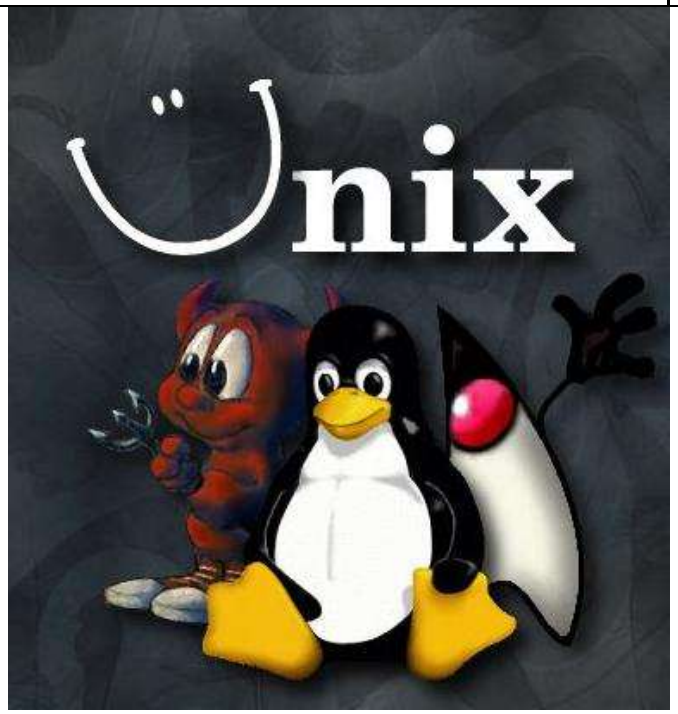
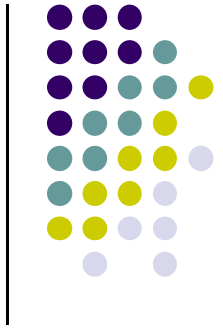
Sistemas de Computação

