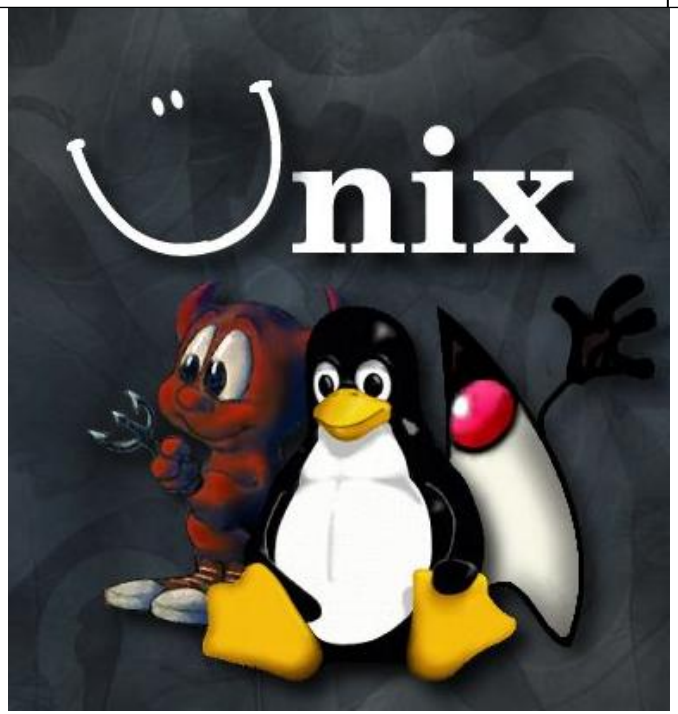
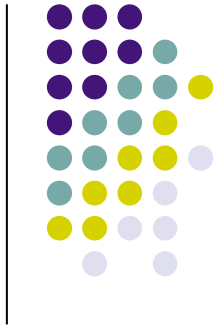
Sistemas de Computação

