DCA-0125 Sistemas de Tempo Real

Luiz Affonso Guedes
www.dca.ufrn.br/~affonso
affonso@dca.ufrn.br

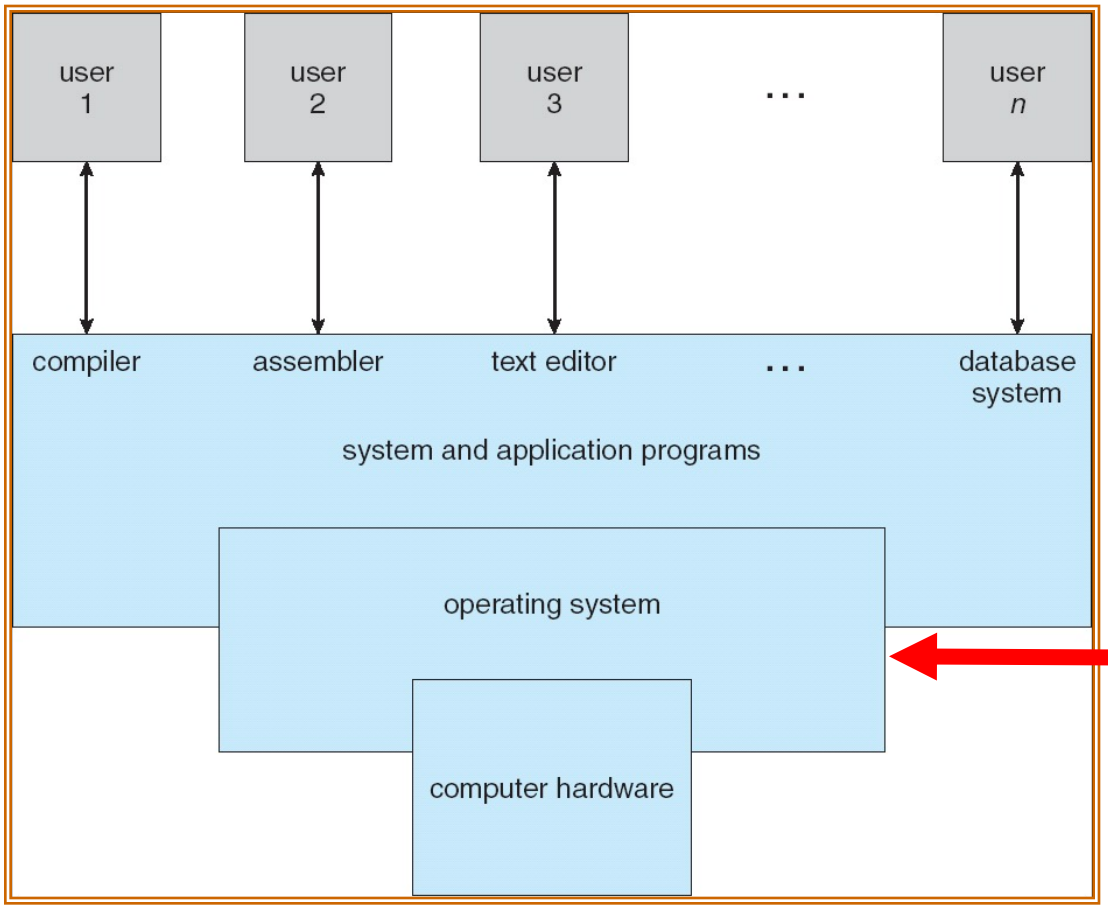
Conceitos de Processos

Conteúdo

- Definição de processos no Sistema Operacional
- Gerência de processos
  - Criação, uso e eliminação.
  - Árvore hierárquica de processos
- Exemplos em C/C++

Recordando SO

Os 4 componentes de sistema computacional



Nosso objetivo de estudo

Recordando SO

Objetivos de sistema operacional

- Executar programas de forma conveniente para o usuário.
- Gerenciar os recursos de software e hardware como um todo.
- Utilizar os recursos de hardware de forma eficiente e segura.

Recordando SO

Visões do sistema operacional

- Como alocador de recursos
  - Gerenciar todos os recursos
  - Resolver conflitos e concorrências por recursos de modo a melhorar a eficiência do sistema como um todo.
- Sistema de Programa de Controle de Recursos
  - Controlar a execução de programas para prevenir erros e melhorar o uso do computador.