O Sistema Operacional Unix

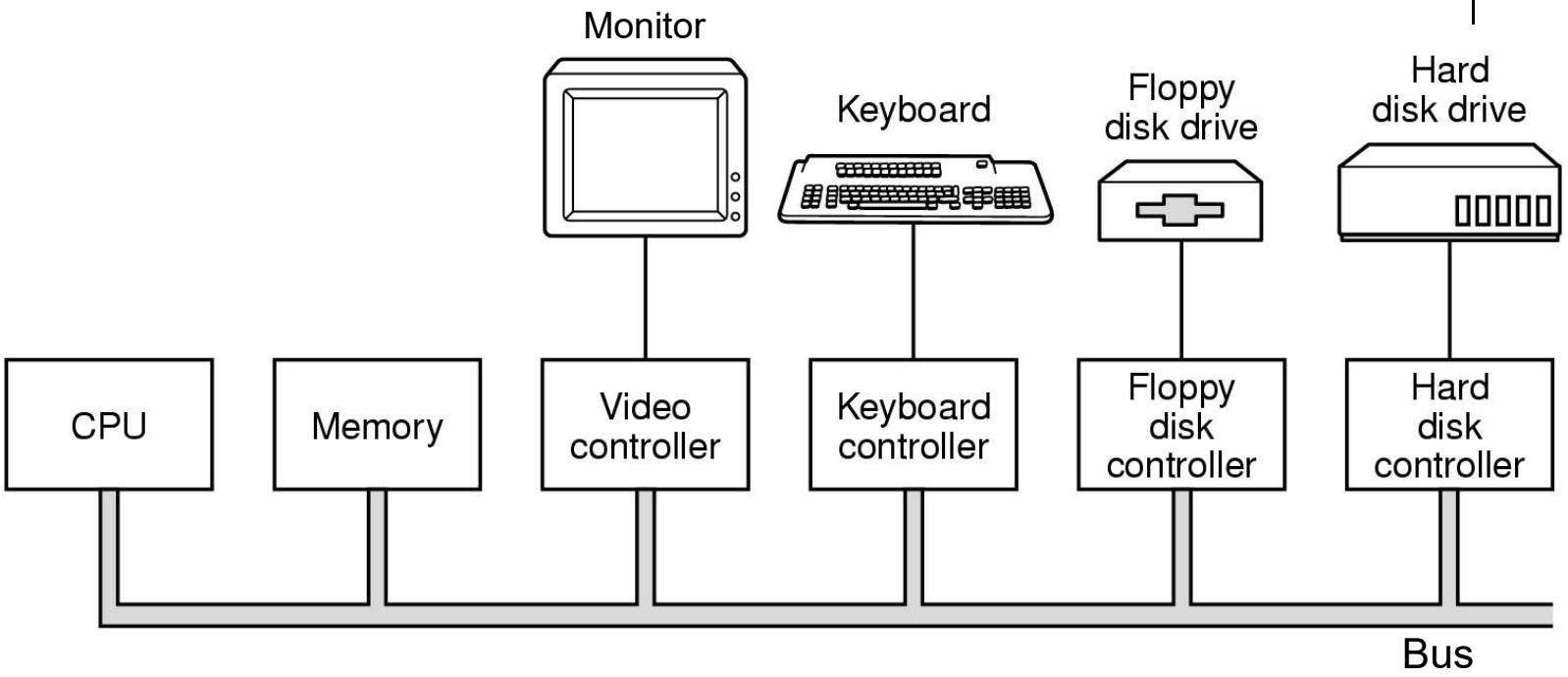


Introdução



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

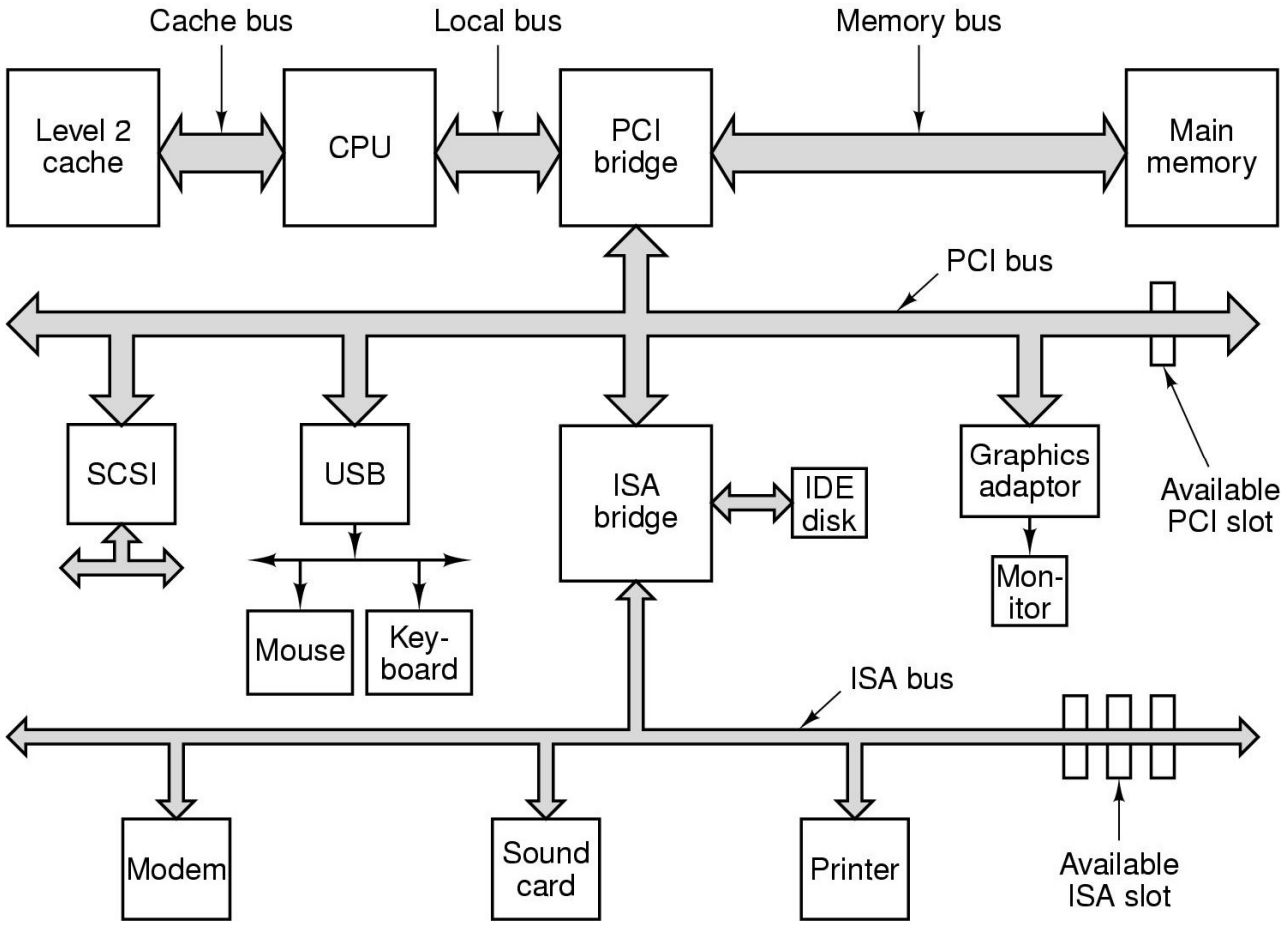
Visão simplificada do Hardware



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



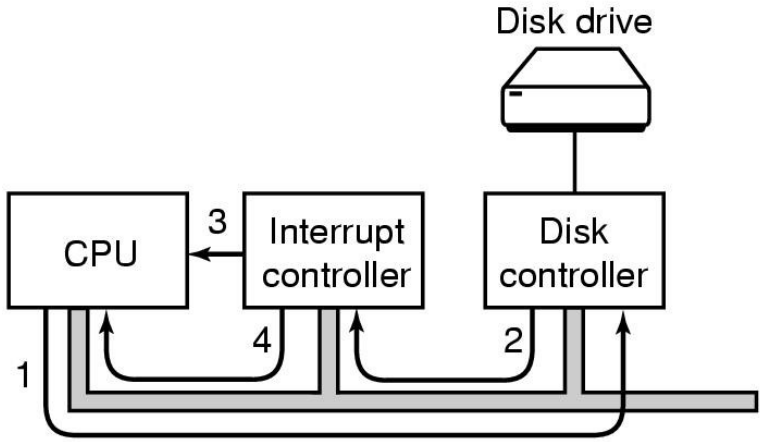
Barramentos com diferentes velocidades

Interação Dispositivos-CPU através de Interrupções de HW

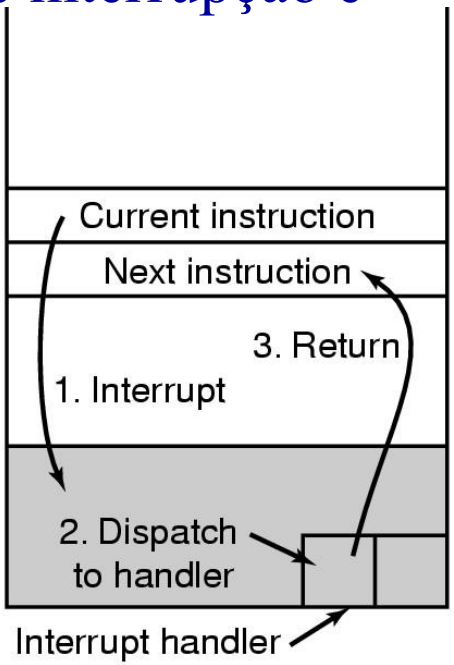
Processamento de Interrupções

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)

