Conceitos de sistemas operacionais

- Processos
- Espaços de endereçamento
- Arquivos
- Entrada/Saída
- Proteção
- O interpretador de comandos (shell)

Processos

Definição:

Programa em execução ao qual estão associados:

– **Espaço de endereçamento**: lista de posições na memória associadas ao processo (onde este pode ler e escrever)
Composto do programa executável, das variáveis e da pilha

– **Recursos**:

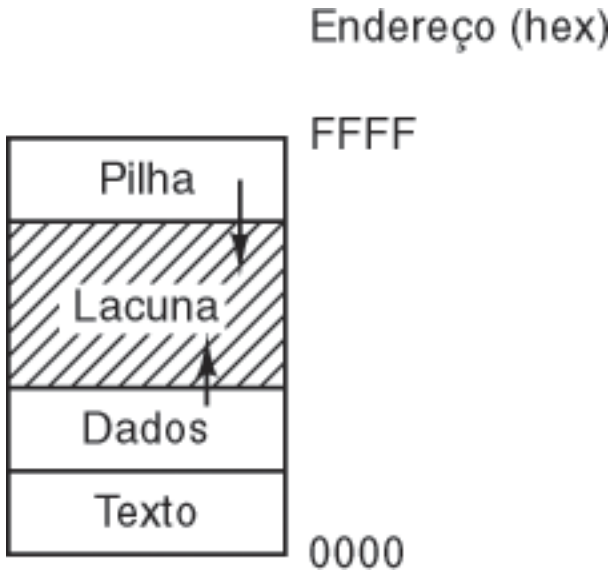
-- registradores de uso geral;

-- PC – contador de programa;

-- SP – endereço de retorno (caso ocorra um desvio)

– **Lista de arquivos abertos, processos relacionados e todas as informações necessárias para a sua execução.**

Processos – espaço de endereçamento



O espaço de endereçamento têm três segmentos: texto, dados e pilha

Processos

Tabela de processos:

Armazena informações sobre o estado de cada processo:

-- se está ou não em execução;

-- contador de programa;

-- ponteiro de pilha;

-- conteúdo de seus registradores;

-- ponteiros para arquivos abertos;

-- alocação da memória;

-- tipo de escalonamento

..... *e todas as informações que devem ser salvas sobre o processo quando este passar do estado de* ***em execução*** *para o estado de* ***bloqueado****.*

Processos

Multiprogramação:

sistema com múltiplas tarefas carregadas na memória que evita a ociosidade da CPU, mantendo-a 100% ocupada (mesmo qdo existe E/S) ⇒ exige proteção

Ex: **vários programas aplicativos abertos:**

Navegador Web, e-mail, editor de texto, etc

⇒ SO controla a execução de cada um compartilhando o tempo de CPU

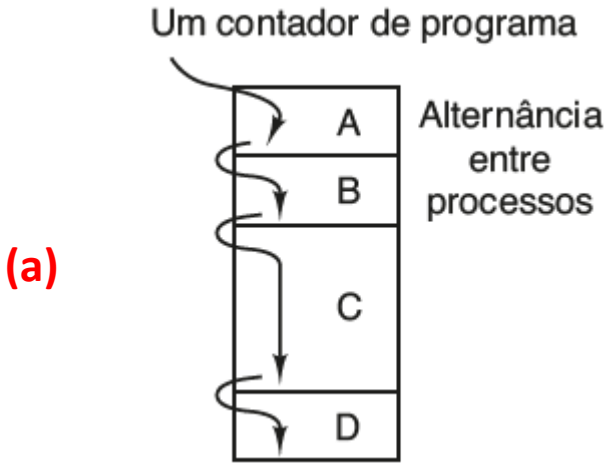
Obs: ***Quando um processo é suspenso temporariamente, todas as informações referentes a ele deverão ser salvas para que ele volte a ser executado posteriormente*** ***(salvar o contexto)***

O modelo de processo

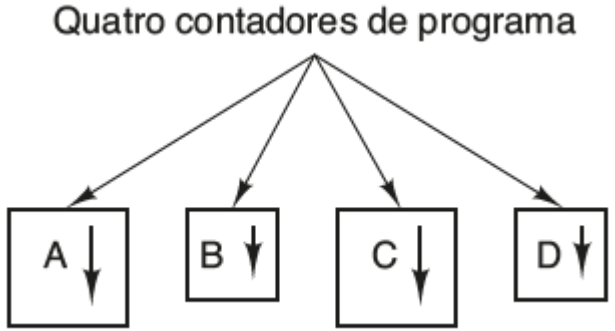
- **Figura (a):** computador multiprogramado com quatro programas na memória.

- **Figura (b):** quatro processos, cada um com seu próprio fluxo de controle e sendo executado independente dos outros.

