Sistemas Operacionais
Processos

Carlos Ferraz (cagf@cin.ufpe.br)
Jorge Cavalcanti Fonsêca (jcbf@cin.ufpe.br)

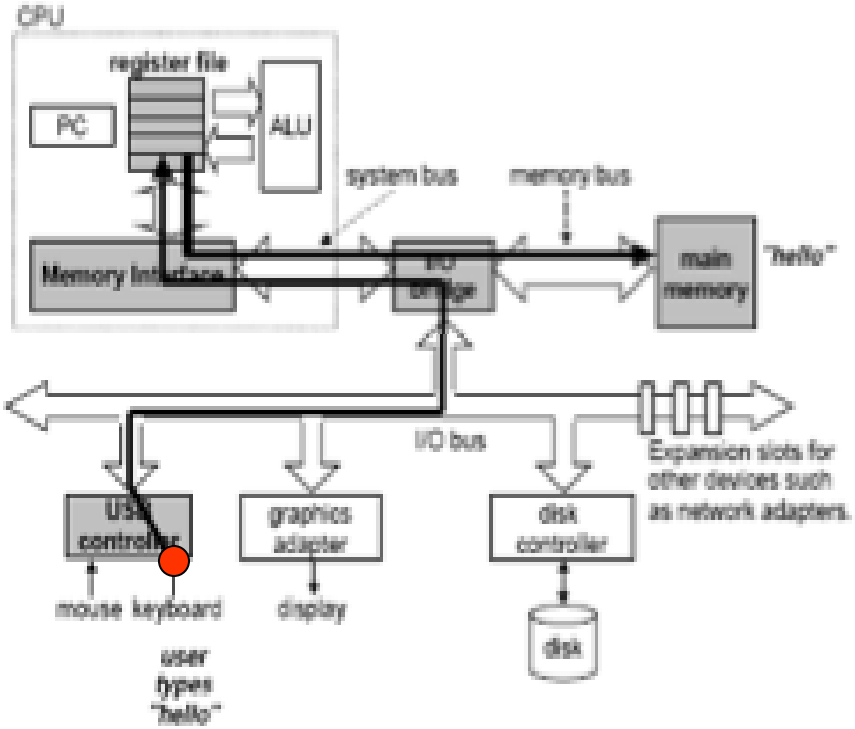
Copyright

Carlos Ferraz

Processo

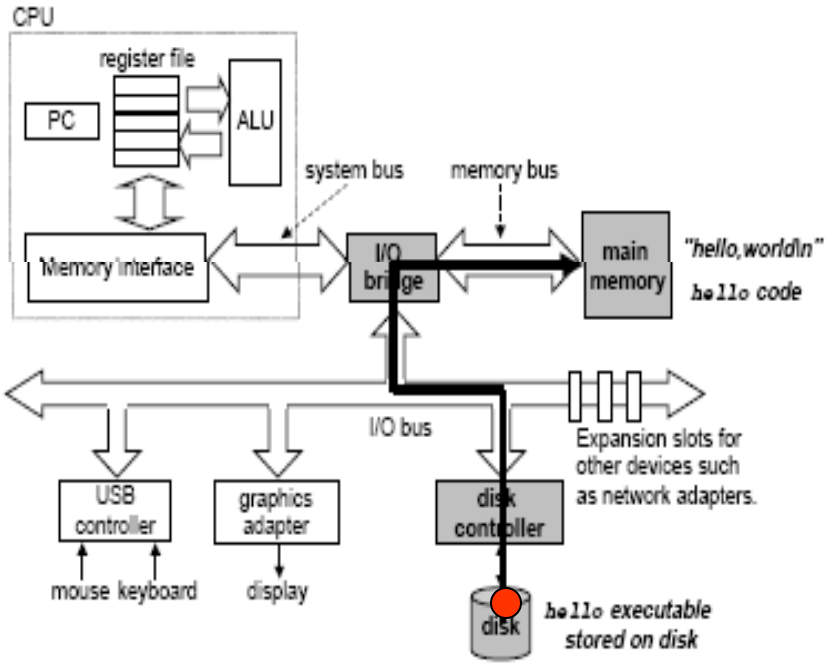
- **Conceito**: Um programa em execução

1. Ao digitar “hello”, os caracteres são passados para um registrador e depois para memória principal



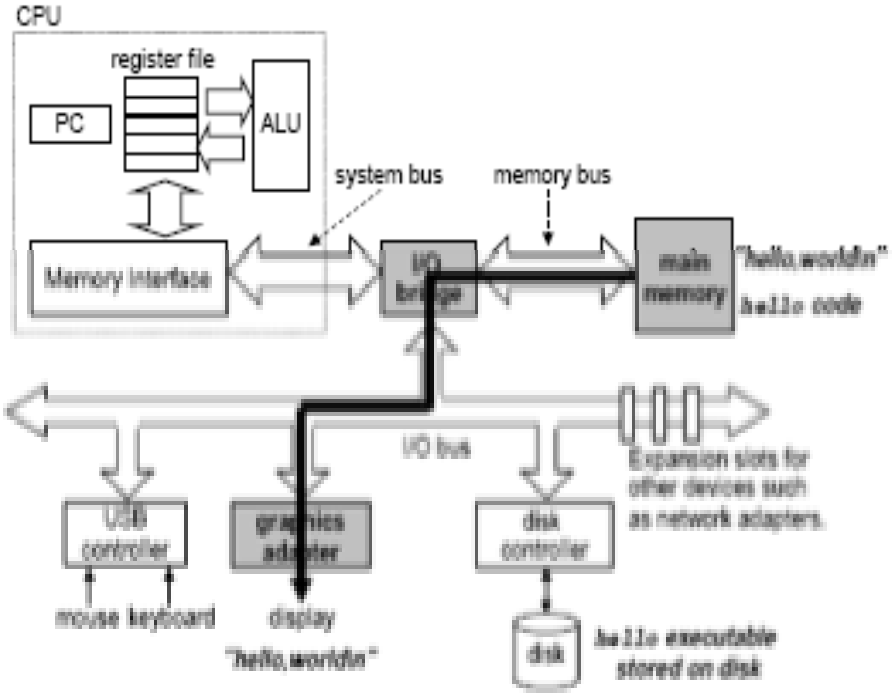
Programa em execução

