O sistema operacional

Memória virtual

- A memória virtual foi usada pela primeira vez em alguns computadores durante a década de 1960, a maioria deles associada com projetos de pesquisa na área de sistemas de computação.

- No início da década de 1970, a memória virtual já estava disponível na maioria dos computadores.

- Agora, até computadores de um só chip, incluindo o Core i7 e a CPU ARM do OMAP4430, têm sistemas de memória virtual altamente sofisticados.

Paginação

- A ideia de separar o espaço de endereço e os endereços de memória é a seguinte.
- Em qualquer instante, 4.096 palavras de memória podiam ser acessadas diretamente, mas elas não precisam corresponder a endereços de memória 0 a 4.095.
- A técnica de sobreposição automática é denominada **paginação**.
- Como o programador pode programar como se a paginação não existisse, diz-se que o mecanismo de paginação é **transparente**.

Paginação

- Os primeiros 64 KB do espaço de endereço virtual divididos em 16 páginas, cada página com 4 K.

Quadro de página	Endereços virtuais
≈	≈
15	61440 – 65535
14	57344 – 61439
13	53248 – 57343
12	49152 – 53247
11	45056 – 49151
10	40960 – 45055
9	36864 – 40959
8	32768 – 36863
7	28672 – 32767
6	24576 – 28671
5	20480 – 24575
4	16384 – 20479
3	12288 – 16383
2	8192 – 12287
1	4096 – 8191
0	0 – 4095

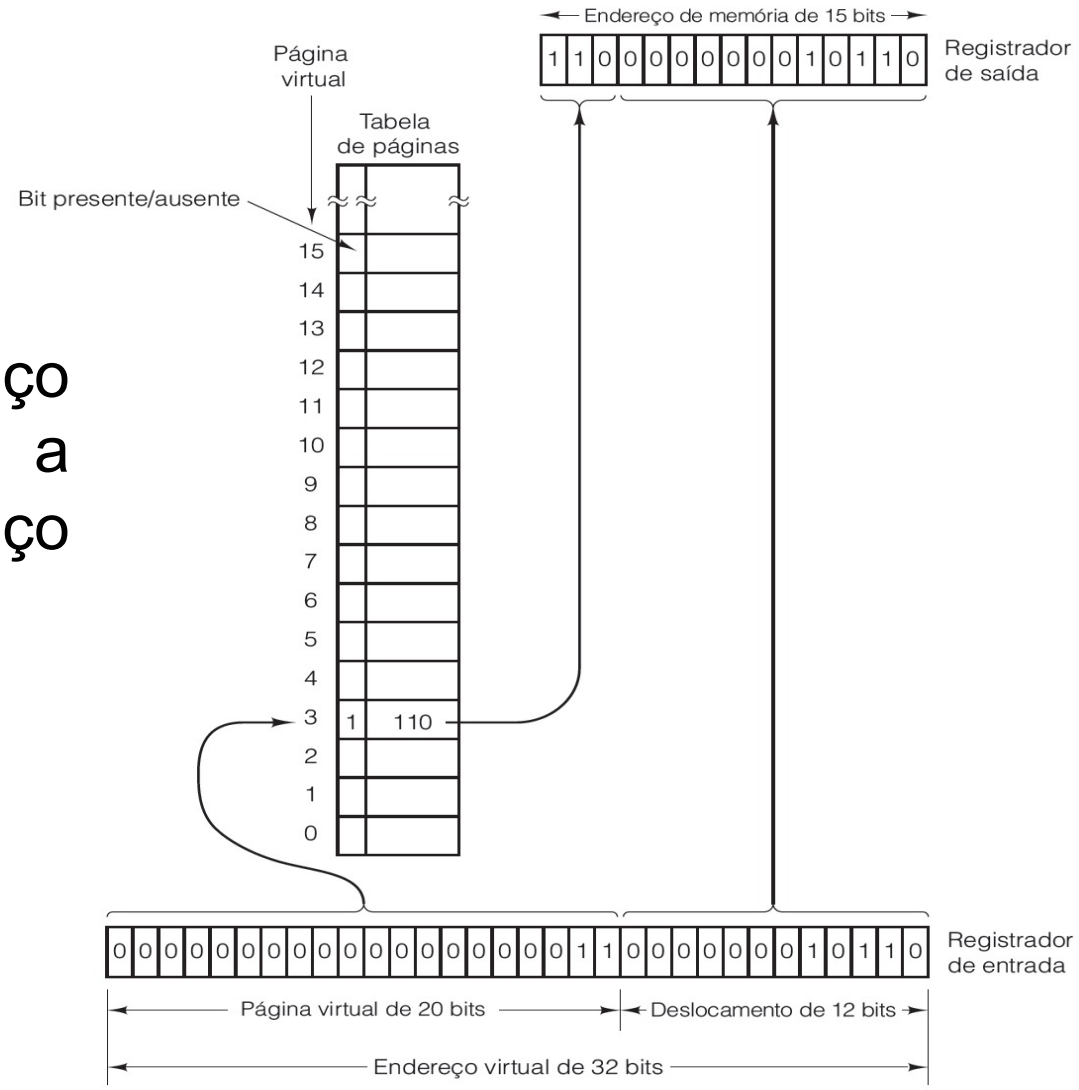
Paginação

- Memória principal de 32 KB dividida em oito quadros de página de 4 KB cada.

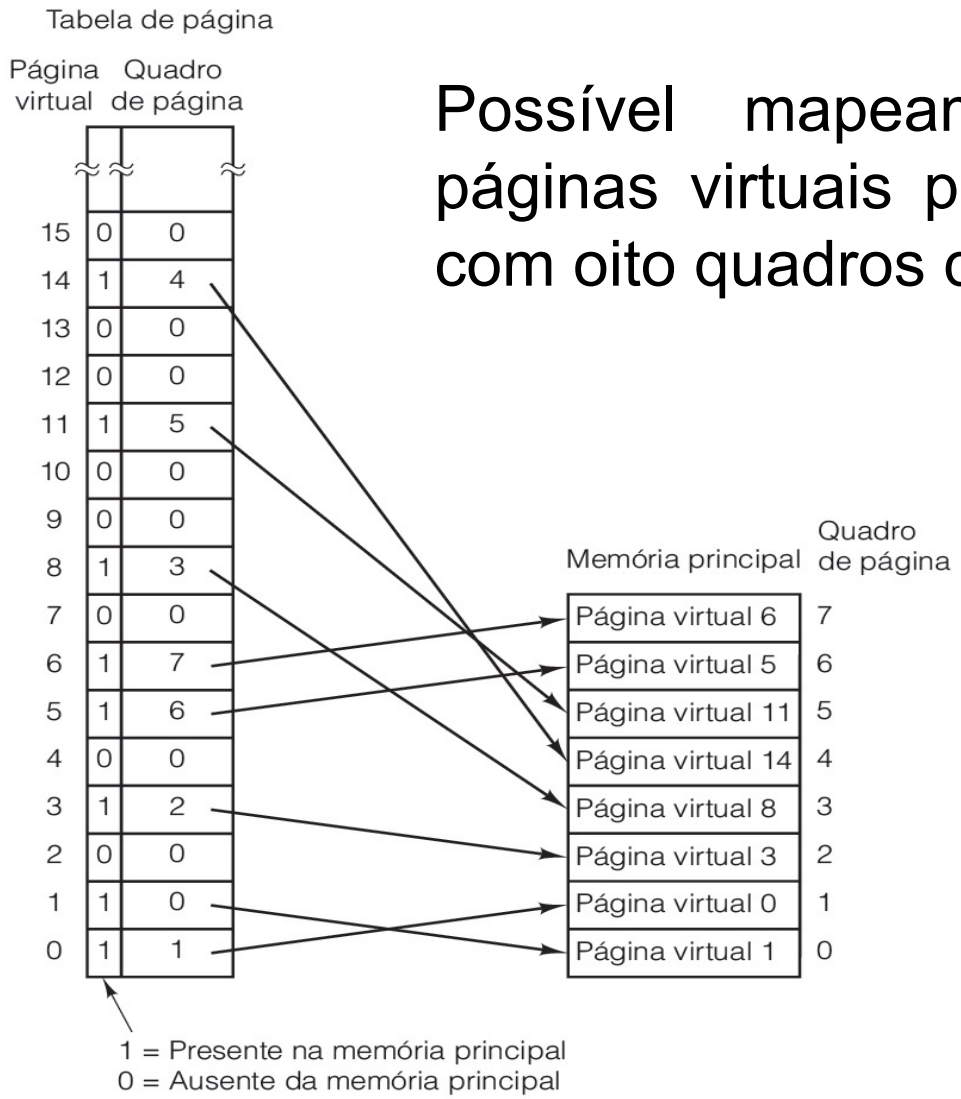
Quadro de página	32 KB da parte inferior da memória principal Endereços físicos
7	28672 – 32767
6	24576 – 28671
5	20480 – 24575
4	16384 – 20479
3	12288 – 16383
2	8192 – 12287
1	4096 – 8191
0	0 – 4095

Paginação

Formação de um endereço de memória principal a partir de um endereço virtual.



