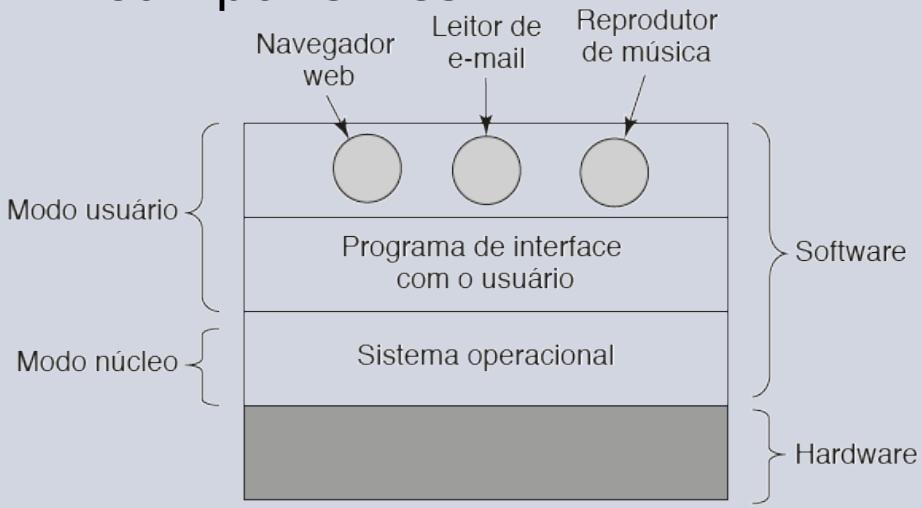
Capítulo 1:
Introdução

- A função do sistema operacional é fornecer aos programas do usuário um modelo do computador melhor, mais simples e mais limpo, assim como lidar com o gerenciamento de todos os recursos mencionados.
- A maioria dos computadores tem dois modos de operação: modo núcleo e modo usuário.

- O sistema operacional opera em modo núcleo (também chamado modo supervisor). Nesse modo, ele tem acesso completo a todo o hardware e pode executar qualquer instrução que

Observe, na figura a seguir, uma visão geral simplificada dos principais componentes:



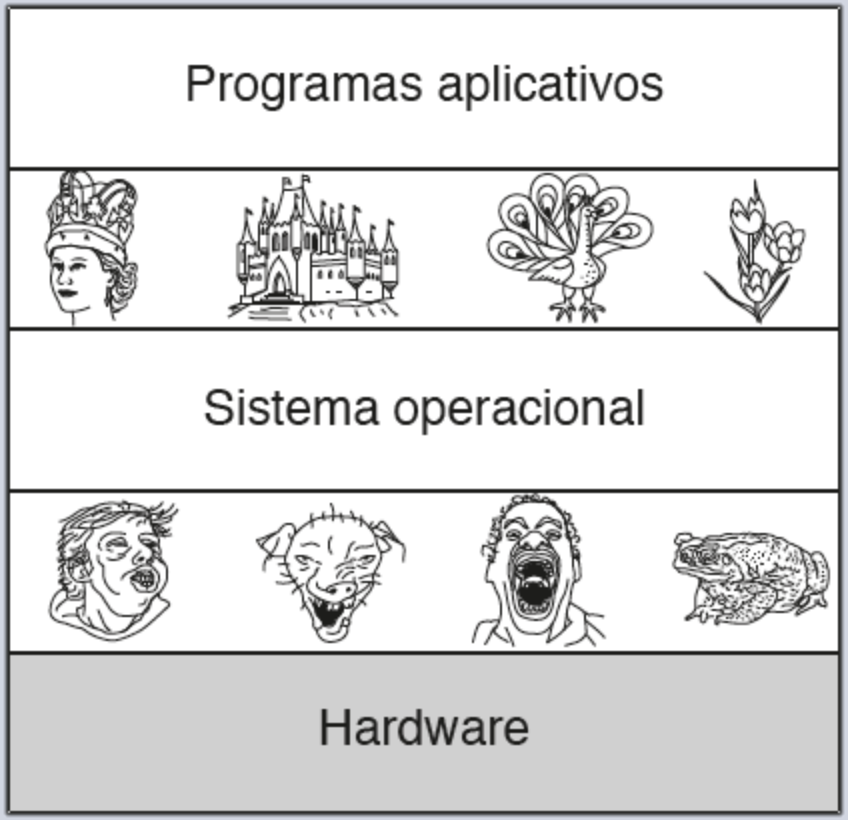
- O resto do software opera em modo usuário, no qual apenas um subconjunto das instruções da máquina está disponível.
- A diferença entre os modos exerce papel crucial na maneira como os sistemas operacionais funcionam.
- Os sistemas operacionais são enormes, complexos e têm vida longa. O código-fonte do coração de um sistema operacional como Linux ou Windows tem cerca de cinco milhões de linhas.

O que é um sistema operacional?

- Os sistemas operacionais realizam duas funções essencialmente não relacionadas: fornecer a programadores de aplicativos (e programas aplicativos, claro) um conjunto de recursos abstratos limpo em vez de recursos confusos de hardware, e gerenciar esses recursos de hardware.

O que é um sistema operacional?

Sistemas operacionais transformam o feio em belo, como mostrado na figura:



O que é um sistema operacional?

- O conceito de um sistema operacional como fundamentalmente fornecendo abstrações para programas aplicativos é uma visão top-down (abstração de cima para baixo). Uma visão alternativa, bottom-up (abstração de baixo para cima), sustenta que o sistema operacional está ali para gerenciar todas as partes de um sistema complexo.

O que é um sistema operacional?

- O gerenciamento de recursos inclui a multiplexação (compartilhamento) de recursos de duas maneiras diferentes: no tempo e no espaço.
- Quando um recurso é multiplexado no tempo, diferentes programas ou usuários se revezam usando-o.
- O outro tipo é a multiplexação de espaço. Em vez de os clientes se revezarem, cada um tem direito a uma parte do recurso.