O Sistema Operacional Unix

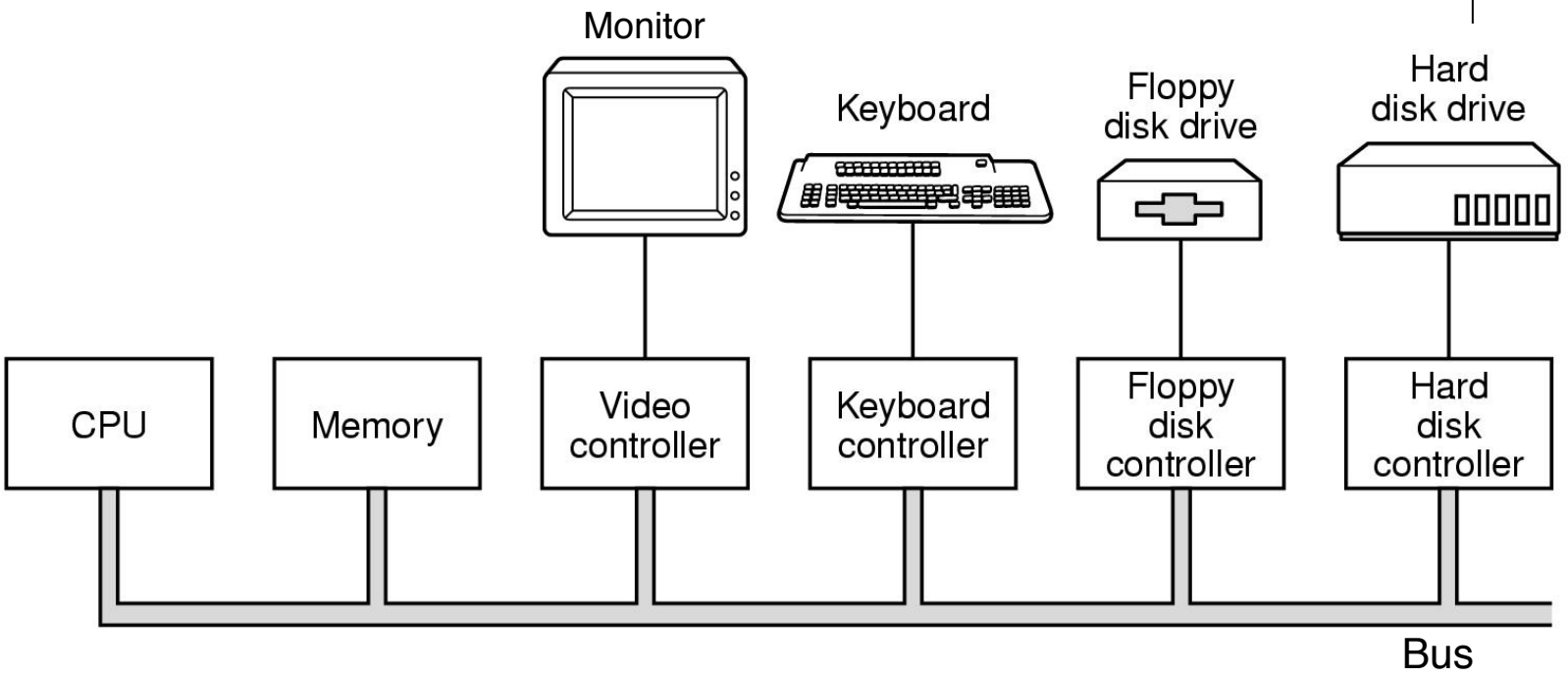


Introdução



- Interrupções de hardware
- Execução de uma Chamada de Sistema
- Alocação de memória
- Chamadas de Sistema típicas
- Arquitetura do Unix

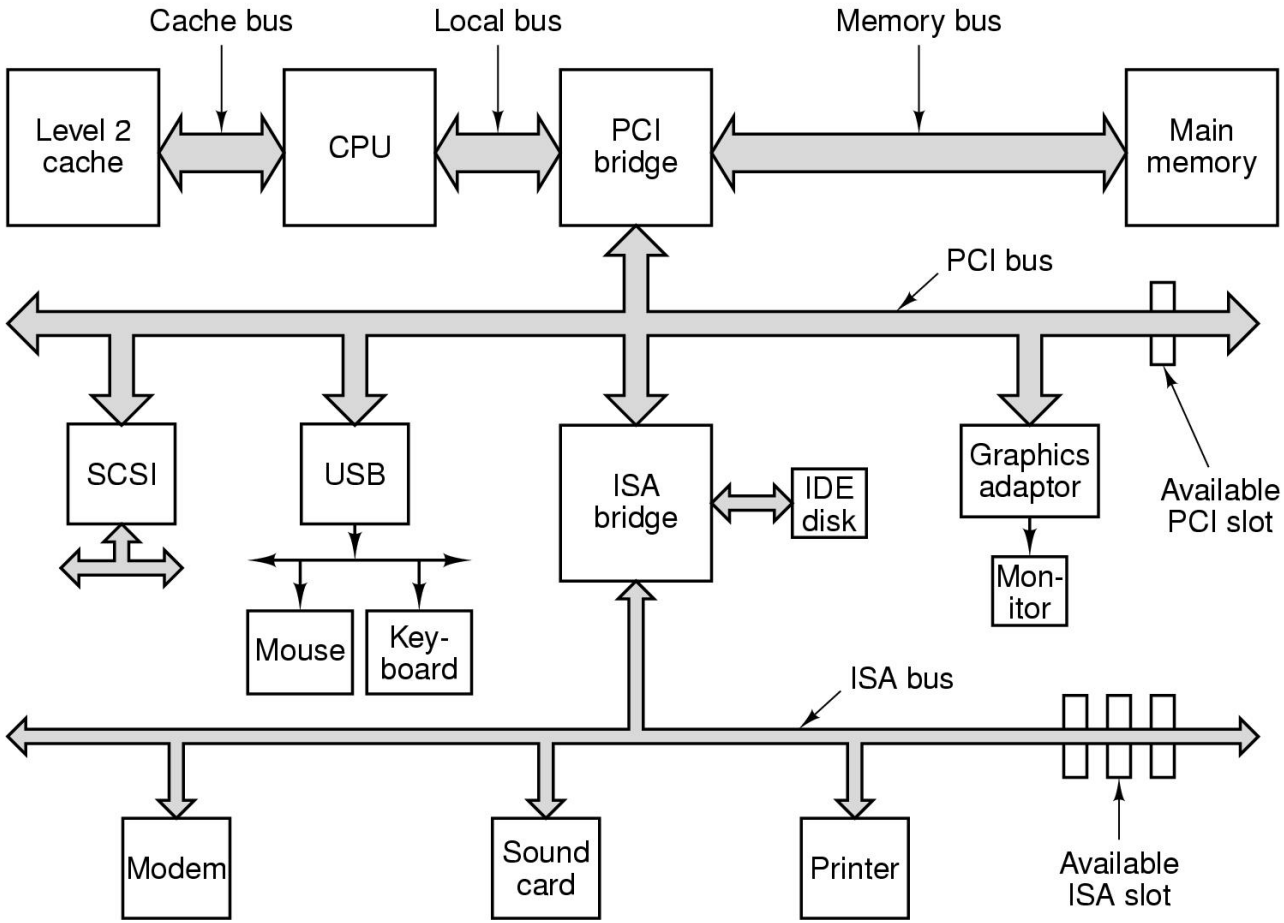
Visão simplificada do Hardware



Através do barramento trafegam:

- Dados
- Endereços (Memória e portas de E/S)
- Instruções de máquina (p. CPU controladores de E/S)

