Sistemas Operacionais

Multiprogramação

Capítulo 2

Multiprogramação

- Tornar mais eficiente o aproveitamento dos recursos do computador
- Execução simultânea* de vários programas
  - Diversos programas são mantidos na memória
  - Conceitos necessários a multiprogramação
    - Processo
    - Interrupção
    - Proteção entre processos

(*) Na realidade a execução é feita de forma concorrente (máquinas monoprocessadas)!!

O conceito de processo (1)

- Diferenciação entre o programa e sua execução
- Programa:
  - Entidade estática e permanente
    - Seqüência de instruções
    - Passivo sob o ponto de vista do sistema operacional
- Processo:
  - Entidade dinâmica e efêmera
    - Altera seu estado a medida que avança sua execução
  - Composto por código, dados e contexto (valores)
  - Identificado por um número único (PID)

O conceito de processo (2)

- Abstração que representa um programa em execução
- Diferentes instâncias
  - Um mesmo programa pode ter várias instâncias em execução, i.e., diferentes processos
  - Mesmo código (programa) porém dados e momentos de execução (contexto) diferentes
- Forma pela qual o sistema operacional “enxerga” um programa e possibilita sua execução

Ciclos de um processo (1)

- Processos são:
  - Criados
    - Momento da execução
    - Chamadas de sistemas
    - Podem ser associados a uma sessão de trabalho
      - e.g. login + senha → shell (processo)
  - Destruídos
    - Término da execução
    - Por outros processos
- Processos executam:
  - Programas de usuários
  - Programas de sistema (daemons)

Ciclos de um processo (2)

- Processos apresentam dois ciclos básicos de operação
  - Ciclo de processador
    - Tempo que ocupa a CPU
  - Ciclo de entrada e saída
    - Tempo em espera pela conclusão de um evento (e.g. E/S)
- Primeiro ciclo é sempre de processador
  - Trocas de ciclos por:
    - Chamada de sistema (CPU → E/S)
    - Interrupção (CPU → E/S ou E/S → CPU)
    - Ocorrência de evento (E/S → CPU)

Ciclos de um processo (3)

- Processos
  - CPU *bound*
    - Ciclo de processador >> ciclo de E/S
  - I/O *bound*
    - Ciclo de E/S >> ciclo de processador
- Sem quantificação exata
- Situação ideal:
  - Misturar processos CPU *bound* com I/O *bound*
    - Benefícios a nível de escalonamento

Relacionamento entre processos (1)

- Processos independentes
  - Não apresentam relacionamentos com outros processos
- Grupo de processos
  - Apresentam algum tipo de relacionamento
    - e.g. filiação
  - Podem compartilhar recursos
  - Definição de uma hierarquia

Relacionamento entre processos (2)

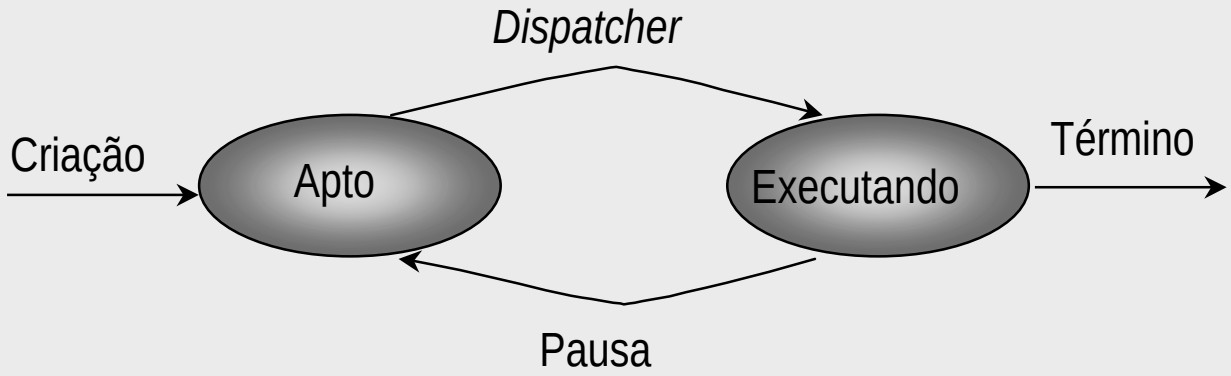
- Hierarquia de processos:
  - Processo criador é processo pai
  - Processo criado é processo filho
- Representação através de uma árvore
  - Evolução dinâmica com o tempo
- Semântica associada: o que fazer na destruição de um processo?
  - Toda a descendência “morre”
  - A descendência é herdada pelo processo “vô”
  - Postergar a destruição efetiva do processo pai até o final de todos processos filhos

