. Gerencia de Armazenamento (Sistemas de Arquivos)

- As informações de um computador são armazenadas nas mídias (HDs, SSDs, entre outras) em unidades chamadas arquivos. O SO deve garantir que estes arquivos não sejam corrompidos em hipótese alguma.

- O SO faz o gerenciamento dos arquivos através do sistema de arquivos, que define toda a política de como os dados serão armazenados no disco.

- Como exemplo de sistemas de arquivos, desde MS-DOS até o Windows 98, forma usados o FAT12, FAT16 e FAT32. A partir do Windows NT o NTFS se tornou o padrão.

- No Linux é possível escolher entre diversos sistemas de arquivos, como a família EXT (versões 2, 3 e 4), JFS, XFS, BTRFS, entre muitos outros.

6. Gerencia de Armazenamento (Sistemas de Arquivos)

- Os arquivos são a menor porção de dados de um sistema. Ele é identificado por um nome e por sua extensão. A extensão na maioria dos SO's possuem 3 caracteres e identificam o tipo do arquivo (por exemplo: JPG é imagem).

- Alguns SO's diferenciam são sensitivos em maiúsculas e minúsculas, e não obrigam o arquivo a ter extensões.

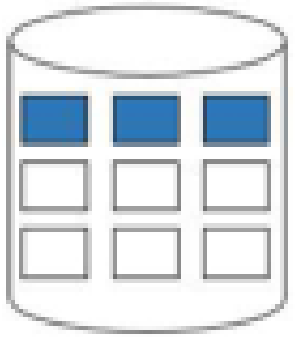
- Os arquivos possuem uma série de metadados com informações como data de criação, alteração, usuário proprietário, permissões de acesso, tamanho, etc.

- É importante reforçar que todas estas características podem mudar de acordo com o sistema de arquivo utilizado.

6. Gerencia de Armazenamento (Sistemas de Arquivos)

- Formas de alocação do disco:

1. **Alocação contígua**: Armazena cada arquivo como uma sequência contígua de blocos. Se uma partição for formatada com blocos de tamanho 4 KB, um arquivo de 40 KB deve alocar 10 blocos consecutivos. Sua grande vantagem é a simplicidade e velocidade na leitura. O problema é a fragmentação.

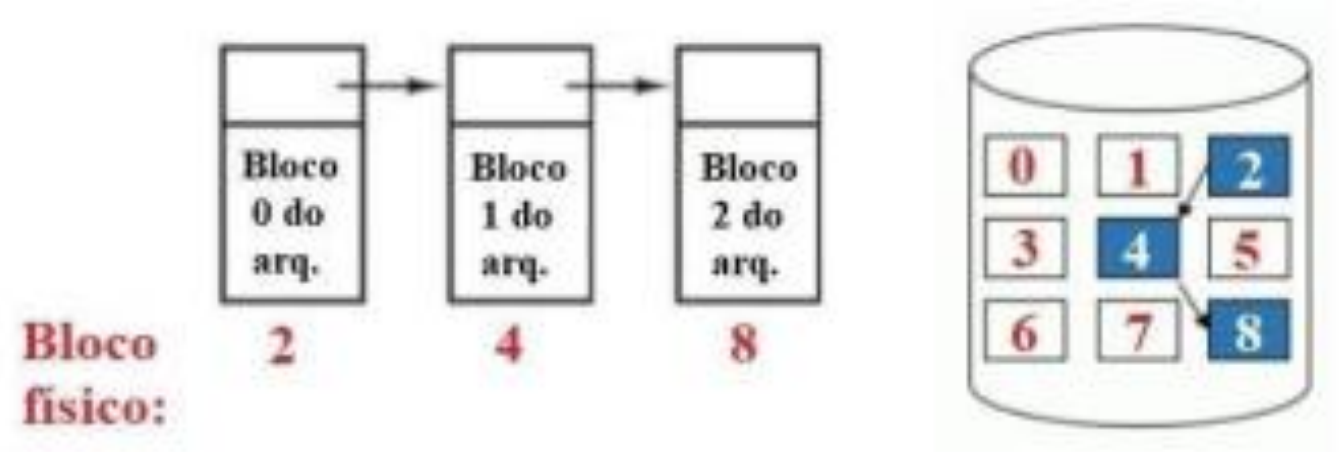


Alocação contígua

6. Gerencia de Armazenamento (Sistemas de Arquivos)

2. **Alocação encadeada**: cada arquivo é mantido como uma lista encadeada de blocos de disco, onde cada bloco possui o endereço do próximo. Desta forma os dados podem ser gravados em qualquer bloco.