Arquitetura Típica do Hardware



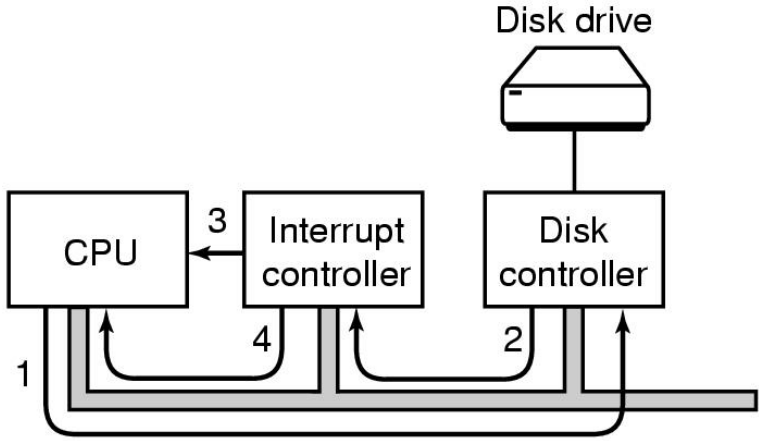
Barramentos com diferentes velocidades

Interação Dispositivos-CPU através de Interrupções de HW

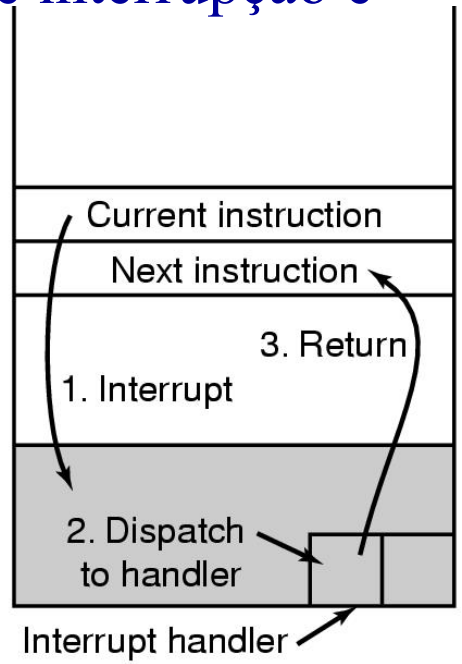
Processamento de Interrupções

Interrupção é a forma do hardware avisar o núcleo que alguma ação precisa ser tomada

O processo atual é interrompido e um tratador de interrupção é ativado.



(a)

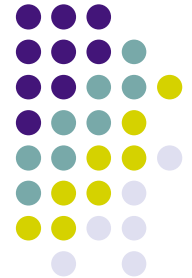


(b)

(a) Iniciando uma E/S e obtendo uma interrupção de hardware

(b) Fluxo de controle no tratamento de uma interrupção pelo núcleo

Chamadas de Sistema



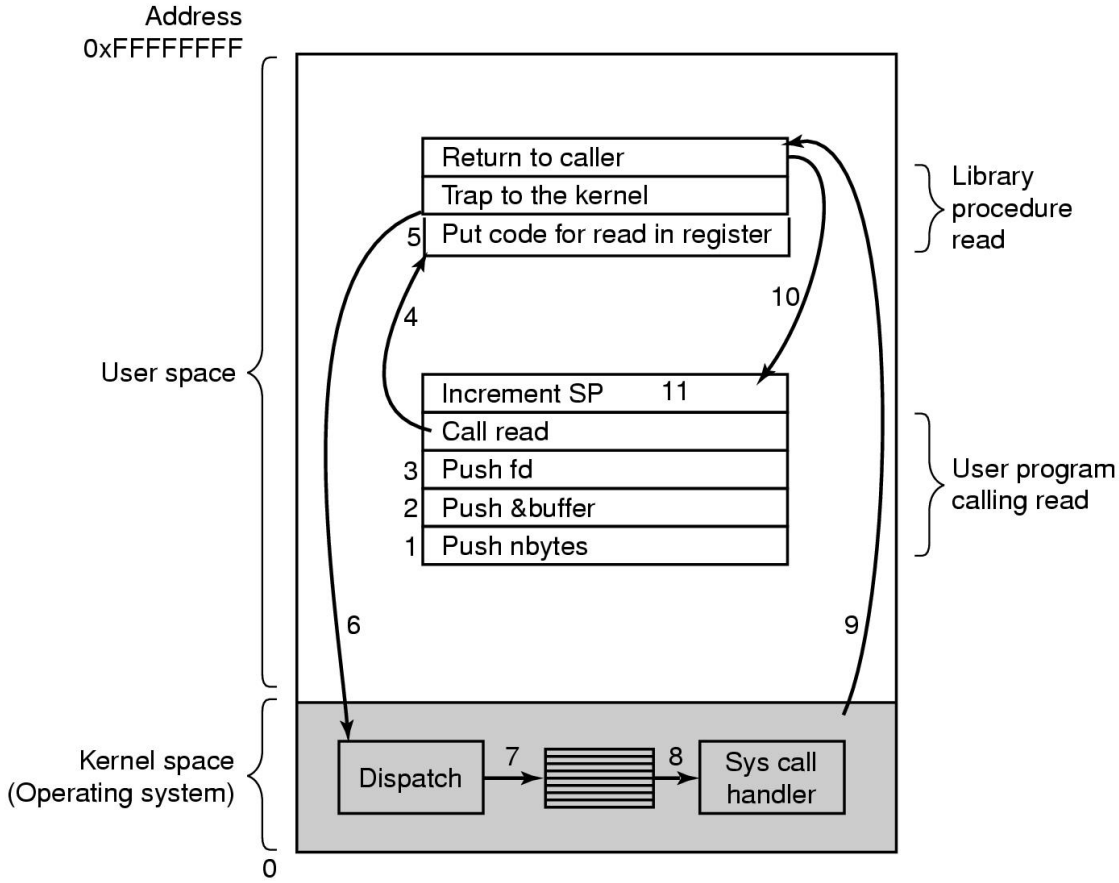
É o conjunto de funções/procedimentos disponibilizadas pelo núcleo (a API) para acesso a recursos da máquina (p.ex. Entrada/Saída)