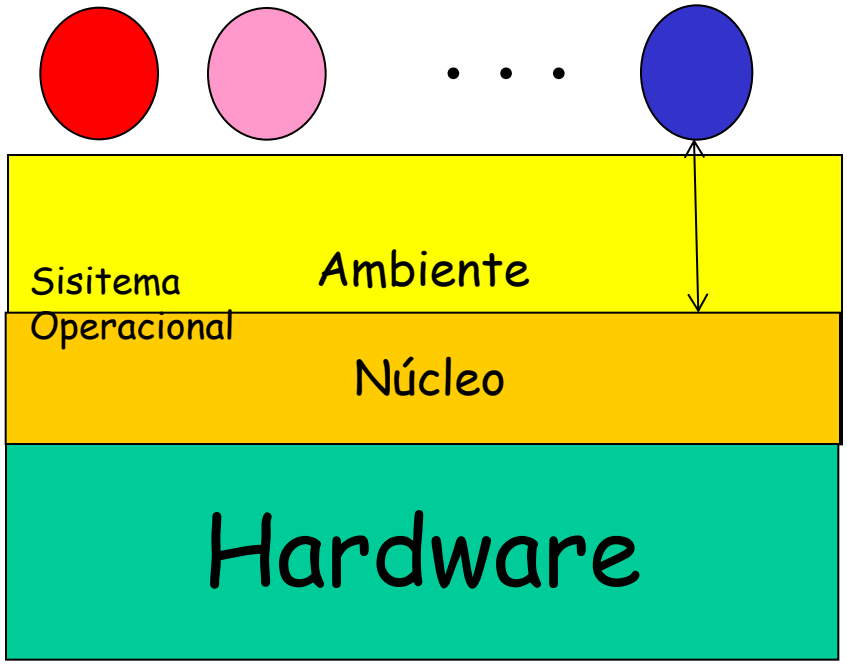
Recordando SO

Visões do sistema operacional

- Visão do usuário
  - Interfaces de acesso a recursos e programas.
- Visão de projeto
  - Organização interna.
  - Mecanismos empregados para gerência de recursos.

Recordando SO

Estrutura e funcionalidades básicas



Ambiente para utilização do sistema (gráfico ou textual)

Ferramentas para a criação de programas

- Editores, depuradores, compiladores

Execução dos programas

Controle de acesso a arquivos

Acesso a recursos de sistema (memória, dispositivos de E/S)

Contabilização de uso de recursos do sistema (cpu, memória, etc)

Detecção de erros

Recordando SO

Núcleo do Sistema Operacional

- Define a abstração de uma máquina virtual sobre uma máquina real
  - Gerência do processador
  - Gerência de memória real e virtual
  - Gerência de E/S
- É acessado via chamada ao sistema (system calls)

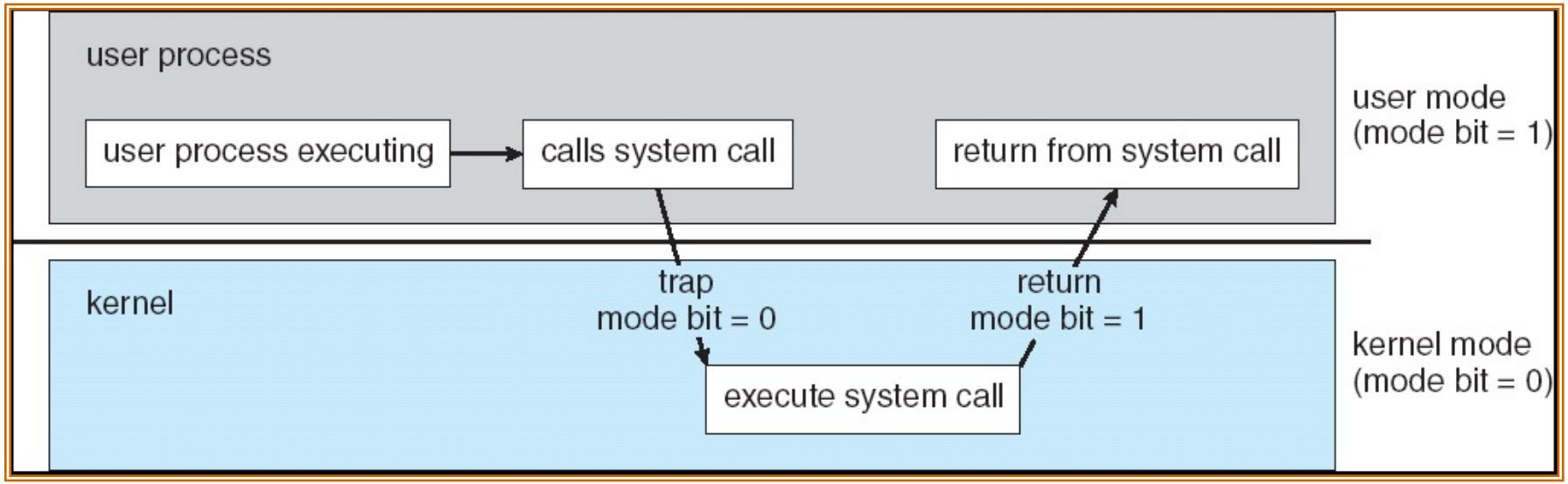
Recordando SO

Programa de Sistema

- Programas executados fora do núcleo e compõem o ambiente.
  - Compiladores, ligadores, bibliotecas
  - Interpretadores de comandos (shell, no Unix)
- Funciona como interface para acesso ao núcleo.