2. Ao clicar “Enter”, sabe-se que acabou o comando e então é realizada uma seqüência de instruções para copiar código e dados do programa `hello` do disco para a memória principal



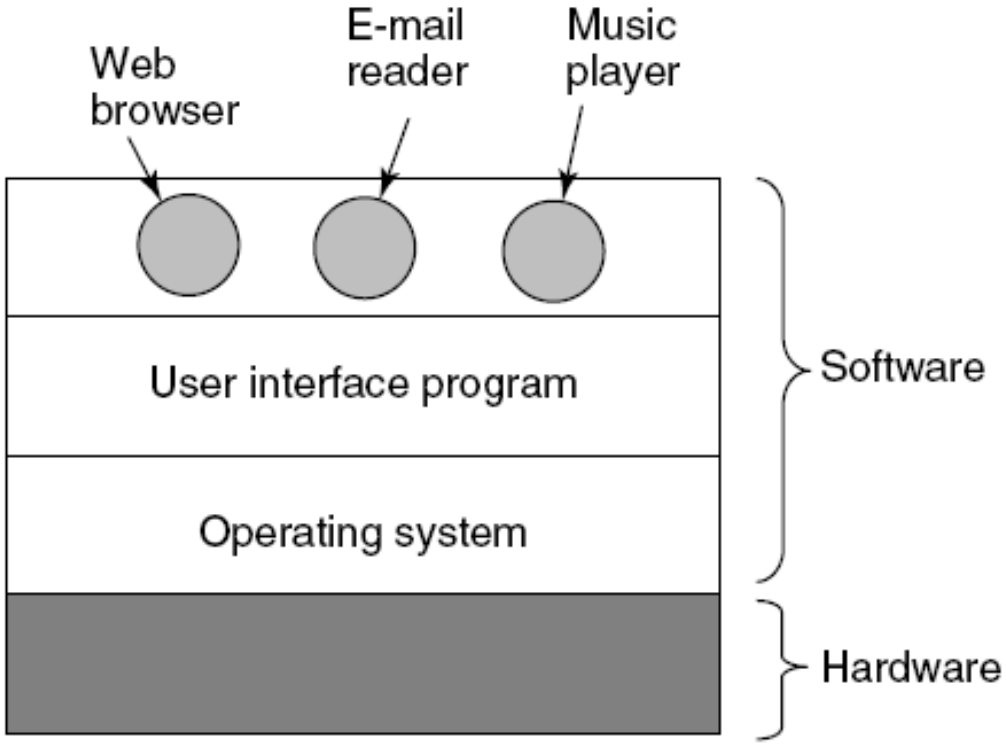
Programa em execução

3. PC aponta para o endereço de memória onde o programa `hello` foi escrito
4. Processador executa instruções em linguagem de máquina da função `main()` do programa



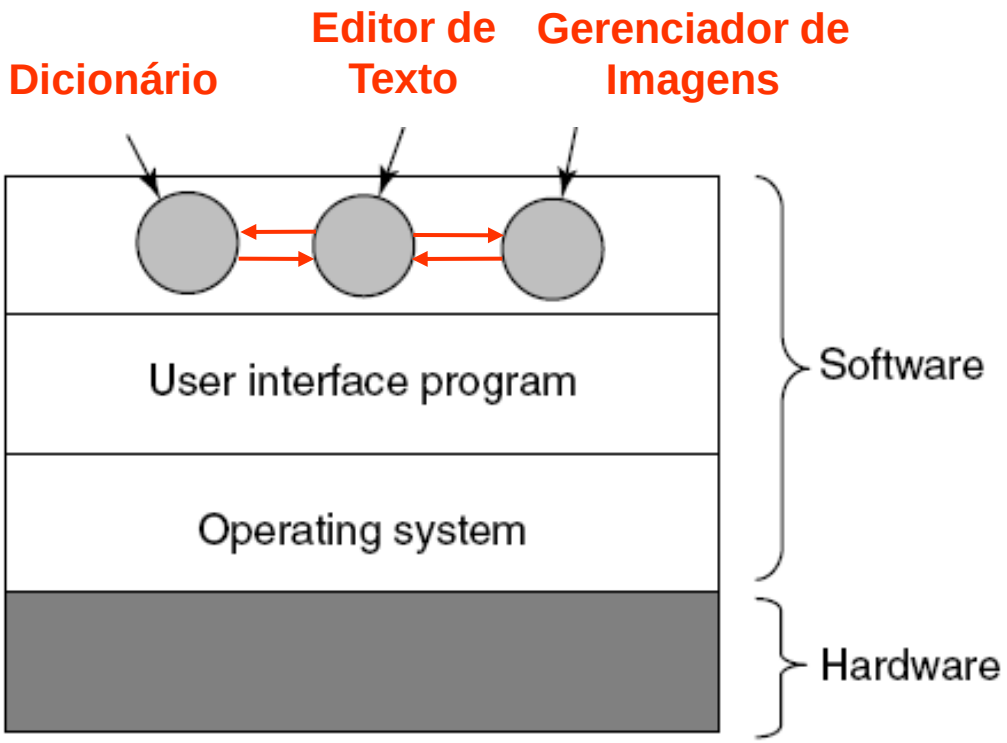
Mais de um programa em execução

- Múltiplos processos *vs.* um (ou [poucos] mais) processador(es) ⇨ **como pode???**