História dos sistemas operacionais

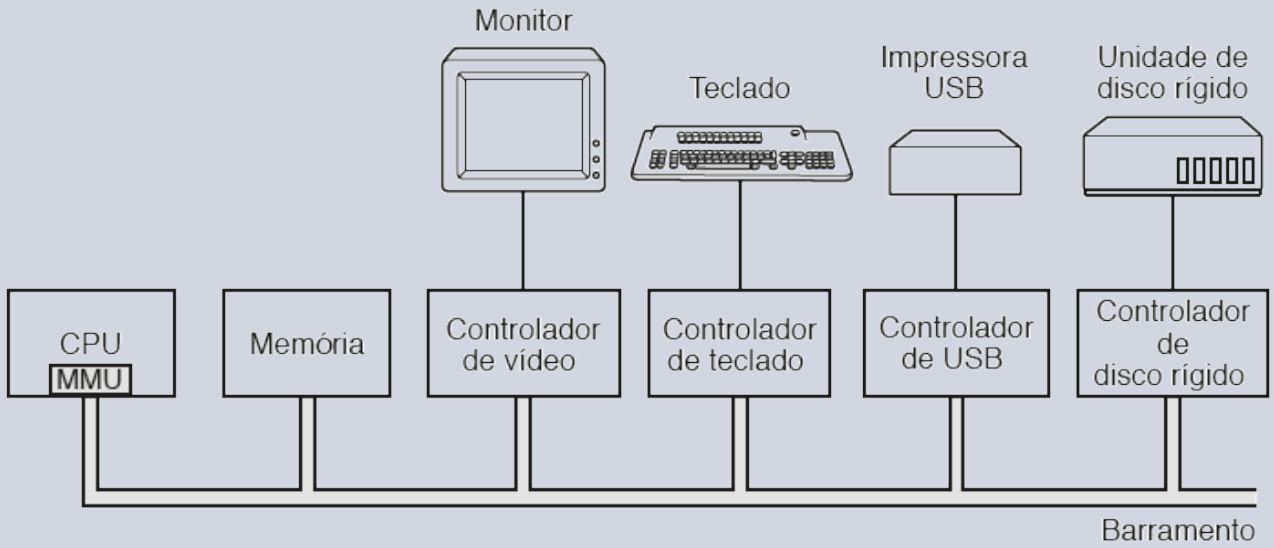
- O primeiro computador verdadeiramente digital foi projetado pelo matemático inglês Charles Babbage (1792–1871). Mas Babbage nunca conseguiu colocá-lo para funcionar para valer porque a máquina era puramente mecânica.
- A solução para esse problema foi dada pela jovem Ada Lovelace, a primeira programadora do mundo. A linguagem de programação Ada® é uma homenagem a ela.

História dos sistemas operacionais

- A primeira geração (1945-1955): válvulas
- A segunda geração (1955-1965): transistores e sistemas em lote (batch)
- A terceira geração (1965-1980): CIs e multiprogramação
- A quarta geração (1980-presente): computadores pessoais
- A quinta geração (1990-presente): computadores móveis

Revisão sobre hardware de computadores

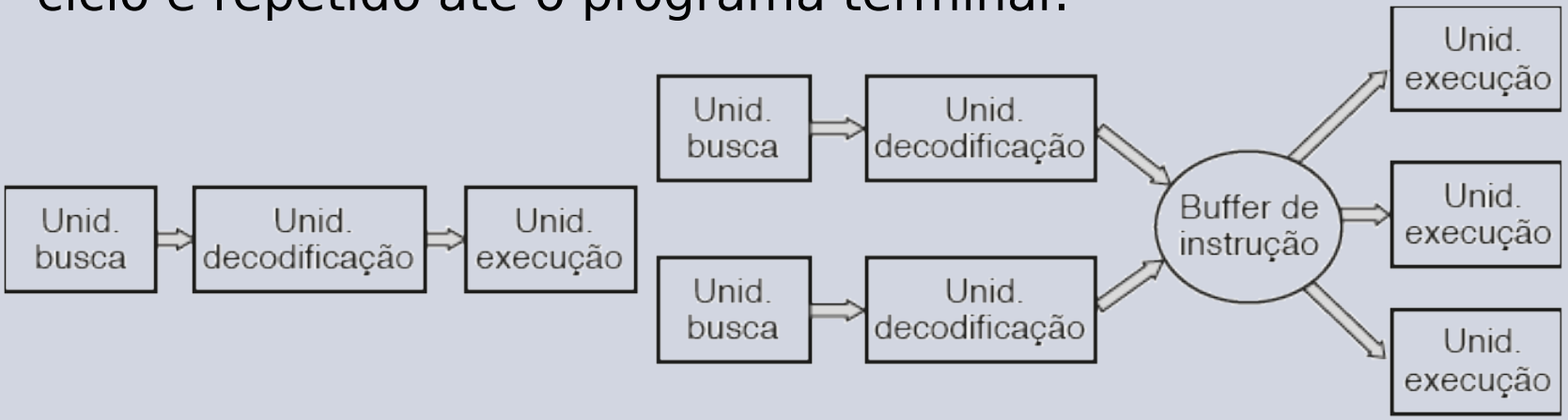
- Um sistema operacional está intimamente ligado ao hardware do computador no qual ele é executado



Alguns dos componentes de um computador pessoal simples.

Revisão sobre hardware de computadores

- O “cérebro” do computador é a CPU. O ciclo básico de toda CPU é buscar a primeira instrução da memória, decodificá-la para determinar o seu tipo e operandos, executá-la, e então buscar, decodificar e executar as instruções subsequentes. O ciclo é repetido até o programa terminar.