Estados de um processo

- Após criado o processo necessita entrar em ciclo de processador
  - Necessita do processador para executar
- Hipotéses (supor caso de uma máquina monoprocessada):
  - Processador não está disponível
  - Vários processos sendo criados
- Que fazer?
  - Criação de uma fila com os processos aptos a ganhar o processador
    - Fila de aptos (*ready queue*)

Modelo simplificado a dois estados

- Manter uma fila de processos aptos a executar
  - Esperando pelo processador ficar livre
- Escalonador (*dispatcher*):
  - Atribui o processador a um processo da fila de aptos
  - Pode prevenir um único processo de monopolizar o processador



Criação de processos

- Submissão de um job
- Logon de usuários
- Processo criado para execução de um determinado serviço (*daemons*)
- Processo criado a partir de um processo já existente (*spawn*)

Término de processos (1)

- Final de execução (normal)
- Situações “anormais”
  - Exceder tempo limite
  - Falta de memória
  - Erros de proteção
    - e.g. escrita em arquivo *read-only*, acesso a áreas de memória não autorizadas, etc...
  - Erros aritméticos
    - e.g. divisão por zero, *overflow, underflow*

Término de processos (2)

- Erro em periféricos de E/S
- Execução de instruções inválidas
  - e.g. tentativa de executar área de dados, instruções privilegiadas
- Intervenção do sistema operacional
  - e.g. ocorrência de blocagens (*deadlocks*)
- *Log off* de usuários