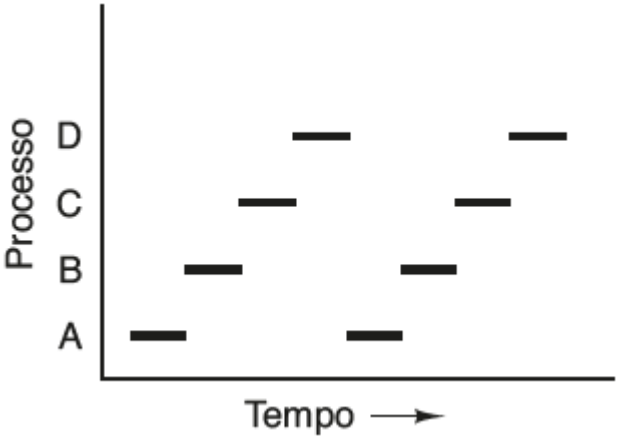
- **Figura (c):** todos os processos tiveram progresso, mas a qualquer dado instante apenas um está sendo de fato executado.



(a)



(b)



(c)

Processos

Chamada de sistema

Forma de um programa do usuário obter um serviço do sistema operacional, através de uma instrução ***TRAP****, que* ***chaveia do modo usuário para o modo núcleo.*** *Após a obtenção do serviço o SO retorna o controle para o programa do usuário, na instrução seguinte à que estava sendo executada quando ocorreu a* ***chamada de sistema***

Processos

Obs: 1. Principais ***chamadas de sistema*** de gerenciamento de processos: criação e término de processos

2. ***Interpretador de comandos (shell)****:* processo que lê um comando de um terminal, cria um novo processo (filho), que se autofinaliza após sua execução

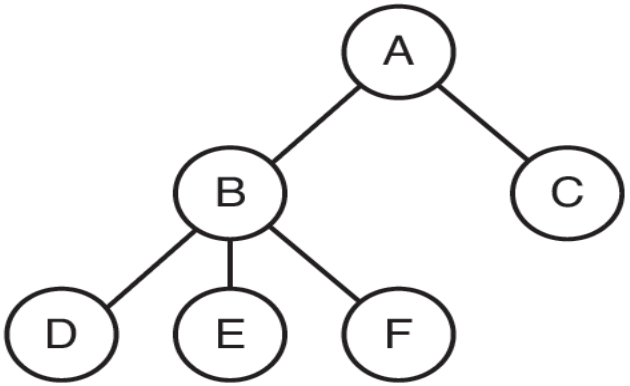


Figura 1.13 Uma árvore de processo. O processo *A* criou dois processos filhos, *B* e *C*. O processo *B* criou três processos filhos, *D*, *E* e *F*.

Comunicação entre Processos

Necessária durante a cooperação entre os processos para realizar uma tarefa e *sincronizar* suas atividades

Outras ***chamadas de sistema*** importantes:

- ✓ requisitar e liberar memória;
- ✓ esperar término de processo filho;
- ✓ temporização (falta de ack, quando existe comunicação entre processos em uma rede) ⇒ gera sinal de alarme

  ⇒ ***tratamento do alarme***

UID – identificação fornecida ao usuário para o qual foi dada permissão de utilizar o sistema;

GID – identificação do grupo

UID especial: superusuário

Espaço de endereçamento

- ***Mecanismo de proteção***, controlado pelo SO. Permite que múltiplos programas estejam carregados na memória ao mesmo tempo.
- ***Gerenciamento do espaço de endereçamento*** de processos:
  - Cada processo: conjunto de endereços de tamanho 0 até uma quantidade máxima
  - Caso mais simples: qtde máxima < tam. da memória principal
  - Computadores com endereços de 32 bits ou 64 bits
    $\Rightarrow 2^{32}$ ou 2^{64} endereços, respectivamente

E se espaço endereçamento for maior que a memória física?

$\Rightarrow$ **memória virtual:**

técnica na qual o SO mantém parte do espaço de endereçamento no disco (o espaço de endereçamento é desacoplado da memória física)

Arquivos

- SO oculta peculiaridades dos discos e de outros dispositivos de E/S
- Fornece **modelo de arquivos** claro e fácil de usar
- Necessárias ***chamadas ao sistema*** para:

  Criação, remoção, abertura, fechamento, leitura e escrita de/em arquivos

Diretório: arquivo especial que agrupa outros arquivos

- Outras chamadas de sistema são necessárias
  - ***Criação e remoção de diretórios***
  - ***Colocar e remover um arquivo em/de um diretório***
- Conteúdo (entradas) de diretórios:
  - **arquivos comuns**
  - **outros diretórios**