As funções fazem parte de uma biblioteca do sistema (ligada a todo programa). Cada função executa um TRAP para trocar para o modo núcleo.

Quando um processo faz uma chamada de sistema, ele abre mão do controle, passando-o para o núcleo.

Etapas de uma Chamada ao Sistema

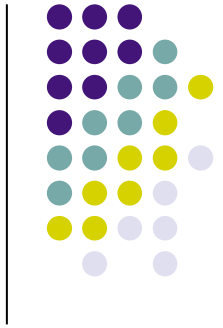
Exemplo: read (fd, buffer, nbytes)



Chamada de sistema = interrupção de software

Fluxo de controle: programa de aplicação >> biblioteca de chamadas de sistema (libc.so) >> núcleo >> biblioteca >> programa de aplicação.

Conceitos Centrais



Processo:: programa em execução

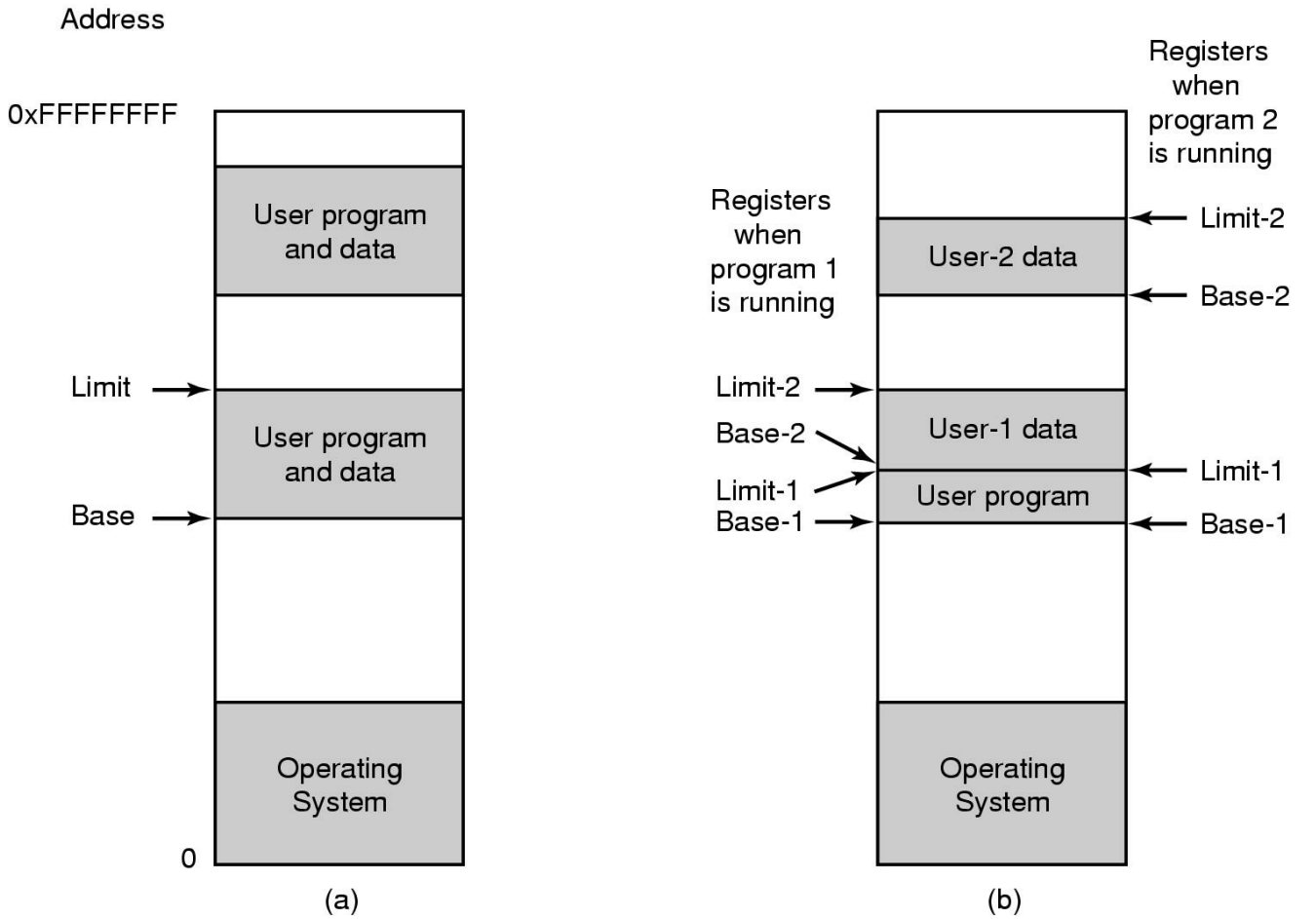
contexto do processo = conteúdo dos registradores da CPU, espaço de endereçameno e contexto de software.