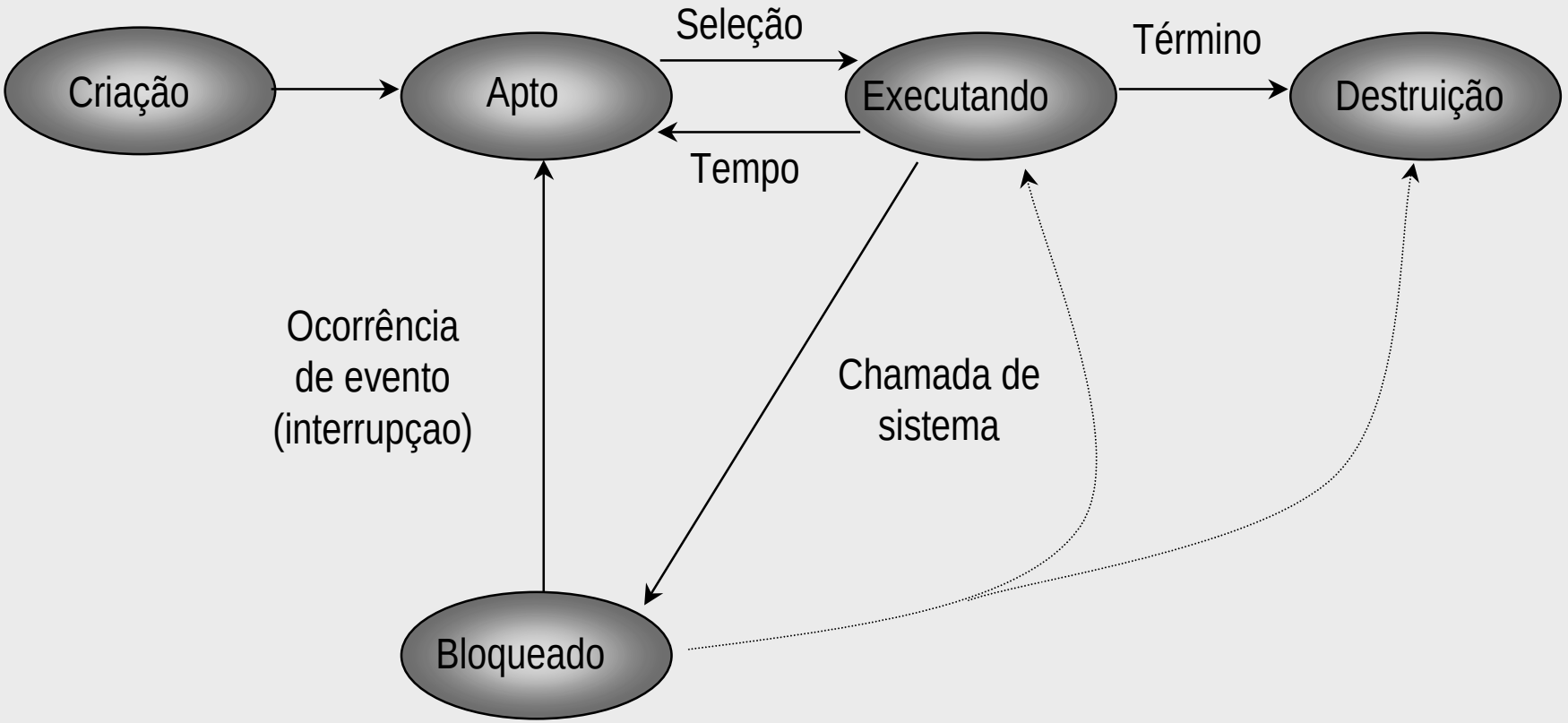
Limitação do modelo simplificado

- Causas para um processo não executar
  - Esperando pelo processador
    - Aguardando na fila aptos a executar
  - Esperando pela ocorrência de um evento externo
    - Processo “bloqueado” a espera desse evento
- Escalonador não pode selecionar um processo bloqueado para executar, logo modelo a dois estados não é suficiente
  - Criação de novos estados

Modelo a 5 estados

- Criação de novos estados e fila de forma a representar diferentes momentos de execução dos processos
- Estados:
  - Executando (*Running*)
  - Apto (*Ready*)
  - Bloqueado (*Blocked*)
  - Criação (*New*)
  - Destruição (*Exit*)
- Necessidade de associar eventos às transições de um estado a outro

