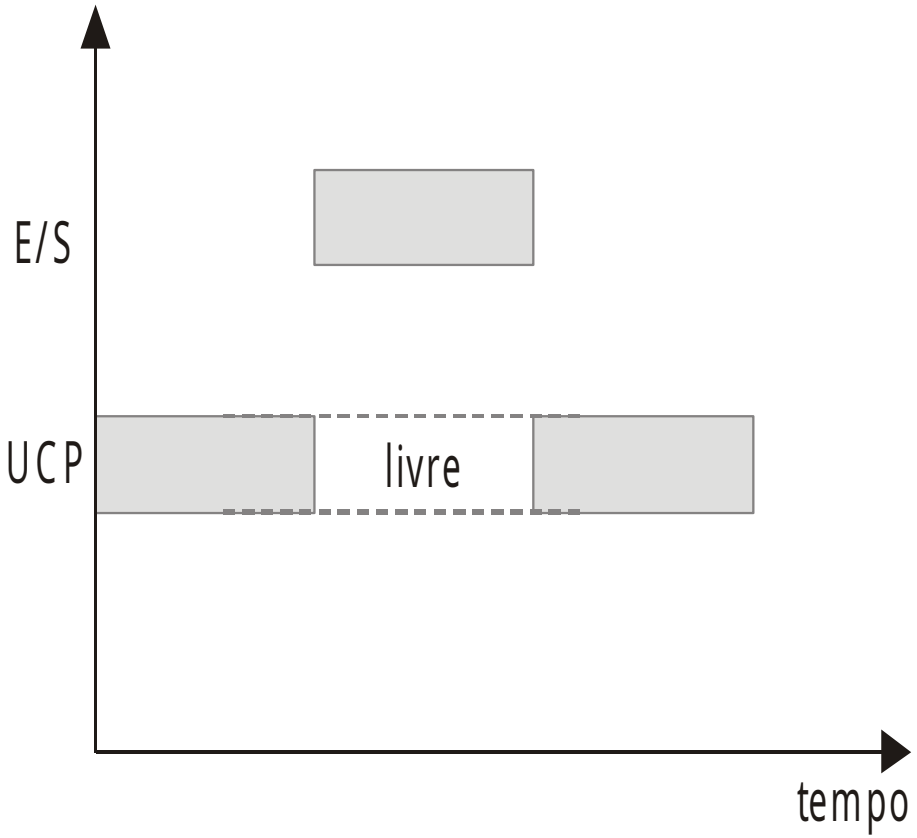
Concorrência e Estrutura do Sistema Operacional

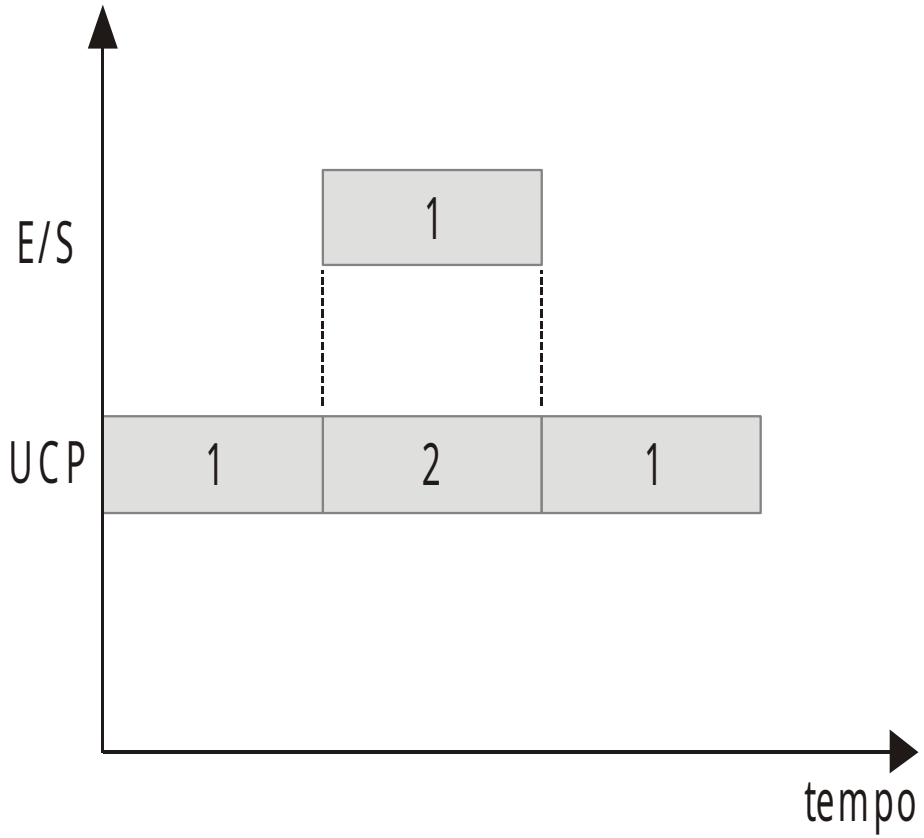
UNIFEI
Universidade Federal de Itajubá
IMC – Instituto de Matemática e Computação

Prof. Carlos Minoru Tamaki

Sistema monoprogramável x Sistema multiprogramável



(a) Sistema Monoprogramável



(b) Sistema Multiprogramável

Exemplo de utilização do sistema

Leitura de um registro	0,0015 s
Execução de 100 instruções	0,0001 s

Total	0,0016 s
% utilização da CPU (0,0001/0,0015) = 0,066 = 6,6%	

- Outro aspecto é a subutilização da memória principal. Um programa que não ocupe totalmente a memória causa áreas livres sem utilização.

