Infra-Estrutura de Software

IF677

Alguns Conceitos Importantes

Tópicos

- Processo
- Página
- Memória virtual
- Escalonamento
- Interrupção
- Chamada ao sistema

Hardware

CPU

Register file

PC

Bus interfac

ry bus E/S

Main memory

Proces-sador

Memória

Disco

Graphics adapter

Display

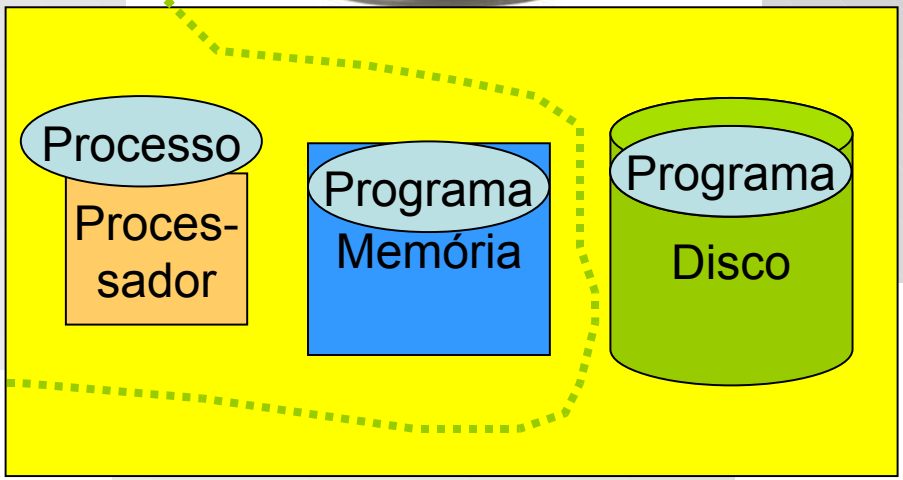
cont

Software

Como rodar um programa?

E/S

O conceito de "Processo"

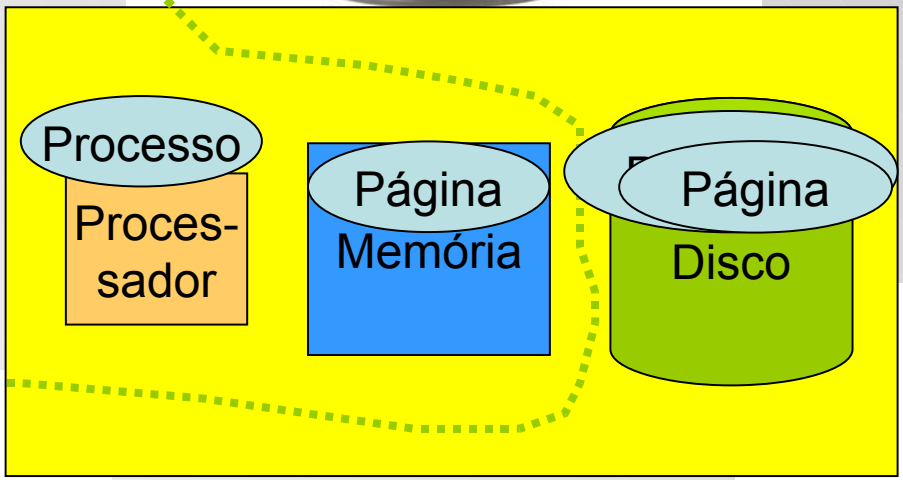


Software

E se o programa for maior do que o espaço de memória disponível?

E/S

O conceito de "Página" e "Memória Virtual"

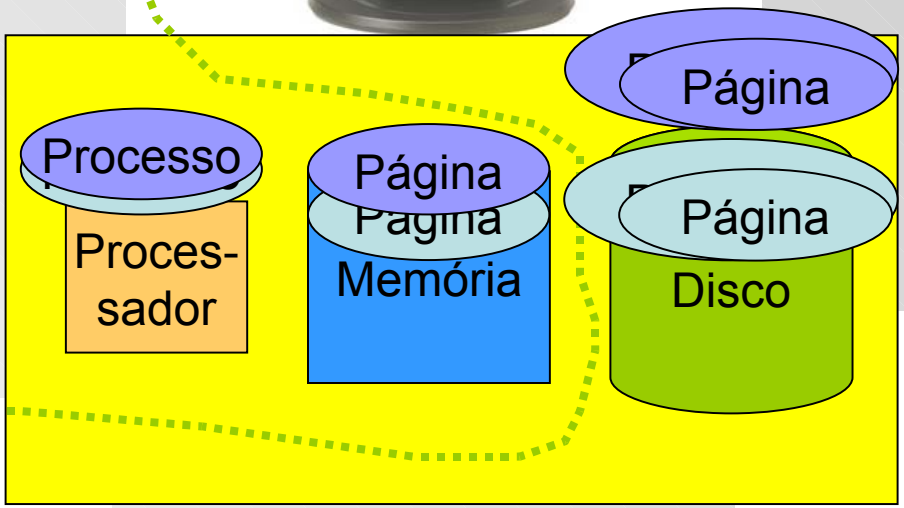


Software

Como rodar mais de um programa?