Paginação



Possível mapeamento das 16 primeiras páginas virtuais para uma memória principal com oito quadros de página.

Paginação

- Quando a CPU tenta buscar a primeira instrução, obtém imediatamente uma falta de página, e isso faz com que a página que contém a primeira instrução seja carregada na memória e registrada na tabela de página.

- Então, a primeira instrução pode começar.

- Se ela tiver dois endereços, e estes estiverem em páginas diferentes, e ambas forem diferentes da página da instrução, ocorrerão mais duas faltas de página e mais duas páginas serão trazidas antes que a instrução possa, por fim, ser executada.

Paginação

- A próxima instrução pode causar mais algumas faltas de página e assim por diante.
- Esse método de operar uma memória virtual é denominado **paginação por demanda**.
- Quando um programa referencia uma página que não está na memória principal, ela deve ser buscada no disco.
- Um algoritmo popular extrai a página menos recentemente usada porque é alta a probabilidade *a priori* de ela não estar no conjunto de trabalho atual. Ele é denominado algoritmo **LRU**.

Paginação

- Outro algoritmo de substituição de página é o **FIFO**.
- O FIFO remove a página menos recentemente carregada, independente de quando essa página foi referenciada pela última vez.
- Se por acaso acontecer de o programa e os dados do usuário preencherem exatamente um número inteiro de páginas, não haverá nenhum espaço desperdiçado quando eles estiverem na memória.
- Caso contrário, haverá algum espaço não utilizado na última página.

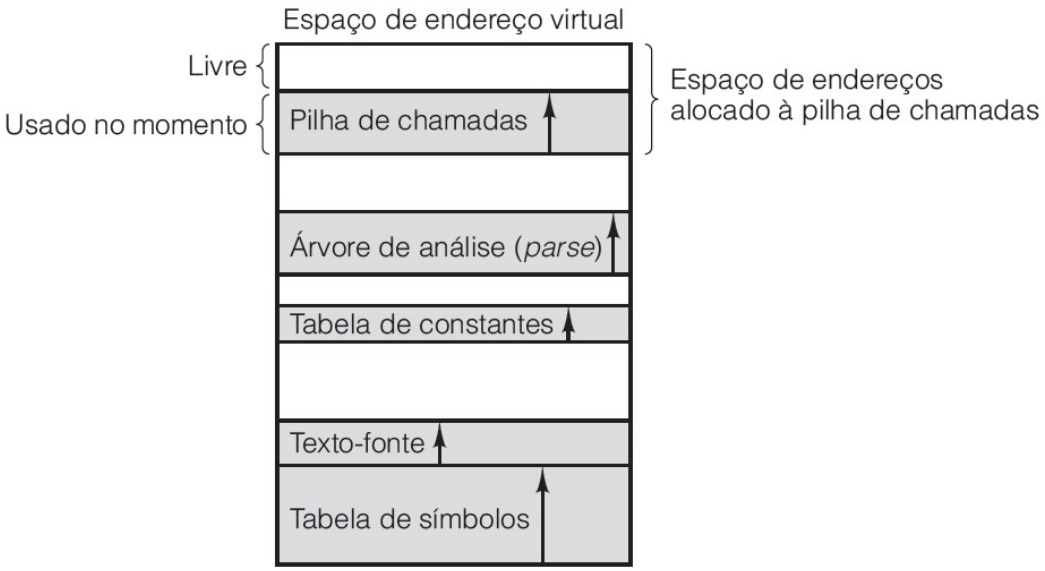
Paginação

- O problema desses bytes desperdiçados é denominado **fragmentação interna**.
- Páginas pequenas fazem uso ineficiente da largura de banda do disco.
- Páginas pequenas também têm a vantagem de, se o conjunto de trabalho consistir em um grande número de regiões pequenas e separadas no espaço de endereço virtual, pode haver menos acessos ao disco (paginação excessiva) com uma página de tamanho pequeno do que com uma de tamanho grande.

Paginação

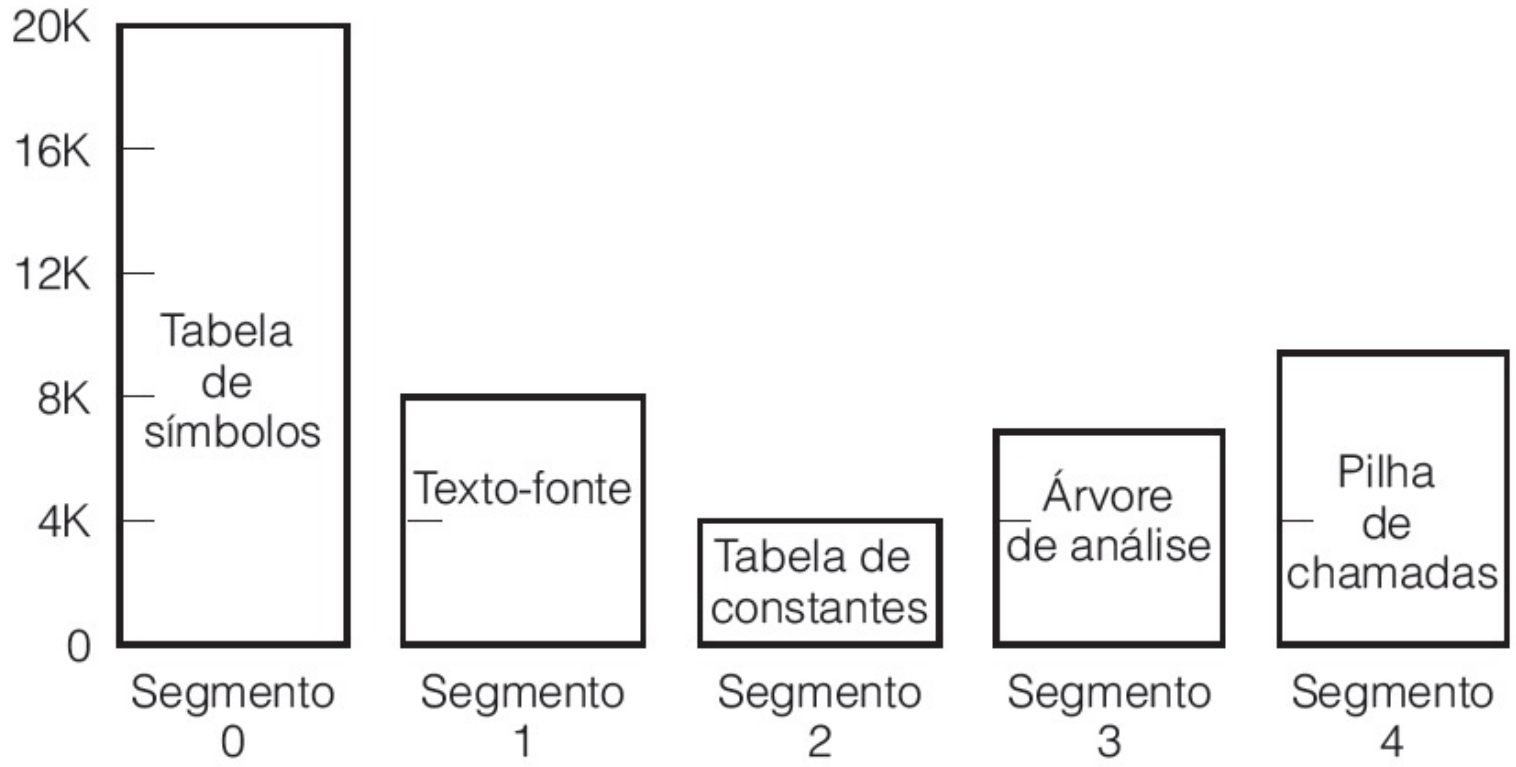
- Há muitos problemas para os quais poderia ser melhor ter dois ou mais espaços de endereços virtuais separados do que ter só um.