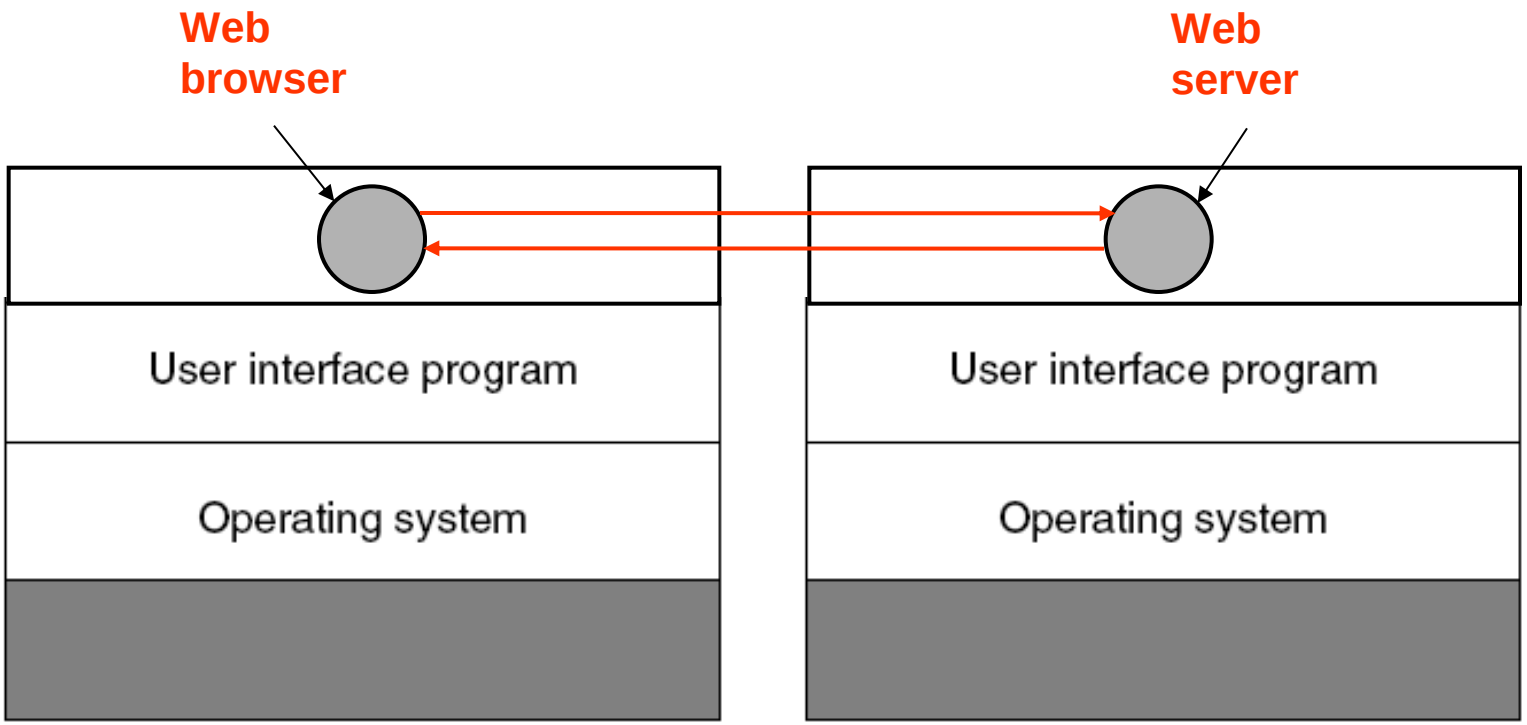
Processos Comunicantes

- **Como pode???**

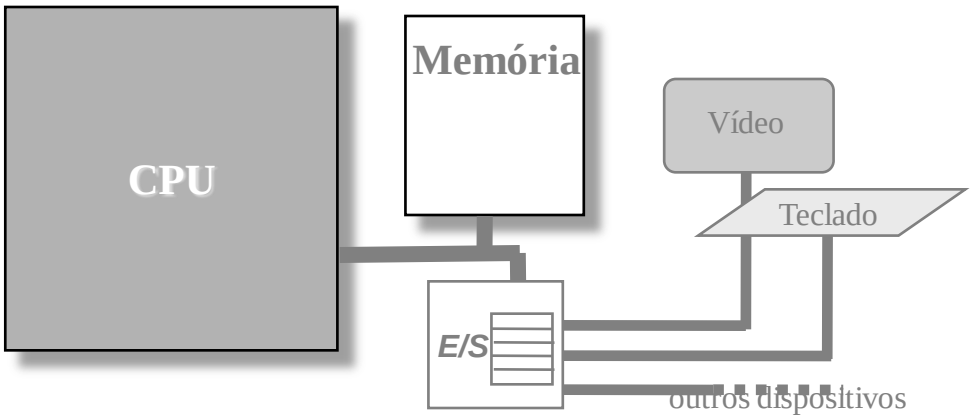


Sistemas Distribuídos

- Processos em máquinas distintas e que se comunicam



Contexto de Processo

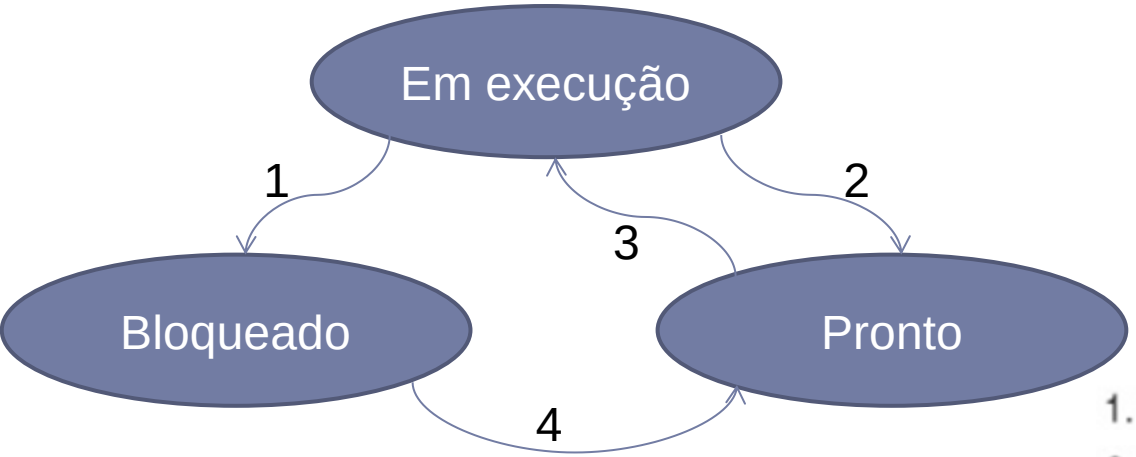


- Informações
  - CPU: Registradores
  - Memória: Posições em uso
  - E/S: Estado das requisições
  - Estado do processo: Rodando, Bloqueado, Pronto
  - Outras informações

Estados de Processos

- Possíveis estados de processos
  - em execução
  - bloqueado
  - pronto

Contexto
ID do Processo
Estado
Prioridade
Program Counter
Ponteiros da Memória
Contexto (outros regs.)
I/O Status
Informações gerais • tempo de CPU • limites, usuário, etc.



1. O processo bloqueia aguardando uma entrada
2. O escalonador seleciona outro processo
3. O escalonador seleciona esse processo
4. A entrada torna-se disponível

Transições 2 e 3 ocorrem sem que o processo saiba disso

Criação de Processos

- Principais eventos que levam à criação de processos
  - Início do sistema
  - Execução de chamada ao sistema de criação de processos
  - Solicitação do usuário para criar um novo processo
  - Início de um job em lote

Término de Processos

- Condições que levam ao término de processos
  - Saída normal (voluntária) — programado
  - Saída por erro (voluntária) — programado
  - Erro fatal (involuntário)
  - Cancelamento por um outro processo (involuntário)

Chamada ao S.O.:

Exit ou ExitProcess
Kill ou terminatePrcess

Hierarquias de Processos

- Processo “pai” cria um processo “filho”, processo filho pode criar seu próprio processo …
- Formando uma hierarquia
  - UNIX chama isso de “grupo de processos”
- Windows não possui o conceito de hierarquia de processos
  - Todos os processos são criados iguais (sem conceito de “pai” e “filho”)

Windows - Handle

Processos

- Com a ideia de processo, torna-se muito mais fácil saber o que está ocorrendo dentro do sistema
  - Processos “de usuário”
  - Processos do S.O.
    - Requisições por serviços de arquivos
    - Gerenciar detalhes do funcionamento de um acionador de disco/fita
    - ...

trata interrupções, escalonamento

Escalonamento de processos

- Quando um ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado primeiro
- A parte do sistema operacional responsável por essa decisão é chamada escalonador, e o algoritmo usado para tal é chamado de algoritmo de escalonamento

- Para que um processo não execute tempo demais, praticamente todos os computadores possuem um mecanismo de relógio (clock) que causa uma interrupção, periodicamente