E/S

O conceito de "Interrupção" e "Escalonamento"



Software

Como rodar mais de um programa em **máquinas diferentes**, formando um **sistema distribuído**?

conceito

Parte II…

Interrupção

O Elo Hardware-Software

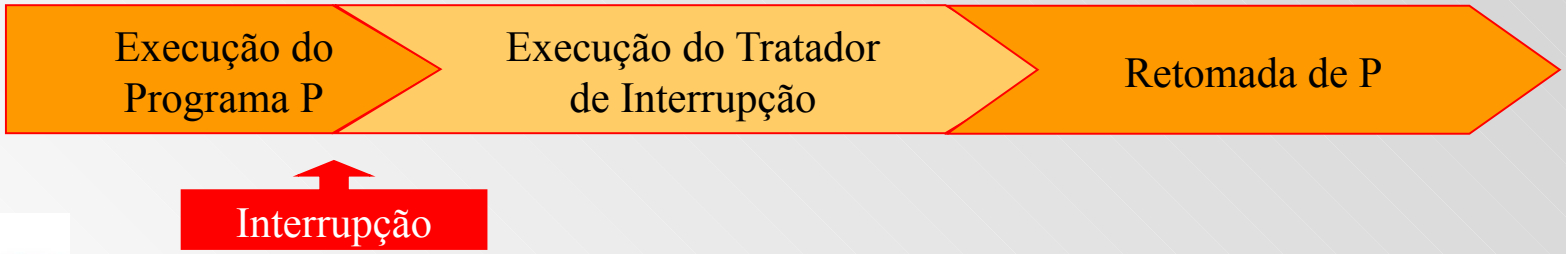
Motivação

- Para controlar entrada e saída de dados, **não é interessante** que a CPU monitore continuamente os dispositivos
  - discos ou teclados
  - **Por que** não é interessante?

- Interrupções permitem que o hardware "chame a atenção" da CPU quando há algo a ser feito

Interrupções de Hardware

- Quando geradas por um dispositivo externo à CPU são chamadas de interrupções de hardware ou *assíncronas*
  - controlador de disco ou teclado
  - independem das instruções que a CPU está executando
- A CPU interrompe o processamento do programa em execução e executa um tratador de interrupção
  - Tipicamente parte do sistema operacional
  - O programa interrompido e o tratador não se comunicam
  - Em geral, após a execução do tratador, a CPU volta a executar o programa interrompido



Interrupção de Relógio

(Um tipo de Interrupção de HW)

- O sistema operacional atribui *quotas de tempos de execução* (*quantum* ou *time slice* – fatias de tempo) para cada um dos processos
  - em um sistema com *multiprogramação*

- A cada interrupção do relógio, o tratador verifica se a fatia de tempo do processo em execução já se esgotou
  - se for esse o caso, suspende-o e aciona o *escalonador* para que escolha outro processo para executar

Interrupções Síncronas ou *Traps*

- *Traps* ocorrem em consequência da instrução sendo executada [no programa em execução]

- Algumas são geradas pelo *hardware*, para indicar, por exemplo, *overflow* em operações aritméticas ou acesso a regiões de memória não permitidas
  - Situações em que o programa não teria como prosseguir
  - O hardware sinaliza uma interrupção para passar o controle para o tratador da interrupção (no só)
    - tipicamente termina a execução do programa

Traps (cont.)

- *Traps* também podem ser geradas explicitamente por instruções do programa
  - Chaveamento entre modo usuário e modo núcleo
  - Forma do programa acionar o sistema operacional, por exemplo, para requisitar um serviço de entrada ou saída
    - Um processo pode ativar um tratador que pode "encaminhar" uma chamada ao sistema operacional
- Programas podem passar parâmetros para o tratador da *trap*
  - Exemplos?