Diagrama de transição do modelo a 5 estados



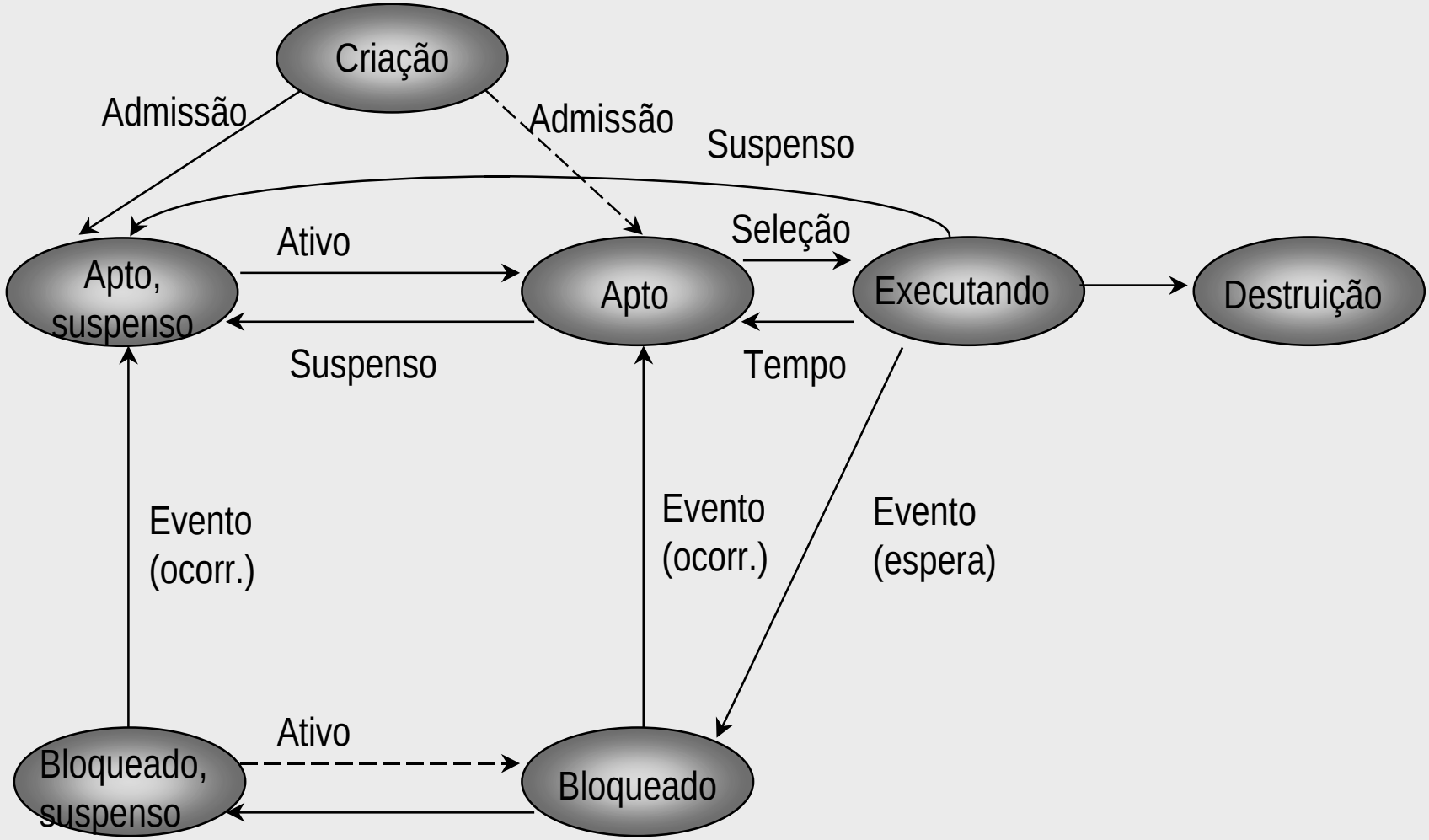
Processos suspensos

- Processador é mais rápido que operações de E/S
  - Possibilidade de todos processos estarem bloqueados esperando por E/S
- Liberar memória ocupada por estes processos
  - Transferidos para o disco (*swap*)
- Estado bloqueado assume duas situações:
  - Bloqueado com processo em memória
  - Bloqueado com processo no disco
- 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 suspender outro processo
  - e.g. sincronização

Diagrama de estados de processos



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

Mecanismo de interrupção (1)

- Sinaliza a ocorrência de algum evento solicitando a atenção do processador para o tratamento desse evento
- Provoca a execução de uma rotina especial:
  - Tratador de interrupção
  - Similar a execução de uma subrotina
- Ciclo de execução de uma interrupção
  - Prepara a transferência de controle para a rotina de tratamento de interrupção (salvamento do contexto de execução)
  - Desvia controle para rotina de tratamento de interrupção
  - Após retorna execução ao ponto que teve sua execução interrompida (restaura contexto de execução)

Mecanismo de interrupção (2)

- Tipos de interrupção
  - Hardware: ocorrência de evento externo
  - Software: execução de uma instrução específica (*traps*)
  - Exceção: erros de execução (*overflow*, *undeflow*...)
- Identificadas por um número
  - Vetor de interrupção
- Prioridades
- Instruções privilegiadas

Proteção entre processos

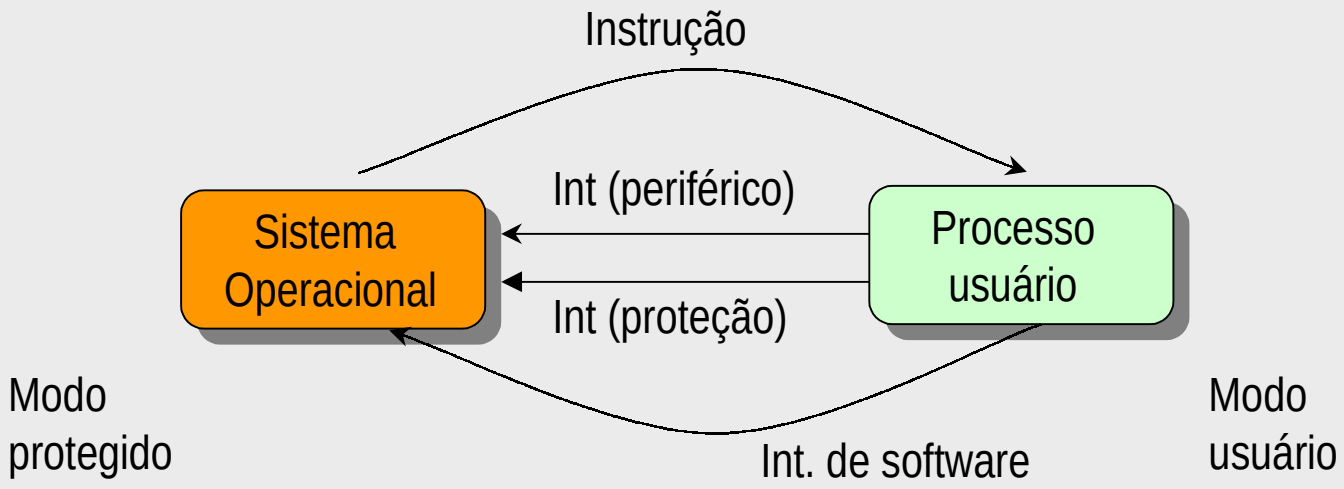
- O compartilhamento de recursos comuns implica em garantir que a execução incorreta de um programa não influencie a execução de outro programa.
- Mecanismos de proteção
  - Modos de operação do processador
  - Proteção de periféricos
  - Proteção de memória

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 → modo protegido)
  - Instrução (modo protegido → modo usuário)