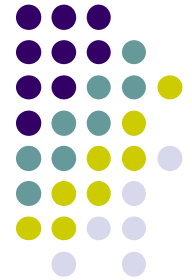


(b)

(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



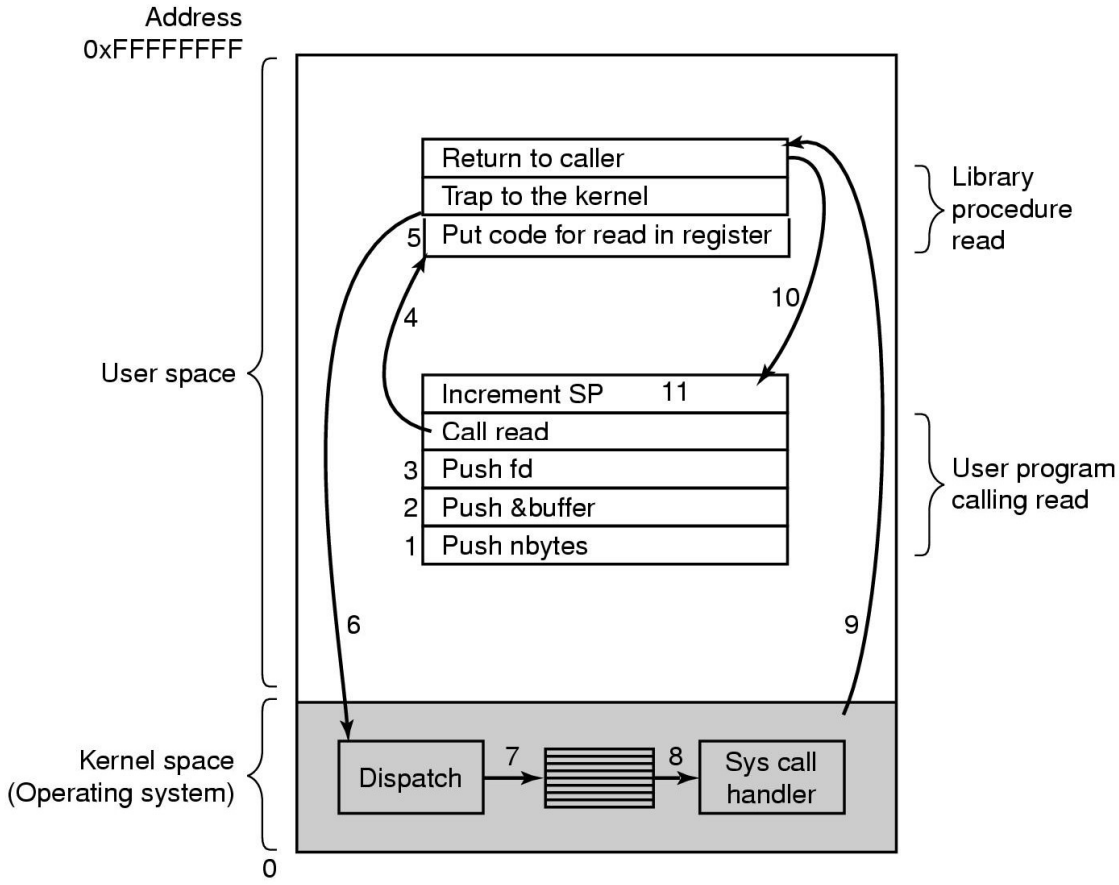
É o conjunto de funções/procedimentos disponibilizadas pelo núcleo (a API) para acesso a recursos da máquina (p.ex. Entrada/Saída)

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

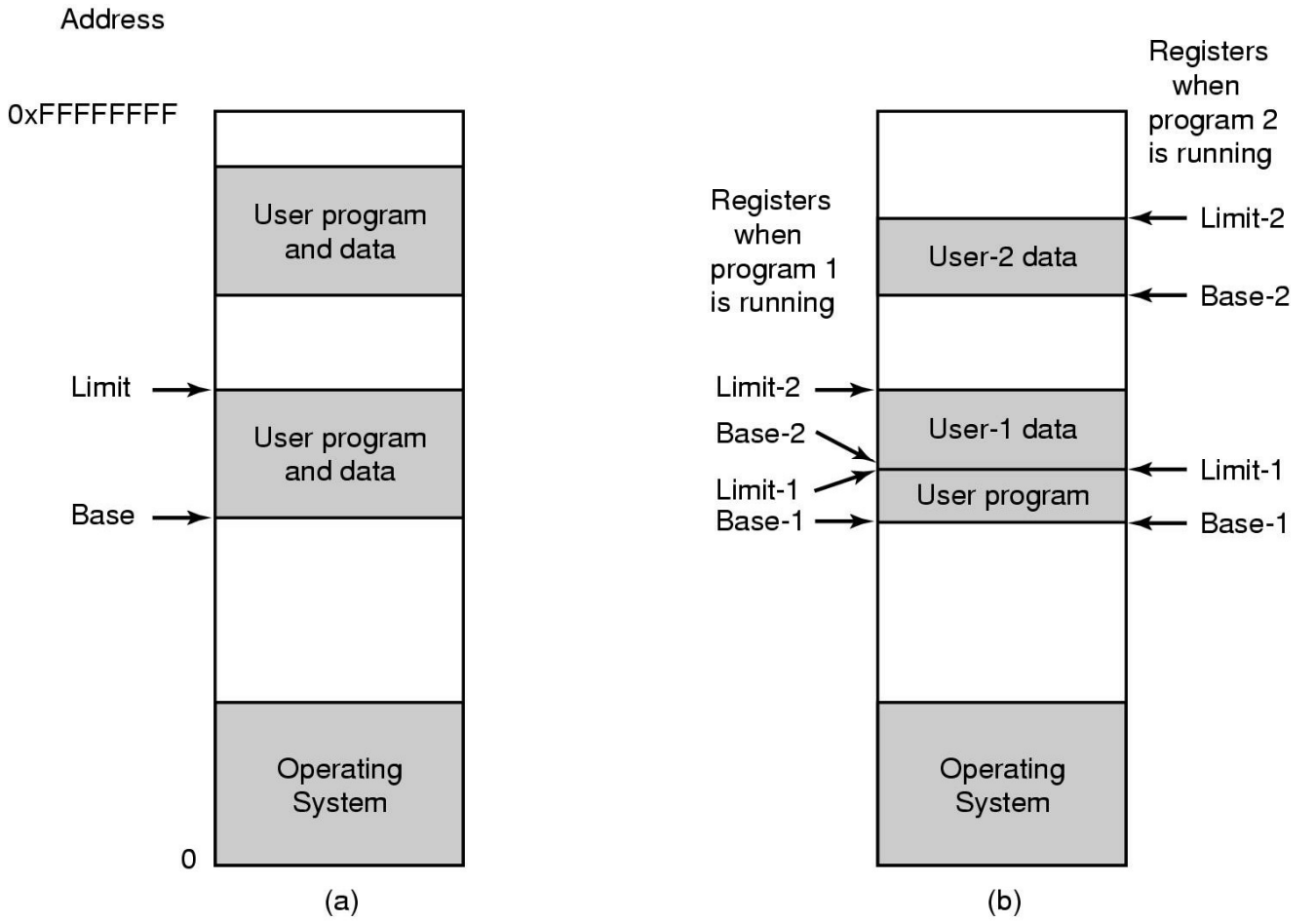
Exemplo: read (fd, buffer, nbytes)