Interrupções

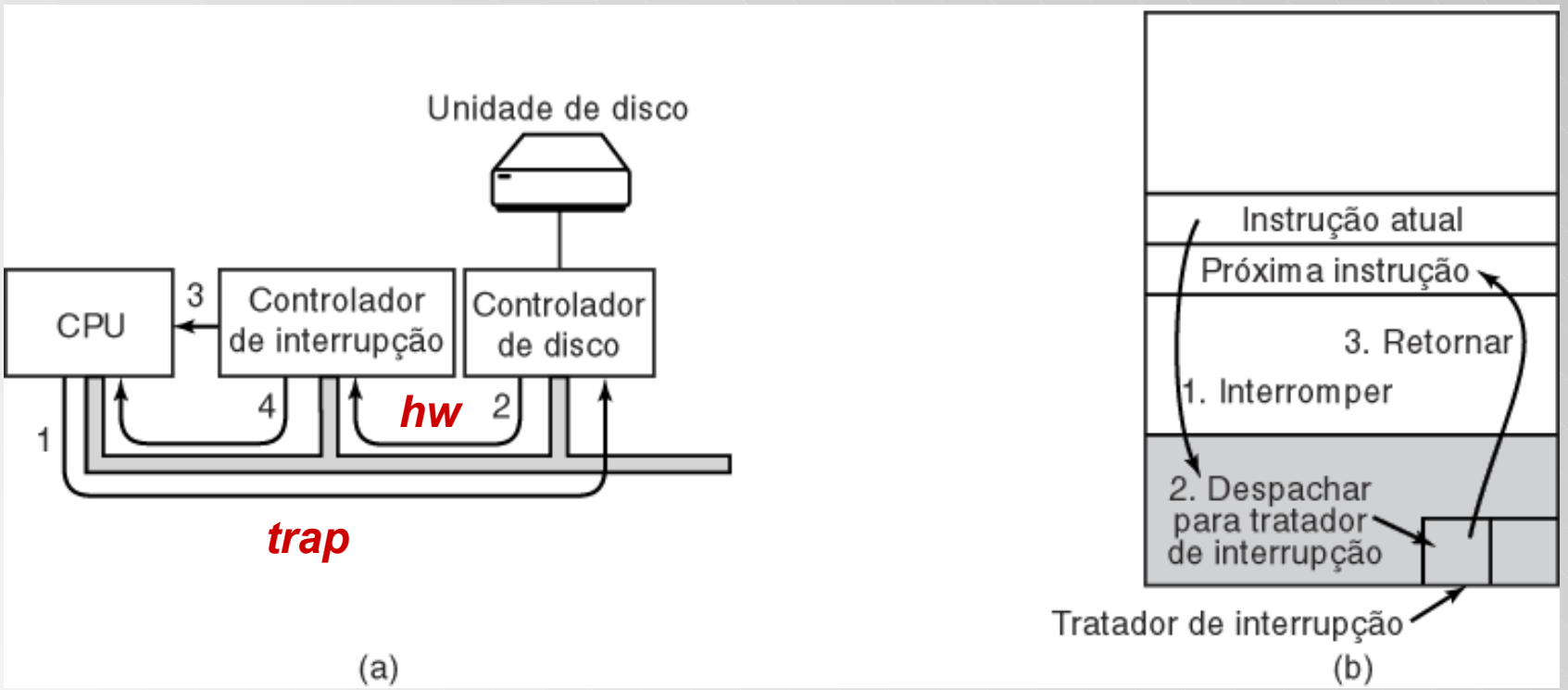
Assíncronas (hardware)

- geradas por algum dispositivo externo à CPU
- ocorrem independentemente das instruções que a CPU está executando
- não há qualquer comunicação entre o programa interrompido e o tratador
- Exemplos:
  - interrupção de relógio, quando um processo esgotou a sua fatia de tempo (*time slice*)
  - teclado, para uma operação de E/S (neste caso, de Entrada)

Síncronas (*traps*)

- Geradas pelo programa em execução, em consequência da instrução sendo executada
- Algumas são geradas pelo hardware: situações em que o programa não teria como prosseguir
- O programa pode passar parâmetros para o tratador
- Exs.: READ, *overflow* em operações aritméticas ou acesso a regiões de memória não permitidas