Características de execução de programas

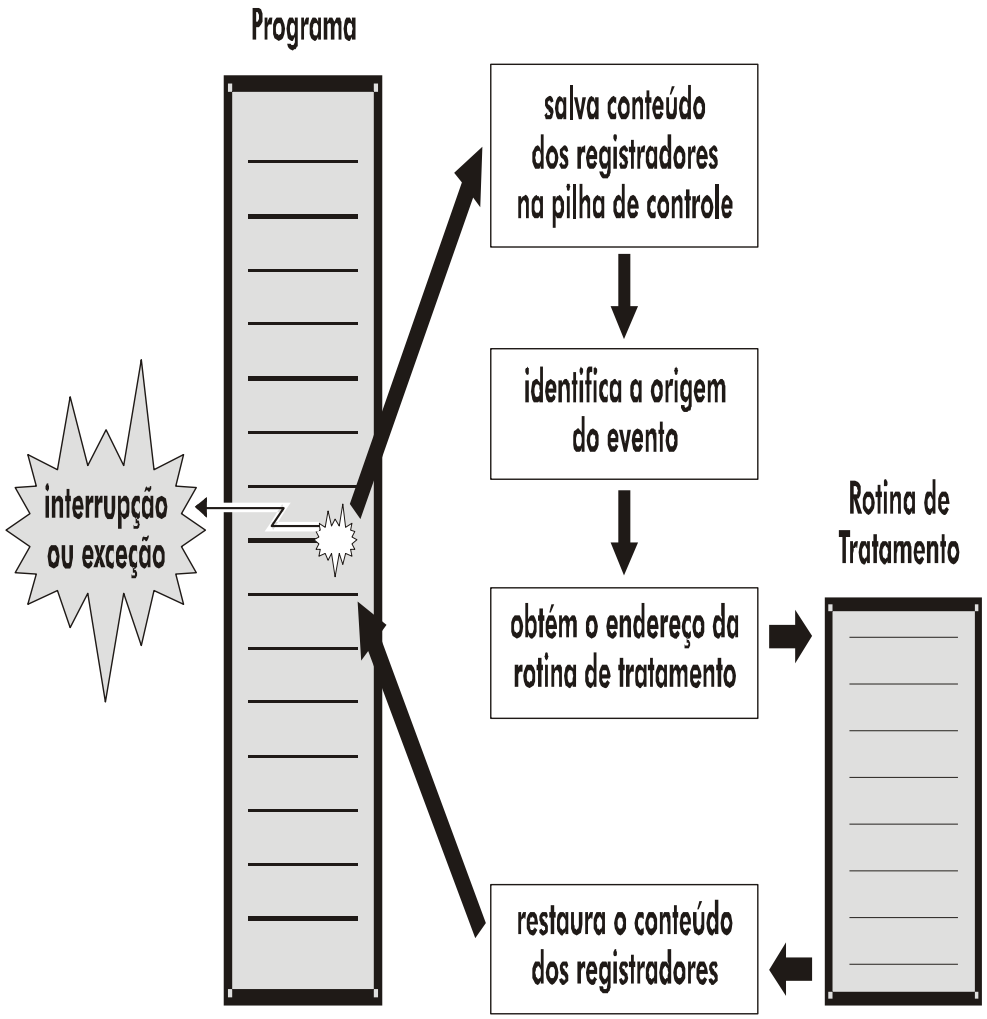
Características	Prog1	Prog2	Prog3
Utilização da CPU	Alta	Baixa	Baixa
Operações de E/S	Poucas	Muitas	Muitas
Tempo de processamento	5 min	15 min	10 min
Memória utilizada	50 Kb	100 Kb	80 Kb
Utilização do disco	Não	Não	Sim
Utilização de terminal	Não	Sim	Não
Utilização de Impressora	Não	Não	Sim

Características de execução de programas

	Monoprogramação	Multiprogramação
Utilização da CPU	17%	33%
Utilização de Memória	30%	67%
Utilização do disco	33%	67%
Utilização de Impressora	33%	67%
Tempo de processamento	30 min	15 min
Taxa de throughput	6 prog./hora	12 prog./hora

Interrupção e Exceção

- A interrupção é um mecanismo que torna a implementação da concorrência nos computadores.
- Uma interrupção é sempre gerada por um evento externo ao programa e independe da instrução que está em execução

