

AULA 05 – Processos

Através da execução “*simultânea*” de vários programas, a multiprogramação torna mais eficiente o aproveitamento dos recursos do computador, tais como: tempo do processador, espaço de memória, etc. Na realidade a execução é feita de forma **concorrente** (máquinas monoprocessadas). Em um sistema multiprogramado vários programas são mantidos na memória ao mesmo tempo.

O que acontece em um sistema multiprogramado quando uma chamada de sistema é solicitada por algum processo? Basicamente, o escalonador interrompe o processo que executou a chamada de sistema e escolhe outro da fila de apto para utilizar a CPU.

Os seguintes conceitos são necessários para implementar o conceito de multiprogramação: processos, interrupção e proteção entre processos. Nesta nota de aula iremos discutir estes temas e apresentar detalhes sobre as entidades denominadas processos.

Processos

Em sistema operacional é conveniente diferenciar um programa de sua execução. Assim, podemos dizer que:

Programa: é uma entidade estática e permanente composto por uma seqüência de instruções: passivo sob o ponto de vista do sistema operacional.

Processo: é uma entidade dinâmica e efêmera, que altera seu estado a medida que avança sua execução. Assim, o processo é uma abstração que representa um programa em execução. Um processo é composto por: programa, dados, contexto (valores).

Ciclos de um processo

São várias as razões para que um processo seja criado e/ou destruído. O momento e a forma como isso acontece pode depender do sistema operacional considerado. Por exemplo:

Criação de Processos:

- ✍ Momento da execução
- ✍ Chamadas de sistemas
- ✍ Podem ser associados a uma sessão de trabalho, exemplo: login + senha _ shell (processo)

Finalização de Processos:

- ✍ Término da execução
- ✍ Chamadas de sistemas
- ✍ Por outros processos

Os processos podem executar programas de usuários ou rotinas de sistema (*daemons*). Basicamente, os processos apresentam dois ciclos de operação:

1. *Ciclo de processador*: tempo que ocupa a CPU executando algum programa.
2. *Ciclo de entrada e saída*: tempo em espera pela conclusão de um evento (E/S).

O primeiro ciclo é sempre de processador, pois para entrar em um ciclo de E/S necessariamente é preciso executar pelo menos uma instrução. As trocas de ciclos podem acontecer através:

1. Chamada de sistema (CPU ✍ E/S)
2. Interrupção (CPU ✍ E/S ou E/S ✍ CPU)
3. Ocorrência de evento (E/S ✍ CPU)

Dessa forma, os processos podem ser classificados de acordo com taxa de utilização da CPU ou E/S em:

1. Processos CPU *bound*:

Ciclo de processador >> ciclo de E/S

Exemplo: processo que faz multiplicação de matrizes.

2. Processos I/O *bound*:

Ciclo de E/S >> ciclo de processador

Exemplo: processo que faz cópia de arquivos ou um processo de Banco de Dados.

Relacionamento entre processos

Os processos podem, ou não, apresentar alguma forma de relacionamento ou dependência em relação a sua criação e execução no sistema. Por isso, os processos ativos dentro do sistema computacional podem ser:

1. Independentes

- ✍ Não apresentam relacionamentos com outros processos
- ✍ Não existe nenhum vínculo entre o processo que acabou de ser criado com seu criador
- ✍ A criação exige uma nova entrada na PCB

2. Subprocessos

- ✍ Criados em uma estrutura hierárquica
- ✍ O processo criador é chamado de pai e o novo de filho ou subprocesso
- ✍ Apresentam algum tipo de relacionamento: dependência do processo filho em relação ao pai
- ✍ Podem compartilhar recursos
- ✍ A criação do filho também exige uma nova entrada na PCB

A representação da estrutura hierárquica acontece através de uma árvore, semelhante a estrutura de diretórios e representa a evolução dinâmica dos processos com o tempo.

Estados de um processo

Inicialmente, após ser criado o processo necessita entrar em ciclo de processador para começar a execução das instruções que compõem o programa. No entanto, devemos entender que vários processos são criados e todos irão disputar o uso da CPU. Neste caso, o que o SO deve fazer?