espaço de endereçamento :: faixa de memória ocupada, contendo executável, dados do programa e pilha de execução

contexto de software :: identificacão, uid, conj. de arquivos abertos, sinais pendentes, processos relacionados, quotas (arquivos, memória, buffer de E/S), e privilégios, diretório corrente, etc.

Todas essas informações são mantidas pelo núcleo para cada processo, para reiniciá-lo exatamente do ponto em que foi interrompido.

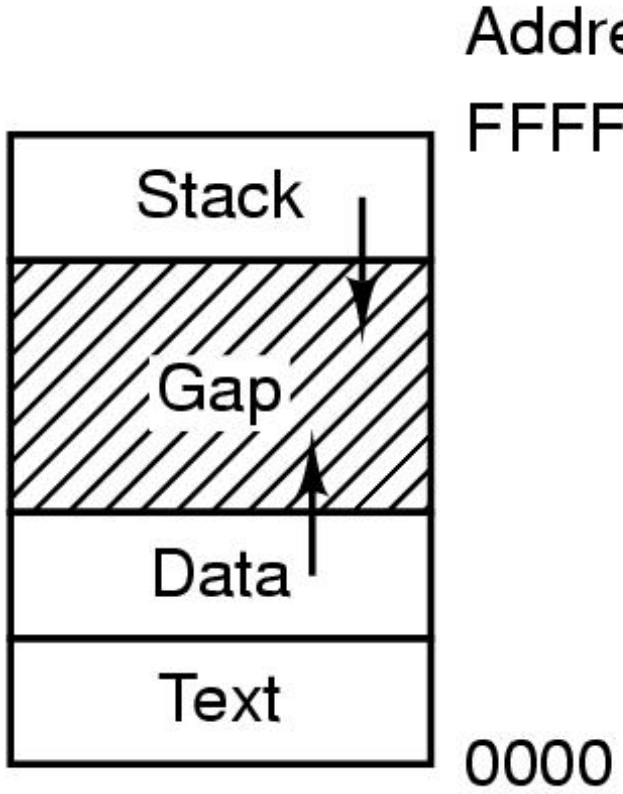
Ocupação da Memória



Cada processo ocupa uma região própria (e isolada) na memória e endereços são traduzidos

Pode ser em uma região contígua (a) ou não (b)

Um processo na memória



- Processos possuem 3 segmentos: text (instruções), dados dinâmicos, e pilha
- Dados e pilha crescem em sentidos opostos
- Através da chamada de sistema BRK(newDataLimitAddr) processo pode requisitar mais espaço de memória.

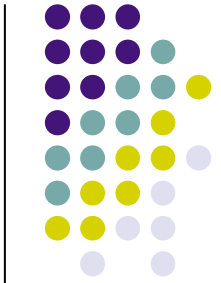
Chamadas de Sistema: Process Management

Gerenciamento de processos

Chamada	Descrição
pid = fork( )	Crie um processo filho idêntico ao processo pai
pid = waitpid(pid, &statloc, options)	Aguarde um processo filho terminar
s = execve(name, argv, environp)	Substitua o espaço de endereçamento do processo
exit(status)	Termine a execução do processo e retorne o estado

- **pid = fork()** cria uma copia do processo pai, que executará o mesmo programa a partir do mesmo ponto de execução (mas terá outro PID)
- O **exec(comand, parameters)** carrega e inicia execução de um novo programa executável (comand), com o mesmo PID
- **Pid = wait(pid, &status)** faz o pai esperar pelo exit/ término forçado de algum processo filho.