- O problema é que o acesso aleatório é mais lento que o sequencial, além disso o ponteiro ocupa um espaço do bloco. Isso exige mais operações de E/S.



6. Gerencia de Armazenamento (Sistemas de Arquivos)

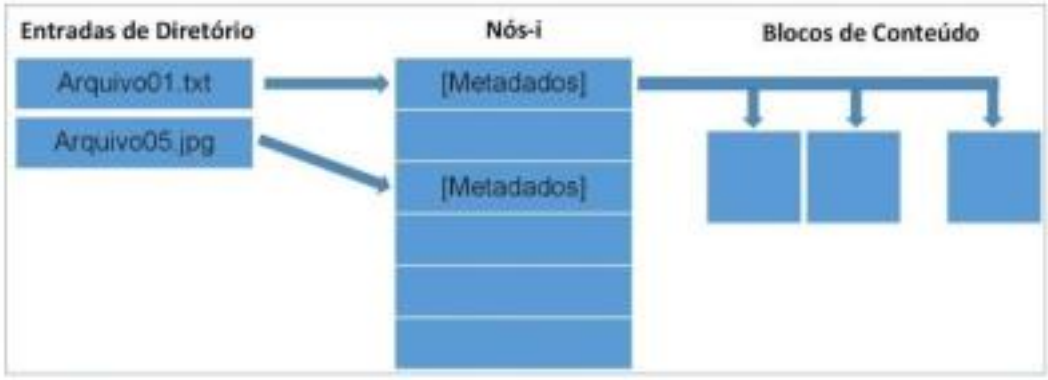
3. Alocação encadeada com tabela na memória: Ao invés de um ponteiro, o sistema possui uma tabela, chamada de FAT (File Allocation Table), com o endereço de cada bloco). O problema desta abordagem é que toda a tabela precisa estar na RAM.

Bloco físico	Aponta para
0	
1	

6. Gerencia de Armazenamento (Sistemas de Arquivos)

4. Alocação indexada: utiliza uma estrutura de índices para mapear diretórios e arquivos em blocos de dados no disco. Essa abordagem é comumente encontrada em sistemas de arquivos modernos, como NTFS. A alocação indexada oferece vantagens em termos de organização e eficiência na localização de dados.

Cada arquivo é associado a uma estrutura de dados denominada i-node (nó-índice), a qual lista os atributos e endereços de blocos do disco, conforme mostra a figura abaixo.



6. Gerencia de Armazenamento (Sistemas de Arquivos)

- Uma das principais funções do SO é garantir a segurança e consistência dos arquivos. Por esta razão, os sistemas modernas utilizam a técnica de **Journaling**.

- Journaling consiste em fazer o registro (log) das atividades que o sistema de arquivos irá executar (mas antes de executar). Desta forma, se algo der errado, é possível voltar ao momento anterior ao erro e desfazer a operação.

- Journaling não evita problemas, mas tenta recuperar caso algo errado ocorra. Em contrapartida, existe um custo computacional ao se utilizar esta técnica.

6. Gerencia de Armazenamento (Sistemas de Arquivos)

- Nem todos os sistemas de arquivos implementam journaling, como mostra a imagem:

  - No Windows:
    - "família" FAT (FAT12, FAT16, FAT32, exFAT): NÃO possuem *journaling*;
    - NTFS: possui *journaling*;
    - ReFS (*Resilient File System*): possui *journaling*. O ReFS é menos conhecido, foi introduzido inicialmente no Windows Server 2012 e no Windows 8. Foi projetado para oferecer maior resiliência e integridade de dados, incluindo mecanismos de detecção e correção automática de erros, bem como a capacidade de realizar verificações de integridade em segundo plano.
  - No Linux:
    - EXT2: NÃO possui *journaling*;
    - EXT3, EXT4: possuem journaling.
    - ReiserFS, XFS, JFS: possuem journaling.