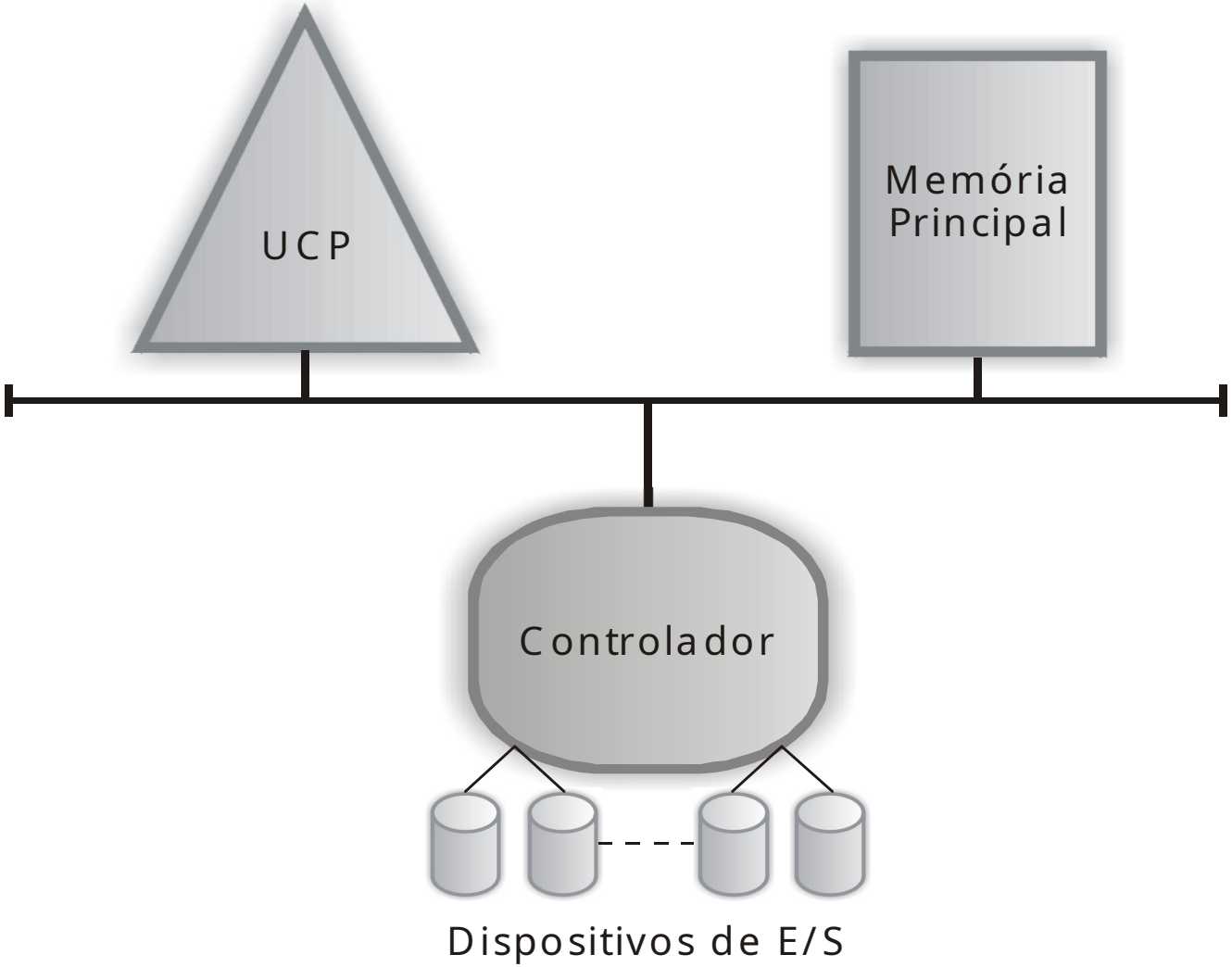


Mecanismo de interrupção

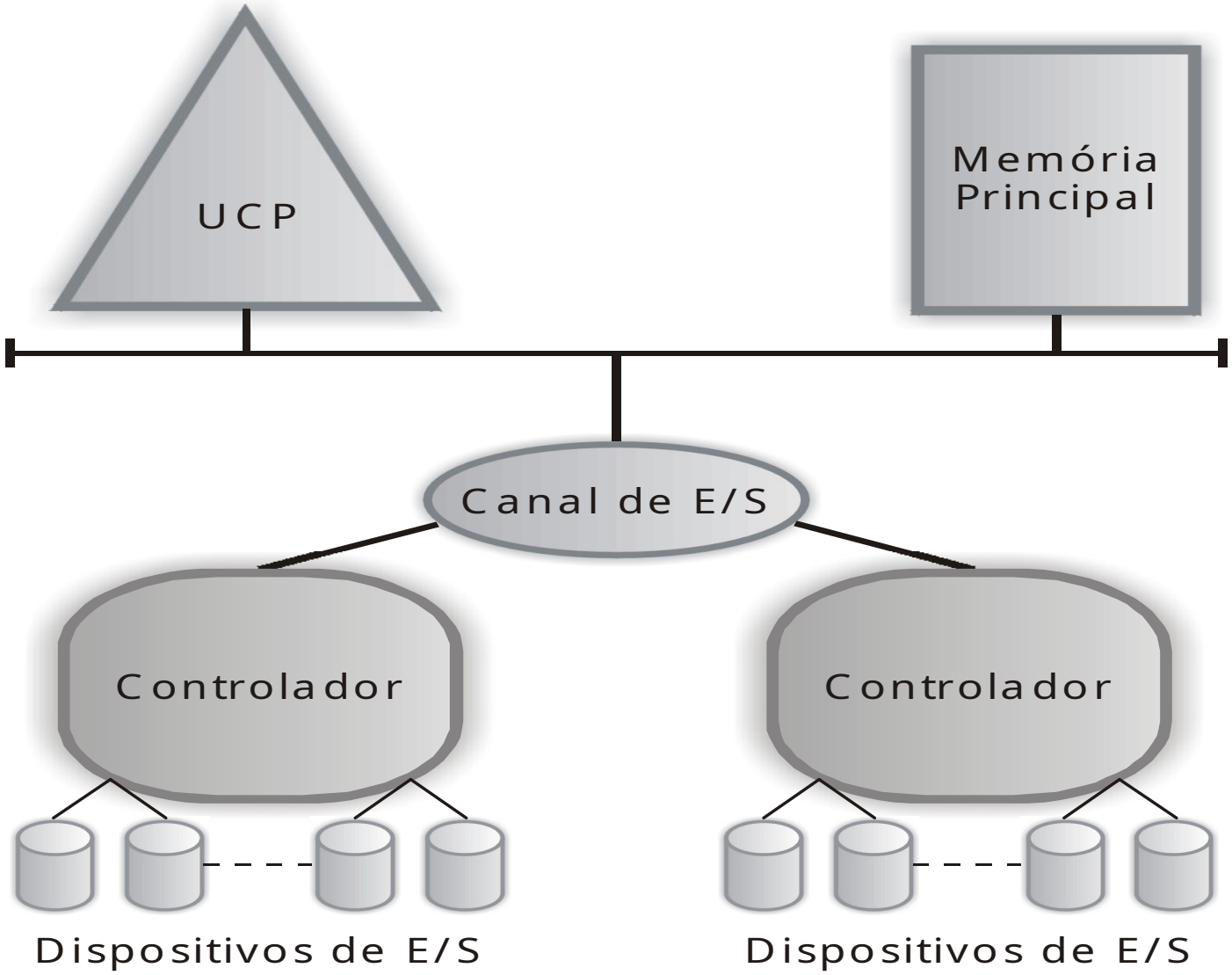
Via Hardware	1. Um sinal de interrupção é gerado para o processador; 2. após o término da instrução corrente, o processador identifica o pedido de interrupção; 3. os conteúdos dos registradores PC e de status são salvos; 4. o processador identifica qual rotina de tratamento que será executada e carrega o PC com o endereço inicial desta rotina;
Via Software	5. a rotina de tratamento salva o conteúdo dos demais registradores do processador na pilha de controle do programa; 6. a rotina de tratamento é executada; 7. após o término da execução da rotina de tratamento, os registradores de uso geral são restaurados, além do registrador de status e o PC, retornando à execução do programa interrompido.