Esboço de uma shell



```
while (TRUE) {                                  /* repeat forever */
    type_prompt( );                             /* display prompt */
    read_command (command, parameters)          /* input from terminal */

if (fork() != 0) {                              /* fork off child process */
    /* Parent code */
    waitpid( -1, &status, 0);                   /* wait for child to exit */
} else {
    /* Child code */
    execve (command, parameters, 0);            /* execute command */
 }
}
```

Obs: **fork()** retorna 0 para o processo filho (criado), e o Pid do filho para o processo pai,

Chamada fork( ) / exec( )

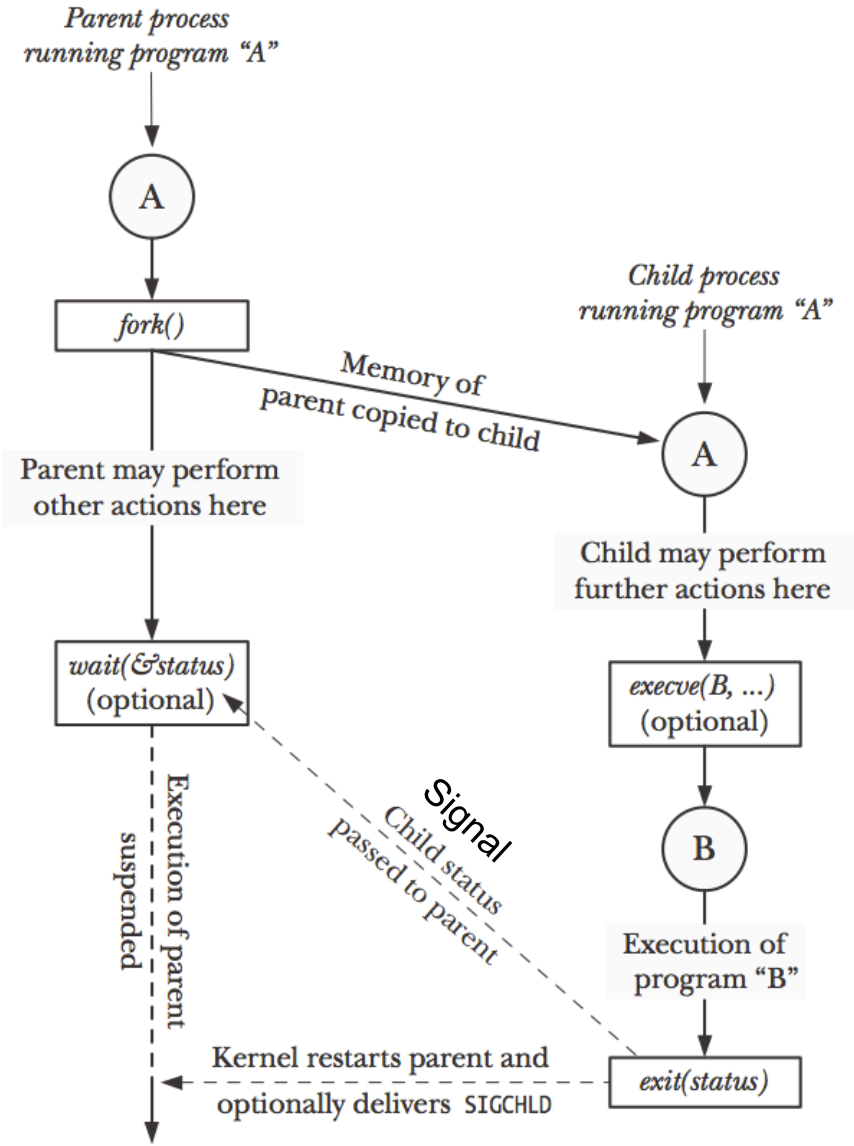


Figure 24-1: Overview of the use of *fork()*, *exit()*, *wait()*, and *execve()*

Sinais

Um sinal é uma notificação assíncrona enviada a um processo.

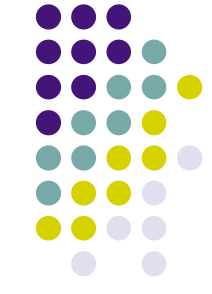
Sinais podem ser enviados:

- Entre processo de um mesmo grupo (com mesmo userID, real ou efetivo) ou processo de root
- Pelo núcleo para um processo (devido a ocorrência de um erro: divisão por zero, segmentation fault, etc.) ou como consequência de um comando do usuário CTRL-C), CTRL-\,

Em Unix existem aprox. 64 sinais: e.g. keyboard interrupt, erro em um processo, start/stop process, kill, eventos da rede, sinal de timer, etc.