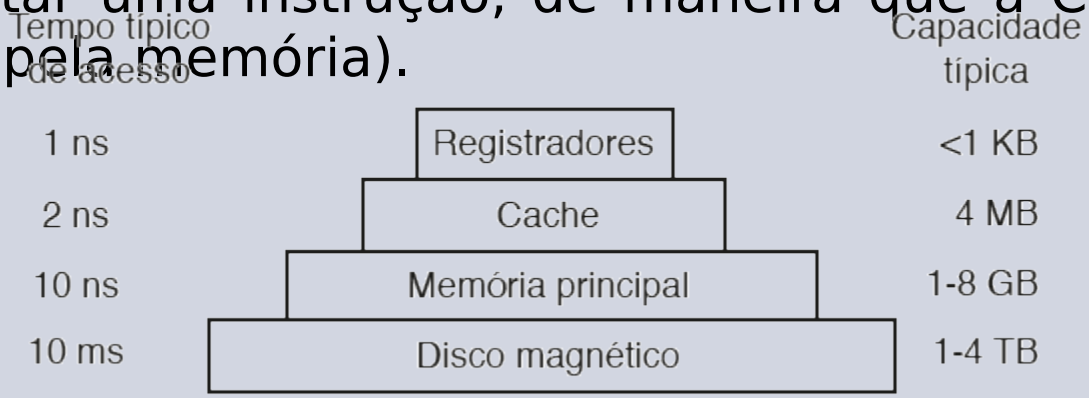


Um pipeline com três estágios. Uma CPU superescalar.

Revisão sobre hardware de computadores

- *Chips multithread e multinúcleo:* o Pentium 4 da Intel introduziu a propriedade chamada multithreading ou hyperthreading (o nome da Intel para ela), ao processador x86 e vários outros chips de CPU também o têm.

- *Memória:* é o segundo principal componente em qualquer computador, o qual deve ser rápido ao extremo (mais rápida do que executar uma instrução, de maneira que a CPU não seja atrasada pela memória).

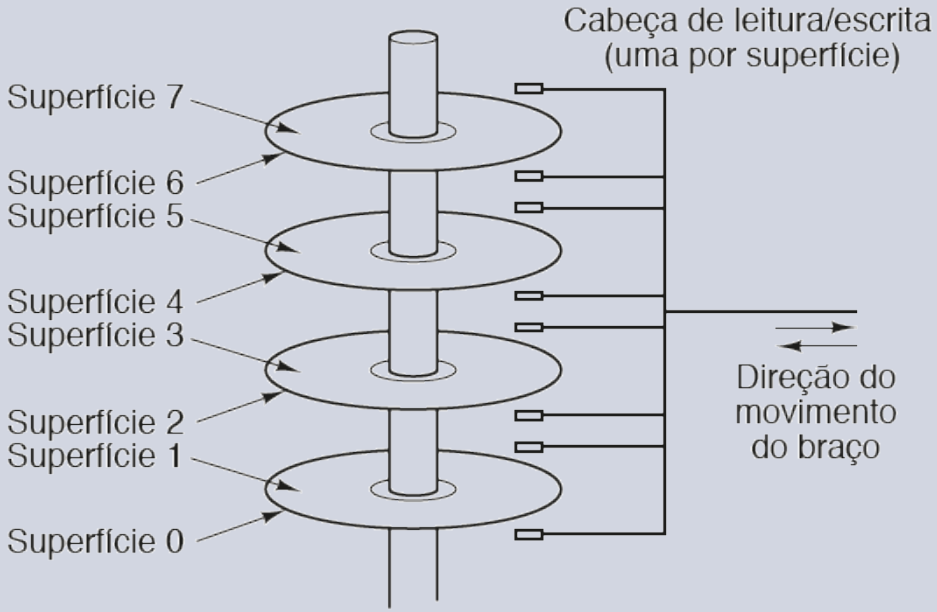


Uma hierarquia de memória típica.

Os números são apenas aproximações.

Revisão sobre hardware de computadores

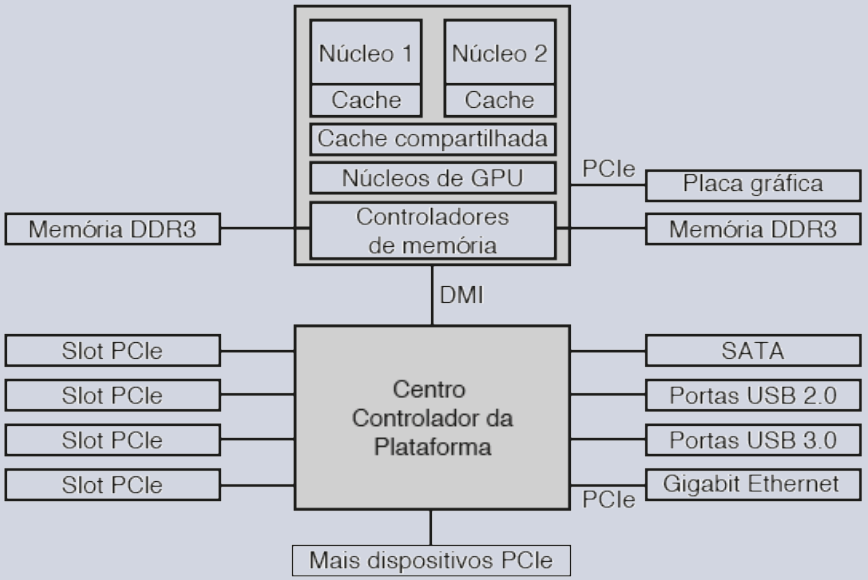
- *Discos:* um disco consiste em um ou mais pratos metálicos que rodam a 5.400, 7.200, 10.800 RPM, ou mais. Um braço mecânico move-se sobre esses pratos a partir da lateral, como o braço de toca-discos de um velho fonógrafo de 33 RPM para tocar discos de



Estrutura de uma unidade de disco.

Revisão sobre hardware de computadores

- *Barramentos:* à medida que os processadores e as memórias foram ficando mais rápidos, a capacidade de um único barramento de lidar com todo o tráfego foi exigida até o limite. Barramentos adicionais foram acrescentados, tanto para dispositivos de E/S mais rápidos quanto para o tráfego CPU para memória.



A estrutura de um sistema x86 grande.