Chamada de sistema = interrupção de software

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

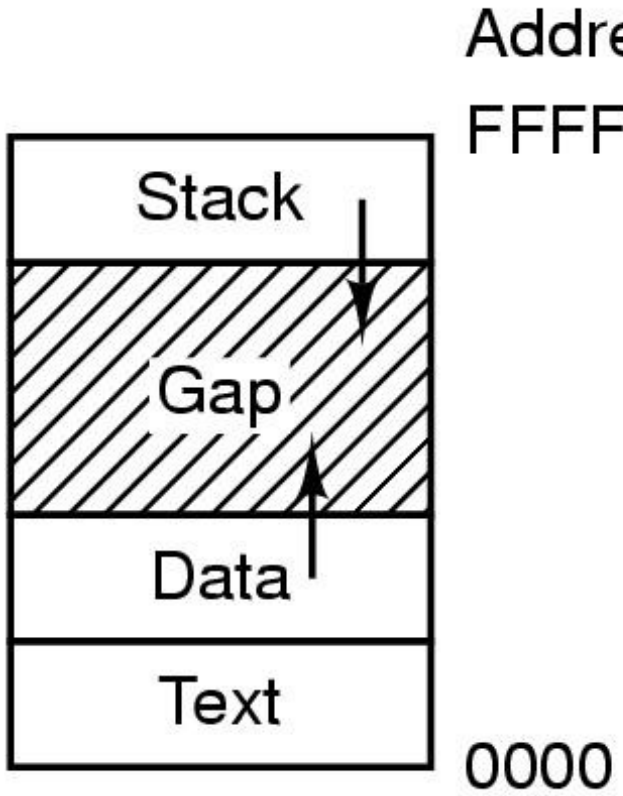
Ocupação da Memória



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

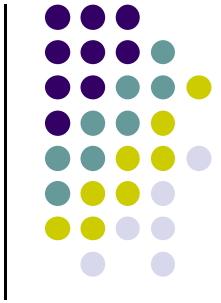
Pode ser em uma região contígua (a) ou não (b)

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.