Recordando SO

Interação entre núcleo ambiente

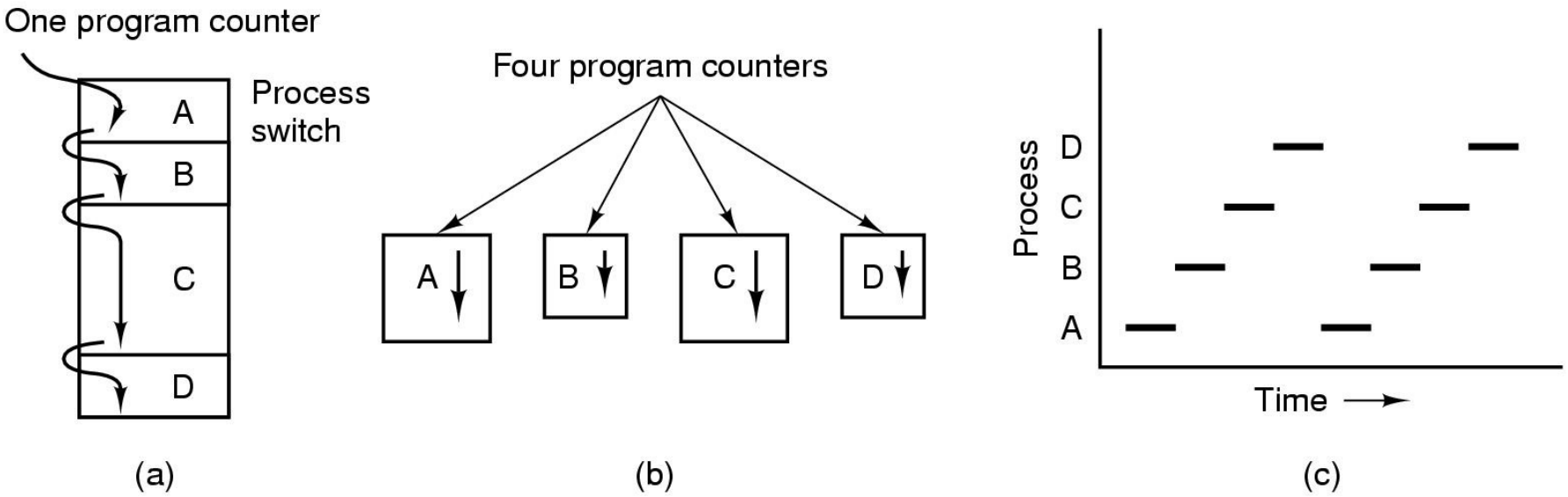


Exemplos de Chamadas do Sistema

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

Objetivos da Multiprogramação

- Tornar mais eficiente o aproveitamento dos recursos do computador.
- Execução "simultânea" de vários programas.
  - Diversos programas são mantidos na memória.
- O próprio SO é um programa!



Conseqüências da Multiprogramação

- Necessidade de controle e sincronização dos diversos programas.
- Necessidade de se criar conceitos e abstração novas
  - Modelagem
  - Implementação