Traps e interrupções de hardware



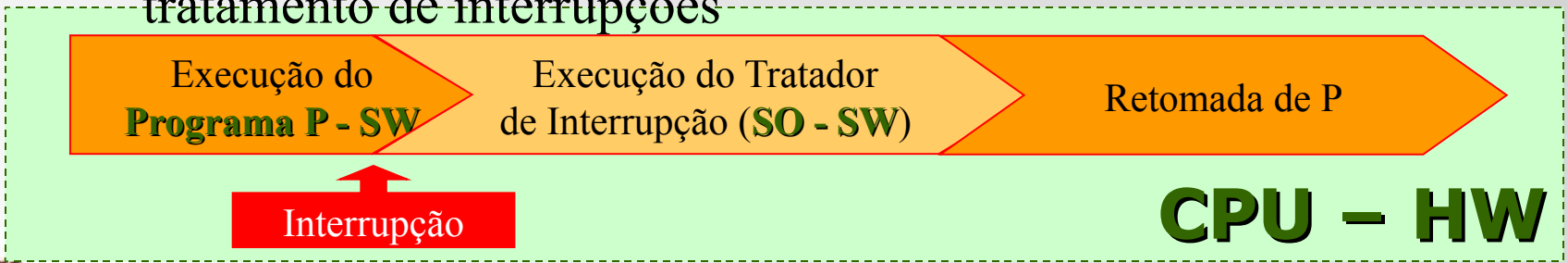
- Passos para iniciar um dispositivo de E/S e obter uma interrupção
- Como a CPU é interrompida

Interrupção: Suporte de HW

- Tipicamente, o hardware detecta que ocorreu uma interrupção,
  - aguarda o final da execução da instrução corrente
  - salva o contexto de execução do processo interrompido
  - aciona o tratador

Interrupção: Suporte de HW

- Tipicamente, o hardware detecta que ocorreu uma interrupção,
  - aguarda o final da execução da instrução corrente
  - salva o contexto de execução do processo interrompido
- Para poder reiniciar o processo mais tarde, é necessário salvar o *program counter* (PC) e outros registradores de status
  - Os registradores com dados do programa devem ser salvos pelo próprio tratador (ou seja, por software), que em geral os utiliza
  - Para isso, existe uma pilha independente associada ao tratamento de interrupções



Interrupção: Passo-a-passo

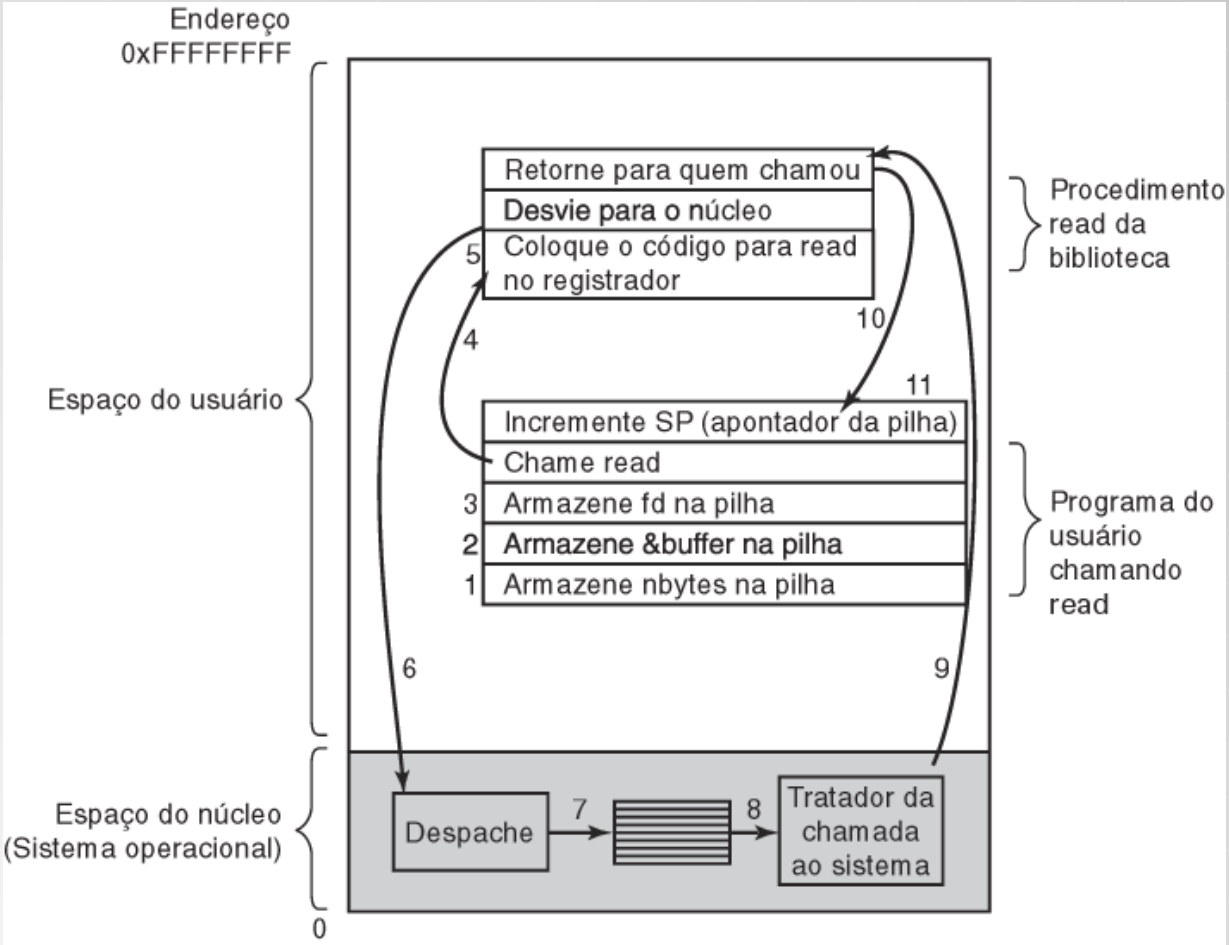
1. O hardware empilha o contador de programa etc.
2. O hardware carrega o novo contador de programa a partir do vetor de interrupção.
3. O procedimento em linguagem de montagem salva os registradores.
4. O procedimento em linguagem de montagem configura uma nova pilha.
5. O serviço de interrupção em C executa (em geral lê e armazena temporariamente a entrada).
6. O escalonador decide qual processo é o próximo a executar.
7. O procedimento em C retorna para o código em linguagem de montagem.
8. O procedimento em linguagem de montagem inicia o novo processo atual.