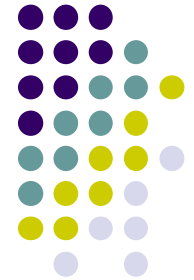
Chamadas de Sistema: Process Management



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

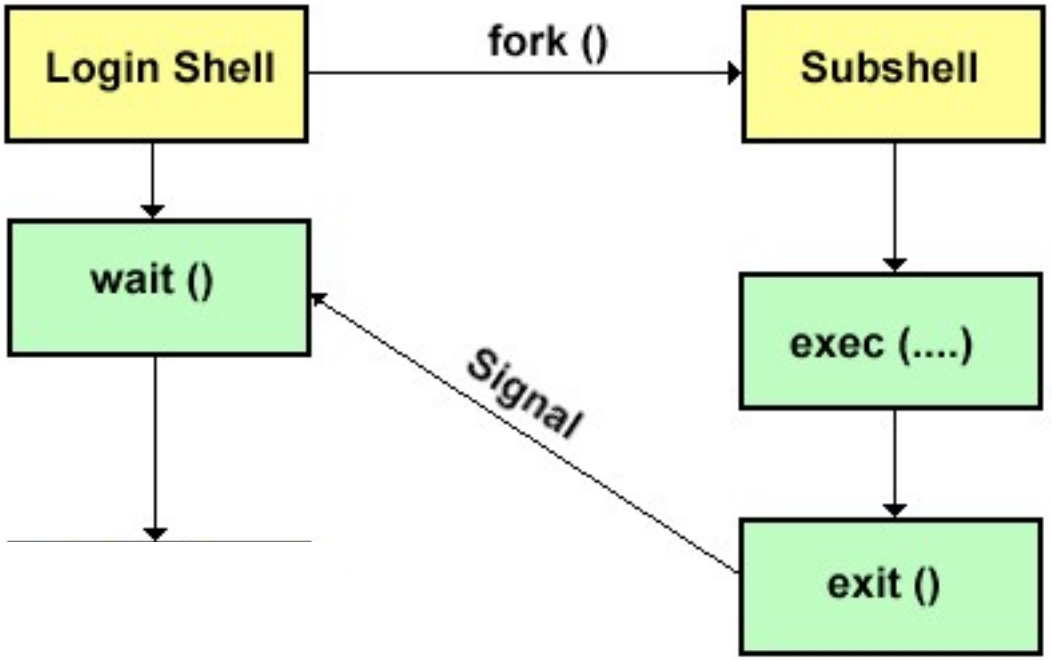
Esboço de uma shell



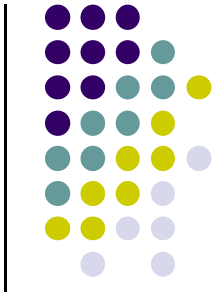
```
while (TRUE) {                                  /* repeat forever */
    type_prompt( );                             /* display prompt */
    read_command (command, parameters)          /* input from terminal */

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 */
 }
}
```

Chamada fork( ) / exec( )