- Em um espaço de endereço unidimensional com tabelas que aumentam, uma tabela pode encostar em outra.



Paginação

- Como cada segmento constitui um espaço de endereço separado, diferentes segmentos podem se expandir ou encolher independentemente, sem que um afete o outro.



Paginação

- Comparação entre paginação e segmentação.

Consideração	Paginação	Segmentação
O programador precisa estar ciente dela?	Não	Sim
Quantos espaços de endereços lineares há?	1	Muitos
O espaço de endereço virtual pode ser maior do que o tamanho da memória?	Sim	Sim
Tabelas de tamanhos variáveis podem ser manipuladas com facilidade?	Não	Sim
Por que a técnica foi inventada?	Para simular memórias grandes	Para fornecer vários espaços de endereço

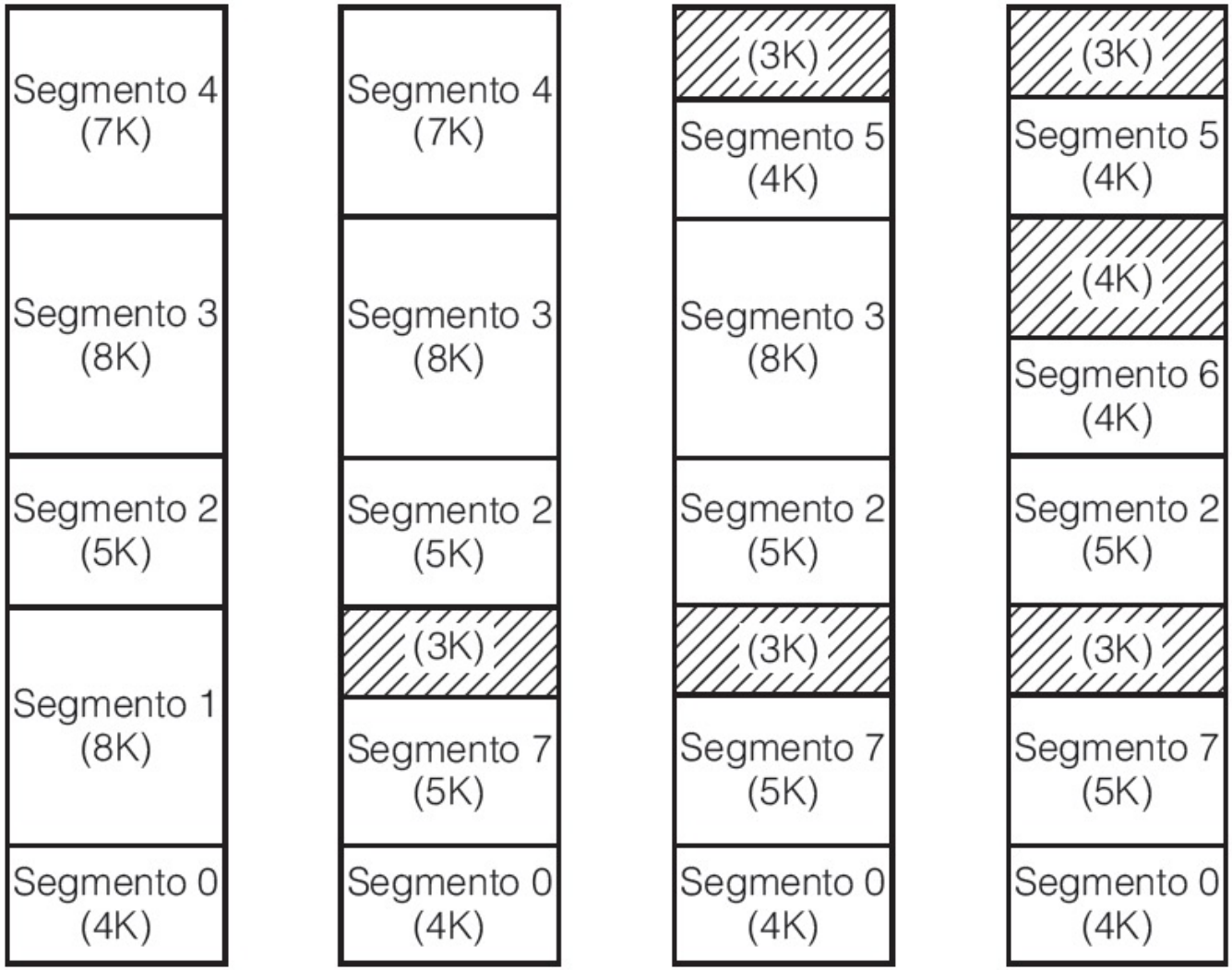
Paginação

- Há uma diferença essencial entre a implementação de segmentação e a de paginação: páginas têm tamanho fixo, mas segmentos não.

- O desenvolvimento de fragmentação externa é exibido na figura a seguir.

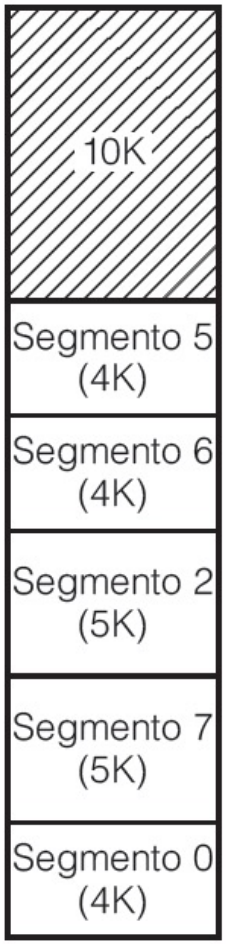
- Um modo de evitar a fragmentação externa é o seguinte: toda vez que aparecer uma lacuna, mover os segmentos que a seguem para mais perto do local 0 da memória, o que elimina aquela lacuna, mas deixa uma grande no final.