O zoológico dos sistemas operacionais

- Sistemas operacionais de computadores de grande porte
- Sistemas operacionais de servidores
- Sistemas operacionais de multiprocessadores
- Sistemas operacionais de computadores pessoais
- Sistemas operacionais de computadores portáteis
- Sistemas operacionais embarcados
- Sistemas operacionais de nós sensores (senso r-node)
- Sistemas operacionais de tempo real
- Sistemas operacionais de cartões inteligentes (smartcard)

Conceitos de sistemas operacionais

- Processos
- Espaços de endereçamento
- Arquivos
- Entrada/Saída
- Proteção
- O interpretador de comandos (shell)
- A ontogenia recapitula a filogenia

Processo

- Programa em execução que ocorre num espaço de endereçamento na memória (lista de posições).
- Para cada processo há um conjunto de recursos, incluindo registradores (contador de programa e ponteiro de pilha), lista de arquivos abertos e processos relacionados.
- Num sistema multiprogramação poderíamos ter 3 processos ativos: o SO pode executá-los simultâneamente ou parar de executar um e começar a executar o outro.
- Se um processo é suspenso temporariamente ele deve ser reiniciado, no exato estado em que estava.
- Informações do processo são salvas durante a suspensão. Um ponteiro fornece o número do Byte a ser lido em seguida.

Processo

- A tabela de processos guarda informações a respeito de cada processo.
- Chamadas de sistema de gerenciamento de processos lidam com sua criação e término.
- Os processos são estruturados em árvore (processo pai/filho).
- A sincronia de atividades envolve comunicação entre processos. Tal comunicação pode ocorrer de forma remota (processo remoto).
- Quando um processo A é finalizado o SO pode notificar o processo B. Este sinal pode fazer com que o processo B seja suspenso e seus regitradores seriam salvos na pilha de execução.
- Um sinal de software é análogo ao conceito de interrupção em hardware. Um sinal pode ser emitido quando um temporizador expira, por exemplo.

Espaços de endereçamento

- SO simples: um programa por vez.
- SO sofisticado: múltiplos programas compartilham memória. Neste caso o SO protege a memória principal, gerenciando espaços de endereçamento.
- Caso mais simples:
  - espaço de endereçamento do(s) processo(s) < memória principal;
- Caso complexo:
  - espaço de endereçamento do(s) processo(s) > memória principal;
  - Memória virtual: parte do endereçamento na memória e parte no disco (abstração de espaço de endereçamento).

Arquivos

- Uma das suas funções é esconder as peculiaridades dos discos.
- Chamadas de sistemas criam, removem, leêm, escrevem, abrem e fecham arquivos.
- Os diretórios mantêm os arquivos agrupados e as chamadas de sistemas também criam e removem diretórios. Tal modelo dá origem a hierarquia do sistema de arquivos.
- Hierarquia de processo: não é profunda e dura pouco (minutos).
- Hierarquia de arquivo: é profunda e pode durar muito.
- Antes de ser lido, um arquivo precisa ser aberto. Momento em que as permissões são checadas.

Arquivos

- Montagem do sistema de arquivos: quando um sistema de arquivo de uma mídia removível, por exemplo, é agregado a árvore principal.
- O pipe é uma espécie de pseudoarquivo usado para conectar 2 processos ou arquivos.

Entrada/Saída

- Refere-se aos dispositivos físicos para obter entradas e produzir saídas.
  - Ex.: teclados, monitores, impressoras...
- Subsistema de E/S gerencia os dipositivos de E/S:
  - Há softwares E/S independentes de dispositivos;
  - Outros softwares, como os Drivers, são específicos para cada dipositivo.

Proteção

- Arquivos acessíveis somente por usuários autorizados;
- Código de proteção binária de 9 bits (rwx):
  - usuários, membros do grupo e outros;
  - para um diretório o x indica permissão de busca;
  - Traço significa permissão ausente
- Outras proteções: intrusos (humanos e vírus).

Interpretador de comandos (shell)

- SO é o código que executa as chamadas de sistema;
- Programas utilitários e interpretadores de comando não fazem parte do SO;
- O Shell faz uso intensivo do SO: principal interface de usuário.

Conceitos de sistemas operacionais

- *Memórias grandes:* Os primeiros computadores de grande porte tinham uma memória limitada. Um IBM 7090 ou um 7094 completamente carregados, que eram os melhores computadores do final de 1959 até 1964, tinha apenas um pouco mais de 128 KB de memória. Em sua maior parte, eram programados em linguagem de montagem e seu sistema operacional era escrito nessa linguagem para poupar a preciosa memória.

Conceitos de sistemas operacionais

- *Hardware de proteção:* Os primeiros computadores de grande porte inicialmente não tinham hardware de proteção e nenhum suporte para multiprogramação, então sistemas operacionais simples eram executados neles. Esses sistemas lidavam com apenas um programa carregado manualmente por vez. Mais tarde, eles adquiriram o suporte de hardware e sistema operacional para lidar com múltiplos programas ao mesmo tempo, e então capacidades de compartilhamento de tempo completas.

Conceitos de sistemas operacionais

- *Discos:* Os primeiros computadores de grande porte eram em grande parte baseados em fitas magnéticas. Eles liam um programa a partir de uma fita, compilavam-no e escreviam os resultados de volta para outra fita. Não havia discos e nenhum conceito de um sistema de arquivos. Isso começou a mudar quando a IBM introduziu o primeiro disco rígido — o RAMAC (RAndoM ACcess) em 1956.

Conceitos de sistemas operacionais

- *Memória virtual*: A memória virtual proporciona a capacidade de executar programas maiores do que a memória física da máquina, rapidamente movendo pedaços entre a memória RAM e o disco. Ela passou por um desenvolvimento similar, primeiro aparecendo nos computadores de grande porte, então passando para os minis e os micros.

Chamadas de sistema

- Chamadas de sistema para gerenciamento de processos
- Chamadas de sistema para gerenciamento de arquivos
- Chamadas de sistema para gerenciamento de diretórios
- Chamadas de sistema diversas
- A API Win32 do Windows

Chamadas de sistema

- **POSIX – Portable Operating System Interface**
  - X: representa a herança que a interface tem do SO UNIX
  - International Standard 9945-1:1990 → norma definida pela IEEE para manutenção de compatibilidade entre os SO's.
  - Para gerenciar os recursos computacionais, há cerca de 100 chamadas de sistema POSIX divididas em 4 categorias: processos; arquivos; sistema de diretórios e arquivo; e diversas.
  - SO's devem seguir o padrão POSIX.
  - A maioria das rotinas POSIX invoca chamadas de sistema para o modo núcleo.