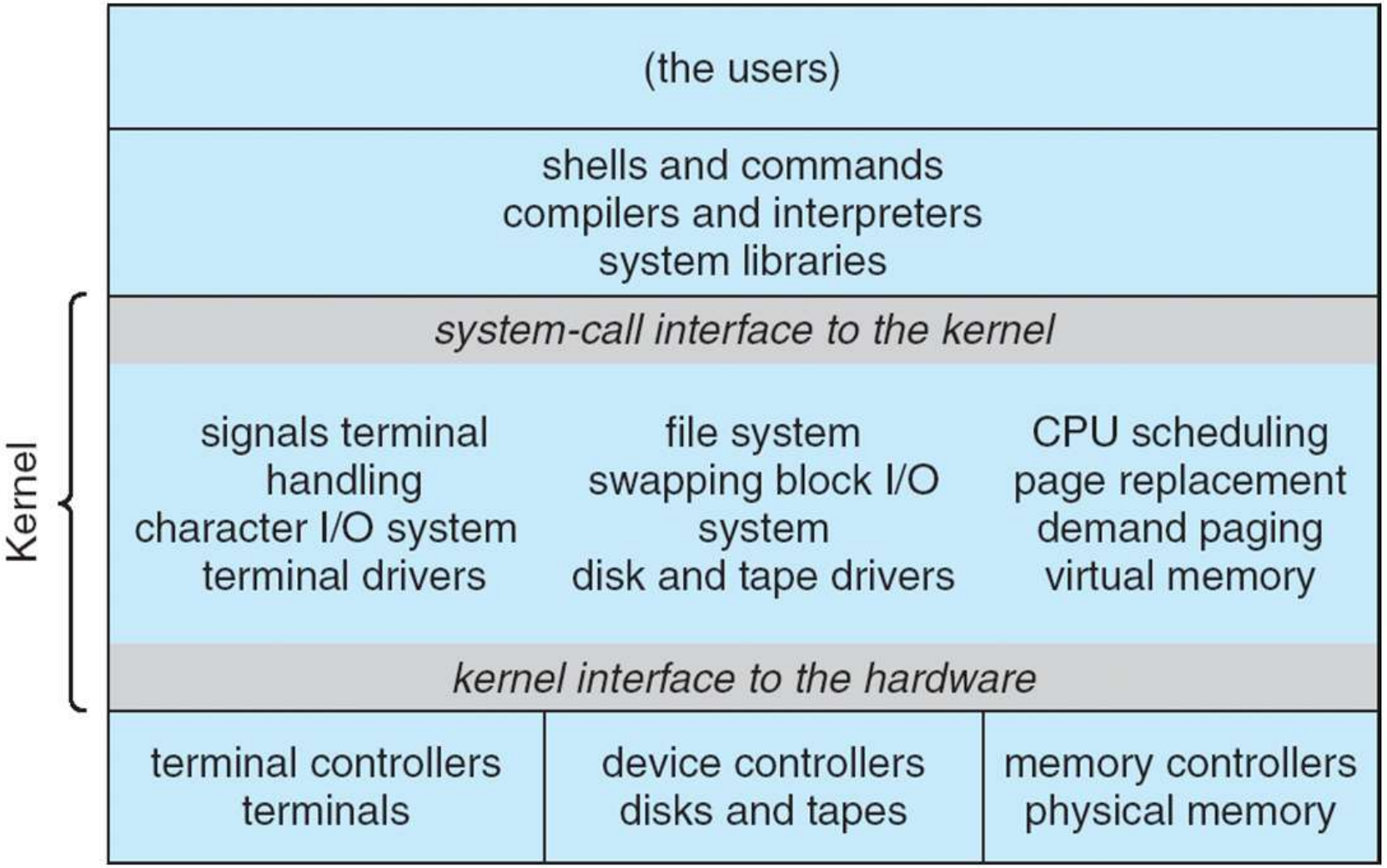
Chamadas de Sistema: File Management



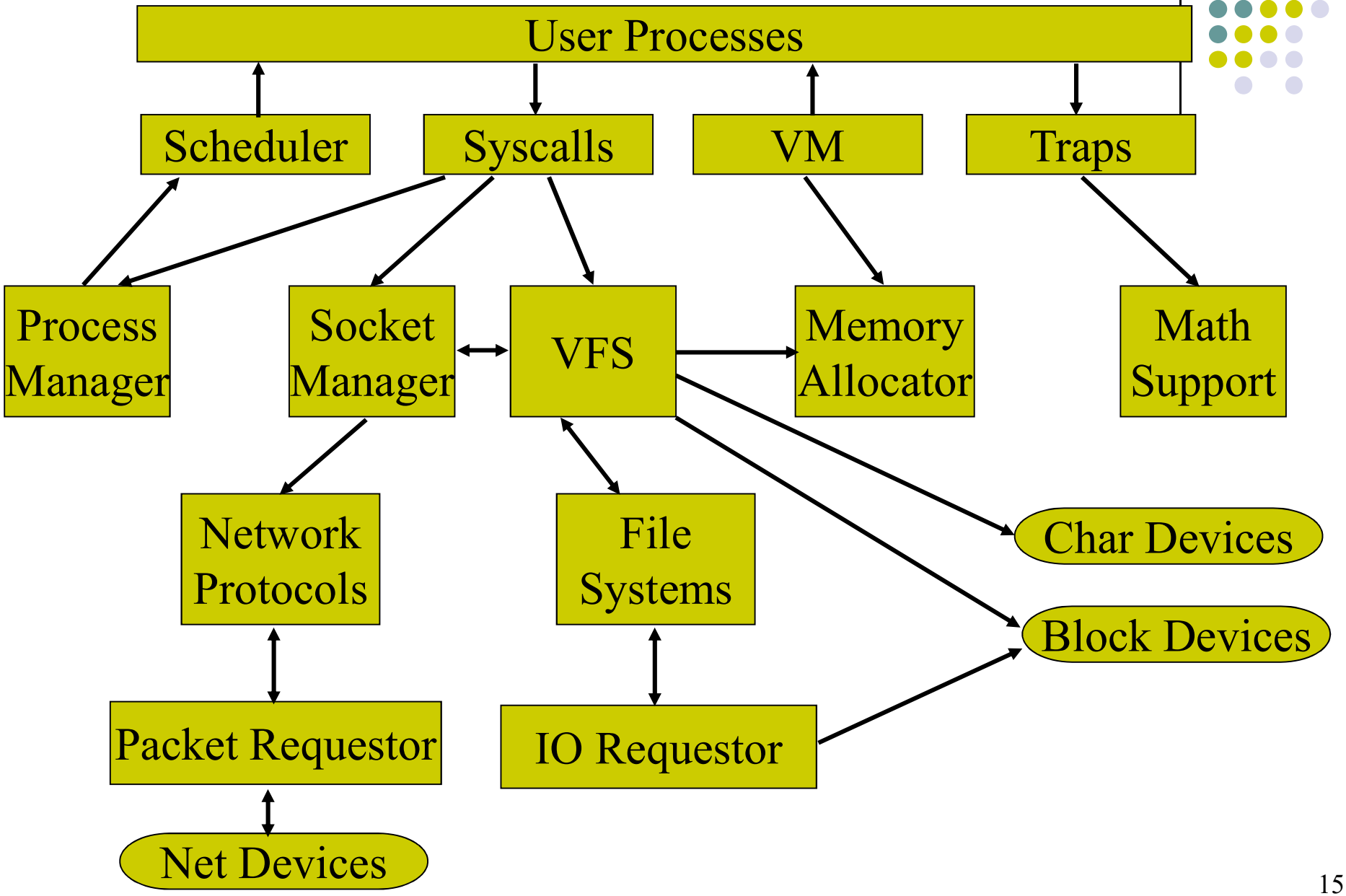
Gerenciamento de arquivos

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

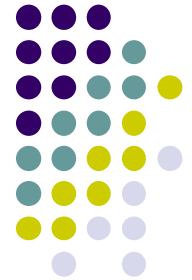
Estrutura Tradicional de UNIX



Estrutura do Linux



Conceitos básicos de UNIX



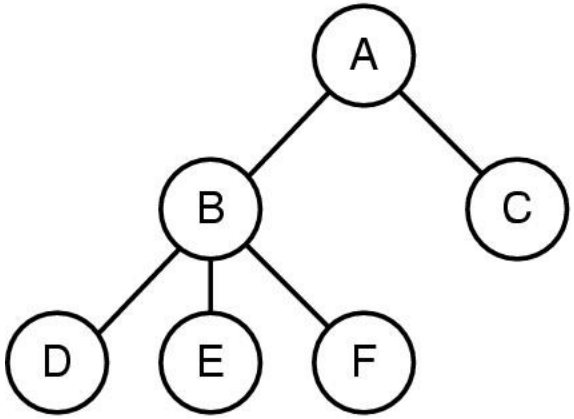
- Usuário invoca comandos (i.e. executa programas) através de interpretador de comandos em linha de comando (*shell*)
- Comandos podem ser executados em 1º ou 2º plano
- Shell tem o terminal como entrada e saida padrão (STDIN e STDOUT)
- A shell e os comandos herdam o UserID e um GroupID do programa login
- Programas acessam, criam e modificam arquivos, que estão organizados em uma hierarquia, e têm associados a eles atributos e permissões
- Dispositivos de E/S (impressora, pen-drive, cdrom…) são representados como arquivos
- Pode-se alterar permissões dos próprios arquivos e diretório

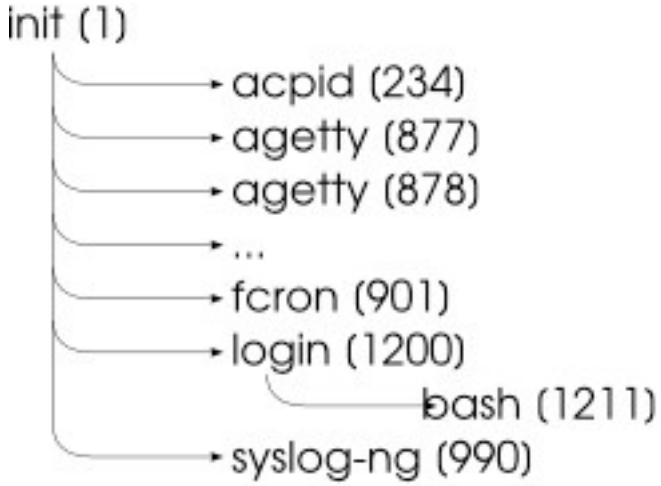
Processos

Um processo é um programa em execução. Cada comando da shell é um programa.