6. Gerencia de Armazenamento (Sistemas de Arquivos)

- **Slack space (file slack):** O termo “slack space” (espaço ocioso) aplicado a um arquivo se refere ao espaço não utilizado em um cluster (unidade de armazenamento) de dados.

- Geralmente ocorre em qualquer sistema de arquivos, quando a quantidade de dados armazenada em um cluster não preenche completamente o espaço disponível.

- Ex.: cluster de tamanho 4 KB e um arquivo com tamanho 9 KB, são alocados 3 clusters, sendo que no terceiro haverá um slack space de 3 KB. Neste slack space pode haver dados já excluídos de um arquivo antigo que ocupava a mesma área, útil para a aplicação de perícia digital/computação forense.

6. Gerencia de Armazenamento (Sistemas de Arquivos)

- **Partição de Disco:** Uma partição de disco é como se fosse uma divisão lógica dentro de um disco rígido (HDD) ou SSD. Mesmo que o computador tenha um único disco físico, pode separá-lo em partes independentes, cada uma funcionando como se fosse um “disco separado”.

- Existem diversos softwares que permitem fazer o particionamento (inclusive trocando o sistema de arquivos) sem perder dados do disco, como o **Gparted** e **Particion Magic**.

7. Comandos Linux de Gerenciamento de Memória

- O comando "**free –h**" é o ponto inicial para gerenciamento da memória, mostra um resumo simples do uso da memória:

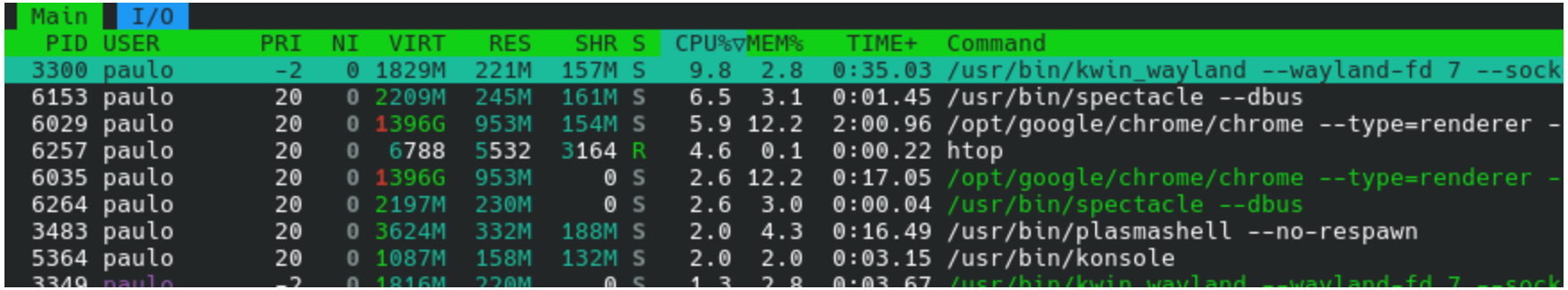
```
               total    usado    livre  compart  buff/cache  disponivel
Mem:            15Gi     6.2Gi    2.1Gi    500Mi      6.7Gi        8.5Gi
Swap:          2.0Gi     0.5Gi    1.5Gi
```

7. Comandos Linux de Gerenciamento de Memória

- O comando "**htop**" é uma interface moderna que mostra os processos rodando. Apertando F6 é possível ordenar por gasto de memória, processador, entre outros.

```
#pode precisar instalar
sudo apt install htop

#htop mostra os processos com seu uso de recursos
htop
```



7. Comandos Linux de Liberação de Cache

- O comando grava o conteúdo gravado em cache que ainda não foi gravado no disco, e libera a cache. Na prática, não precisa ser feito regularmente, tendo em vista que a cache é benéfica ao desempenho.