FIGURA 1.18 Algumas das principais chamadas de sistema POSIX. O código de retorno *s* é –1 se um erro tiver ocorrido. Os códigos de retorno são os seguintes: *pid* é um processo id, *fd* é um descritor de arquivo, *n* é um contador de bytes, *position* é um deslocamento no interior do arquivo e *seconds* é o tempo decorrido. Os parâmetros são explicados no texto.

Gerenciamento de processos

Chamada	Descrição
pid = fork( )	Cria um processo filho idêntico ao pai
pid = waitpid(pid, &statloc, options)	Espera que um processo filho seja concluído
s = execve(name, argv, environp)	Substitui a imagem do núcleo de um processo
exit(status)	Conclui a execução do processo e devolve status

Gerenciamento de arquivos

Chamada	Descrição
fd = open(file, how, ...)	Abre um arquivo para leitura, escrita ou ambos
s = close(fd)	Fecha um arquivo aberto
n = read(fd, buffer, nbytes)	Lê dados a partir de um arquivo em um buffer
n = write(fd, buffer, nbytes)	Escreve dados a partir de um buffer em um arquivo
position = lseek(fd, offset, whence)	Move o ponteiro do arquivo
s = stat(name, &buf)	Obtém informações sobre um arquivo

Gerenciamento do sistema de diretório e arquivo

Chamada	Descrição
s = mkdir(name, mode)	Cria um novo diretório
s = rmdir(name)	Remove um diretório vazio
s = link(name1, name2)	Cria uma nova entrada, name2, apontando para name1
s = unlink(name)	Remove uma entrada de diretório
s = mount(special, name, flag)	Monta um sistema de arquivos
s = umount(special)	Desmonta um sistema de arquivos

Diversas

Chamada	Descrição
s = chdir(dirname)	Altera o diretório de trabalho
s = chmod(name, mode)	Altera os bits de proteção de um arquivo
s = kill(pid, signal)	Envia um sinal para um processo
seconds = time(&seconds)	Obtém o tempo decorrido desde 1º de janeiro de 1970

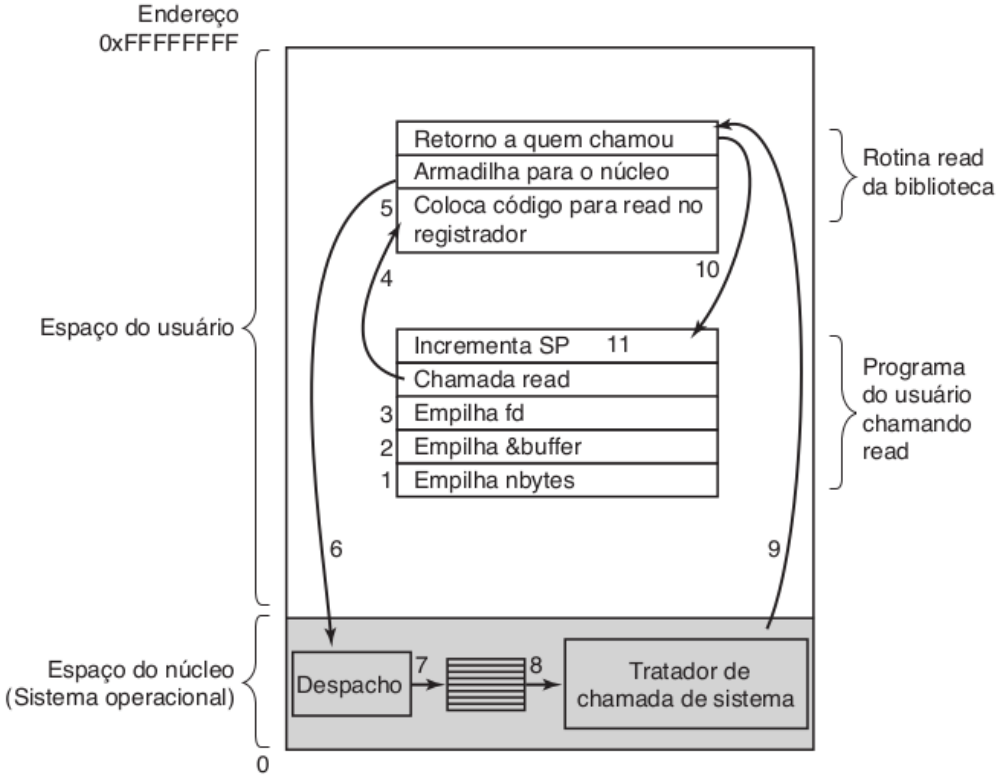
Chamadas de sistema

- Se um processo em modo usuário precisa de um serviço de sistema ele tem de executar uma instrução de armadilha (trap) para transferir o controle para o SO – do modo usuário para o modo núcleo.

- **Rotinas POSIX invocam chamadas de:**
  - Biblioteca (rotina): ocorre em modo usuário. Em geral, apresenta melhor desempenho.
  - Sistema (tipo especial de rotina): ocorre em modo núcleo.

- Os programas devem sempre conferir os resultados de uma chamada de sistema para ver se um erro ocorreu.

FIGURA 1.17 Os 11 passos na realização da chamada de sistema read (fd, buffer, nbytes).



Chamadas de sistema

- **Instrução trap:**
  - direciona chamada de sistema do modo usuário para o núcleo;
  - não pode saltar para um endereço arbitrário;
  - Dependendo da arquitetura do SO/Hardware:
    - salta para um único local fixo; ou
    - salta para um campo de 8 bits fornecendo um índice para uma tabela de ponteiros na memória contendo endereços para novos saltos.
  - O número da chamada de sistema é examinado e despachado para o tratador correto da chamada de sistema indexado pela tabela de ponteiros.

Chamadas de sistema

- O controle pode ser retornado para a rotina de biblioteca no espaço do usuário;
- Para terminar a tarefa, o programa do usuário tem de limpar a pilha;
- O programa está livre agora para fazer o que quiser.