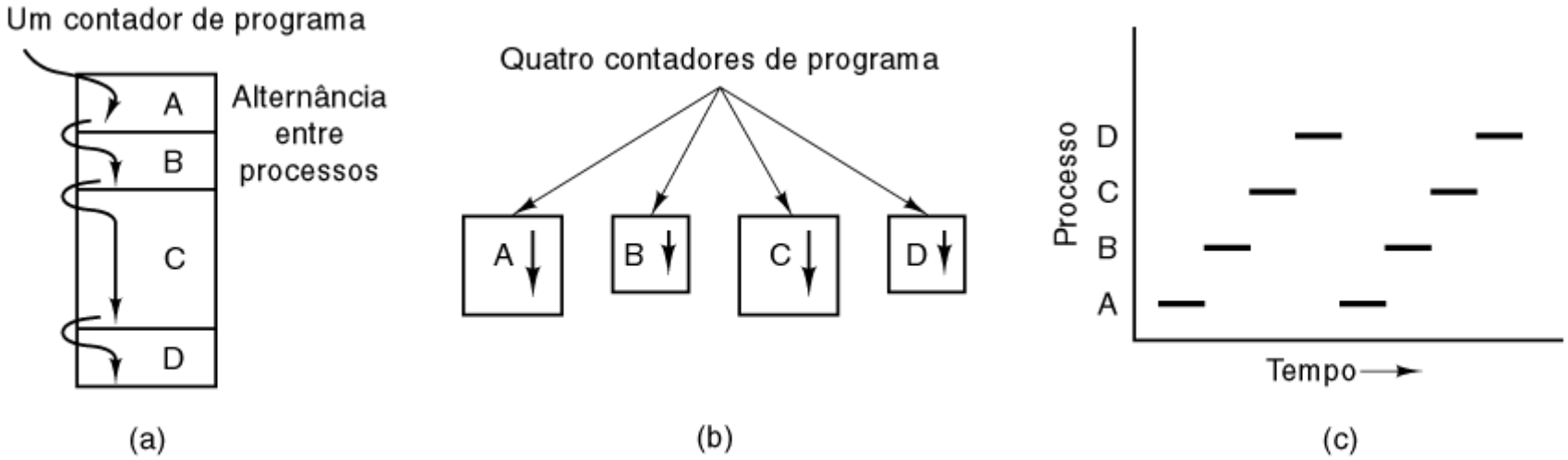
Centro de Informática UFPE

Implementação de Processos (tabela)

Gerenciamento de processos	Gerenciamento de memória	Gerenciamento de arquivos
Registradores	Ponteiro para o segmento de código	Diretório-raiz
Contador de programa	Ponteiro para o segmento de dados	Diretório de trabalho
Palavra de estado do programa	Ponteiro para o segmento de pilha	Descritores de arquivos
Ponteiro de pilha		Identificador (ID) do usuário
Estado do processo		Identificador (ID) do grupo
Prioridade		
Parâmetros de escalonamento		
Identificador (ID) do processo		
Processo pai		
Grupo do processo		
Sinais		
Momento em que o processo iniciou		
Tempo usado da CPU		
Tempo de CPU do filho		
Momento do próximo alarme		

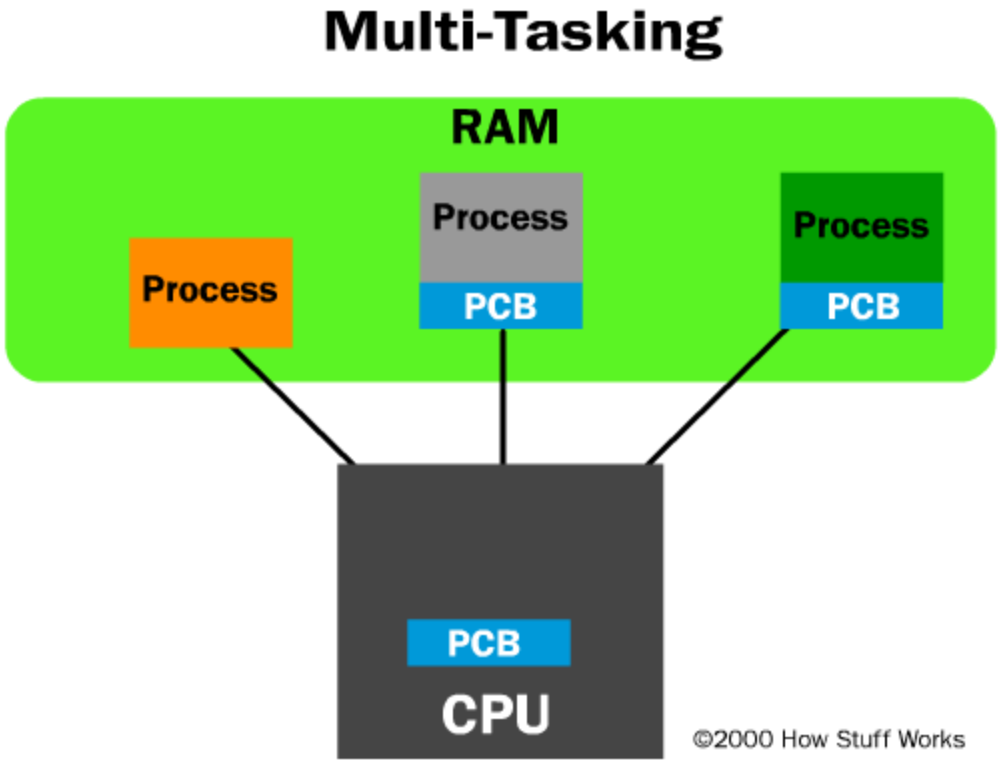
Campos da entrada de uma tabela de processos

Conceito: Multiprogramação



a) Multiprogramação de quatro programas
b) Modelo conceitual de 4 processos sequenciais, independentes, mas
c) Somente um processo está ativo a cada momento ⇨ escalonamento

Multiprogramação



Classificação de Processos

- I/O Bound (intensivos em E/S)
  - Mais tempo esperando por E/S
  - Muitas rajadas curtas de processamento
- CPU Bound (intensivos em processamento)
  - Passa mais tempo em processamento
  - Períodos longos “de CPU”

- E o que ocorre quando existe processos que executam/esperam operações de I/O ?
  - Interrupção
    - uma interrupção é um sinal de um dispositivo que tipicamente resulta em uma troca de contextos, isto é, o processador para de fazer o que está fazendo para atender o dispositivo que pediu a interrupção.