#sync grava a cache no disco. Echo 3 limpa a cache e tee grava no arquivo de cache para liberar
sync; echo 3 | sudo tee /proc/sys/vm/drop_caches

8. Comandos Linux para Gerenciamento de Disco

- Monitorar o uso do disco (velocidade de leitura e escrita) são atividades importantes. O quadro abaixo mostra o comando "iotop":

#iotop mostra o uso de leitura e escrita dos processos
sudo apt install iotop
sudo iotop

```
Total DISK READ:         0.00 B/s | Total DISK WRITE:       262.71 K/s
Current DISK READ:       0.00 B/s | Current DISK WRITE:     154.75 K/s
    TID  PRIO  USER     DISK READ  DISK WRITE>    COMMAND
    373 be/4 root        0.00 B/s  147.55 K/s systemd-journald
    324 be/3 root        0.00 B/s   89.97 K/s [jbd2/nvme0n1p2-8]
   4136 be/4 paulo       0.00 B/s   21.59 K/s renderD128 --crashpad-handler-p
   1833 be/4 pcp         0.00 B/s    3.60 K/s pmproxy -F -A
      1 be/4 root        0.00 B/s    0.00 B/s init
      2 be/4 root        0.00 B/s    0.00 B/s [kthreadd]
      3 be/4 root        0.00 B/s    0.00 B/s [pool_workqueue_release]
      4 be/0 root        0.00 B/s    0.00 B/s [kworker/R-kvfree_rcu_reclaim]
      5 be/0 root        0.00 B/s    0.00 B/s [kworker/R-rcu_gp]
```

8. Comandos Linux para Gerenciamento de Disco

- O comando "iostat" mostra estatísticas do uso do disco. O valor "tps" é o número de operações por segundo. Leitura e escrita por segundo também são parâmetros importantes. O "dscd" é o descarte de dados (não muito importante).

```
#iotop mostra o uso de leitura e escrita dos processos
sudo apt install sysstat
iostat
```

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           6,49    0,00    1,31    0,13    0,00   92,08

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
loop0             0,02         0,15         0,00         0,00        363          0          0
loop1             0,02         0,29         0,00         0,00        697          0          0
loop2             0,02         0,15         0,00         0,00        354          0          0
loop3             0,24         8,43         0,00         0,00      20318          0          0
loop4             0,00         0,01         0,00         0,00         14          0          0
nvme0n1          48,67       777,69       606,00         0,00    1874931    1461005          0
```

9. Benchmark no Disco com a Ferramenta FIO

- O **`fio`** é a ferramenta mais completa pra benchmark de disco no Linux. O **fio** (*Flexible I/O Tester*) é uma ferramenta para simular cargas reais de disco, testar **SSD, HDD, NVMe** e medir velocidade (MB/s). Diferente do dd, ele simula cenários reais (tipo banco de dados, servidor, etc.)

- Sua estrutura básica:

```
fio --name=teste --rw=randrw --bs=4k --size=1G --numjobs=4 --runtime=60 --group_reporting
```

`--name:` Nome do teste
--rw: tipo de operação
--bs: tamanho do bloco. Bloco pequeno simula BD, blocos grandes simula transferência de arq.
--numjobs: número de threads (vários processos e/ou usuários)
--size: tamanho dos dados
--runtime: tempo de teste
--ioengine=libaio (padrão do linux)
--group_reporting junta resultados para facilitar leitura

9. Benchmark no Disco com a Ferramenta FIO

- A tabela abaixo mostra como definir o parâmetro rw:

Valor	Significado
`read`	leitura
`write`	escrita
`randread`	leitura aleatória
`randwrite`	escrita aleatória
`randrw`	leitura + escrita aleatória

9. Benchmark no Disco com a Ferramenta FIO

- Vejamos alguns exemplos prontos:

```
#Teste sequencial (arquivos grandes)
fio --name=seq_read --rw=read --bs=1M --size=1G --runtime=30 --group_reporting

#Teste que simula serviços como banco de dados
fio --name=rand --rw=randrw --bs=4k --size=1G --numjobs=4 --runtime=60 --group_reporting

#Teste de escrita
fio --name=write --rw=write --bs=1M --size=1G --runtime=30 --group_reporting
```

9. Benchmark no Disco com a Ferramenta FIO

- O resultado é mostrado como na imagem abaixo, onde "bw" é velocidade, "IOPS" é operaçõe spor segundo e "lat" é tempo de resposta.