  ⇒ ***hierarquia: Sistema de arquivos***

Arquivos

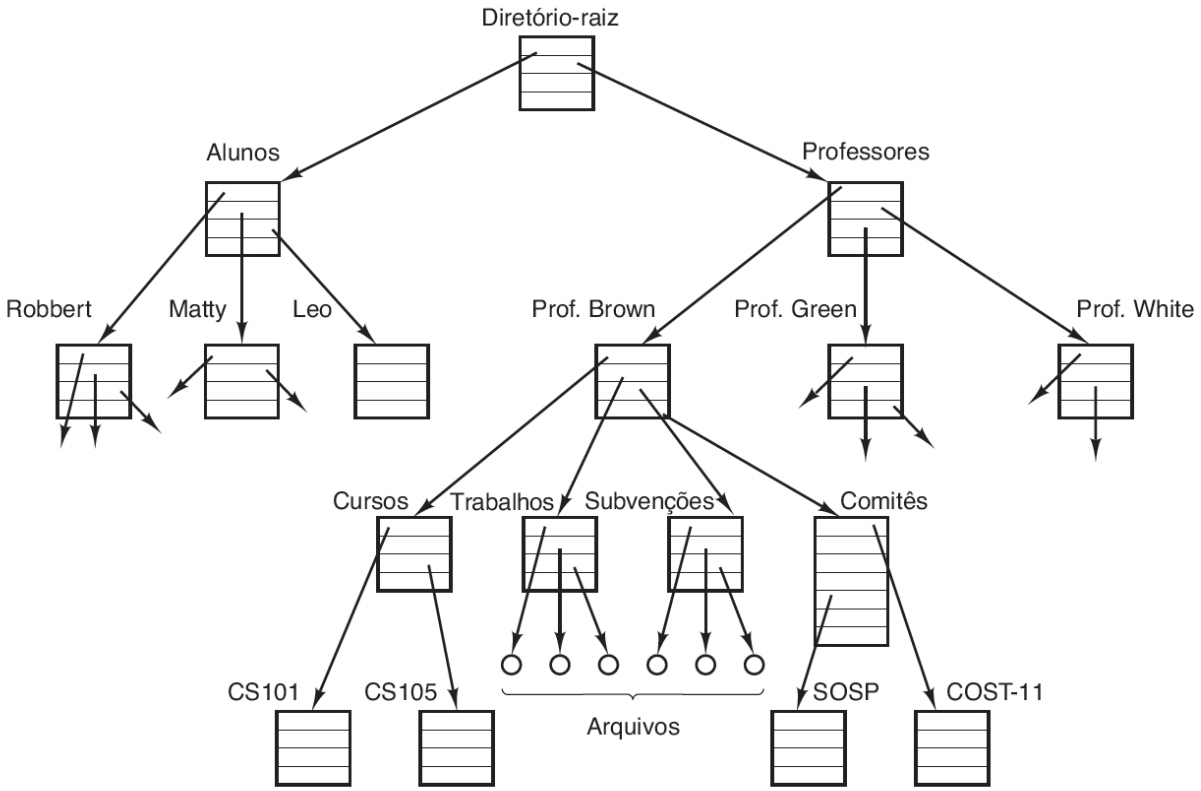


Figura 1.14 Sistema de arquivos para um departamento universitário.

Arquivos

Hierarquia de processos vs hieraquia de arquivos

Processos:

- comumente máximo 3 níveis;
- pouco tempo de vida
- só um processo pai tem o controle

Arquivos:

- pode ser maior que 5 níveis
- pode existir por anos
- vários usuários podem ter acesso

- Noções de caminho absoluto, relativo, diretório raiz, diretório de trabalho
- Um processo está associado a um diretório de trabalho atual (necessária um chamada ao sistema para alteração)

Arquivos

Descritor de arquivo:

valor retornado pela chamada ao sistema no momento da abertura do arquivo (permissões devem ser verificadas). É um ponteiro usado em operações subsequentes.

Montagem de um sistema de arquivos:

permite a agregação de um sistema de arquivos de um DVD ou CD-ROM, por exemplo, ao sistema de arquivos do disco rígido (árvore principal)

- Chamada ao sistema: ***mount***
- *Ex:* ***mount /b*** "monta" (integra) o sistema de arquivos do CD-ROM no diretório /b (que deve estar vazio, caso contrário os arquivos dentro de b ficarão inacessíveis enquanto o CD-ROM estiver montado)

Obs:

dispositivos USB e múltiplos discos rígidos podem também ser montados numa única árvore

Arquivos

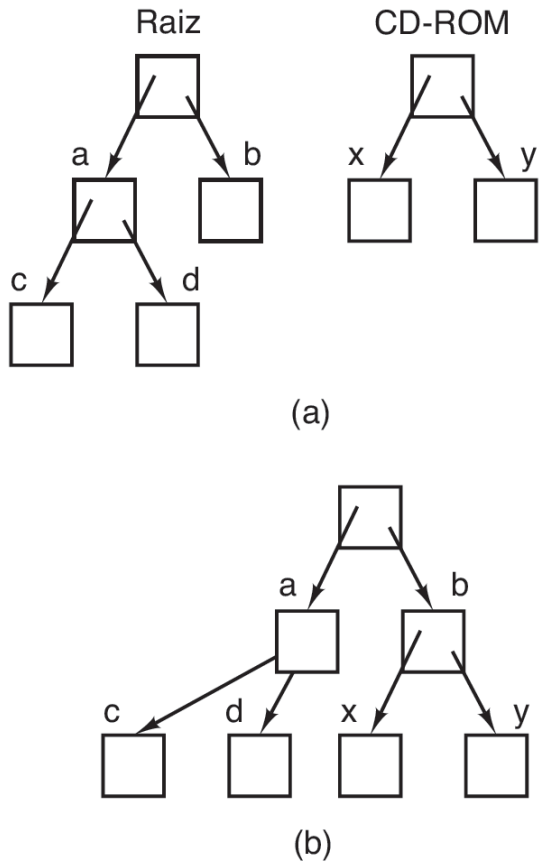


Figura 1.15 (a) Antes da montagem, os arquivos no CD-ROM não estão acessíveis. (b) Após a montagem, tornam-se parte da hierarquia de arquivos.