Interrupção: Motivação

- Para controlar entrada e saída de dados, **não é interessante** que a CPU tenha que ficar continuamente monitorando o status de dispositivos como discos ou teclados

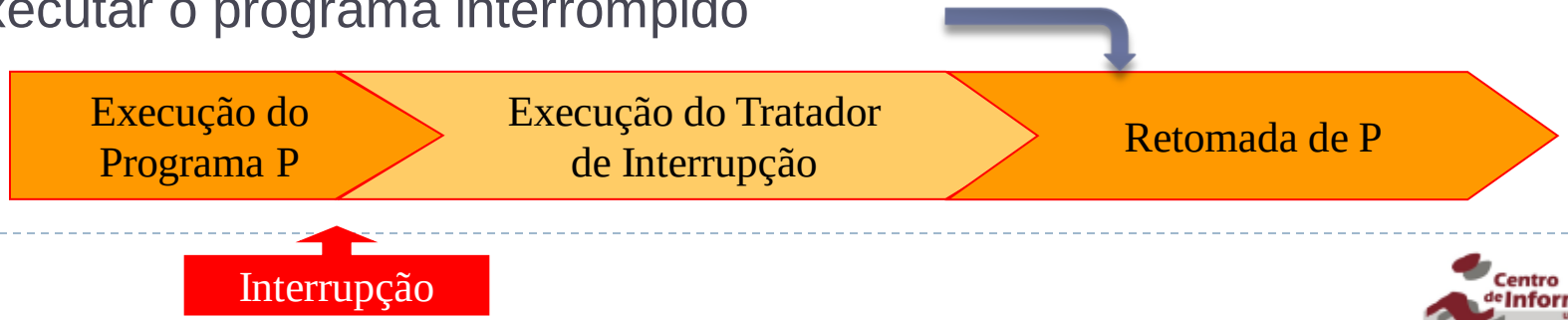
- O mecanismo de interrupções permite que o hardware "chame a atenção" da CPU quando há algo a ser feito

- Interrupção
  - O Elo Hardware-Software

Interrupções de Hardware

- Interrupções geradas por algum dispositivo externo à CPU, como teclado ou controlador de disco, são chamadas de interrupções de hardware ou *assíncronas* [ocorrem independentemente das instruções que a CPU está executando]

- Quando ocorre uma interrupção, a CPU interrompe o processamento do programa em execução e executa um pedaço de código (tipicamente parte do sistema operacional) chamado de ***tratador de interrupção***
  - não há qualquer comunicação entre o programa interrompido e o tratador (parâmetros ou retorno)
  - em muitos casos, 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 e, se for esse o caso, suspende-o e aciona o *escalonador* para que esse escolha outro processo para colocar em execução

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
  - Essas são situações em que o programa não teria como prosseguir
  - O hardware sinaliza uma interrupção para passar o controle para o tratador da interrupção (no SO), que tipicamente termina a execução do programa

Traps (cont.)

- *Traps* também podem ser geradas, explicitamente, por instruções do programa
  - Essa é uma forma do programa acionar o sistema operacional, por exemplo, para requisitar um serviço de entrada ou saída
    - Ex. Read
  - Um programa não pode chamar diretamente uma rotina do sistema operacional, já que o SO é um processo a parte, com seu próprio espaço de endereçamento...
    - Através do mecanismo de interrupção de software, um processo qualquer pode ativar um tratador que pode "encaminhar" uma chamada ao sistema operacional
- Como as interrupções síncronas ocorrem em função da instrução que está sendo executada (ex. READ – uma chamada ao sistema), nesse caso o programa passa algum parâmetro para o tratador

Interrupções

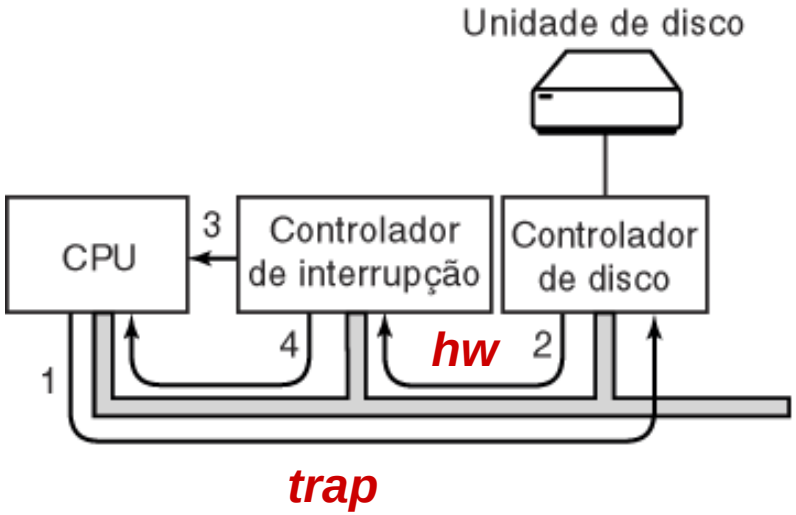
Assíncronas (hardware)

- 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*) no uso compartilhado do processador
  - teclado, para uma operação de E/S (neste caso, de Entrada)

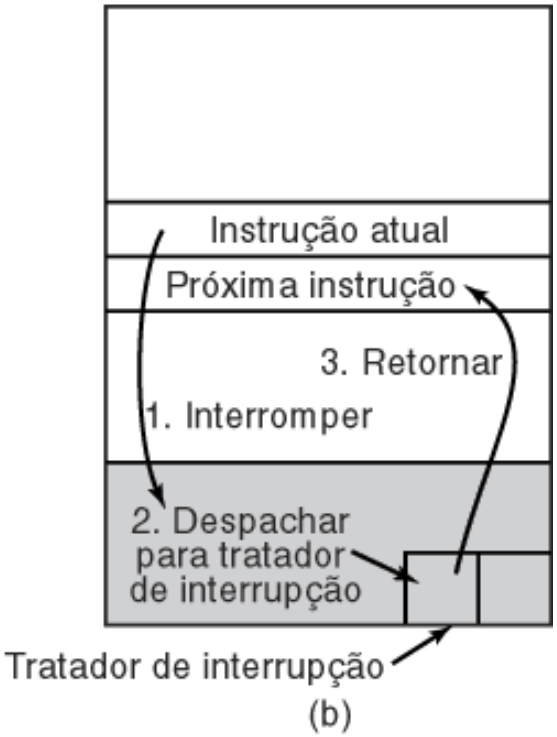
Síncronas (*traps*)