Paginação



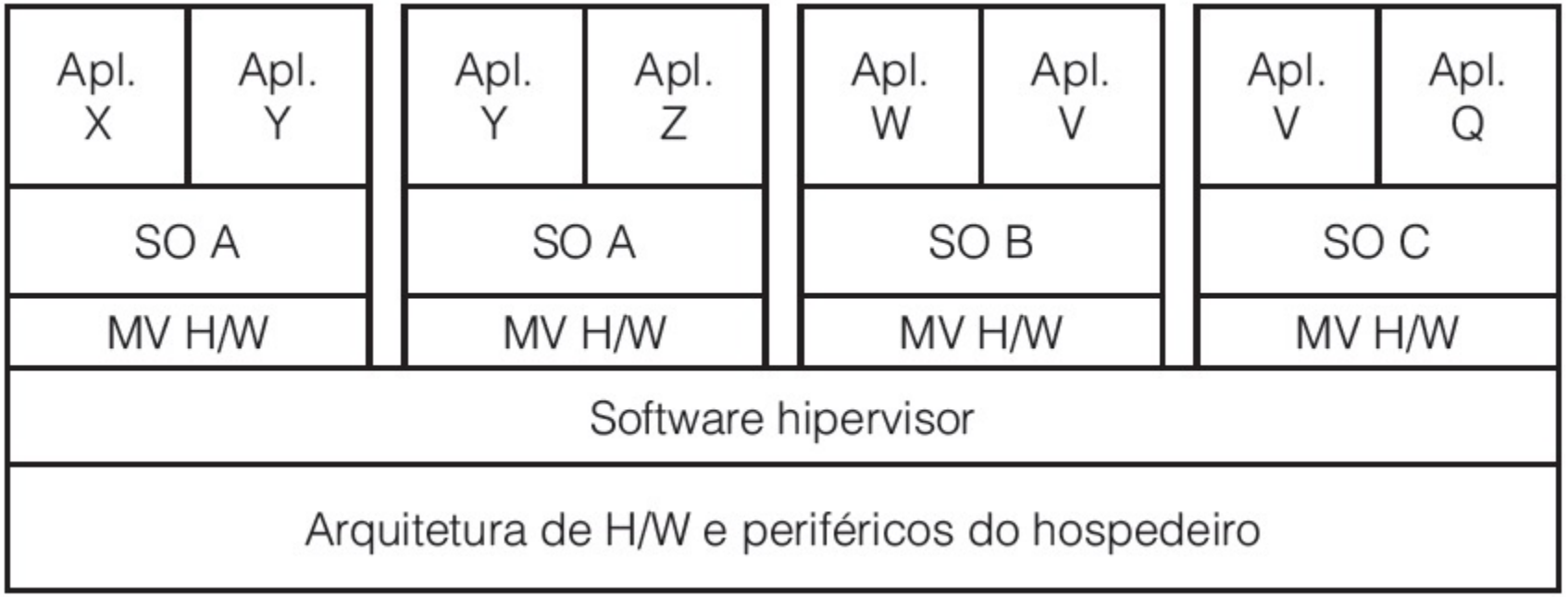
Paginação

- Remoção da fragmentação externa por compactação.



Virtualização do hardware

- A **virtualização do hardware**, ilustrada abaixo, é uma combinação de suporte de hardware e software que permite a execução simultânea de múltiplos sistemas operacionais em um único computador físico.



Instruções de E/S de nível OSM

- O conjunto de instruções de nível OSM contém grande parte das instruções de nível ISA com a adição de algumas instruções novas, porém importantes, e a remoção de algumas poucas instruções potencialmente perigosas.

- Um modo de organizar a E/S virtual é usar uma abstração denominada **arquivo**.

- Em sua forma mais simples, um arquivo consiste em uma sequência de bytes escrita em um dispositivo de E/S.

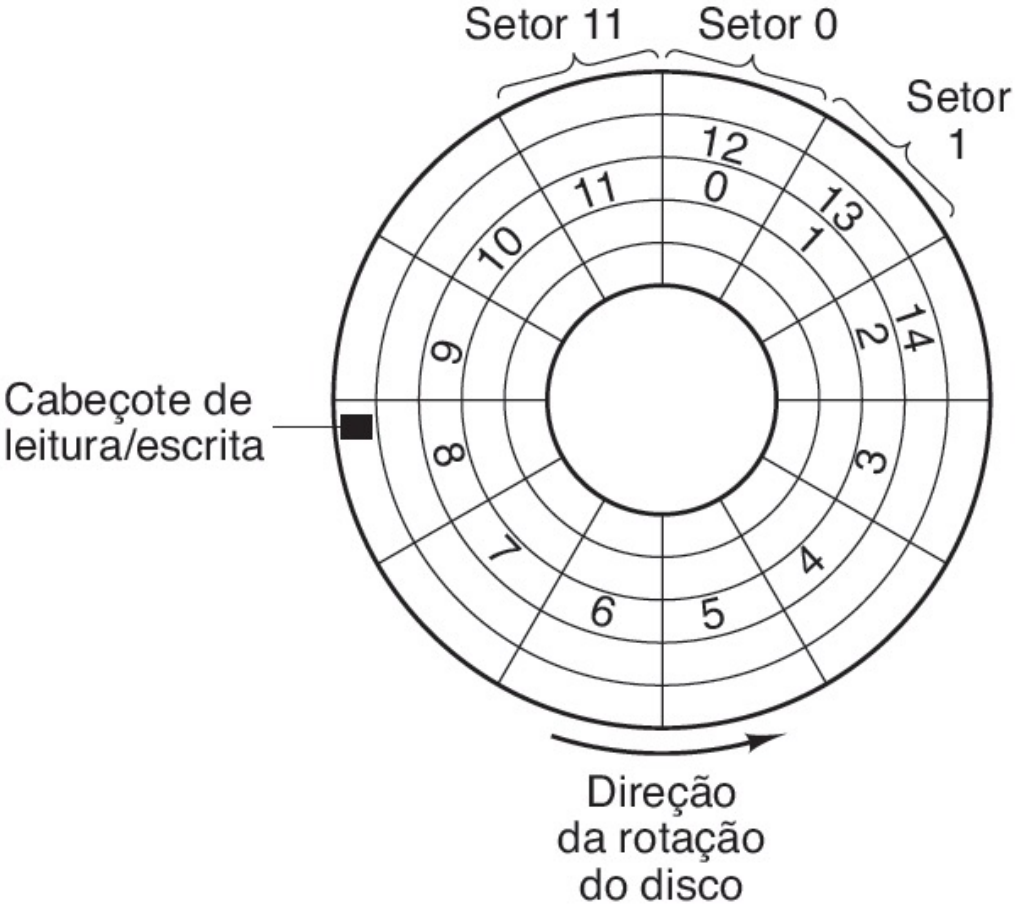
- Para o sistema operacional, um arquivo é em geral apenas uma sequência de bytes.

Instruções de E/S de nível OSM

- Para entender como são realizadas instruções de E/S virtuais, é necessário examinar como os arquivos são organizados e armazenados.
- Há uma questão básica que todos os sistemas de arquivo devem encarar: a alocação de armazenamento.
- Uma questão fundamental é se um arquivo é armazenado em unidades de alocação consecutivas ou não.
- Há uma importante distinção entre a visão que um programador de aplicação tem de um arquivo e a visão que o sistema operacional tem desse arquivo.

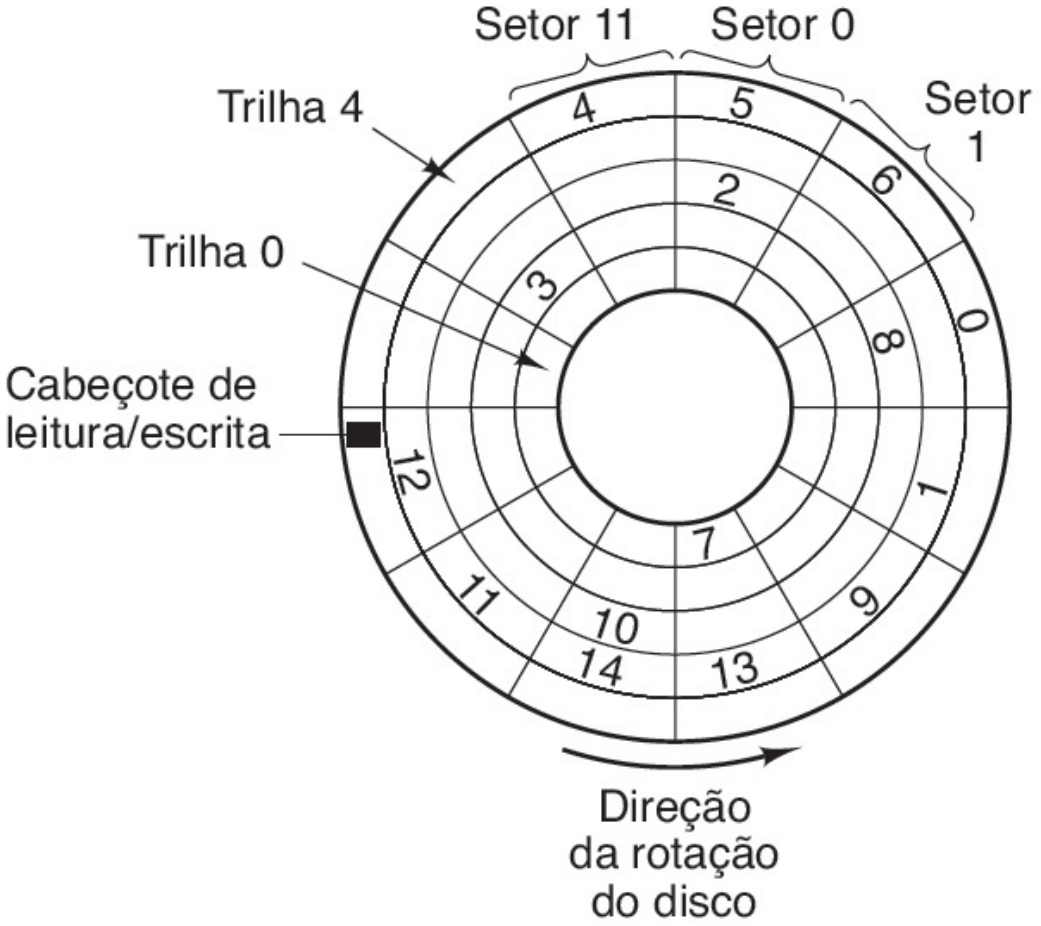
Instruções de E/S de nível OSM

- Estratégias de alocação de disco. Arquivo em setores consecutivos.