Sinais podem ser tratados, bloqueados ou ignorados pelo processso alvo. O tratador pode ser um procedimento do próprio programa de usuário.

Chamadas de Sistema: File Management



Gerenciamento de arquivos

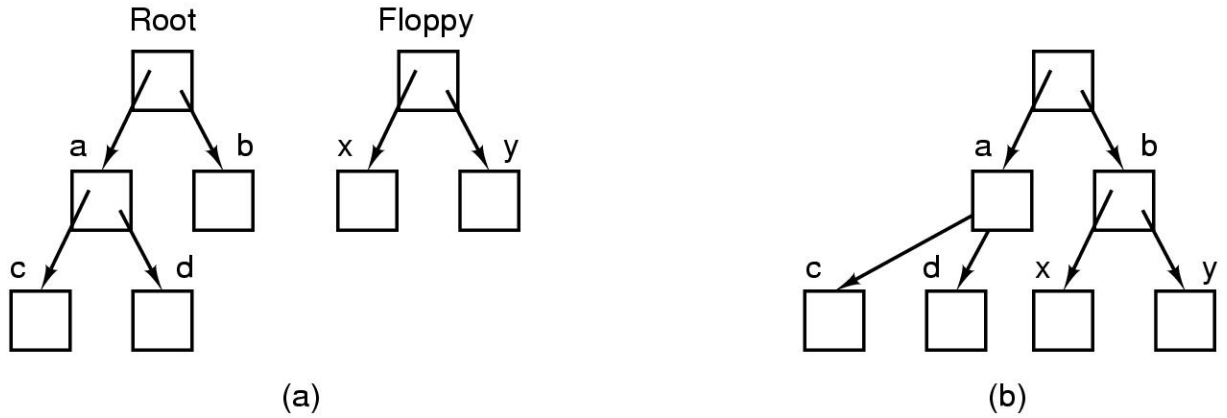
Chamada	Descrição
fd = open(file, how, ...)	Abra um arquivo para leitura, escrita ou ambas
s = close(fd)	Feche um arquivo aberto
n = read(fd, buffer, nbytes)	Leia dados de um arquivo para um buffer
n = write(fd, buffer, nbytes)	Escreva dados de um buffer para um arquivo
position = lseek(fd, offset, whence)	Mova o ponteiro de posição do arquivo
s = stat(name, &buf)	Obtenha a informação de estado do arquivo

Chamadas de Sistema: Directory Management

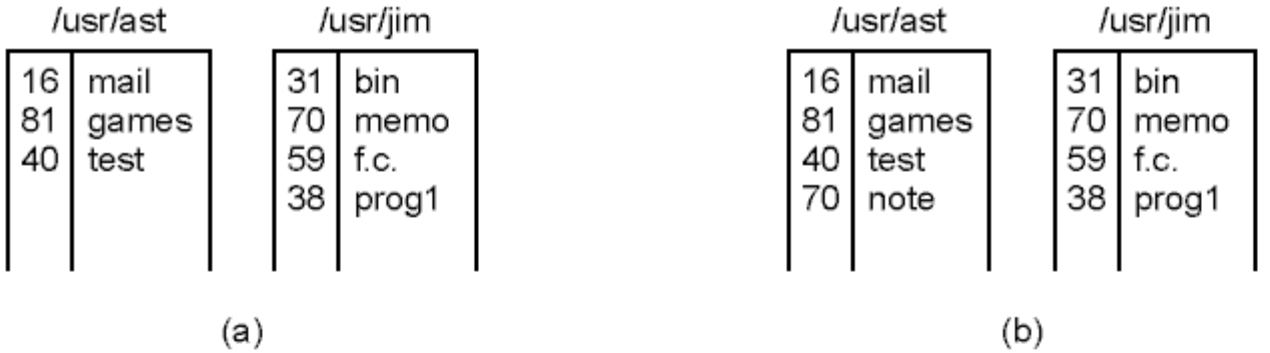
Gerenciamento do sistema de diretório e arquivo

Chamada	Descrição
s = mkdir(name,mode)	Crie um novo diretório
s = rmdir(name)	Remova um diretório vazio
s = link(name1, name2)	Crie uma nova entrada, name2, apontando para name1
s = unlink(name)	Remova uma entrada de diretório
s = mount(special,name, flag)	Monte um sistema de arquivo
s = umount(special)	Desmonte um sistema de arquivo

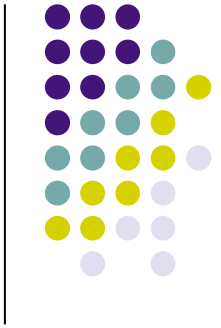
Conexão entre Sist. de Arquivos (montagem)



- **`mount /etc/floppy b`** – integra a árvore do disquete ao sistema principal
- **`umout b`** - desfaz a conexão