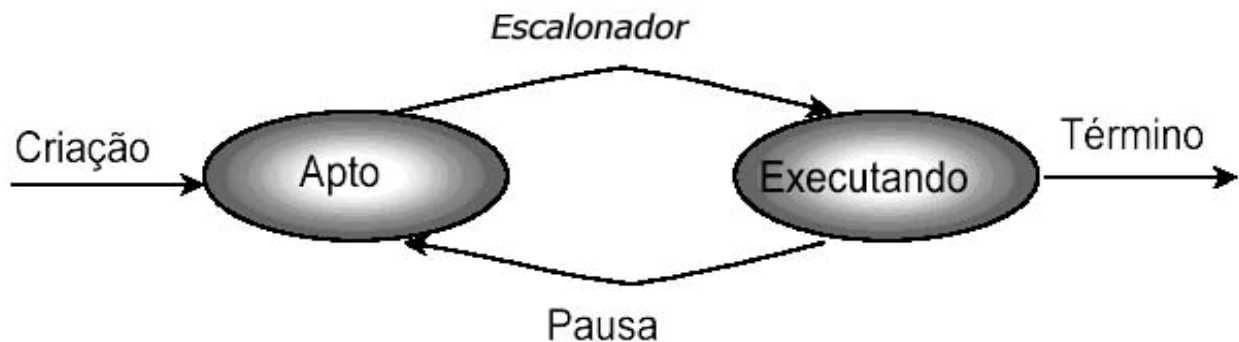
- ✍ Criar uma fila com os processos aptos a disputar o uso do processador: Fila de Aptos (*ready queue*).
- ✍ Gerenciar o escalonamento através de políticas implementadas pelo SO para que todos possam ser executados.

Modelo simplificado de Dois Estados

Este diagrama de estado representa o modelo inicial para ilustrar os possíveis estados que um processo pode alcançar, desde a sua criação até o seu término. Dois estados são destacados: apto e executando. Apto representa o processo em memória aguardando o escalonador para entrar em execução; e executando caracteriza a aplicação utilizando a CPU.

O escalonador de processos é um módulo do SO que deve realizar as seguintes funções básicas:

- ✍ Atribui o processador a um processo da fila de aptos
- ✍ Prevenir que um único processo monopolize o processador



Criação de Processos

Os processos podem ser criados através das seguintes ações:

- ✍ Inicialização de um programa
- ✍ Logon de usuários
- ✍ Processo criado para execução de um determinado serviço do SO
- ✍ Chamadas de sistemas, como o fork no Linux

Término de processos

Os processos podem ser finalizados pelas seguintes ações:

- ✍ Final de execução (normal)

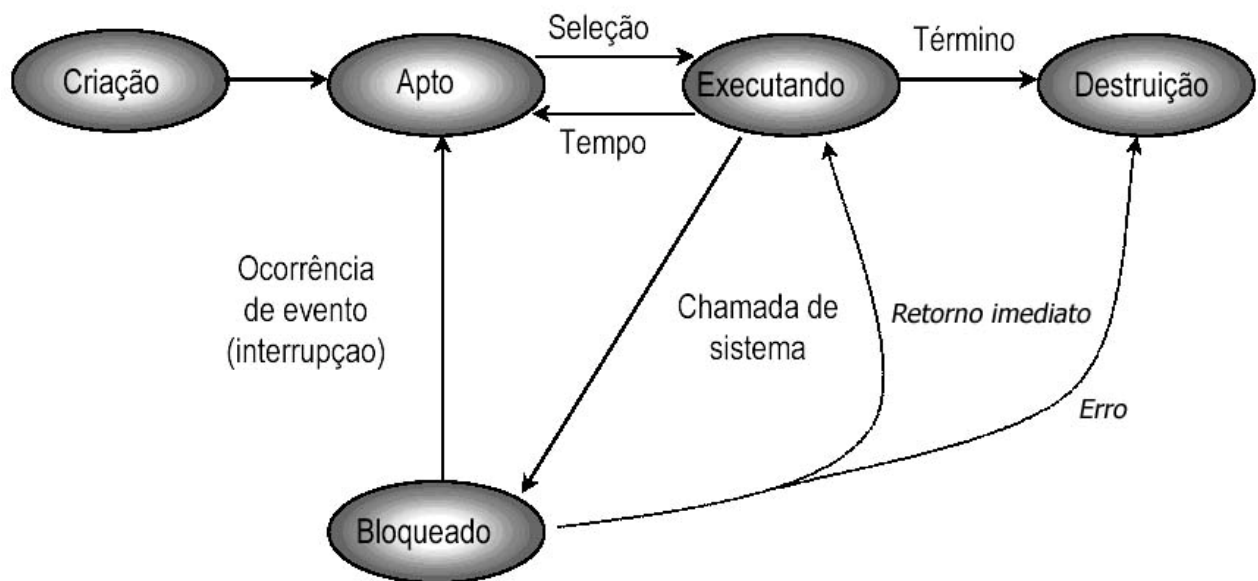
- ✍ Situações “anormais”: Exceder tempo limite de espera, Falta de memória, Erros de proteção (Ex. escrita em arquivo *read-only*, acesso a áreas de memória não autorizadas), Erros aritméticos (Ex. divisão por zero, *overflow*, *underflow*)
- ✍ Erro em periféricos de E/S
- ✍ Execução de instruções inválidas (Ex. tentativa de executar área de dados, instruções privilegiadas)
- ✍ Intervenção do sistema operacional
- ✍ Ex. ocorrência de blocagens (*deadlocks*)
- ✍ *Log off* de usuários