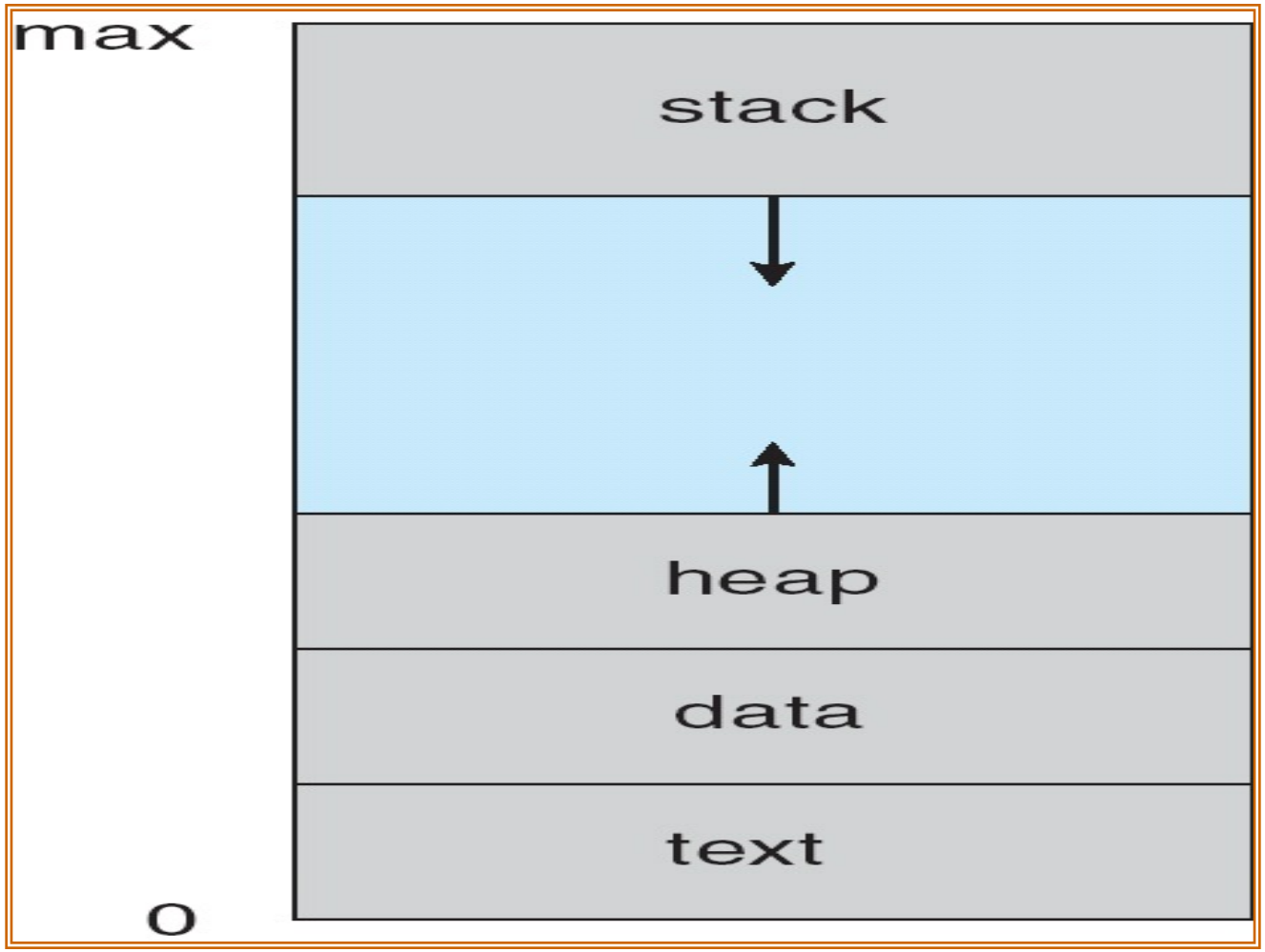
Conceitos Fundamentais

- Processos
- Interrupção
- Sincronização, comunicação, controle e proteção entre processos

Definição de Processo

- Processo é o mesmo de Programa?
- Programa
  - Estrutura estática
    - Instruções + Dados
- Processo
  - Entidade Ativa
  - Instância de um Programa em execução.
  - Processos = Programa + Identificador + Entrada + Saída + Estado
  - Dois Processos podem executar instâncias diferentes do mesmo Programa.

Estrutura de Processo na Memória

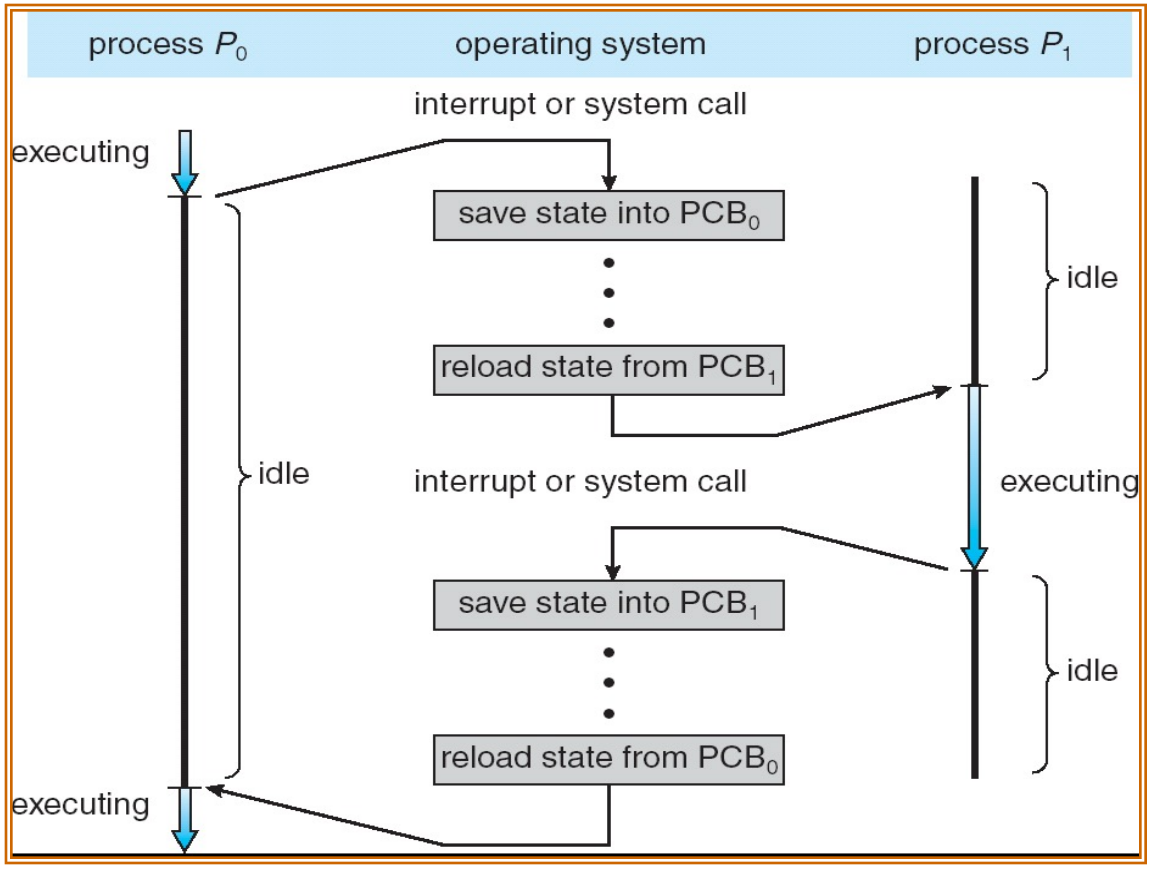


Definição de Processo

- Processo é então um programa em execução!
- O SO trata com processos e não com programas.
- Tipos de Processos
  - Do usuário
  - Do SO, daemons