Operações de Entrada/Saída

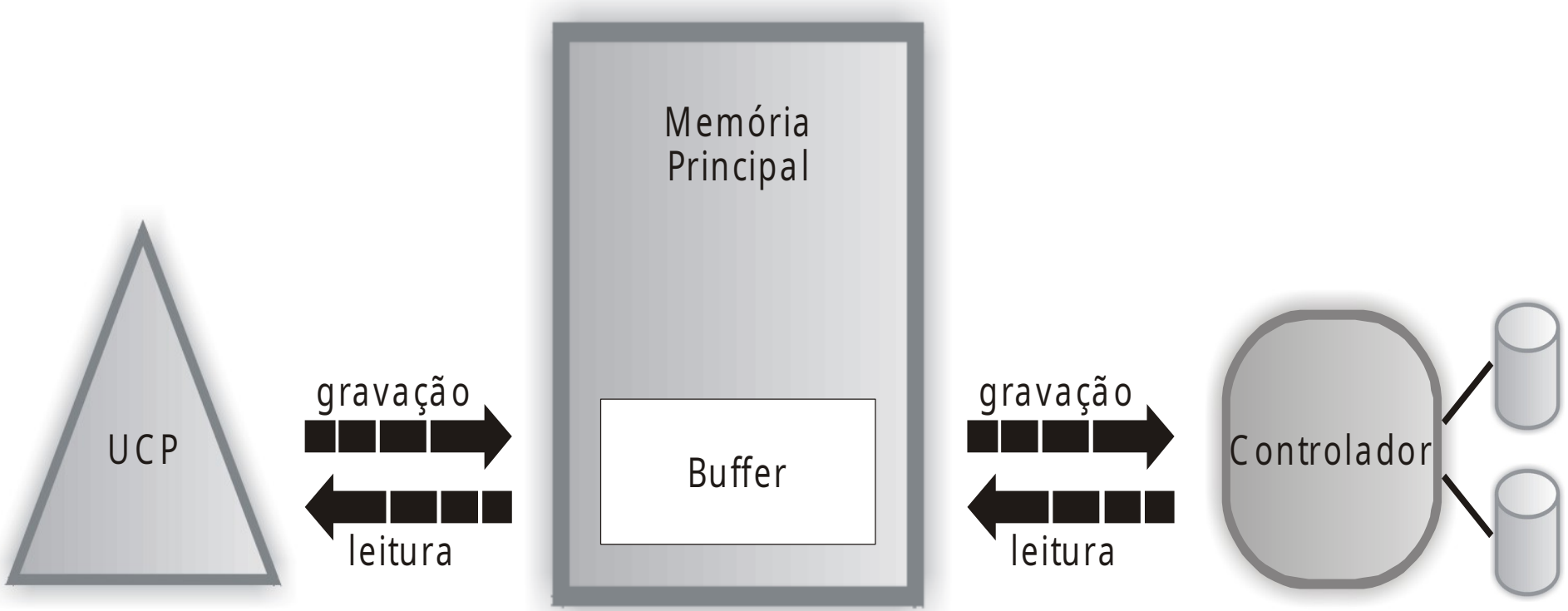
- Controlador



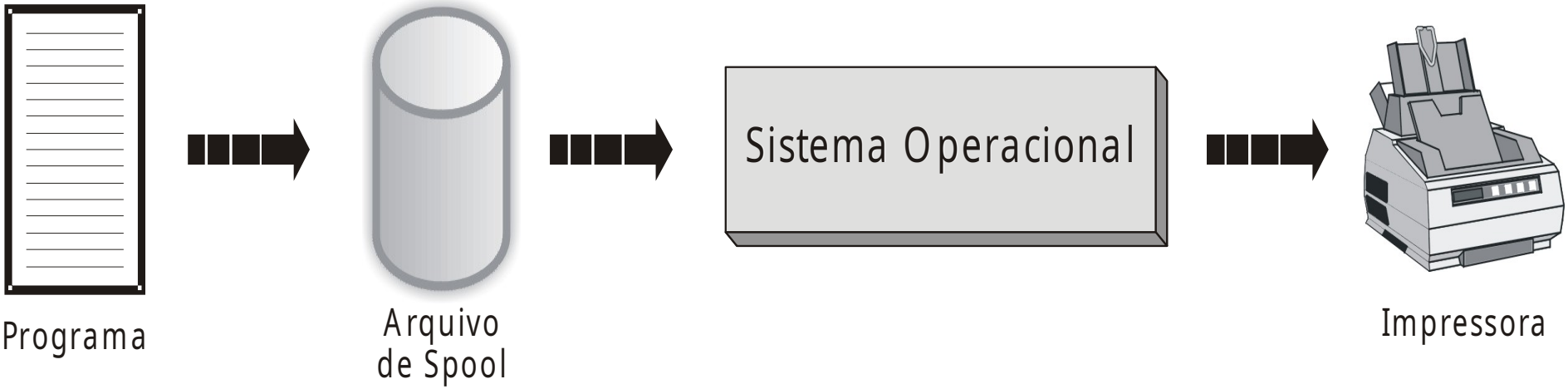
Canal de E/S


UCP
Memória Principal
Canal de E/S
Controlador
Controlador
Dispositivos de E/S
Dispositivos de E/S