Instruções de E/S de nível OSM

- Estratégias de alocação de disco. Arquivo em setores não consecutivos.



Instruções de E/S de nível OSM

- Dois modos de monitorar setores disponíveis. Lista de livres.

Trilha	Setor	Número de setores na lacuna
0	0	5
0	6	6
1	0	10
1	11	1
2	1	1
2	3	3
2	7	5
3	0	3
3	9	3
4	3	8

Instruções de E/S de nível OSM

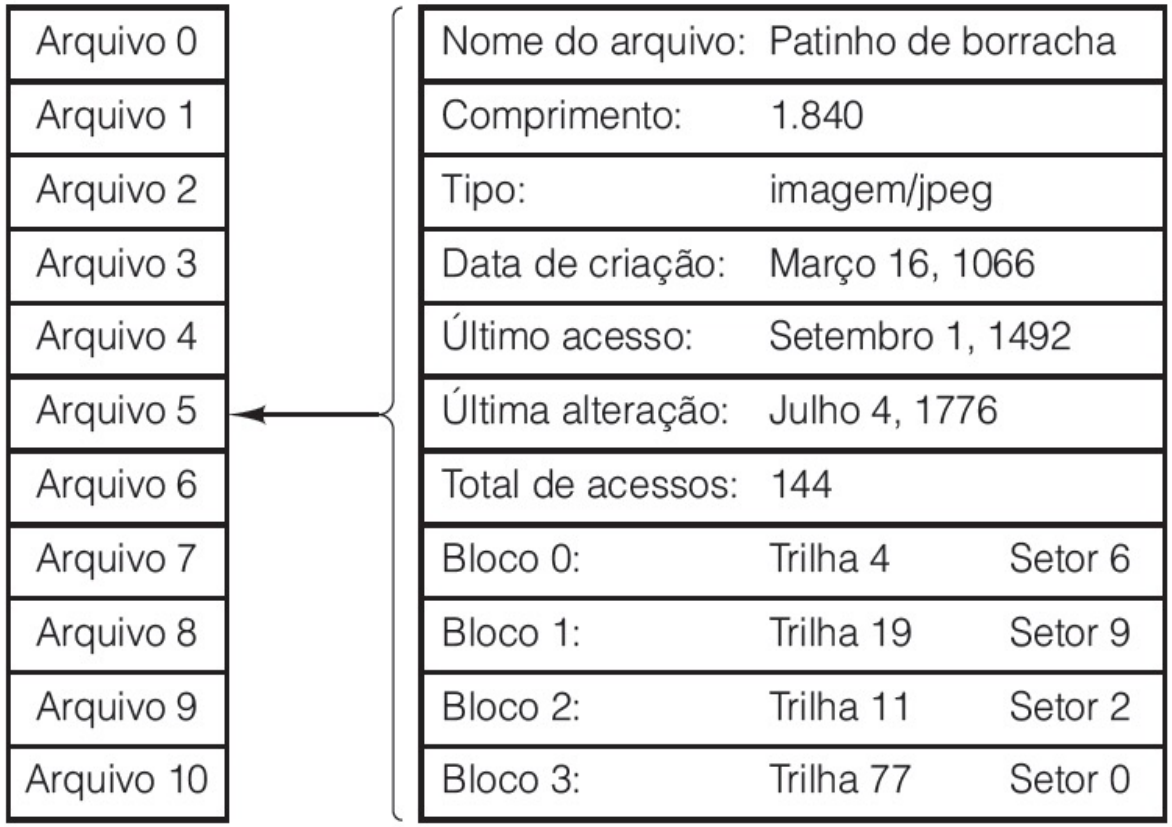
- Dois modos de monitorar setores disponíveis. Mapa de bits.

	Setor											
Trilha	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0
2	1	0	1	0	0	0	1	0	0	0	0	0
3	0	0	0	1	1	1	1	1	1	0	0	0
4	1	1	1	0	0	0	0	0	0	0	0	1

Instruções de E/S de nível OSM

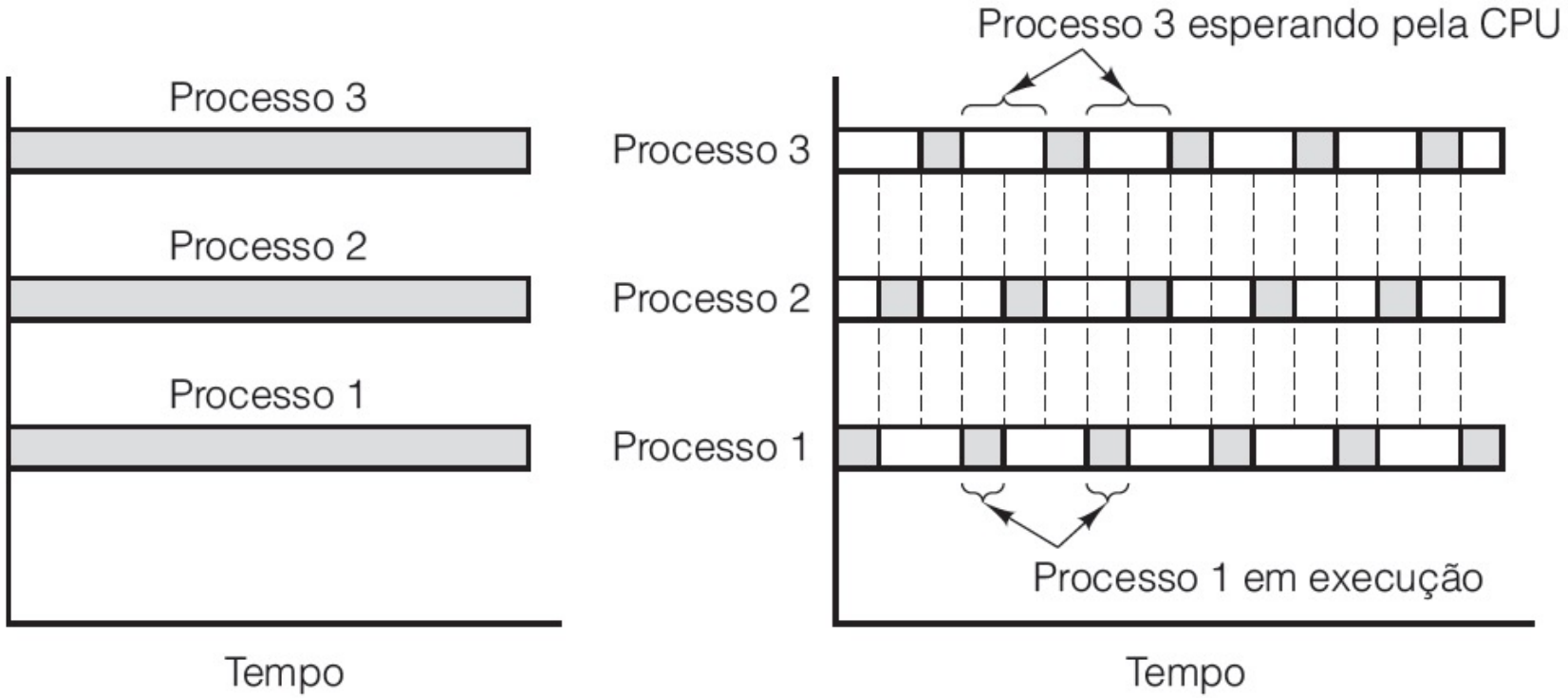
- O modo usual de um sistema operacional organizar arquivos em linha é agrupá-los em **diretórios**.