Multiprocessos

- Necessidade de mudança de contexto entre processos



Ciclo de Vida de Processos

- Como são programas em execução, eles têm começo, meio e fim.
  - Início (criação)
  - Executando
  - Término

Término de Processos

- Crie dos Terminais de Shell
- Execute num dos terminais o programa vitima
  - g++ vitima.cpp
  - ./vitima
- No outro terminal termine o processo vitima
  - ???

Criação de Processo

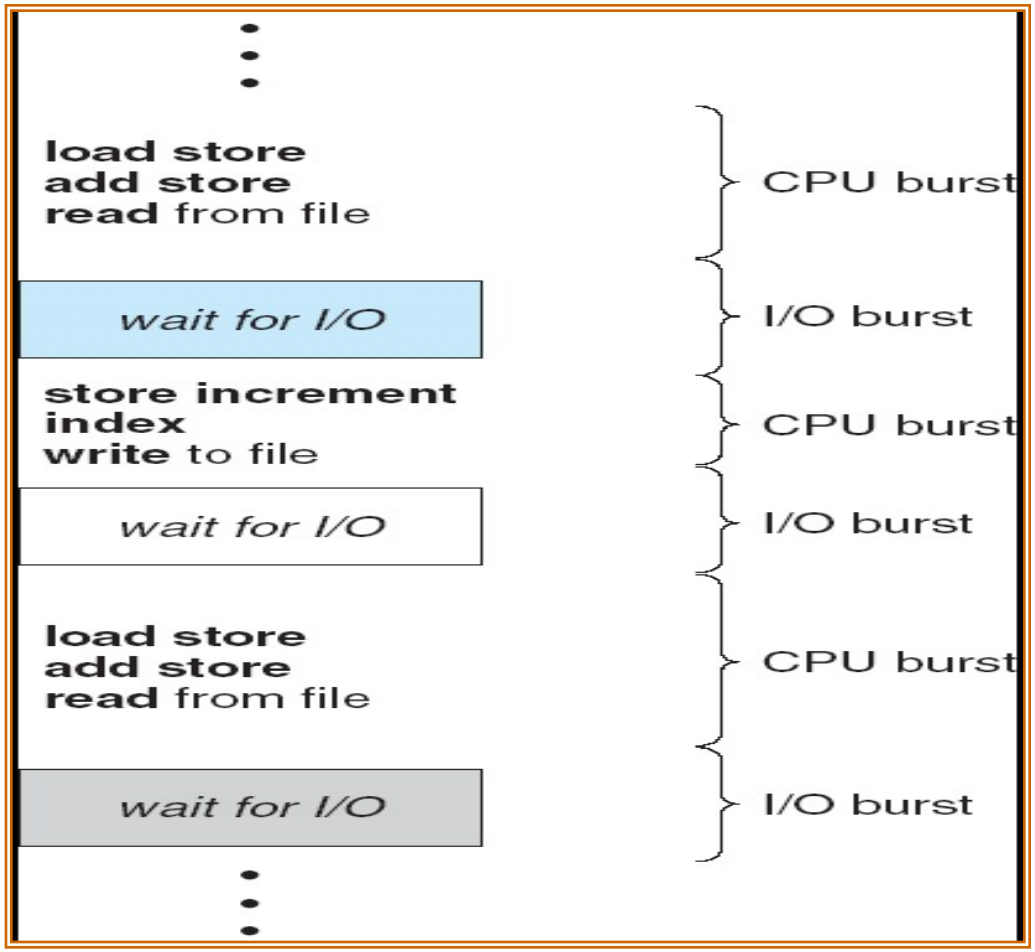
- Momento do início da sua execução
- Associar recursos ao processo
  - Identificador único (PID) → caracterizado por um número inteiro
  - Associar com um programa.
  - Registrar o processo no SO (Tabela de Processos)
    - Essa tabela contém todos os dados necessários para se gerência um processos
  - Exemplo de criação
    - fork( ); ← (unix)
    - CreateProcess ( ); ← windows32

Criação de Processo

- Ao se iniciar o computador, o SO cria vários processo.
- Em unix, ao se iniciar uma seção, o SO cria o processo init (PID=1)