Esqueleto do que o nível mais baixo do SO faz quando ocorre uma interrupção

Chamadas ao Sistema

(*System Calls*)

Passos de uma Chamada ao Sistema



Os 11 passos para fazer uma chamada ao sistema

Ex. `read (fd, buffer, nbytes)`

Centro de Informática UFPE

Algumas Chamadas ao Sistema para Gerenciamento de Processos

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

Algumas Chamadas ao Sistema para Gerenciamento de Arquivos

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

Algumas Chamadas ao Sistema para Gerenciamento de Diretório

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

Algumas Chamadas ao Sistema para Tarefas Diversas

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

Chamadas ao Sistema

- 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 ao Sistema

Unix	Win32	Descrição
fork	CreateProcess	Crie um novo processo
waitpid	WaitForSingleObject	Pode esperar um processo sair
execve	(none)	CrieProcesso = fork + execve
exit	ExitProcess	Termine a execução
open	CreateFile	Crie um arquivo ou abra um arquivo existente
close	CloseHandle	Feche um arquivo
read	ReadFile	Leia dados de um arquivo
write	WriteFile	Escreva dados para um arquivo
lseek	SetFilePointer	Mova o ponteiro de posição do arquivo
stat	GetFileAttributesEx	Obtenha os atributos do arquivo
mkdir	CreateDirectory	Crie um novo diretório
rmdir	RemoveDirectory	Remova um diretório vazio
link	(none)	Win32 não suporta ligações (link)
unlink	DeleteFile	Destrua um arquivo existente
mount	(none)	Win32 não suporta mount
umount	(none)	Win32 não suporta mount
chdir	SetCurrentDirectory	Altere o diretório de trabalho atual
chmod	(none)	Win32 não suporta segurança (embora NT suporte)
kill	(none)	Win32 não suporta sinais
time	GetLocalTime	Obtenha o horário atual

Algumas chamadas da interface API Win32

Conceitos

- **Processo**: um programa em execução
- **Página**: parte de um programa capaz de caber na memória
- **Memória virtual**: espaço de armazenamento de páginas em disco
- **Escalonamento**: quando um ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado

Conceitos

- **Processo**: um programa em execução
- **Página**: parte de um programa capaz de caber na memória
- **Memória virtual**: espaço de armazenamento de páginas em disco
- **Escalonamento**: quando um ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado

- **Interrupção**
  - **Por hardware**
    - Algum dispositivo externo à CPU (ex. teclado)
    - Relógio (para suspender um processo)
  - **Por software (*trap*)**
    - Execução de intrução de programa (ex. READ)
    - situações em que o programa não teria como prosseguir (ex. *overflow* em operações aritméticas)
- **Chamadas ao sistema** formam a interface entre o SO e os programas de usuário