Na inicialização do sistema, alguns processos do sistema são criados (mas executam em modo usuário).

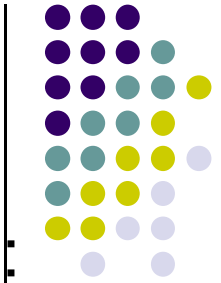
Processo pai pode ou não esperar pelo término do processo filho (se não esperar, diz-se que processo filho executa em 2º. plano)





- Exemplo:
  - A = init, B = /etc/rc (script que inicia diversos drivers)
  - C= login → shell → comandos do usuário
- Maioria dos S.O. executa vários processos concorrentes

Redirecionamento de E/S

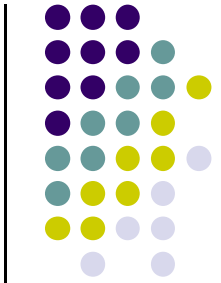


Todos os programas em primeiro plano possuem:

- **STDOUT** (standard output), default é a tela
- **STDIN** (standard input), default é o teclado
- **STDERR** (standard error), default é também a tela

- Redirecionamento de STDIN/STDOUT. Exemplos:
  - **`who > file`**,
  - **`date >> arq`**,
  - **`sort < infile > outfile`**
- Pipe é uma conexão entre STDOUT e STDIN de dois processos filho
  - Ex: **`who | sort`** - saida de **`who`** será exibida ordenada na tela

Variáveis de Ambiente



A shell acessa uma série de variáveis do ambiente, que definem o comportamento dos programas/comandos executados. Uma das principais é a $PATH, $CLASSPATH

```
>echo $PATH
/home/d/da/darin/bin:/opt/local/bin:/opt/local/bin/pbmutil
   s:/usr/bin:/usr/sbin:/opt/SUNWspro/bin:/usr/ccs/bin:/op
   t/local/X11/bin:/usr/dt/bin:/usr/openwin/bin:/opt/local
   /gnu/bin:/opt/local/games/bin:/usr/ucb:./
```

Outras variáveis:

- HOST what computer you're logged into
- PAGER program used display man pages
- PWD current directory
- GROUP what group you're in
- USER your login

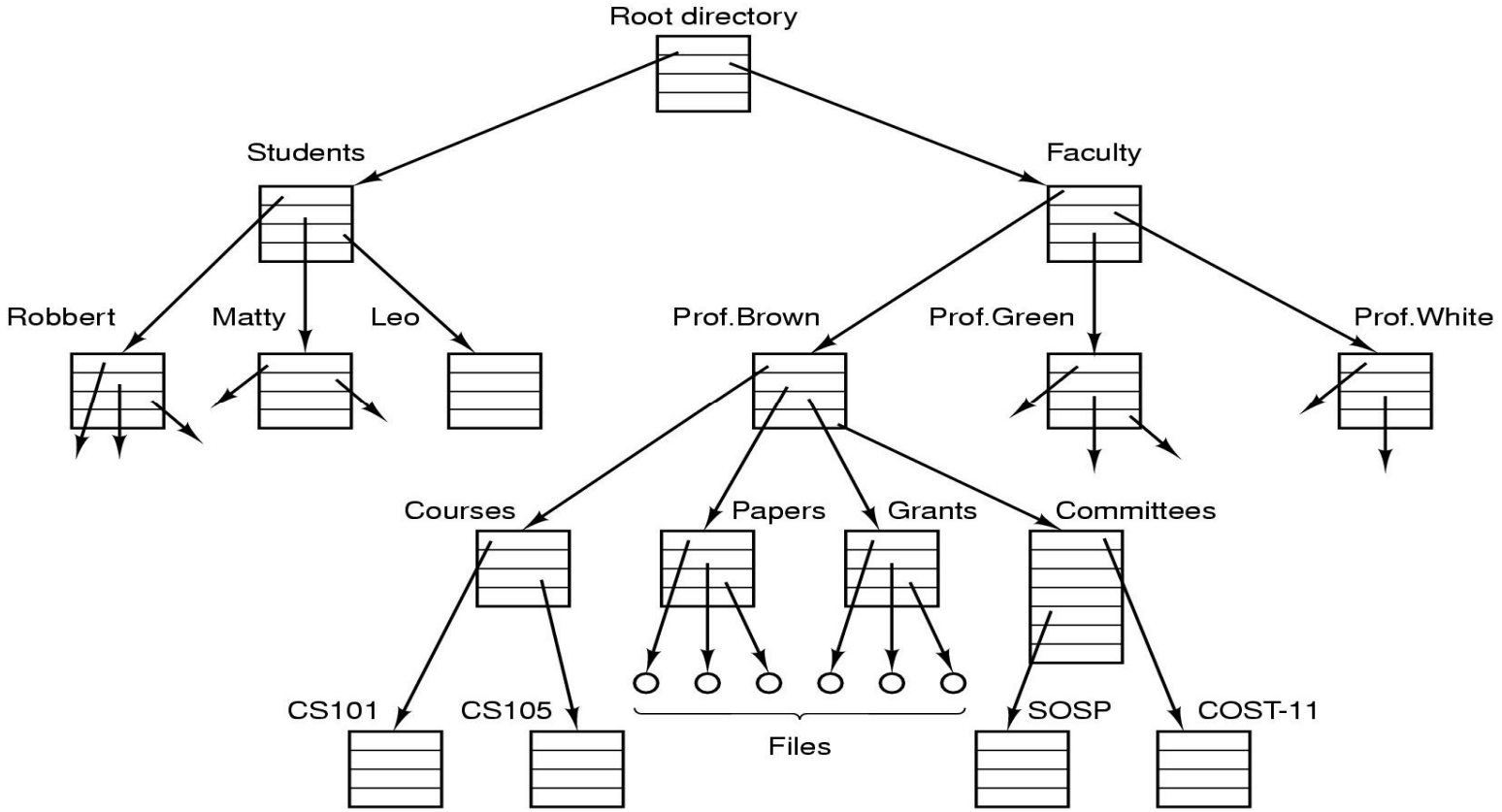
É possível criar novas variáveis e/ou modificá-las, e.g. `setenv` ou `set` (comando específico, depende da shell)