Buffering

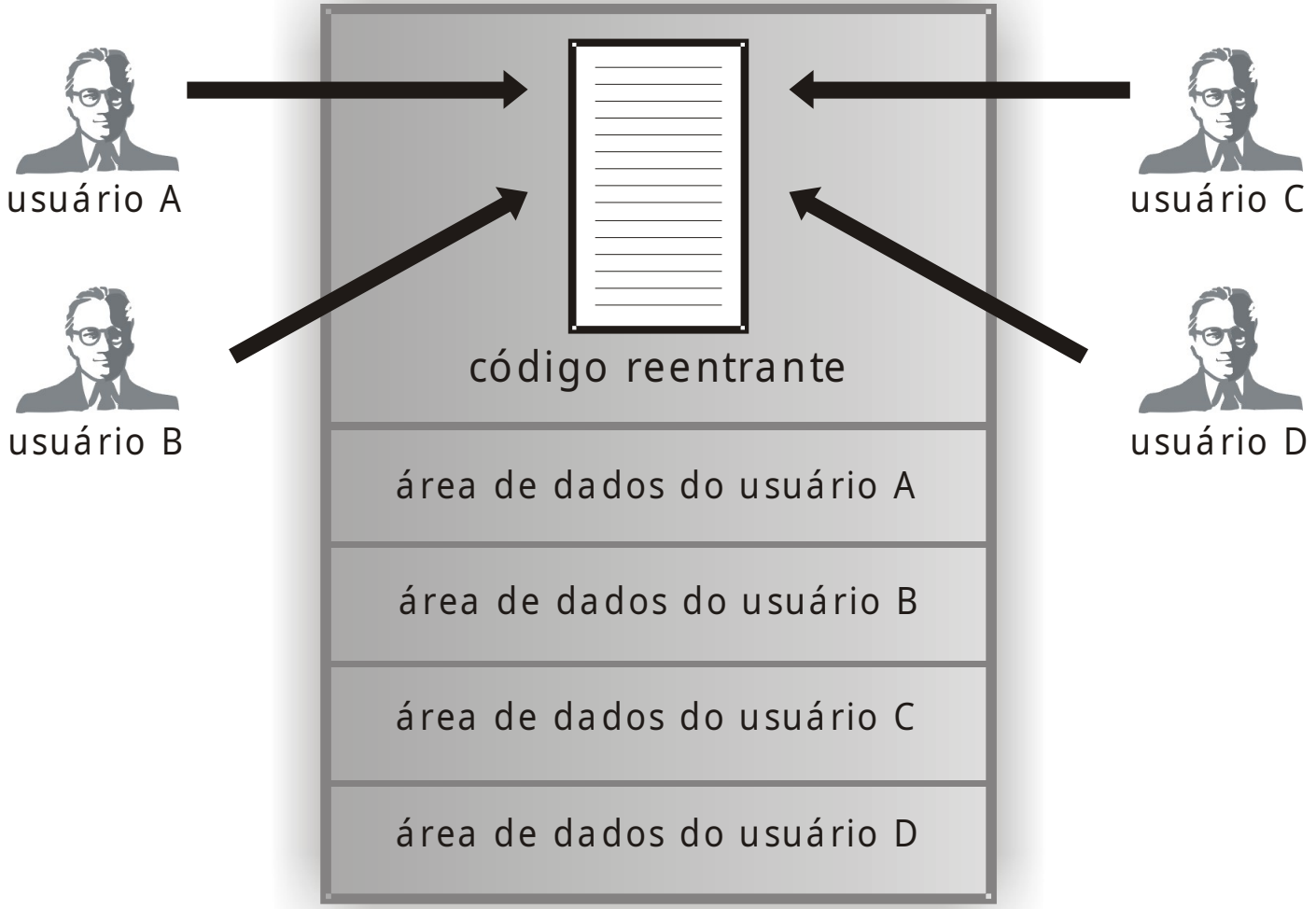

Memória
Principal
UCP
gravação
leitura
Buffer
gravação
leitura
Controlador

Spooling

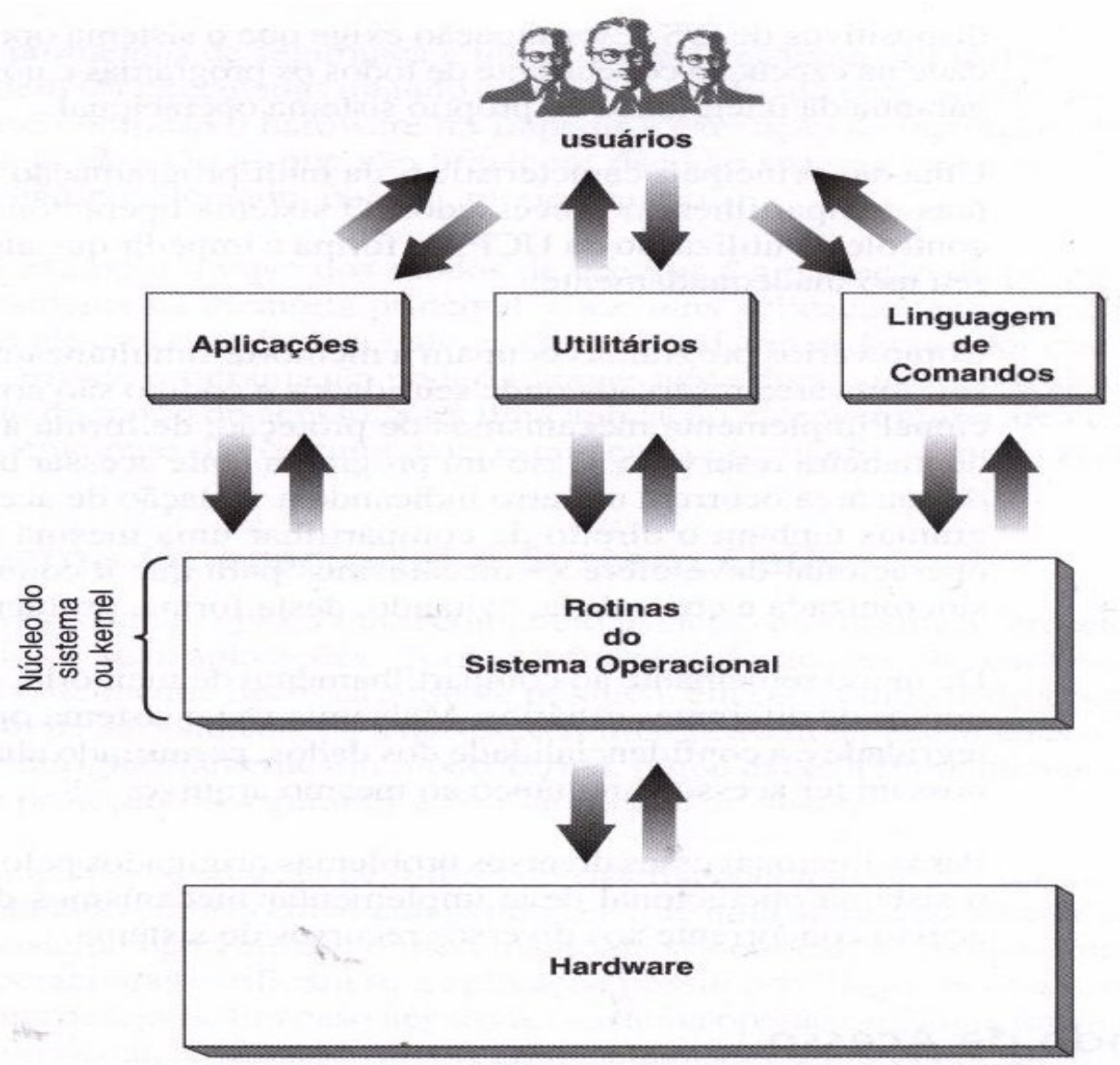


SPOOL (Simultaneous Peripheral Operations On-Line)

Reentrância



Estrutura do Sistema Operacional



Principais Funções do Núcleo do Sistema

- Tratamento de interrupções e exceções
- Criação e eliminação de processos e threads
- Sincronização e comunicação entre processos e threads
- Escalonamento e controle dos processos e threads
- Gerência de memória
- Gerência do sistema de arquivos
- Gerência dos dispositivos de E/S
- Suporte a redes locais e distribuídas
- Contabilização do uso do sistema