Chamadas de sistema

- **Para Gerenciamento de Processos**
  - Em POSIX, cria-se um processo com a chamada fork.
  - Trata-se de uma cópia do processo pai, incluindo descritores de arquivos e resgistradores.
  - Há 3 segmentos num processo: código, dados e pilha de execução.
  - No momento do fork os processos (pai e filho) são idênticos. Após o fork ambos seguem caminho distintos, podendo haver mudanças no segmento de dados (variáveis). No entanto, o código não é passível de alteração.
  - A chamada fork retorna:
    - valor zero no processo filho;
    - PID do processo filho no processo pai.

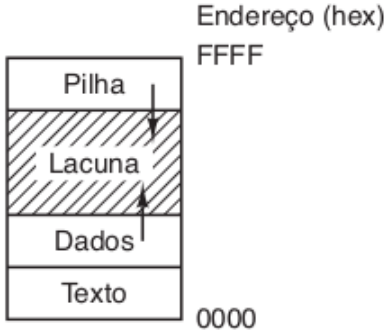
FIGURA 1.19 Um interpretador de comandos simplificado. Neste livro, presume-se que *TRUE* seja definido como 1.*

```
#define TRUE 1

while (TRUE) {                                  /* repita para sempre */
        type_prompt( );                         /* mostra prompt na tela */
        read_command(command, parameters);      /* le entrada do terminal */

        if (fork( ) != 0) {                     /* cria processo filho */
             /* Codigo do processo pai. */
             waitpid(–1, &status, 0);           /* aguarda o processo filho acabar */
        } else {
             /* Codigo do processo filho. */
             execve(command, parameters, 0);    /* executa o comando */
        }
}
```

FIGURA 1.20 Os processos têm três segmentos: texto, dados e pilha.



Chamadas de sistema

- **Para Gerenciamento de Processos**
  - Quando um comando é digitado, o shell cria um novo processo. Esse processo filho tem de executar o comando de usuário. Ele o faz usando a chamada de sistema execve.
  - execve possui três parâmetros: nome do arquivo a ser localizado e executado (cp); ponteiro para o arranjo de argumentos; e ponteiro para o arranjo de ambiente.
    - cp fd1 fd2

Chamadas de sistema

- **Para Gerenciamento de Processos**
  - cp contém a declaração main(argc, argv, envp), onde:
    - argc: contagem do número de itens na linha de comando, incluindo o nome do programa.
    - argv: ponteiro para um arranjo, sendo argv[0] referindo-se aos caracteres “cp”; argv[1] aponta para “fd1”; e argv[2] aponta para “fd2”.
    - envp: ponteiro para o ambiente, um arranjo de cadeias de caracteres contendo atribuições da forma nome = valor usadas para passar informações como tipo de terminal e o nome do diretório home.
      - Se nenhum ambiente é passado para o processo filho, então envp é zero.

Chamadas de sistema

- **Para Gerenciamento de Arquivos**
  - A chamada de sistema open especifica:
    - caminho absoluto ou relativo do arquivo a ser aberto, incluindo seu nome; e
    - código de aberto para leitura (O_RDONLY); escrita (O_WRONLY); ou ambos (O_RDWR).
  - Para criar um novo arquivo executa-se O_CREAT: o descritor de arquivo pode ser usado para leitura ou escrita.
  - O arquivo pode ser fechado por close, que torna o descritor disponível para ser reutilizado em um open subsequente.

Chamadas de sistema

- **Para Gerenciamento de Arquivos**
  - Associado a cada arquivo há um ponteiro que indica a posição atual do arquivo.
  - Lendo/escrevendo ele aponta para o próximo Byte a ser lido/escrito.
  - A chamada lseek muda o valor do ponteiro de posição. Assim, chamadas subsequentes de leitura/escrita podem começar em qualquer parte no arquivo.
  - position = lseek(fd, offset, whence), onde:
    - fd: descritor de arquivo;
    - offset: posição do arquivo; e
    - whence: se a posição do arquivo é relativa ao começo, à posição atual ou ao fim do arquivo.
    - Retorno: posição absoluta no arquivo (em bytes) após mudar o ponteiro.

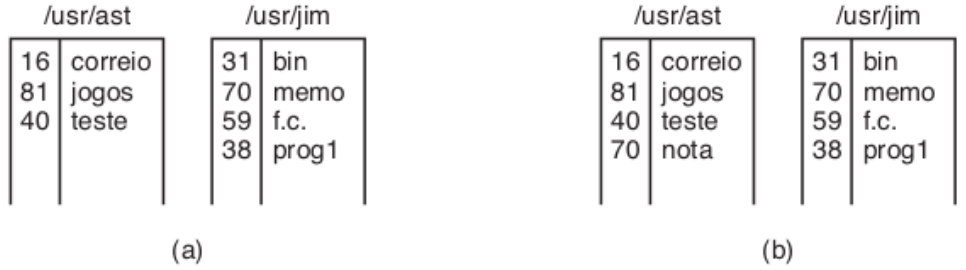
Chamadas de sistema

- **Para Gerenciamento de Arquivos**
  - Exemplo do funcionamento de um descritor de arquivos:
    - https://pt.stackoverflow.com/questions/339073/o-que-s%C3%A3o-descritores-de-arquivos-e-diret%C3%B3rios
    - https://www.geeksforgeeks.org/python-os-open-method/

Chamadas de sistema

- **Para Gerenciamento de Diretórios**
  - ✓ mkdir e rmdir, criam e removem diretórios vazios
  - ✓ link permite o compartilhamento de arquivos (diferente de cópia)
    - ➢ Exemplo: https://www.tutorialspoint.com/python/os_link.htm#

FIGURA 1.21 (a) Dois diretórios antes da ligação de */usr/jim/memo* ao diretório *ast*. (b) Os mesmos diretórios depois dessa ligação.



Chamadas de sistema

- **Para Gerenciamento de Diretórios**
  - ✓ mount permite que dois sistemas de arquivos sejam fundidos em um.
  - ✓ A chamada mount torna possível integrar mídias removíveis (pendrives, por exemplo) em uma única hierarquia de arquivos integrada.
  - ✓ Quando um sistema de arquivos não é mais necessário, ele pode ser desmontado com a chamada de sistema umount.