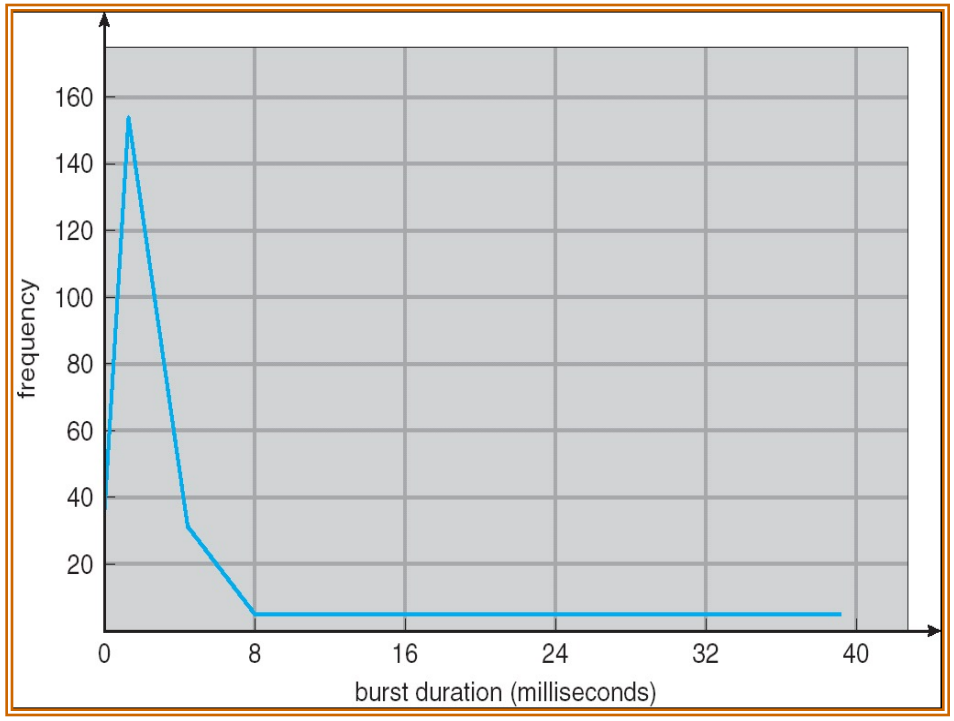
Processo em Execução

- Quando em execução, há basicamente dois modos de operação:
  - Ciclo de processador → quando está utilizando a cpu
  - Ciclo de E/S → quando está esperando por algum dado de E/S



Processo em Execução

- Característica de Processos
  - CPU-Bound
    - Ciclo de CPU >> Ciclo de E/S
    - Aplicações científicas.
  - I/O-Bound
    - Ciclo de E/S >> Ciclo de CPU
    - Acesso a banco de dados.
  - Processos mistos



Término de Processo

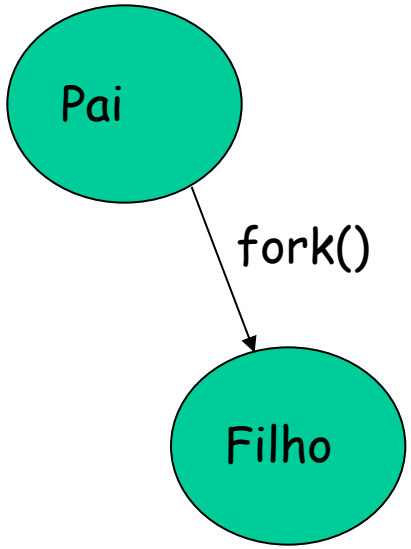
- Final de Execução
  - Normal
  - Por erro
    - Overflow, divisão por zero, falta de memória
- Necessidade de liberar os recursos alocados ao processo.
- Como matar processos
  - Por outro processo
    - Comando kill no Unix
  - Log-off do usuário

Relacionamento entre Processos

- Processos independentes
  - Não há nenhum vínculo entre eles.
  - Não compartilham arquivo, dados, etc.
- Grupo de Processos
  - Há alguma relacionamento.
    - Filiação
    - Compartilhamento de recursos
    - Dependência

Hierarquia entre Processos

- Por exemplo, no Unix cria-se um processo via a primitiva fork();
  - O criador é o Pai
  - O processo criado é o Filho



Hierarquia entre Processos

- Execute o programa fork1
  - g++ fork1.cgg –o fork1
  - ./fork1
- Analise o resultado desse programa

Comando Fork()

```
int main ()
{
// declarações de variáveis

  pid_t pid;  // identificador de processo - inteiro long

  pid = fork();   // dividindo o processo em dois
  switch(pid)
    {
      case -1:        // erro na abertura do processo filho
       CÓDICO DE ERRO
      exit(1);

      // ---------Começo do Processo Filho
      case 0:         // Parte a ser executada pelo processo Filho1
        CÓDIGO DO PROCESSO FILHO
      break;

      //  -------  Início da Parte do Processo Pai  --------------
      default:        // parte a ser executada pelo processo Pai
        CÓDIGO DO PROCESSO PAI
      break;
    }
  exit (0);  // executado pelos processos Pai e Filho
}
```