- Auditoria e segurança do sistema

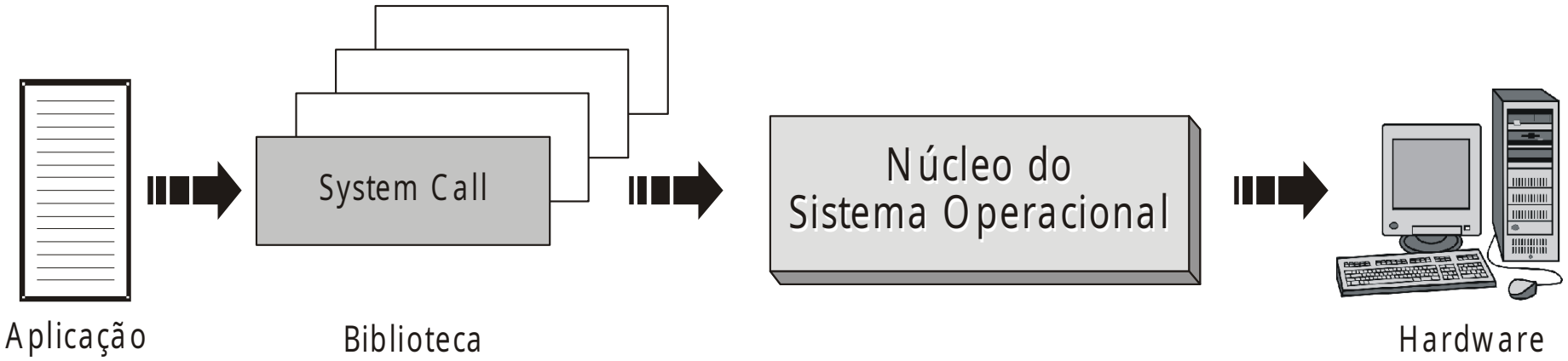
Modos de Acesso

- Uma preocupação no projeto de sistema operacional é a implementação de mecanismo de proteção ao núcleo do sistema e de acesso aos seus serviços
- Muitas das principais implementações de segurança de um sistema operacional utilizam um mecanismo presente no hardware dos processadores, conhecido como **modo de acesso**.
- Os processadores possuem 2 modos de acesso: **modo usuário** e **modo kernel**.
- As **instruções privilegiadas** não devem ser utilizadas de maneira indiscriminada pelas aplicações.

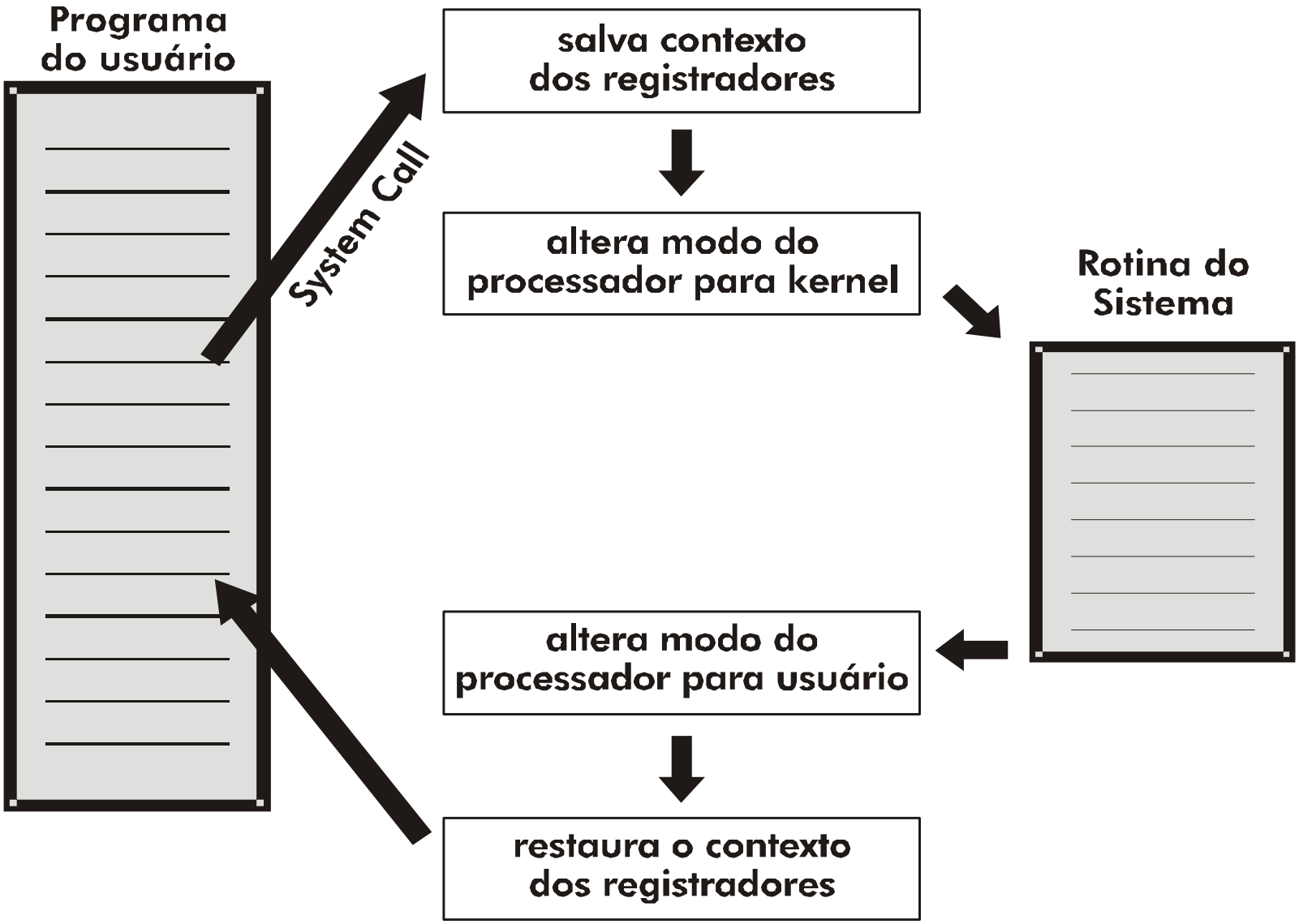
Rotinas do Sistema Operacional e System Calls