Arquivos

Arquivos especiais:

permitem que dispositivos de E/S sejam considerados como arquivos

Dois tipos:

- ***Arquivos especiais de blocos:*** modelam dispositivos que formam uma coleção de blocos. Ex: ***discos***

- ***Arquivos especiais de caracteres:*** modelam dispositivos que recebem e enviam caracteres serialmente. Ex: ***impressoras e modems***

Obs: São mantidos no diretório */dev.*
Ex: ***/dev/lp*** pode ser a impressora

Arquivos

Pipe (|):

arquivo especial usado para a comunicação de dois processos

Processo A escreve no ***Pipe*** como se fosse um arquivo de saída

Processo B lê do ***Pipe*** como se fosse um arquivo de entrada

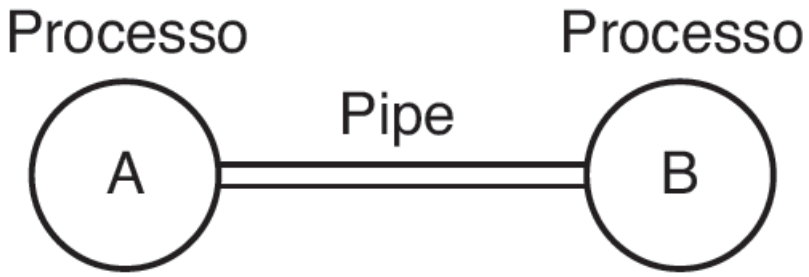


Figura 1.16 Dois processos conectados por um pipe.

E/S e Segurança

SO deve gerenciar seus dispositivos de E/S:

- Alguns softwares de E/S são independentes de dispositivo
- Outros, como os ***drivers*** são específicos de cada dispositivo

SO deve gerenciar a segurança do sistema

Ex.: alguns arquivos devem ser acessíveis somente a usuários autorizados (superusuários)

No Linux: código de 9 bits para cada arquivo:

rwx|r-x|r-x

u g o

O interpretador de comandos - shell

- Interface principal entre o usuário e o SO (exceto se existe uso de GUI, como Gnome ou KDE)
- Não é parte do SO, mas faz uso intensivo dele
- Exemplos: *sh*, *csh*, *ksh*, *tcsh*, *bash*
- O ***shell*** é iniciado na conexão do usuário
- Quando um comando é digitado, um processo filho é criado para executar o programa com aquele nome
- O ***shell*** tem o terminal como entrada e saída padrão. Pode-se alterá-las.

Ex. de alteração de E/S padrão:

sort < arq1 > arq2

Ex. de saída de um programa como entrada de outro

Cat file1 file2 | sort > /dev/lp

Chamadas de sistema

- Realizam a interface entre os comandos do usuário e o sistema operacional
- São semelhantes às chamadas de uma sub-rotina
- Os comandos são rotinas de bibliotecas, que fazem a chamada de sistema
- Chamadas de sistema fazem entrar no modo núcleo (controle do SO) ao contrário das sub-rotinas comuns
- Serão utilizadas, como exemplo, as chamadas do sistema UNIX (Linux)

Tipos de 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

Chamada de sistema READ

Possui 3 parâmetros: arquivo a ser lido (fd), ponteiro para um buffer (onde colocar os dados lidos), nº de bytes que devem ser lidos

Ex: chamada a partir de uma biblioteca em C

contador = read(fd, buffer, nbytes)

Retorna o nº de bytes realmente lidos (normalmente = nbytes)

Passos:

- 1,2,3. Armazena parâmetros na pilha
- 4. Chama rotina
- 5. Coloca nº da chamada de sistema em um registrador
- 6. passa para o modo núcleo (executa instrução TRAP), que contém um índice para uma tabela que indica o tipo de chamada (contém o endereço da rotina a ser chamada)

Obs: a partir daqui, execução do código do SO

Chamada de sistema READ

Passos (contin.):

- 7. SO verifica o nº de chamada do sistema e desvia (despacha) para a rotina correta
- 8. Executa rotina de tratamento
- 9. Retorna o controle para a rotina de biblioteca no espaço do usuário
- 10. Retorna controle ao programa do usuário
- 11. Programa do usuário limpa a pilha

Obs:

A chamada de sistema (passo 8) pode bloquear quem a chamou. Por exemplo, se for uma instrução de leitura do teclado e ninguém tiver digitado nada. Somente após a digitação os passos 9, 10 e 11 seriam executados.