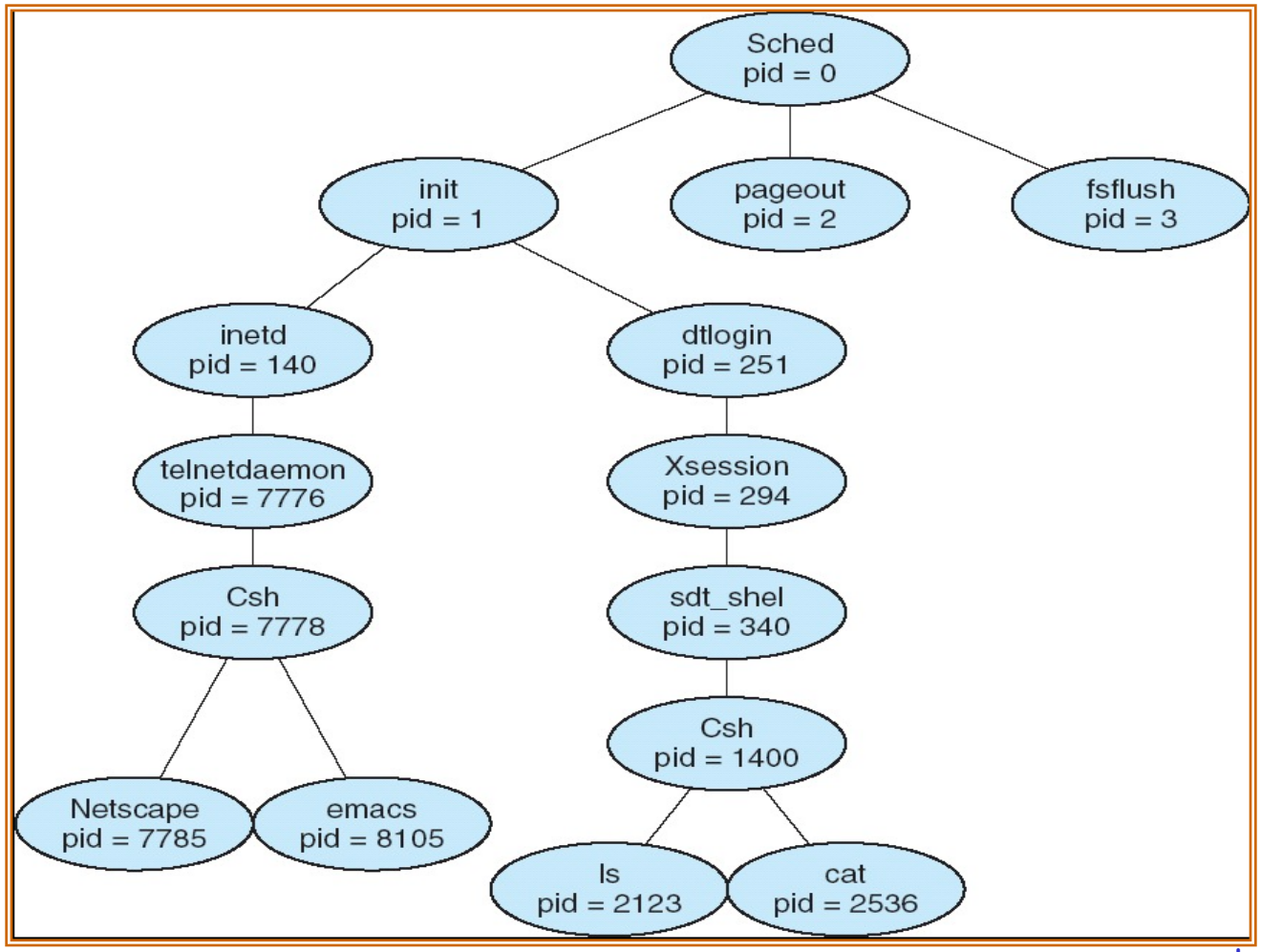
Comando Fork()

```
int main ()
{
// declarações de variáveis

  pid_t pid;  // identificador de
processo - inteiro long

  pid = fork();   // dividindo o
processo em dois
  switch(pid)
    {
      case -1:        // erro na
abertura do processo filho
       CÓDICO DE ERRO
      exit(1);

      // ---------Começo do Processo
Filho
      case 0:         // Parte a ser
executada pelo processo Filho1
        CÓDIGO DO PROCESSO
FILHO
      break;

      //  -------  Início da Parte do
Processo Pai  --------------
      default:        // parte a ser
executada pelo processo Pai
        CÓDIGO DO PROCESSO PAI
      break;
    }
  exit (0);  // executado pelos
processos Pai e Filho
}
```

```
int main ()
{
// declarações de variáveis

  pid_t pid;  // identificador de
processo - inteiro long

  pid = fork();   // dividindo o
processo em dois
  switch(pid)
    {
      case -1:        // erro na
abertura do processo filho
       CÓDICO DE ERRO
      exit(1);

      // ---------Começo do Processo
Filho
      case 0:         // Parte a ser
executada pelo processo Filho1
        CÓDIGO DO PROCESSO
FILHO
      break;

      //  -------  Início da Parte do
Processo Pai  --------------
      default:        // parte a ser
executada pelo processo Pai
        CÓDIGO DO PROCESSO PAI
      break;
    }
  exit (0);  // executado pelos
processos Pai e Filho
}
```

Hierarquia entre Processos

- Árvore de processos típica do Solares



Hierarquia entre Processos

- Execute o programa fork2
  - g++ fork2.cpp –o fork2
  - ./fork2

- Analise os comandos:
  - getpid(.)
  - getppid(.)