Sistema de Arquivos

Arquivo = conceito abstrato para *repositório durável* de informação binária, que é *independente do meio*

Além do conteúdo, arquivo possui uma série de atributos (metadados)

Diretórios/Pastas: agrupamentos de arquivos e sub-diretórios



Sistema de Arquivos: alguns comandos

- **`pwd`** - report your current directory
- **`cd`** <*dir*> - change your current directory
- **`ls`** <*dir*> - list contents of directory
- **`ls`** -a/l - list also dot-files (-a), list all details (-l)
- **`cp`** <*old file*> <*new file*> - copy
- **`mv`** <*old file*> <*new file*> - move (or rename)
- **`rm`** <*file*> - delete a file
- **`mkdir`** <*new dir name*> - make a directory
- **`rmdir`** <*dir*> - remove an empty directory
- **`rm -r`** <dir> - recursive removal
- **`cp -r`** <dir> - recursive copy

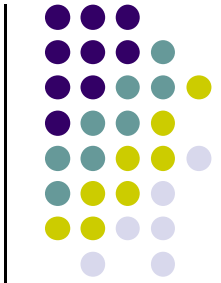
Filosofia UNIX é de ser um SO extensível: quase todos os comandos são programas utilitários

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

Usuários & Grupos



- Unix permite o cadastro de vários usuários, através do administrador (username **`root`**)
  - Lista de usuários contida em **`/etc/passwd`** com senha cifrada
- Cada user pode pertencer a 1 ou mais grupos

  `useradd, groupadd, groupmod, grouprem`
- A cada instante, um usuário está em um grupo, e pode trocar usando **`newgrp`**

  `groups` *user* lista todos os grupos aos quais **`user`** pertence

  Ex: **`root`** pertence aos grupos **`bin, daemon, sys, adm, disk,...`**
- Permissões sobre os arquivos estão definidas em termos do dono, seu grupo corrente e os outros (**`bits rwx`**)
- Cada processo criado recebe o **`userID`** e **`groupID`** do seu processo pai

Permissões de Arquivos e Diretórios

Cada arquivo/diretório tem atributos de controle de acesso:

Bits rwx rwx rwx (relativos ao dono, seu grupo e outros), onde “1/0” significam com/sem permissão

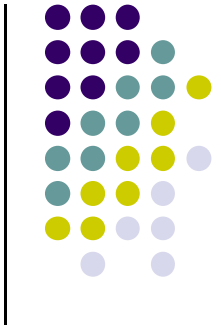
`chmod g+r` <*filename*> - acrescentando direito de leitura aos membros do grupo

`chmod 744` <*filename*>- direitos plenos apenas para dono, demais apenas leitura

`chown` <*user*> <*filename*> - mudando o dono do arquivo

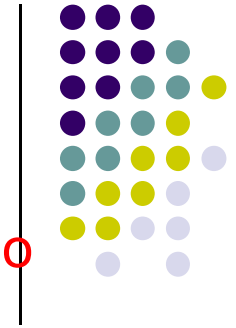
`chgrp` <*group*> <*filename*> - alterando a qual grupo o arquivo pertence

Chamadas de Sistema: Outras tarefas



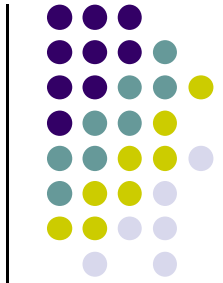
Diversas	
Chamada	**Descrição**
s = chdir(dirname)	Altere o diretório de trabalho
s = chmod(name, mode)	Altere os bits de proteção do arquivo
s = kill(pid, signal)	Envie um sinal a um processo
seconds = time(&seconds)	Obtenha o tempo decorrido desde 1º de janeiro de 1970

Outros comandos úteis



- **`grep`** <*expr*> <*arq*> - busca as linhas que contém <*expr*> no arquivo
- **`diff`** <*arq1*> <*arq2*> - mostra as diferenças entre o conteúdo dos dois arquivos
- **`vi`** OU **`pine`** - editores simples de arquivos
- **`sed , awk`** – filtros/ processadores de texto
- **`Perl`** - linguagem para scripts
- **`make`** – automatizar a geração de programas a partir de vários componentes interdependentes (dependências em Makefile), e verificando se algum foi modificado
- **`man`** <*termo*> – manual on-line
- **`ftp`** <*user@rem-host*> - transferência de arquivos
- **`rsh`** <*user@rem-host*> - login remoto

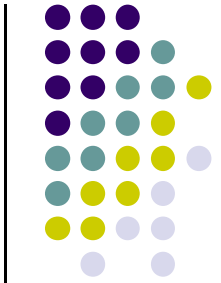
Bibliografia



Para mais detalhes sobre o uso de Unix consulte os livros:

- “Aprenda em 24 horas – Programação para Unix” Warren Gay
- “The Unix Programming environment”, Kerninghan & Pike
- (ou páginas similares na Web)

Tendências em Arquiteturas de S.O.



- Hierarquia de vários níveis garantem maior isolamento e reduzem complexidade (cebola)
- 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?