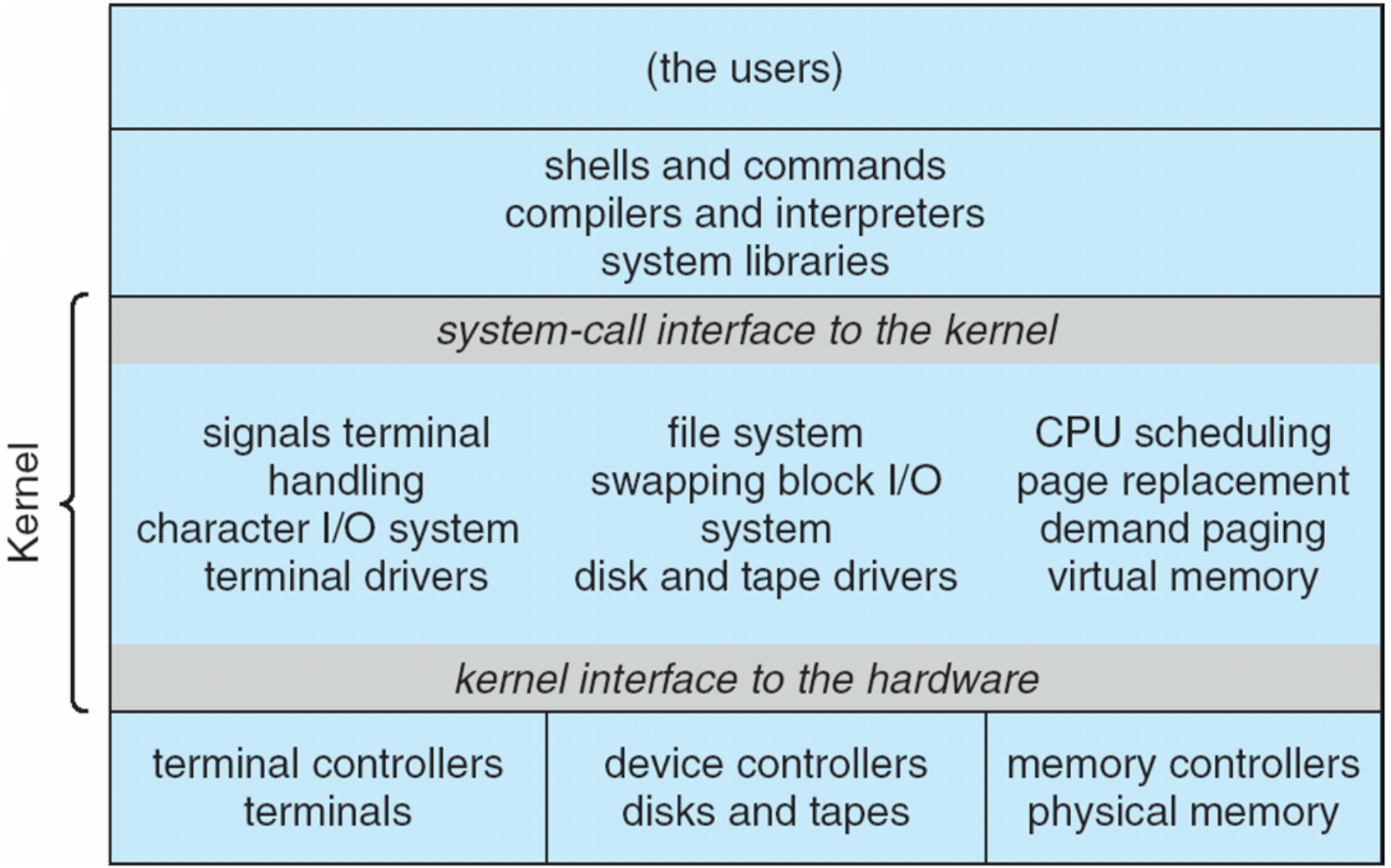


Chamadas de Sistema: Outras tarefas

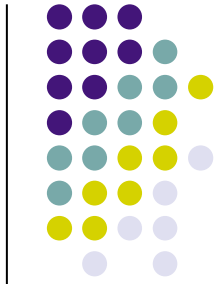


Diversas	
Chamada	**Descrição**
s = chdir(dirname)	Altere o diretório de trabalho
s = chmod(name, mode)	Altere os bits de proteção do arquivo
s = kill(pid, signal)	Envie um sinal a um processo
seconds = time(&seconds)	Obtenha o tempo decorrido desde 1º de janeiro de 1970

Estrutura Tradicional de UNIX



Tendências em Arquiteturas de S.O.



- Hierarquia de vários níveis garantem maior isolamento e reduzem complexidade
- Executar drivers e servidores em modo usuário
- Reduzir ao máximo parte dependente do Hardware (micro-núcleo)
- Virtualização do Hardware
- Sistemas de Arquivos heterogêneos e distribuídos

Perguntas?