Diretório de arquivos de usuário e o conteúdo de uma entrada típica em um diretório de arquivo.



Instruções de nível OSM para processamento paralelo

- A figura abaixo mostra a diferença entre processamento verdadeiramente paralelo, com mais de um processador físico, e paralelo simulado, com só um processador físico.



Instruções de nível OSM para processamento paralelo

- Quando um programa deve ser executado, ele deve rodar como parte de algum processo.
- Esse processo é caracterizado por um estado e um espaço de endereço por meio do qual o programa e os dados podem ser acessados.
- Em muitos casos, processos paralelos precisam se comunicar e sincronizar de modo a realizar seu trabalho.

Exemplos de sistemas operacionais

UNIX

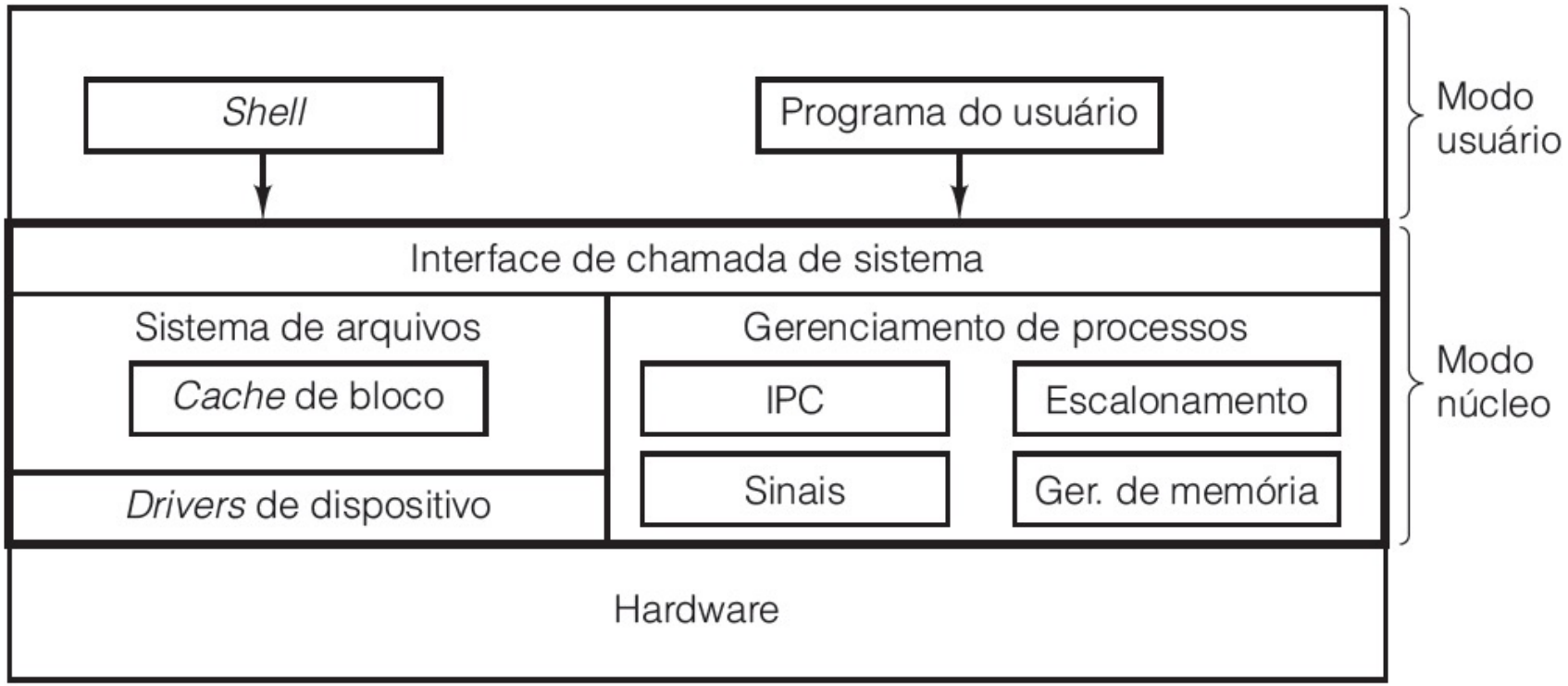
- Divisão aproximada de chamadas de sistema do UNIX.

Categoria	Alguns exemplos
Gerenciamento de arquivo	Abrir, ler, escrever, fechar e bloquear arquivos
Gerenciamento de diretório	Criar e apagar diretórios; mover arquivos de um lado para outro
Gerenciamento de processo	Gerar, encerrar, monitorar e sinalizar processos
Gerenciamento de memória	Compartilhar memória entre processos; proteger páginas
Obter/ajustar parâmetros	Obter ID de usuário, grupo, processo; definir prioridade
Datas e horários	Definir horários de acesso a arquivos; usar temporizador de intervalo; execução de perfil
Rede	Estabelecer/aceitar conexão; enviar/receber mensagens
Diversos	Habilitar contabilidade; manipular cotas de disco; reiniciar o sistema

Exemplos de sistemas operacionais

UNIX

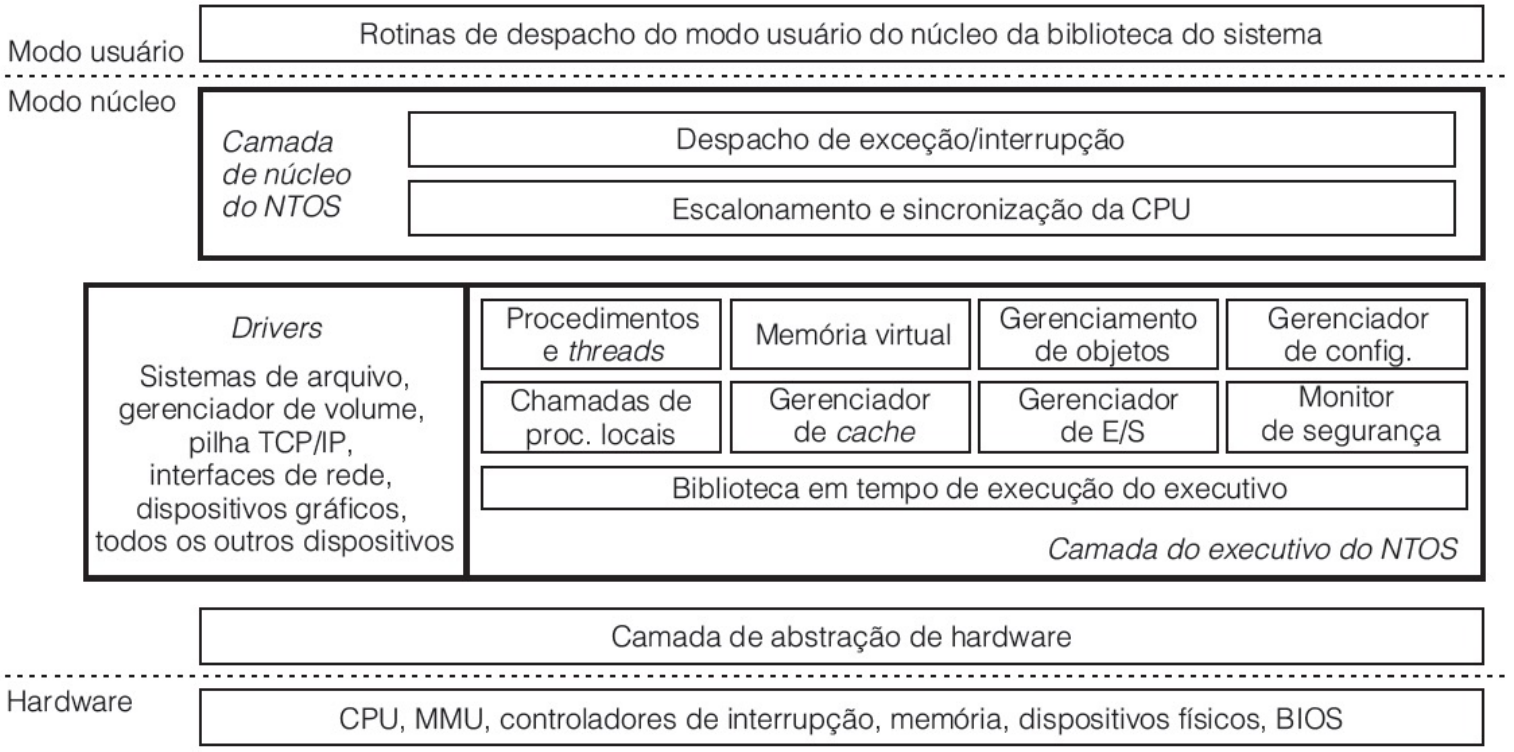
- Estrutura de um sistema UNIX típico.