Chamadas de sistema READ

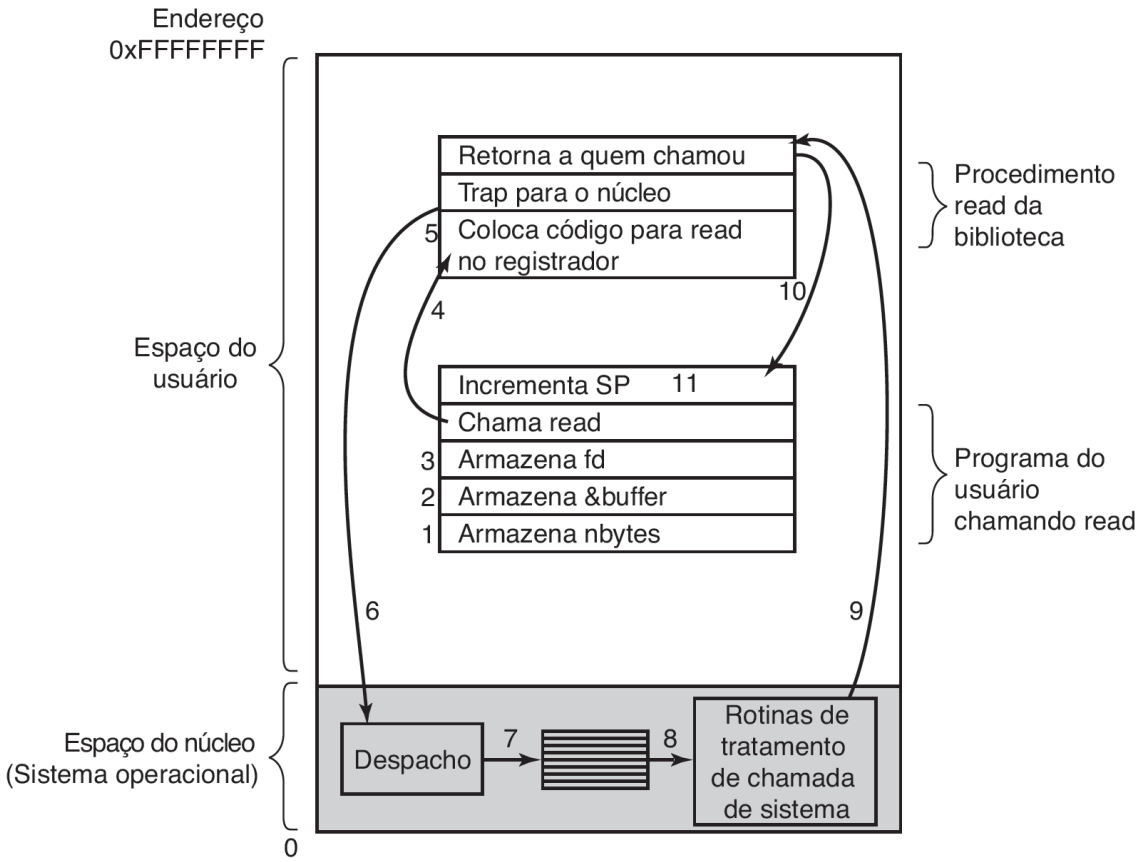


Figura 1.17 Os 11 passos na realização da chamada de sistema read (fd, buffer, nbytes).

Chamadas de Sistema para gerenciamento de processos

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

Tabela 1.1 Algumas das principais chamadas de sistema do 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 uma compensação no interior do arquivo e *seconds* é o tempo decorrido. Os parâmetros são explicados no texto.