```
READ: bw=1200MiB/s, IOPS=300k, lat=0.5ms
WRITE: bw=800MiB/s, IOPS=200k, lat=1.2ms
```

10. Como o Windows Gerencia e Memória

- Tanto o Windows como o Linux dividem a memória em **blocos pequenos** (geralmente 4 KB) chamados **páginas.** O sistema fica movendo páginas entre RAM (rápida) e Disco (lento).

- O Windows 11 é muito eficaz e automatizado no gerenciamento de memória, baseado no conceito de **working set** (páginas que ele está usando agora). Exemplo: Chrome aberto com 10 abas, mas 2 abas estão sendo usadas, as outras podem sair da RAM.

- Quando o Windows precisa liberar a RAM ele segue as etapas:

1 - Diminui o working set de processos

2 - Remove páginas menos usadas

3 – Decide enetre comprimir ou mandar pro disco.

10. Como o Windows Gerencia e Memória

- Antes de usar o disco o Windows **comprime os dados na própria RAM,** buscando economizar espaço sem usar disco. O Windows evita o uso do disco ao máximo.

- Se não houver mais o que fazer, o Windows faz o swap, que basicamente é enviar os dados da RAM para um arquivo chamado **pagefile.sys.**

- O arquivo pagefile.sys no Windows 11 fica localizado na raiz da unidade do sistema, geralmente em C:\. Ele é um arquivo oculto e protegido do sistema, utilizado como memória virtual (RAM).

- Para visualizá-lo, é necessário habilitar a opção "Mostrar arquivos, pastas e unidades ocultas" e desmarcar "Ocultar arquivos protegidos do sistema operacional" nas opções de pasta do Explorador de Arquivos.

10. Como o Debian Gerencia e Memória

- O Debian 13 segue uma filosofia diferente, **usar RAM ao máximo e decidir agressivamente** o que fazer em caso de falta de espaço na memória.

- O Linux usa a técnica de Page Cahce, ou seja, os arquivos acessados ficam na RAM para acelerar o sistema. RAM “cheia” nem sempre é ruim.

- O Debian trabalha com um parâmetro chamado swappiness, que define quando usar swap (arquivo /etc/sysctl.conf):

1 - baixo (10–20): evita swap

2 - médio (60): padrão

3 - alto (80–100): usa swap cedo

10. Como o Debian Gerencia e Memória

- Quando precisa liberar RAM, o Debian remove cache (se possível), move páginas pouco usadas para swap e continua usando RAM para coisas importantes.

- Linux NÃO gosta de travar o sistema, se a memória acabar (RAM e swap), ele usa o OOM Killer. Isso consiste em escolher um processo e matar para liberar memória. Exemplo: Fecha um navegador pesado automaticamente.

- O **swappiness** é um número de **0 a 100** que diz: “Quão cedo eu devo começar a usar swap?”. Lembrando que **Swappiness NÃO significa “quanto usar swap”, significa “quando começar a usar swap”.**

0 → quase nunca usa swap

10–20 → evita swap ao máximo

60 → padrão (equilíbrio)

80–100 → usa swap bem cedo

10. Como o Debian Gerencia e Memória

- Exemplo prático:

🟢 **swappiness = 10**

- Linux tenta manter tudo na RAM
- só usa swap quando está quase sem saída

👉 Resultado:

- menos uso de disco
- risco maior de OOM Killer

🟡 **swappiness = 60 (padrão)**

- começa a mover coisas pouco usadas para swap antes de travar

👉 Resultado:

- sistema mais equilibrado

🔴 **swappiness = 90**

- usa swap cedo
- mantém RAM mais livre

👉 Resultado:

- mais uso de disco
- menos risco de OOM