FIGURA 1.22 (a) O sistema de arquivos antes da montagem. (b) O sistema de arquivos após a montagem.

Chamadas de sistema

- **Aplicações**
  - UNIX e derivados: em geral, para cada chamada de sistema (ex.: read) há uma rotina de biblioteca (ex.: read) para invocar chamada de sistema.
  - Windows: chamadas de biblioteca e as chamadas de sistema reais são altamente desacopladas.
    - A Microsoft definiu um conjunto de rotinas chamadas de API (Application Programming Interface) Win32 para acessar os serviços do SO.
    - A Win32 proporciona compatibilidade entre as versões do Windows.
    - Há milhares de chamadas na API Win32. Dentre elas, além das chamadas de sistema (que levam o controle para o núcleo), há também chamadas de biblioteca que são executadas apenas no espaço do usuário.

Estrutura de sistemas operacionais

- Sistemas monolíticos
- Sistemas de camadas
- Micronúcleos
- O modelo cliente-servidor
- Máquinas virtuais
- Exonúcleos

Estrutura de sistemas operacionais

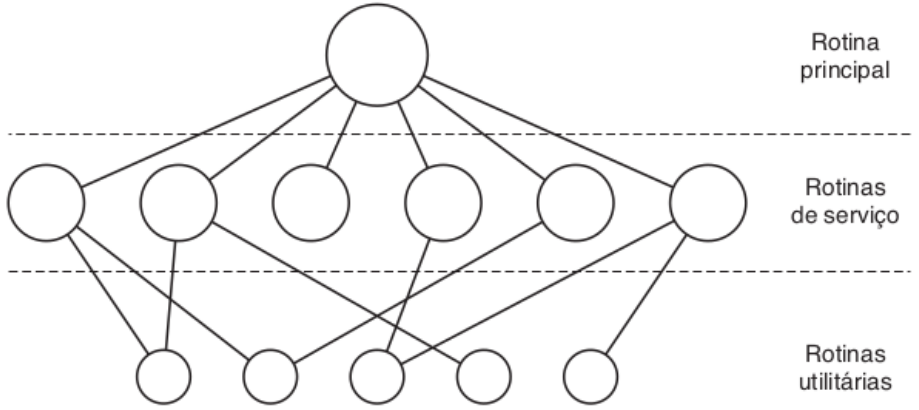
- **Sistemas monolíticos**
  - Estrutura mais comum;
  - Executado como um único programa em modo núcleo;
  - Coleção de rotinas, ligadas a um único grande programa binário executável;
  - Cada procedimento no sistema é livre para chamar qualquer outro;
  - Sistema difícil de lidar e compreender;
  - Uma quebra em qualquer uma dessas rotinas derrubará todo o SO.

Estrutura de sistemas operacionais

- **Sistemas monolíticos**
  - ✓ Rotinas muito acopladas (oposta a uma estrutura modularizada e coesa);
  - ✓ Modelo Lógico:
    1. Chamadas de sistema são requisitadas, utilizando parâmetros de entrada empilhados num local bem definido;
    2. Executa-se a instrução de desvio de controle (trap) que chaveia a máquina do modo usuário para o modo núcleo, transferindo o controle para o SO;
    3. Parâmetros são buscados e determina-se qual chamada de sistema será executada;
    4. O SO indexa uma tabela que contém na linha k um ponteiro para a rotina que executa a chamada de sistema k (Passo 7: Figura 1.17).

- **Sistemas monolíticos**
  - ✓ Estrutura:
    1. Programa principal: invoca as rotinas de serviço requisitadas;
    2. Rotinas de serviço: executam as chamadas de sistema;
    3. Rotinas utilitárias: auxiliam as rotinas de serviço (ex.: buscam dados de programas dos usuários).

FIGURA 1.24 Um modelo de estruturação simples para um sistema monolítico.



Estrutura de sistemas operacionais

- **Sistemas de camadas**
  - Estrutura Implementada pela primeira vez em 1968.
  - Recebeu o nome “Sistema THE”, desenvolvido por Dijkstra e seus alunos.

FIGURA 1.25 Estrutura do sistema operacional THE.

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

Estrutura de sistemas operacionais

- **Sistemas de camadas**

Camadas em destaque	Finalidade
3	Armazenava temporariamente os fluxos de informação que iam ou vinham desses dispositivos de E/S.
2	Encarregava-se da comunicação entre cada processo.
1	Alocação de espaço para processos na memória principal e em um tambor magnético usado para armazenar partes de processos (páginas). O software certificava-se de que as páginas fossem trazidas à memória no momento em que eram necessárias e removidas quando não eram mais.
0	Chaveamento de processos em caso de interrupções ou expiração dos temporizadores. Fornecia a multiprogramação básica da CPU.

Estrutura de sistemas operacionais

- **Micronúcleos**
  - Qual deve ser o limite entre o usuário e o núcleo?
  - Erros no código do núcleo podem derrubar o sistema instantaneamente.
  - Isso justificaria a colocação do mínimo de funcionalidades possíveis no modo núcleo. Assim, processos de usuário poderiam ser configurados para ter menos poder, de maneira que um erro possa não ser fatal.
  - A ideia desta estrutura é atingir uma alta confiabilidade, dividindo o SO em módulos pequenos e bem definidos, apenas um dos quais — o micronúcleo — é executado em modo núcleo e o resto é executado como processos de usuário comuns, relativamente sem poder.

Estrutura de sistemas operacionais

- **Micronúcleos**
  - Ao se executar cada driver de dispositivo e sistema de arquivos como um processo de usuário em separado, um erro em um deles pode derrubar esse componente, mas não consegue derrubar o sistema inteiro.
  - Num sistema monolítico, com todos os *drivers* no núcleo, um *driver* de áudio com problemas pode facilmente referenciar um endereço de memória inválido e provocar uma parada instantânea no sistema.
  - Exemplos de aplicação da estrutura de micronúcleos:
    - OS X, baseado no micronúcleo Mach
    - MINIX 3, baseado em POSIX
    - Aplicações envolvendo tempo real, industriais, de aviônica e militares.