- Geradas pelo programa em execução, em consequência da instrução sendo executada
- Algumas são geradas pelo hardware em situações em que o programa não teria como prosseguir
- Como as interrupções síncronas ocorrem em função da instrução que está sendo executada, nesse caso o programa passa algum parâmetro para o tratador
- Exs.: READ, *overflow* em operações aritméticas ou acesso a regiões de memória não permitidas

Traps e interrupções de hardware



(a)



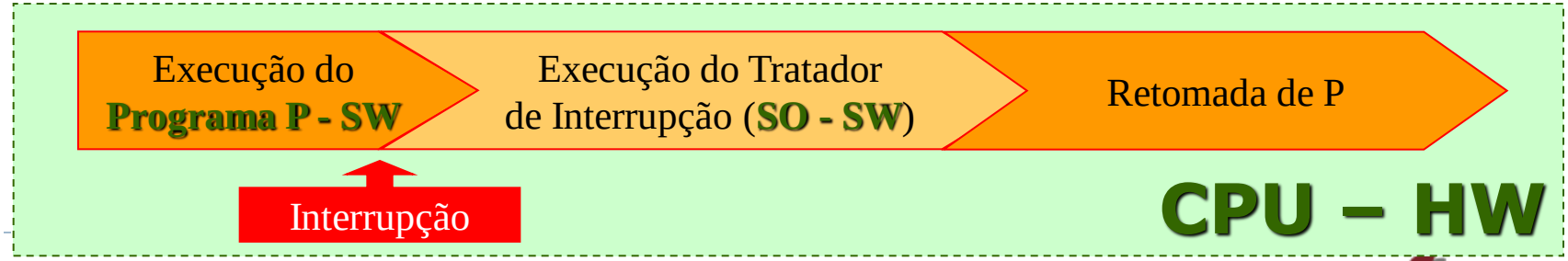
(b)