Exemplos de sistemas operacionais

Windows 7

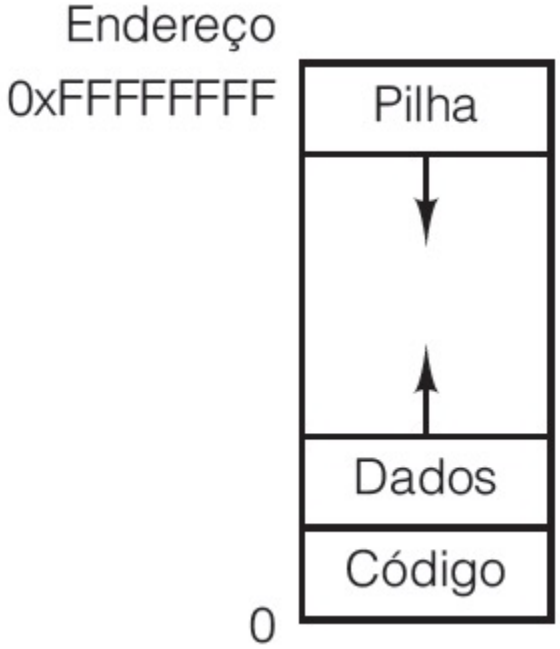
- A estrutura do Windows 7.



Exemplos de sistemas operacionais

Memória virtual do UNIX

- Espaço de endereço de um único processo UNIX.



Exemplos de sistemas operacionais

Memória virtual do Windows 7

- Todo processo do usuário tem seu próprio espaço de endereço virtual.
- Cada página virtual pode estar em um de três estados: livre, reservada ou comprometida.
- Cada página comprometida tem uma página sombra no disco.
- Permite de maneira explícita que dois ou mais processos sejam mapeados para o mesmo arquivo ao mesmo tempo, possivelmente em endereços virtuais diferentes.

Exemplos de sistemas operacionais

E/S em UNIX

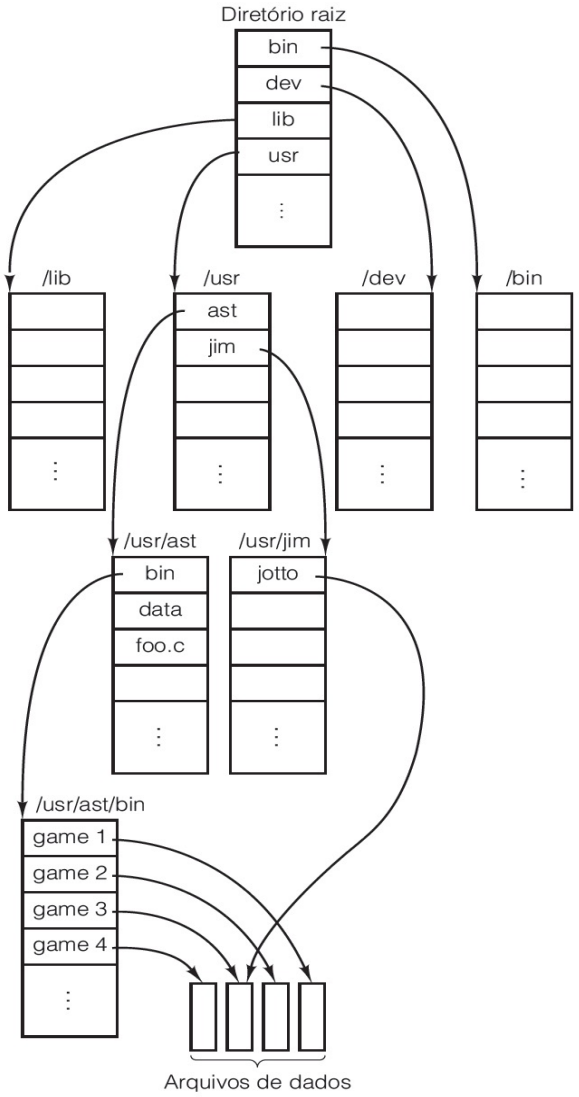
- A figura abaixo ilustra como funcionam as principais chamadas de E/S de arquivo.

```
/* Abre os descritores de arquivo. */
infd = open(˝data˝, 0);
outfd = creat(˝newf˝, ProtectionBits);

/* Laço de cópia. */
do {
        count = read(infd, buffer, bytes);
        if (count > 0) write(outfd, buffer, count);
} while (count > 0);

/* Fecha os arquivos. */
close(infd);
close(outfd);
```

Exemplos de sistemas operacionais

E/S em UNIX

Parte de um sistema de diretório típico do UNIX.



Exemplos de sistemas operacionais

E/S em UNIX

- Principais chamadas de gerenciamento de diretório do UNIX.

Chamada de sistema	Significado
mkdir(name, mode)	Cria um novo diretório
rmdir(name)	Apaga um diretório vazio
opendir(name)	Abre um diretório para leitura
readdir(dirponter)	Lê a próxima entrada em um diretório
close dir(dirponter)	Fecha um diretório
chdir(dirname)	Muda diretório de trabalho para *dirname*
link(name1, name2)	Cria uma entrada de diretório *name2* que aponta para *name1*
unlink(name)	Remove *name* de seu diretório

Exemplos de sistemas operacionais

E/S no Windows 7

- Principais funções da API Win32 para E/S de arquivo.

Função da API	UNIX	Significado
CreateFile	open	Cria um arquivo ou abre um arquivo existente; retorna um manipulador
DeleteFile	unlink	Exclui uma entrada de arquivos existentes de um diretório
CloseHandle	close	Fecha um arquivo
ReadFile	read	Lê dados de um arquivo
WriteFile	write	Escreve dados em um arquivo
SetFilePointer	lseek	Ajusta o ponteiro de arquivo para um local específico no arquivo
GetFileAttributes	stat	Retorna as propriedades do arquivo
LockFile	fcntl	Bloqueia uma região do arquivo para proporcionar exclusão mútua
UnlockFile	fcntl	Desbloqueia uma região do arquivo previamente bloqueada

Exemplos de sistemas operacionais

E/S no Windows 7

- Fragmento de programa para copiar um arquivo usando as funções da API do Windows 7.

```
/* Abra arquivos para entrada e saída. */
inhandle = CreateFile(˝data˝, GENERIC_READ, 0, NULL, OPEN_EXISTING, 0, NULL);
outhandle = CreateFile(˝newf˝, GENERIC_WRITE, 0, NULL, CREATE ALWAYS,
        FILE_ATTRIBUTE_NORMAL, NULL);

/* Copie o arquivo. */
do {
        s = ReadFile(inhandle, buffer, BUF_SIZE, &count, NULL);
        if (s > 0 && count > 0) WriteFile(outhandle, buffer, count, &ocnt, NULL);
} while (s > 0 && count > 0);

/* Feche os arquivos. */
CloseHandle(inhandle);
CloseHandle(outhandle);
```