Modelo de 5 Estados

Este modelo cria novos estados de forma a representar diferentes momentos de execução dos processos.

Estados do diagrama:

- ✍ Executando (*Running*)
- ✍ Apto (*Ready*)
- ✍ Bloqueado (*Blocked*)
- ✍ Criação (*New*)
- ✍ Destruição (*Exit*)



Processos suspensos

1. Processador é mais rápido que operações de E/S

Possibilidade de todos processos estarem bloqueados esperando por E/S

2. Liberar memória ocupada por estes processos

Transferidos para o disco (*swap*)

3. Estado bloqueado assume duas situações:

Bloqueado com processo em memória

Bloqueado com processo no disco

4. Necessidade de novos estados

Bloqueado, suspenso (*Blocked, suspend*)

Apto, suspenso (*Ready, suspend*)

Razões para suspender um processo

Swapping

SO necessita liberar memória para executar um novo processo

Solicitação do usuário

Comportamento típico de depuradores

Temporização:

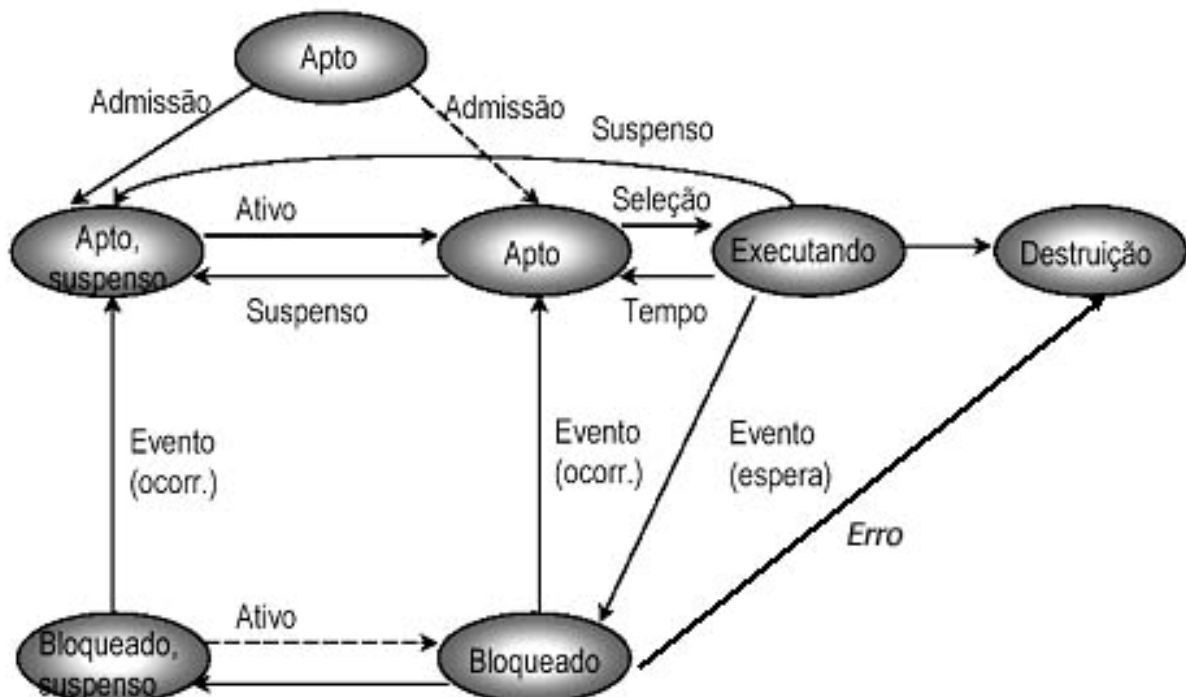
Processo deve ter sua interrupção interrompida por um certo período de tempo