(a) Passos para iniciar um dispositivo de E/S e obter uma interrupção

(b) Como a CPU é interrompida

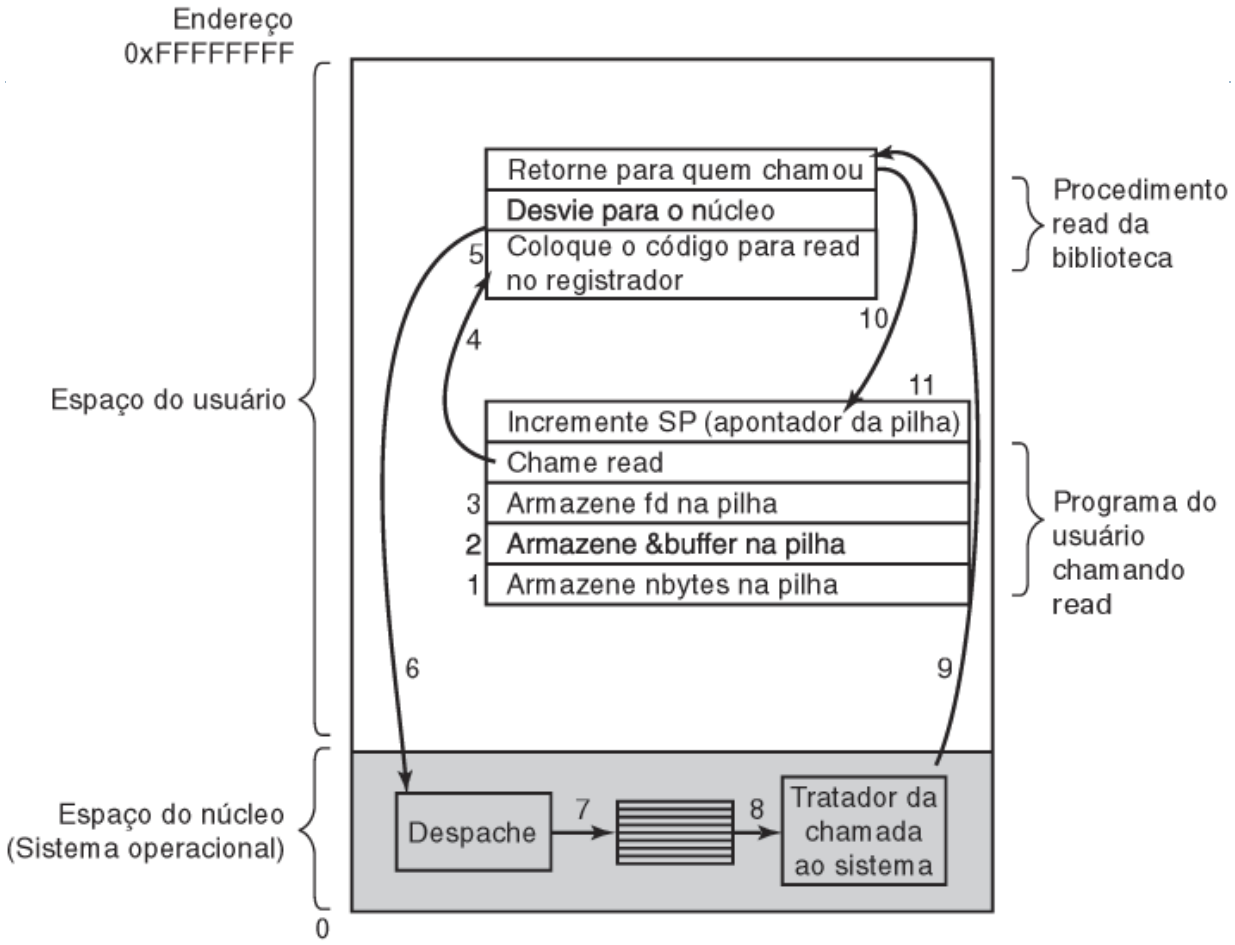
Interrupção: Suporte de HW

- Tipicamente, o hardware detecta que ocorreu uma interrupção,
  - aguarda o final da execução da instrução corrente e aciona o tratador,
  - antes salvando o contexto de execução do processo interrompido
- Para que a execução do processo possa ser reiniciada 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



(*System Calls*)

Chamadas ao Sistema



Os 11 passos para fazer uma chamada ao sistema

Ex. read (fd, buffer, nbytes)

Algumas Chamadas ao Sistema para Gerenciamento de Processos

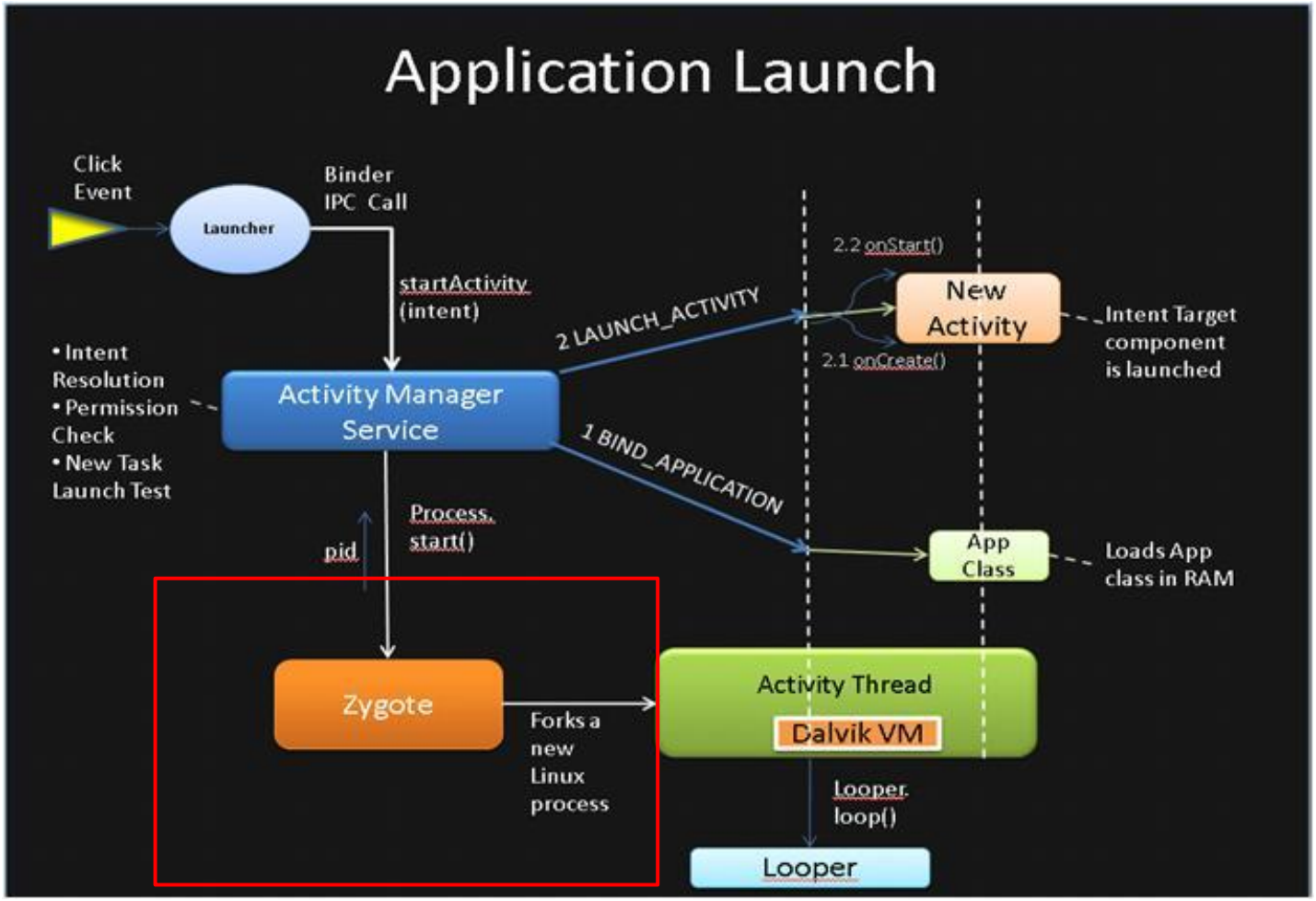
Gerenciamento de processos

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

Fizemos/executamos exemplo usando Fork,
e visualizamos resultados dos processos criados no Linux
Ver arquivo *Aula_04_Processos_Uso_Fork.c*

Centro de Informática UFPE

Algumas Chamadas ao Sistema para Gerenciamento de Processos



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 uma *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 */
        }
}
```

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

Linux SysCall table

- http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html
  - Acessado em 06/04/2015

Sistemas Operacionais
Processos

Carlos Ferraz (cagf@cin.ufpe.br)
Jorge Cavalcanti Fonsêca (jcbf@cin.ufpe.br)