Estrutura de sistemas operacionais

- **Micronúcleos**
  - No MINIX 3, por exemplo:
    - Chamadas de núcleo são controladas processo a processo, assim como a capacidade de enviar mensagens para outros processos.
    - Tais restrições fazem com que cada driver e servidor tenham o poder de fazer exclusivamente o seu trabalho. Isso limita muito o dano que um componente com erro pode provocar.
    - Um algoritmo de escalonamento de processos pode designar prioridade numérica para cada processo (política) para que o núcleo os execute (mecanismo).
    - Política e mecanismo podem ser desacoplados. Assim, o núcleo torna-se menor.

Estrutura de sistemas operacionais

- **Modelo cliente-servidor**
  - Variação da estrutura de micronúcleo, onde há 2 classes de processos:
    - Servidores: prestam algum serviço; e
    - Clientes: usam esses serviços.
  - Comunicação entre clientes e servidores realizada pela troca de mensagens.
  - Processo cliente envia requisição e a envia ao processo servidor.
  - Processo servidor realiza o serviço e envia a resposta ao processo cliente.
  - Abstração usada tanto para uma única máquina quanto para uma rede.

Estrutura de sistemas operacionais

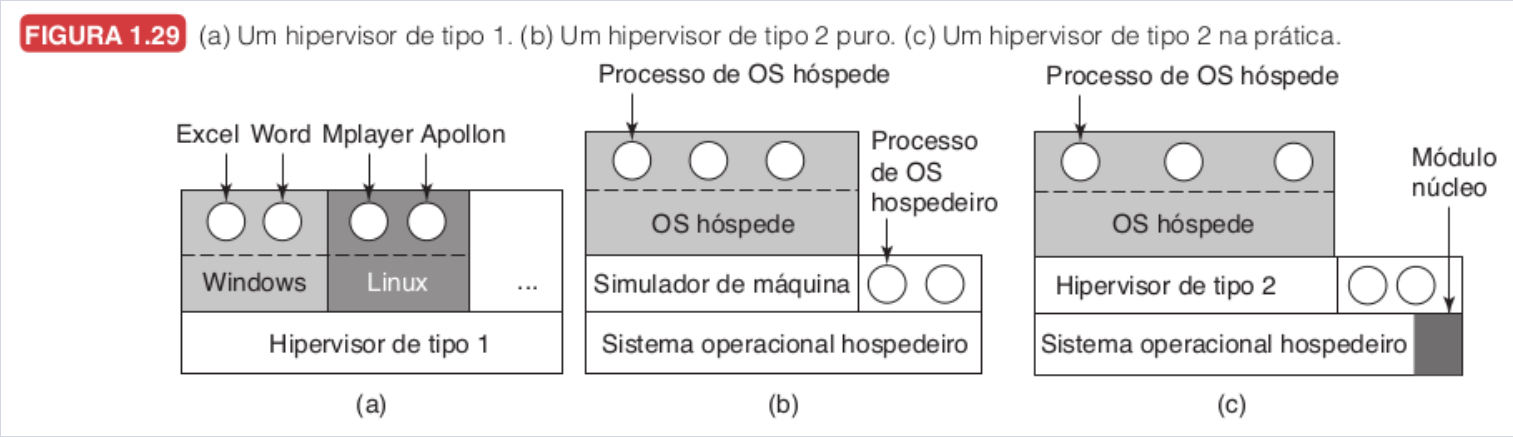
- **Máquinas virtuais**
  - Empresas executavam seus próprios servidores (e-mail, web, FTP, etc) em computadores físicos.
  - No passado, clientes da web escolhiam hospedagem:
    - compartilhada: conta de acesso a um servidor compartilhado, sem nenhum controle sobre o software; ou
    - dedicada: conta de acesso a um servidor exclusivo (flexível, porém caro).
  - Atualmente, alugam-se máquinas virtuais: uma única máquina física pode executar muitas virtuais (redução de custo).

Estrutura de sistemas operacionais

- **Máquinas virtuais**
  - Usuários finais poderiam querer executar 2 sistemas operacionais de forma simultânea (dual boot).
  - Para instalar um software de monitor de máquina virtual (hypervisor) num computador, a CPU precisa ser virtualizável.
  - Ex. de hypervisors: KVM (Linux); VirtualBox (Oracle); e Hyper-V (Microsoft).

Estrutura de sistemas operacionais

- **Máquinas virtuais**
  - Hypervisors:
    - Tipo 1: ausência de SO hospedeiro;
    - Tipo 2: presença de SO hospedeiro.



Estrutura de sistemas operacionais

- **Exonúcleos**
  - Ao invés de clonar a máquina real, como é feito com as máquinas virtuais, outra estratégia é dividi-la, concedendo a cada usuário um subconjunto dos recursos. Desse modo, uma máquina virtual pode obter os blocos de disco.
  - Na camada de baixo, executando em modo núcleo, há um programa chamado exonúcleo.
  - Tarefa do exonúcleo: alocar recursos às máquinas virtuais, assegurando que nenhuma máquina use os recursos de outra.

Estrutura de sistemas operacionais

- **Exonúcleos**
  - Vantagem: poupar uma camada de mapeamento; e separar a multiprogramação (no exonúcleo) do código do sistema operacional do usuário
  - O hypervisor tem de manter tabelas para remapear os endereços de discos (e todos os outros recursos). Com o exonúcleo, esse remapeamento não é necessário.

O mundo de acordo com a linguagem C

- Java, Python e C são todas linguagens imperativas com tipos de dados, variáveis e comandos de controle, por exemplo. Os tipos de dados primitivos em C são inteiros (incluindo curtos e longos), caracteres e números de ponto flutuante. Os tipos de dados compostos em C são similares àqueles em Java, incluindo os comandos if, switch, for e while. Funções e parâmetros são mais ou menos os mesmos em ambas as linguagens.

Pesquisa em sistemas operacionais

- Virtualmente todos os pesquisadores de sistemas operacionais sabem que os sistemas operacionais atuais são enormes, inflexíveis, inconfiáveis, inseguros e carregados de erros, uns mais que os outros (os nomes não são citados aqui para proteger os culpados). Consequentemente, há muita pesquisa sobre como construir sistemas operacionais melhores.