Hierarquia entre Processos

- O que ocorre quando um processo morre?
  - A árvore de processos é alterada?
  - É destruída toda a árvore?
  - O processo Avó herda os netos?
    - E se o Avó já estiver morto?

- Como gerenciar a árvore de processos?

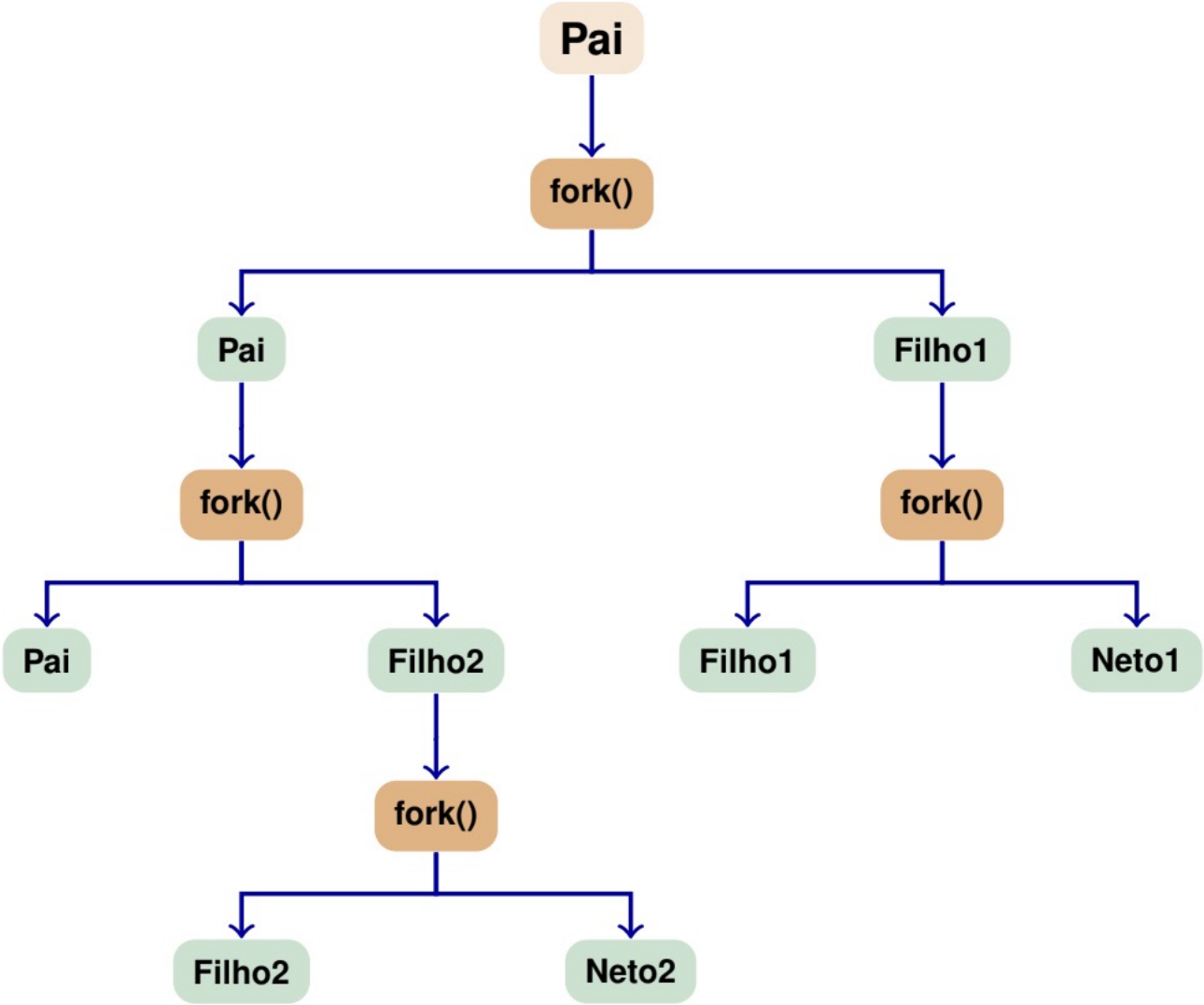
- O que ocorre quando um processo "filho" termina antes do seu "pai"?

Exercício

- Escreva um programa familia.cpp
  - 1 ano corresponde a 1 segundo
  - O pai morre aos 60 anos
  - O pai tem um filho aos 14 anos
  - O pai tem outro filho aos 16 anos
  - O pai é avó aos 26 anos (primeiro filho)
  - O pai e avó novamente (segundo filho) aos 30 anos
  - O primeiro e o segundo filhos morrem ambos aos 30 anos
  - O primeiro neto morre aos 12 anos enquanto que o segundo morre aos 18 anos
  - Imprimir informações sobre os processos e o tempo de vida

Exercício

- Escreva uma programa familia.cpp



Exercício

- Escreva uma programa familia.cpp