Exemplos de sistemas operacionais

E/S no Windows 7

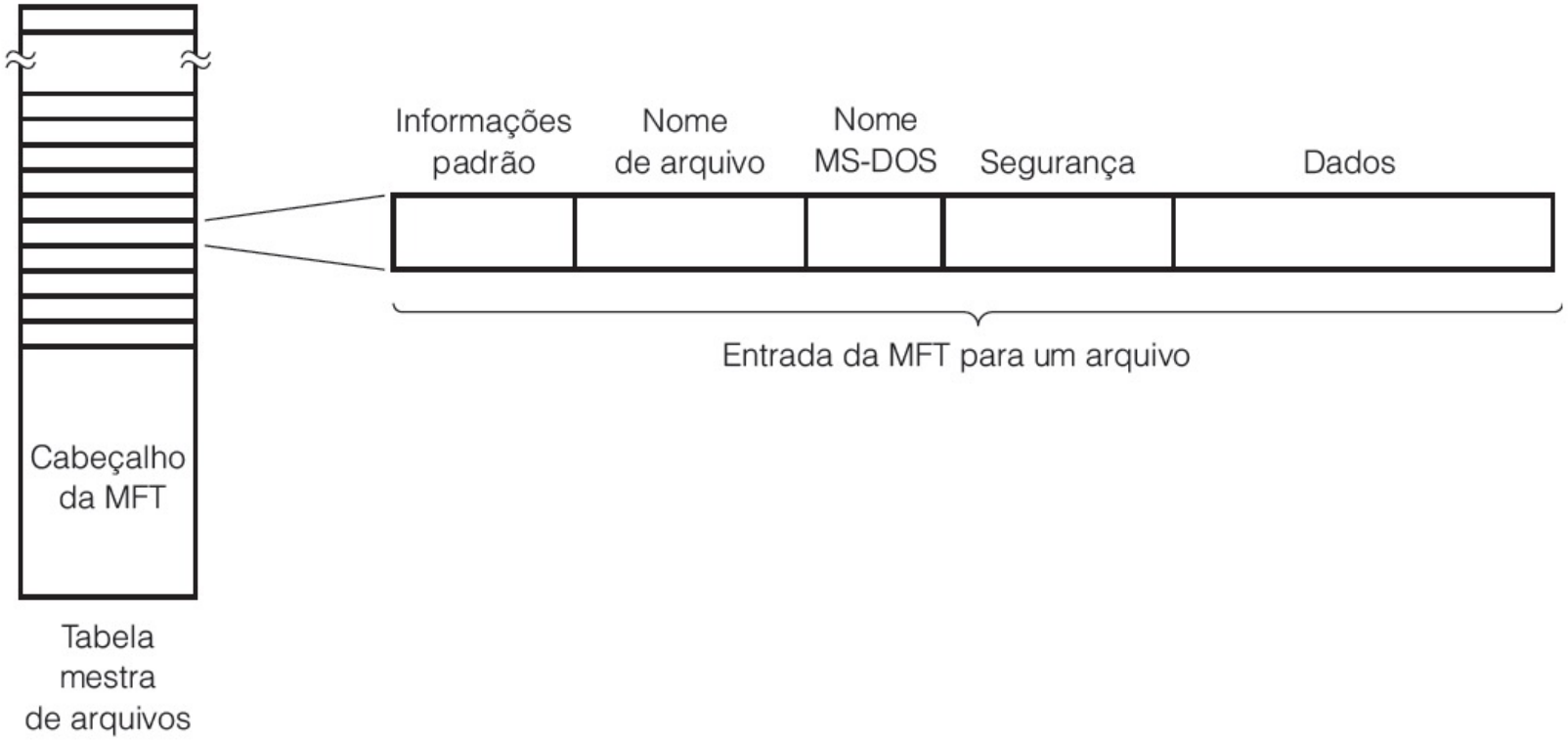
- Principais funções da API Win32 para gerenciamento de diretório.

Função da API	UNIX	Significado
CreateDirectory	mkdir	Cria um novo diretório
RemoveDirectory	rmdir	Remove um diretório vazio
FindFirstFile	opendir	Inicializa e começa a ler as entradas em um diretório
FindNextFile	readdir	Lê a próxima entrada de diretório
MoveFile		Move um arquivo de um diretório para outro
SetCurrentDirectory	chdir	Muda o diretório de trabalho corrente

Exemplos de sistemas operacionais

E/S no Windows 7

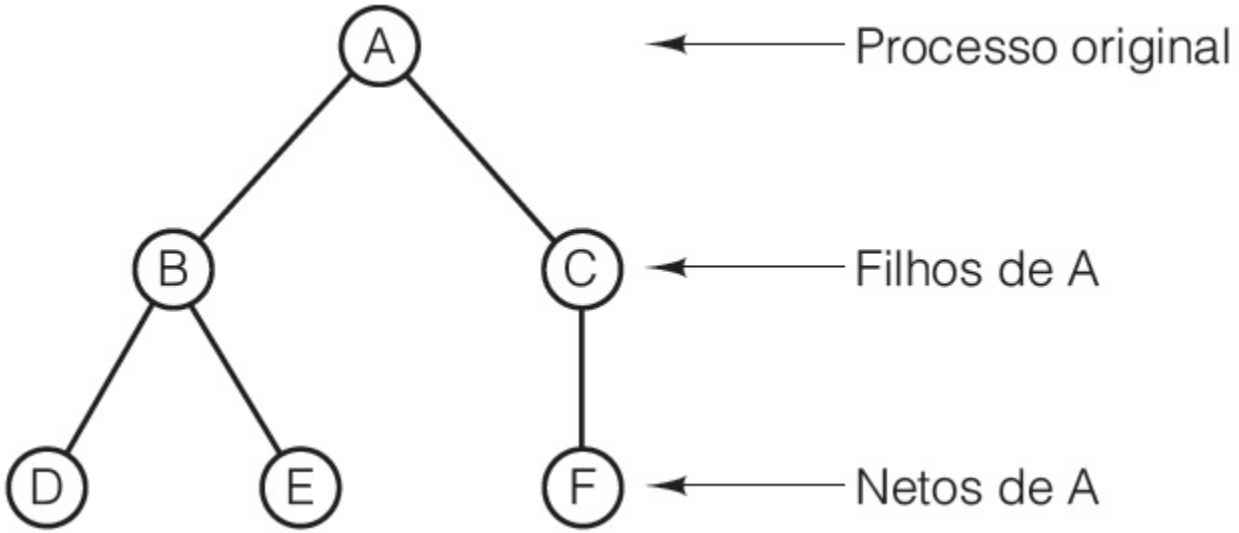
- Tabela mestra de arquivos do Windows 7.



Exemplos de sistemas operacionais

Gerenciamento de processo em UNIX

- Árvore de processos em UNIX.



Exemplos de sistemas operacionais

Gerenciamento de processo em UNIX

- Principais chamadas de thread POSIX.

Chamada de thread	Significado
pthread_create	Cria um novo *thread* no espaço de endereço do chamado
pthread_exit	Encerra o *thread* que está chamando
pthread_join	Espera que um *thread* encerre
pthread_mutex_init	Cria um novo *mutex*
pthread_mutex_destroy	Destrói um *mutex*
pthread_mutex_lock	Bloqueia um *mutex*
pthread_mutex_unlock	Desbloqueia um *mutex*
pthread_cond_init	Cria uma variável de condição
pthread_cond_destroy	Destrói uma variável de condição
pthread_cond_wait	Espera em uma variável de condição
pthread_cond_signal	Libera um *thread* que está esperando em uma variável de condição

Exemplos de sistemas operacionais

Gerenciamento de processo no Windows 7

- Em termos gerais, os dez parâmetros de CreateProcess são os seguintes:

1. Um ponteiro para o nome do arquivo executável.
2. A linha de comando em si (não analisada).
3. Um ponteiro para um descritor de segurança para o processo.

Exemplos de sistemas operacionais

Gerenciamento de processo no Windows 7

4. Um ponteiro para um descritor de segurança para o thread inicial.

5. Um bit que indica se o novo processo herda os manipuladores do criador.

6. Sinalizadores diversos.

7. Um ponteiro para as cadeias de ambiente.

Exemplos de sistemas operacionais

Gerenciamento de processo no Windows 7

8. Um ponteiro para o nome do diretório de trabalho corrente do novo processo.

9. Um ponteiro para uma estrutura que descreve a janela inicial na tela.

10. Um ponteiro para uma estrutura que retorna 18 valores para o chamado.