- As **rotinas do sistema operacional** compõe o núcleo do sistema, oferecendo serviços aos usuários e suas aplicações.
- Para que estas rotinas possam ser executadas o processador deve obrigatoriamente estar em modo kernel, o que exige implementação de mecanismos de proteção.
- Todo o controle de execução de rotinas do sistema operacional é realizado pelo mecanismo conhecido como **system call**.
- O sistema verifica se a aplicação possui privilégio para executar a rotina desejada, em caso negativo o sistema operacional irá impedir o desvio para a rotina. Este é o mecanismo de proteção por **software**, que garante que aplicações só poderão executar rotinas pré autorizadas.

Chamada a uma Rotina do Sistema



System Call


Programa
do usuário
System Call
salva contexto
dos registradores
altera modo do
processador para kernel
Rotina do
Sistema
altera modo do
processador para usuário
restaura o contexto
dos registradores

Chamada de Sistema Read

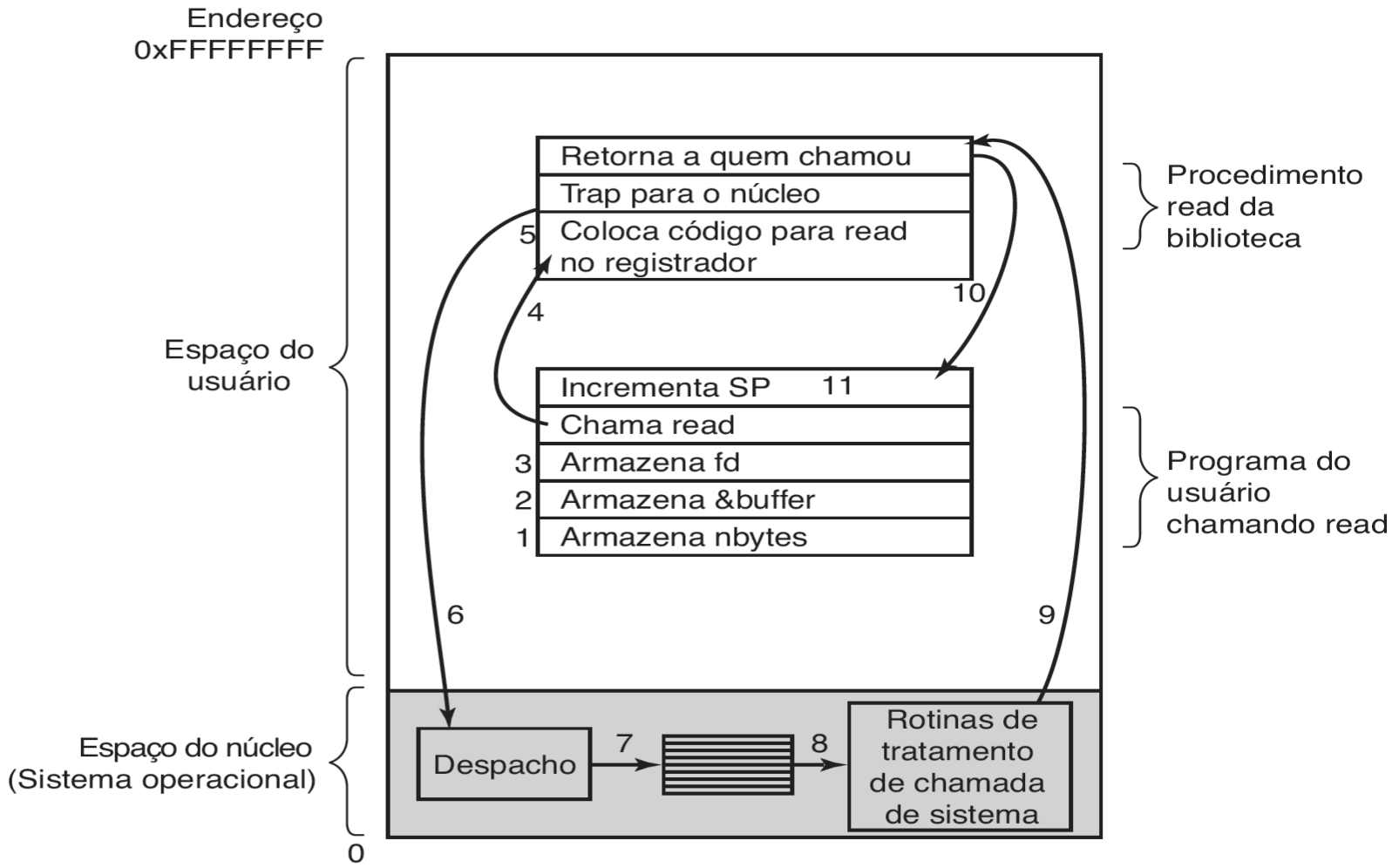


Figura 1.17 Os 11 passos na realização da chamada de sistema read (fd, buffer, nbytes).