Chamadas de Sistema para gerenciamento de processos

O interior de um *shell*:

```
#define TRUE 1

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

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

Chamadas de Sistema para gerenciamento de arquivos

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

Tabela 1.1 Algumas das principais chamadas de sistema do 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 uma compensação no interior do arquivo e *seconds* é o tempo decorrido. Os parâmetros são explicados no texto.

Chamadas ao sistema para gerenciamento de diretórios

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

Tabela 1.1 Algumas das principais chamadas de sistema do 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 uma compensação no interior do arquivo e *seconds* é o tempo decorrido. Os parâmetros são explicados no texto.

Chamadas ao Sistema para gerenciamento de diretórios

a) Sistema de arquivos antes da montagem
Ex: mount(“/dev/sd0”,“/mnt”,0)

b) Sistema de arquivos depois da montagem

Outras chamadas de sistema

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

Tabela 1.1 Algumas das principais chamadas de sistema do 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 uma compensação no interior do arquivo e *seconds* é o tempo decorrido. Os parâmetros são explicados no texto.

Windows Win32 API

No Linux existe quase uma relação de um para um entre as chamadas de sistema e as rotinas de biblioteca

Características:

- A API Win32 (Interface de Programação de Aplicativos) constitue uma interface para as chamadas de sistema reais. Muitas chamadas API Win32 são executadas no espaço do usuário (na verdade, não são chamadas de sistema) e outras fazem chamadas de sistema, ou seja, são "desacopladas"
  $\Rightarrow$ pode-se mudar as chamadas de sistema a qq hora, mantendo-se a compatibilidade entre versões do Windows
- Orientadas a eventos (teclado, mouse, botões, etc)
- Existem Milhares, ao contrário do UNIX (em torno de 100)

Windows Win32 API x Posix (Unix)

UNIX	Win32	Descrição
fork	CreateProcess	Cria um novo processo
waitpid	WaitForSingleObject	Pode esperar que um processo saia
execve	(nenhuma)	CreateProcess = fork + execve
exit	ExitProcess	Conclui a execução
open	CreateFile	Cria um arquivo ou abre um arquivo existente
close	CloseHandle	Fecha um arquivo
read	ReadFile	Lê dados a partir de um arquivo
write	WriteFile	Escreve dados em um arquivo
lseek	SetFilePointer	Move o ponteiro do arquivo
stat	GetFileAttributesEx	Obtém vários atributos do arquivo

(*Continua*)

Windows Win32 API x Posix (Unix)

(*Continuação*)

mkdir	CreateDirectory	Cria um novo diretório
rmdir	RemoveDirectory	Remove um diretório vazio
link	(nenhuma)	Win32 não dá suporte a links
unlink	DeleteFile	Destrói um arquivo existente
mount	(nenhuma)	Win32 não dá suporte a mount
umount	(nenhuma)	Win32 não dá suporte a mount
chdir	SetCurrentDirectory	Altera o diretório de trabalho atual