Processo suspende outro processo

Ex. Sincronização

Diagrama de Estados Ampliado

Neste modelo, novos estados foram inseridos para representar com precisão a localização dos processos durante seu ciclo de vida, diferenciando memória e disco. Os estados com o atributo suspenso indica o processo em disco.



Suporte de hardware à multiprogramação

A implementação da multiprogramação explora características do hardware dos processadores

Mecanismos básicos:

- ✍ Interrupção
- ✍ Dois modos de operação: modo protegido (supervisor) e modo usuário
- ✍ Proteção de periféricos, memória e processador

Proteção entre processos

O compartilhamento de recursos comuns implica em garantir que a execução incorreta de um programa não influencie a execução de outro programa.

Mecanismos de proteção

- ✍ Modos de operação do processador
- ✍ Proteção de periféricos
- ✍ Proteção de memória

Modos de operação do processador

Modo supervisor (protegido)

- ✍ Possibilita a execução de todas as instruções do processador
- ✍ Modo de execução sistema operacional

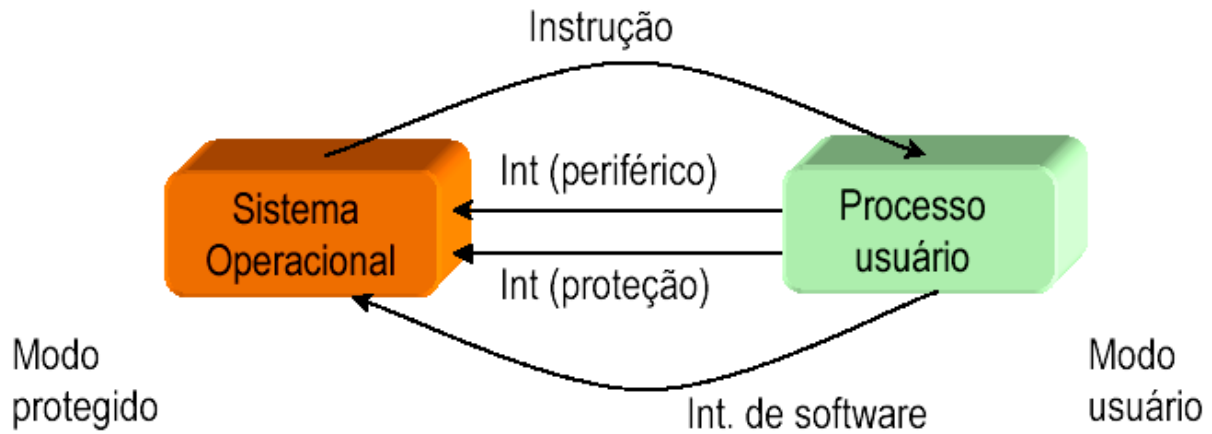
Modo usuário

- ✍ Certas instruções não podem ser executadas
- ✍ Modo de execução dos processos usuários

Chaveamento

- ✍ Interrupção (modo usuário \leftrightarrow modo protegido)
- ✍ Instrução (modo protegido \leftrightarrow modo usuário)