Funções das System Calls

Funções	System Calls
Gerência de processos e threads	Criação e eliminação de processos e threads Alteração das características de processos e threads Sincronização e comunicação entre processos e threads Obtenção de informações sobre processos e threads
Gerência de memória	Alocação e desalocação de memória
Gerência do sistema de arquivos	Criação e eliminação de arquivos e diretórios Alteração das características de arquivos e diretórios Abrir e fechar arquivos Leitura e gravação em arquivos Obtenção de informações sobre arquivos e diretórios
Gerência de dispositivos	Alocação e desalocação de dispositivos Operações de entrada/saída em dispositivos Obtenção de informações sobre dispositivos

Funções do POSIX

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.

Funções do POSIX

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.

Funções do POSIX

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.

Funções do POSIX

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.

Chamadas da API WIN32

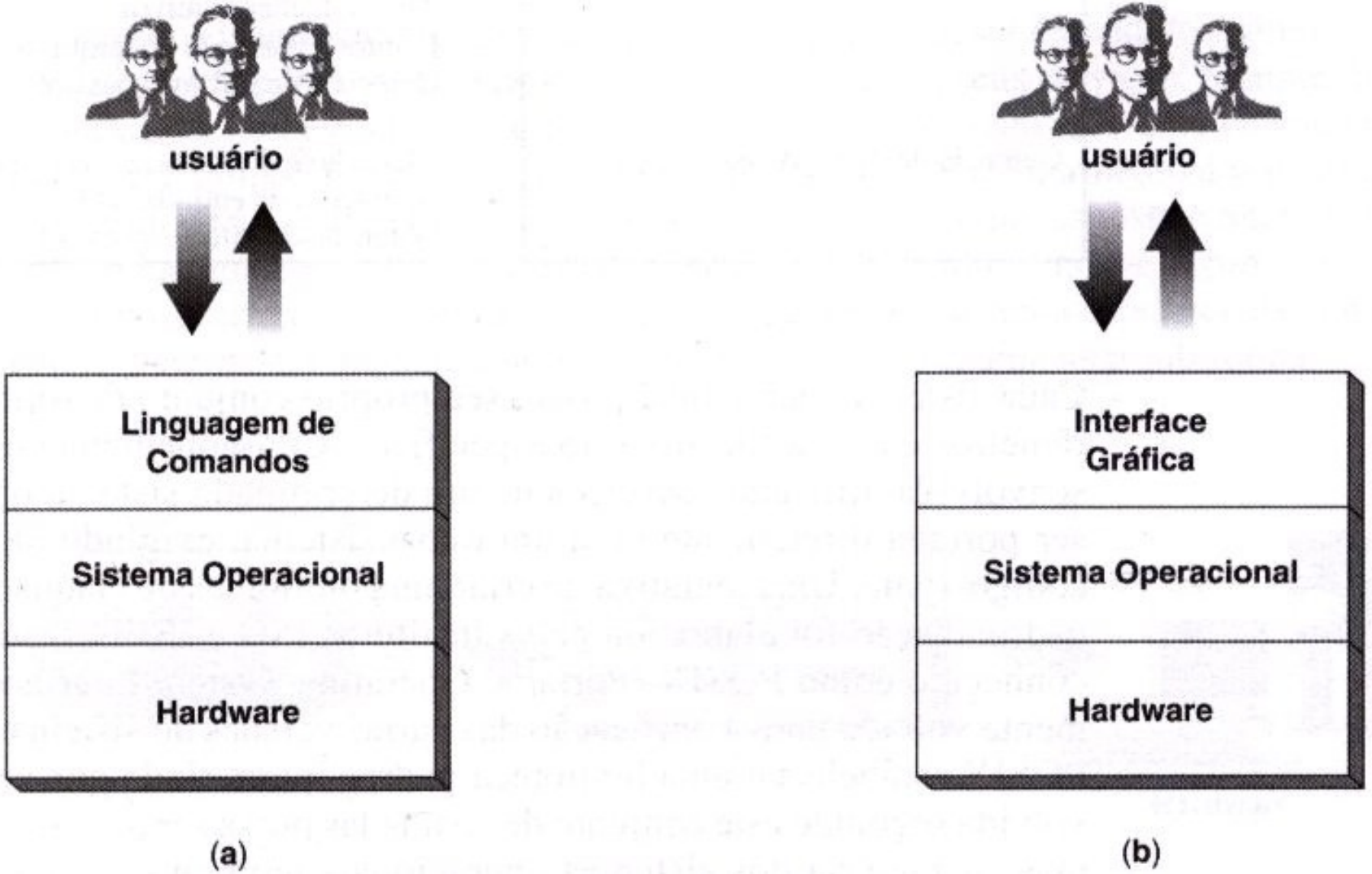
UNIX	Win32	Descrição
fork	CreateProcess	Cria um novo processo
waitpid	WaitForSingleObject	Pode esperar que um processo saia
execve	(nenhuma)	CreateProcess = fork + execve
exit	ExitProcess	Conclui a execução
open	CreateFile	Cria um arquivo ou abre um arquivo existente
close	CloseHandle	Fecha um arquivo
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
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.

Linguagem de Comandos

Exemplos de Comandos do MS Windows (DOS)	
Comando	**Descrição**
dir	Lista o conteúdo de um diretório
cd	Altera o diretório default
type	Exibe o conteúdo de um arquivo
del	Elimina arquivos
mkdir	Cria um diretório
ver	Mostra a versão do Windows

Linguagem de Comandos



Dúvidas?