chmod	(nenhuma)	Win32 não dá suporte a segurança (embora o NT suporte)
kill	(nenhuma)	Win32 não dá suporte a sinais
time	GetLocalTime	Obtém o tempo atual

Tabela 1.2 As chamadas da API Win32 que correspondem aproximadamente às chamadas do UNIX da Tabela 1.1.

Estrutura de sistemas operacionais

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

Estrutura de Sistemas Operacionais

Sistemas monolíticos – estrutura básica:

- Um único programa em modo núcleo executa uma rotina principal
- Um conjunto de rotinas de serviço que executa as chamadas de sistema.
- Um conjunto de rotinas utilitárias que ajudam as rotinas de serviço.

Estrutura de Sistemas Operacionais

Sistemas monolíticos – características e desvantagens:

- Milhares de rotinas que podem chamar umas às outras

  ⇒ Sistema difícil de lidar e compreender

  ⇒ Quebra em uma das rotinas pode derrubar todo o SO

  ⇒ Não oculta informações (toda rotina é visível)

- Todas as rotinas são visíveis umas às outras

- Serviços (chamadas do sistema) são chamados a partir de parâmetros colocados na pilha + *trap*

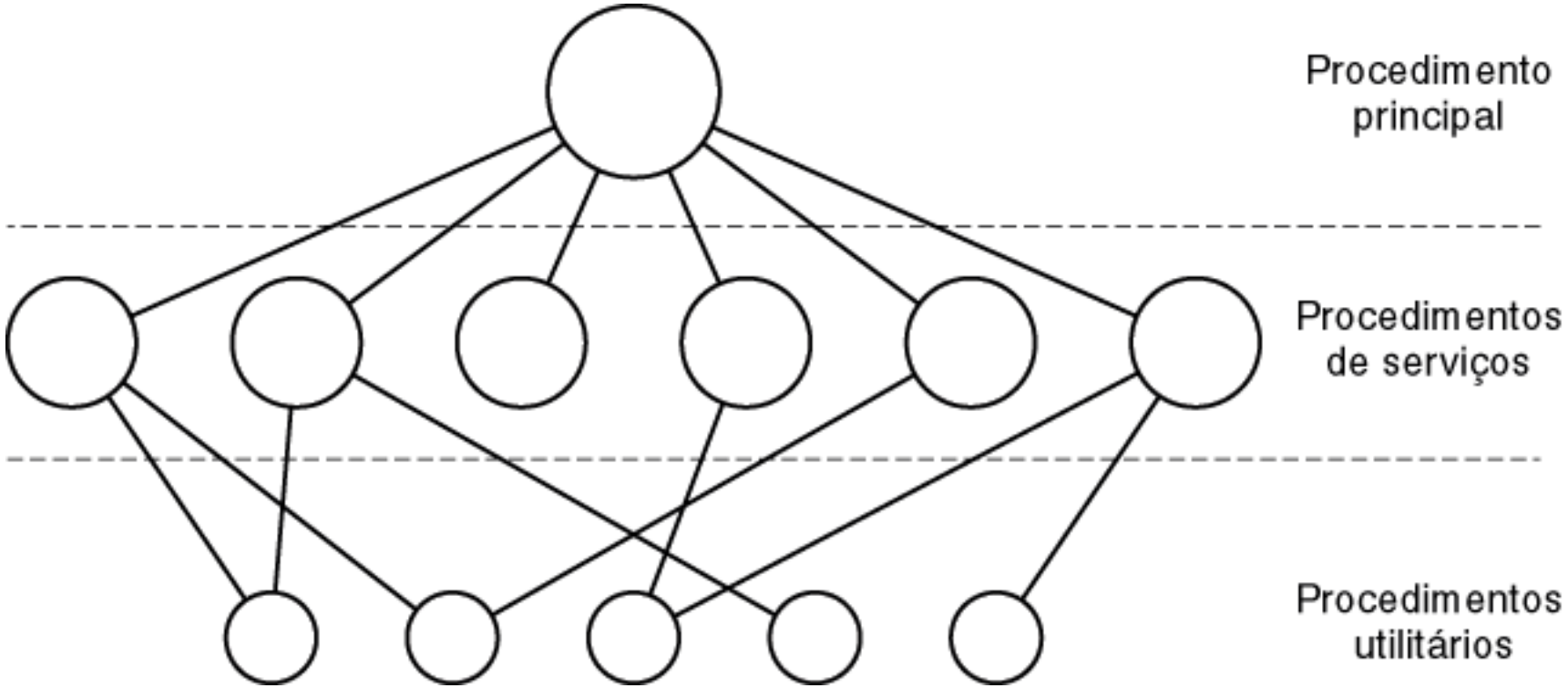
- Para cada chamada de sistema há uma rotina de serviço

  Exemplo de rotina utilitária:

  buscar dados dos programas do usuário

Estrutura de Sistemas Operacionais

Sistemas monolíticos (cont.)



Modelo simples de estruturação de um sistema monolítico

Sistemas de Camadas
Dijkstra – 1968 – Sistema 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

Tabela 1.3 Estrutura do sistema operacional THE.

Sistemas de camadas

Características:

- ✓ Cada camada construída sobre a outra (Sistema THE) ou através de anéis concêntricos (Multics)
- ✓ No Multics, as camadas mais internas têm mais privilégios
- ✓ Rotinas em anéis externos têm que desviar (como uma chamada de sistema TRAP) para os anéis internos para obter algum tipo de serviço que necessite de mais privilégio
- ✓ Construído para um sistema de lote simples
- ✓ Facilmente estendido para estruturar subsistemas de usuários (programas de professores x programas de estudantes)

Sistemas de Camadas

Camada 0:

- ✓ Alocação do processador, realizando o chaveamento de processos quando ocorrem interrupções ou quando expiram temporizadores

- ✓ Fornece a multiprogramação básica da CPU: camada construída sobre a outra (Sistema THE) ou através de anéis concêntricos (Multics)

Acima da camada 0 o sistema fornece abstração de processos sequenciais, sem a preocupação de múltiplos processos executando num só processador.

Sistemas de camadas

Camada 1:

- encarregada do gerenciamento de memória;
- aloca espaços para os processos na memória principal;
- aloca espaço em um tambor magnético para as páginas do processo para as quais não havia espaço na memória principal (memória virtual);
- processos acima desta camada não se preocupam com sua alocação: o sw da camada 1 traz páginas para a memória principal quando necessário e removidas posteriormente (sistema de paginação);