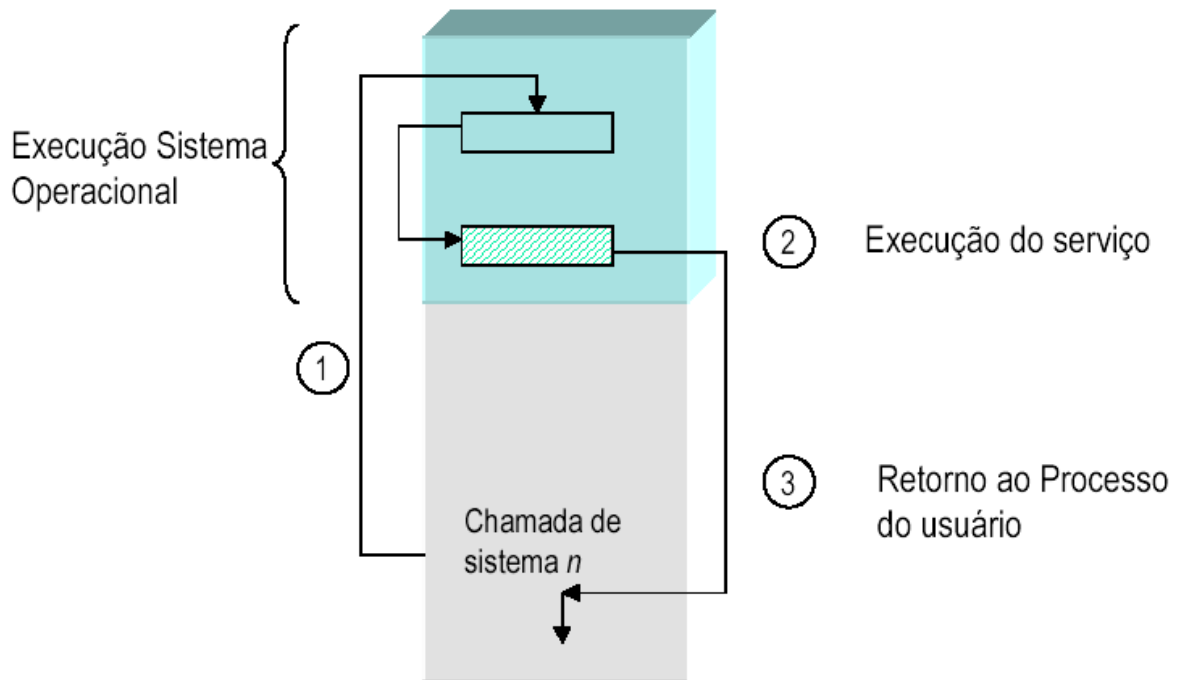
Proteção de periféricos



Como processos usuários realizam operações de E/S já que estas são instruções privilegiadas? Ou ainda, como os processos solicitam serviços para o SO?

Resposta: Através de Chamadas de sistema

- ✍ Normalmente baseada em interrupções de software
- ✍ Aciona a rotina de tratamento de interrupção
 1. Identifica serviço requisitado
 2. Verifica validade dos parâmetros
 3. Executa o serviço
 4. Retorna ao processo do usuário



Exercícios

1. Defina o conceito de processo e explique quais partes o compõe.
2. Como uma aplicação pode implementar concorrência em um ambiente multithread?
3. Como o SO implementa o conceito de processo? Quais as estruturas de dados indicada para organizar os diversos processos na memória principal?
4. Explique as diferenças entre processos foreground, background, cpu-bound e i/o-bound. Dê exemplos de cada um.
5. Por que não faz sentido manter a lista de processos bloqueados em ordem de prioridade?
6. Comente sobre as funções do escalonador e do dispatcher.
7. Apresente situações reais presentes na execução dos processos que justifiquem cada uma das transições apresentadas no diagrama de estados mais completo dos processos.

Bibliografia

OLIVEIRA, R. S.; CARISSIMI, A. S. e TOSCANI, S. S. *Sistemas Operacionais*. Sagra Luzzatto, Porto Alegre, 2001. (Cap. 2)