Camada 2:

- encarregada da comunicação entre cada processo e o console de operação (usuário);

Sistemas de camadas

Camada 3:

- encarregada do gerenciamento de dispositivos de E/S;
- armazena temporariamente os fluxos de informação que iam para ou que vinham desses dispositivos;
- fornece uma interface mais amigável do acesso aos dispositivos de E/S para as camadas superiores;

Camada 4:

- programas de usuário: não se preocupavam com o gerenciamento de processos, de memória ou de dispositivos de E/S.

Camada 5:

- localização do processo operador do sistema

Obs: Na verdade, todas as partes do sistema são agrupadas em um único programa-objeto (sistema de camadas usado para suporte ao projeto)

Sistemas de camadas - Multics

Generalização do sistema de camadas;

Descrito como uma série de anéis concêntricos (anéis internos mais privilegiados);

Chamada de uma rotina externa a um anel interno: TRAP

Ao contrário do THE, onde o mecanismo de camadas é apenas um suporte para o usuário (na prática tudo está num único programa executável), o mecanismo de anéis do MULTICS está presente em tempo de execução (e imposto pelo hw);

Mecanismo de anéis pode ser estendido a programas do usuário (ex: professores no nível n, alunos no nível n+1)

Micronúcleos

Ideia: em qualquer programa, o código está sujeito a erros

- exemplo: em um sistema industrial grande, média de 10 erros a cada 1000 linhas de código
  - SO com 5 milhões de linha de código

    $\Rightarrow$ 50 mil erros no código, apesar de nem todos os erros serem fatais
  - SOs possuem botões de reinicialização

Micronúcleos

Objetivo:

Atingir alta confiabilidade, dividindo o SO em pequenos e bem definidos módulos, onde a maioria é executada em modo usuário e apenas um (o micronúcleo) é executado em modo núcleo.

- ✓ Erro no sistema de arquivos ou em um driver não derruba o SO

  Ex: erro na unidade de áudio não trava o computador (ao contrário de um sistema monolítico)

- ✓ Não é comum em computadores pessoais (exceção: OS X)

- ✓ Usual em sistemas de tempo real, militares, industriais

  Exemplos: QNX, Symbian, Minix 3

Micronúcleos

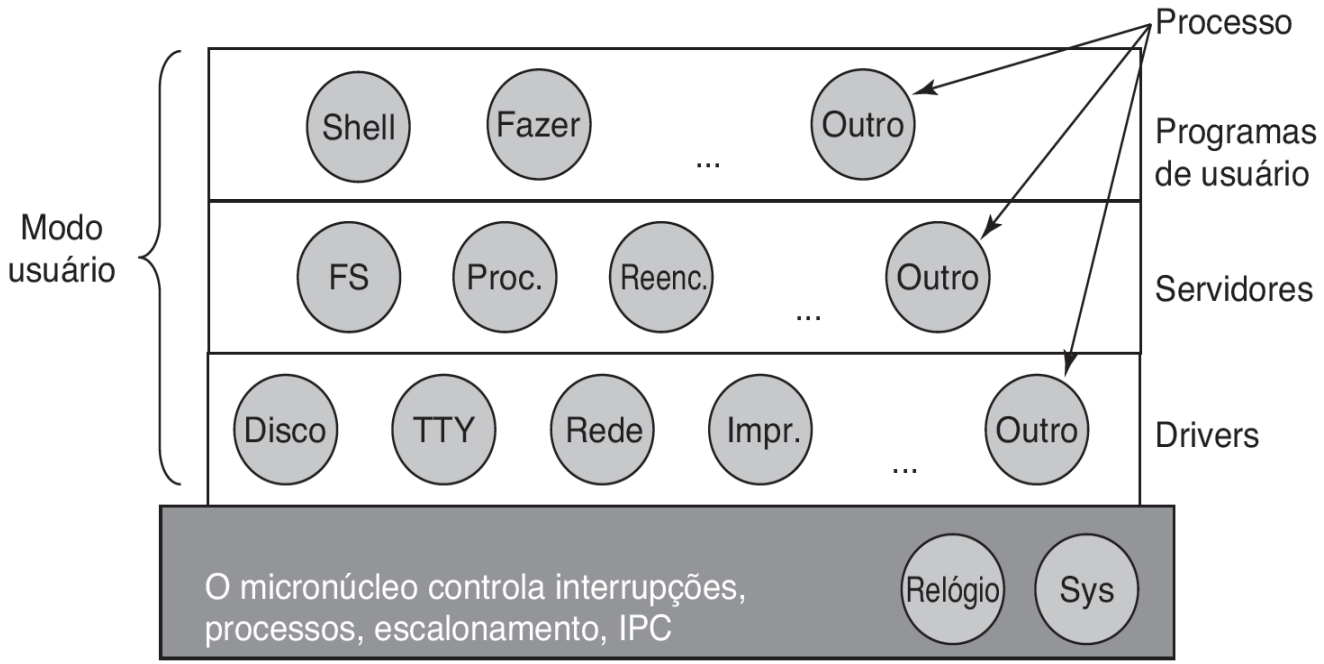


Figura 1.23 Estrutura do sistema MINIX 3.

Sys – tratadores de chamadas do núcleo

Modelo Cliente-Servidor

Divisão entre duas classes de processos:

- **servidores: processos que prestam serviços;**
- **clientes: processos que solicitam e usam esses serviços;**
- **comunicação é realizada através de troca de mensagens;**
  - **processo cliente constrói uma msg e a envia ao servidor apropriado**
  - **processo servidor faz o trabalho e envia msg de resposta**

– abstração que pode ser usada para uma única máquina ou para uma rede de máquinas.

Modelo cliente-servidor

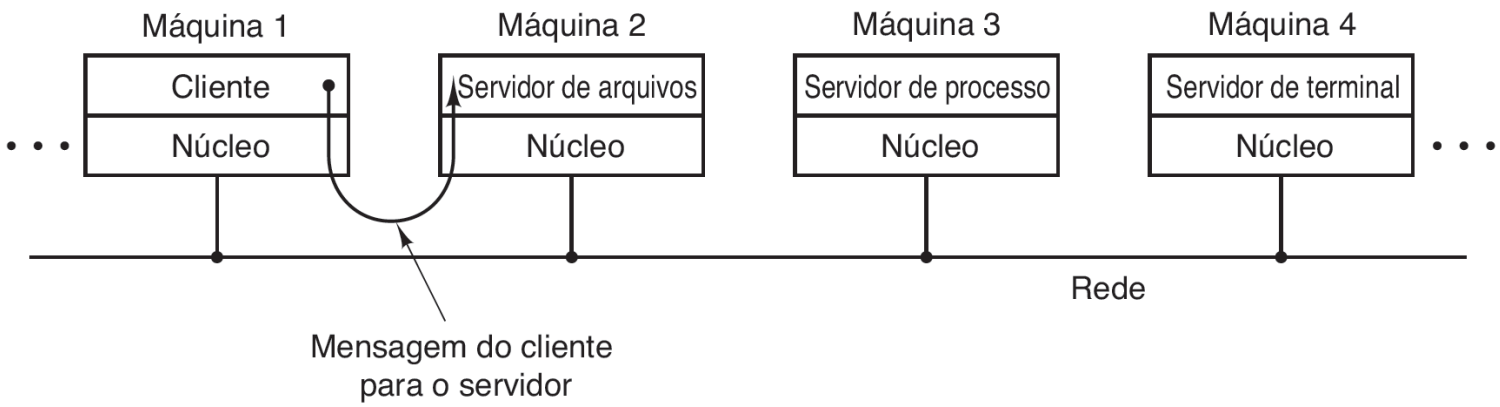


Figura 1.24 O modelo cliente–servidor em uma rede.