Proteção de periféricos (1)

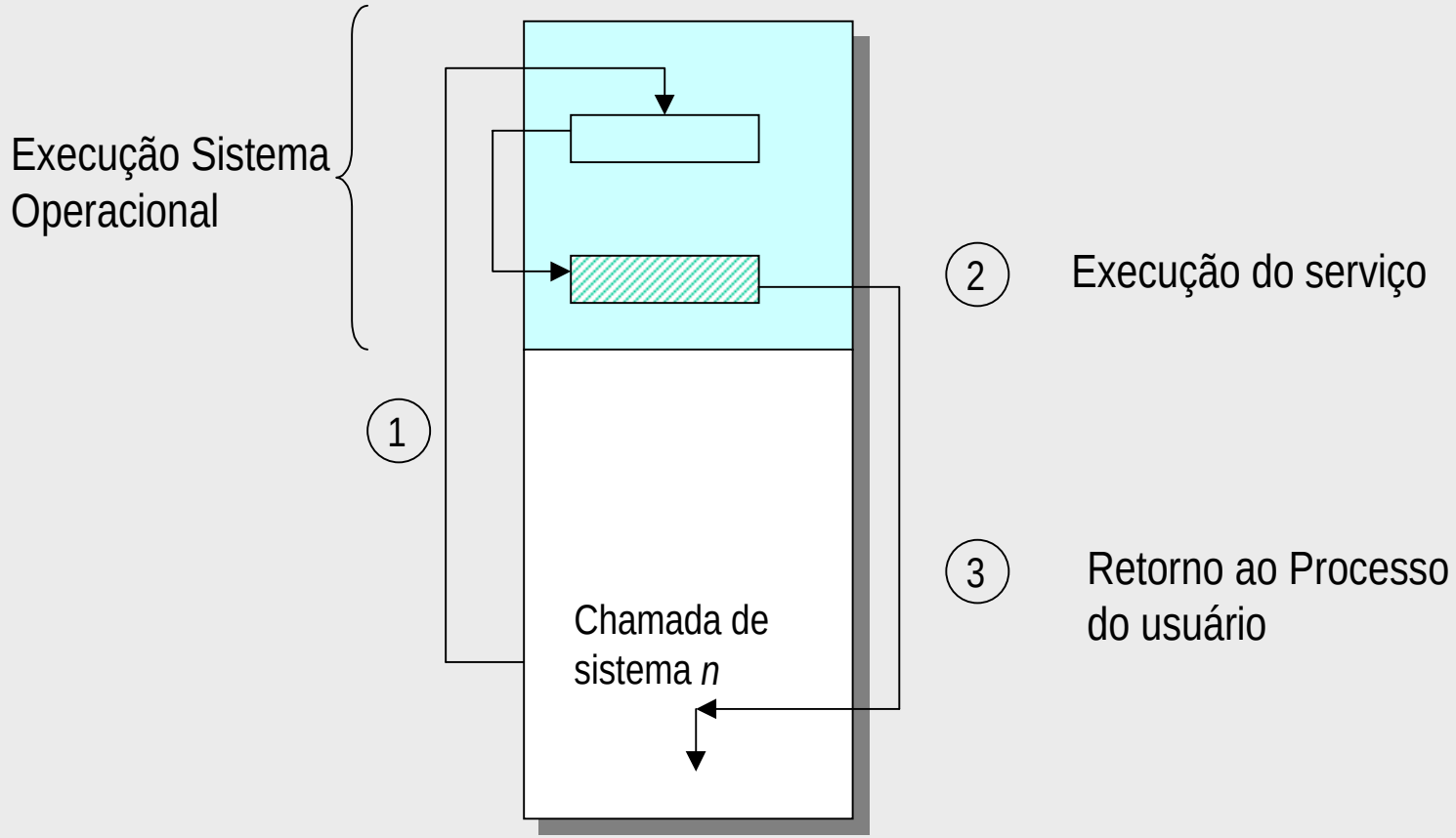
- Instruções de E/S são privilegiadas
- Interrupções de software (*traps*) são utilizadas para realizar chamadas de sistema



Proteção de periféricos (2)

- Como processos usuários realizam operações de E/S já que estas são instruções privilegiadas?
  - Chamadas de sistema
- Chamada de sistema é o método empregado para um processo usuários solicitar serviços ao SO.
  - Normalmente baseada em interrupções de software
  - Aciona a rotina de tratamento de interrupção
    - Identifica serviço requisitado
    - Verifica validade dos parâmetros
    - Executa o serviço
    - Retorna ao processo do usuário

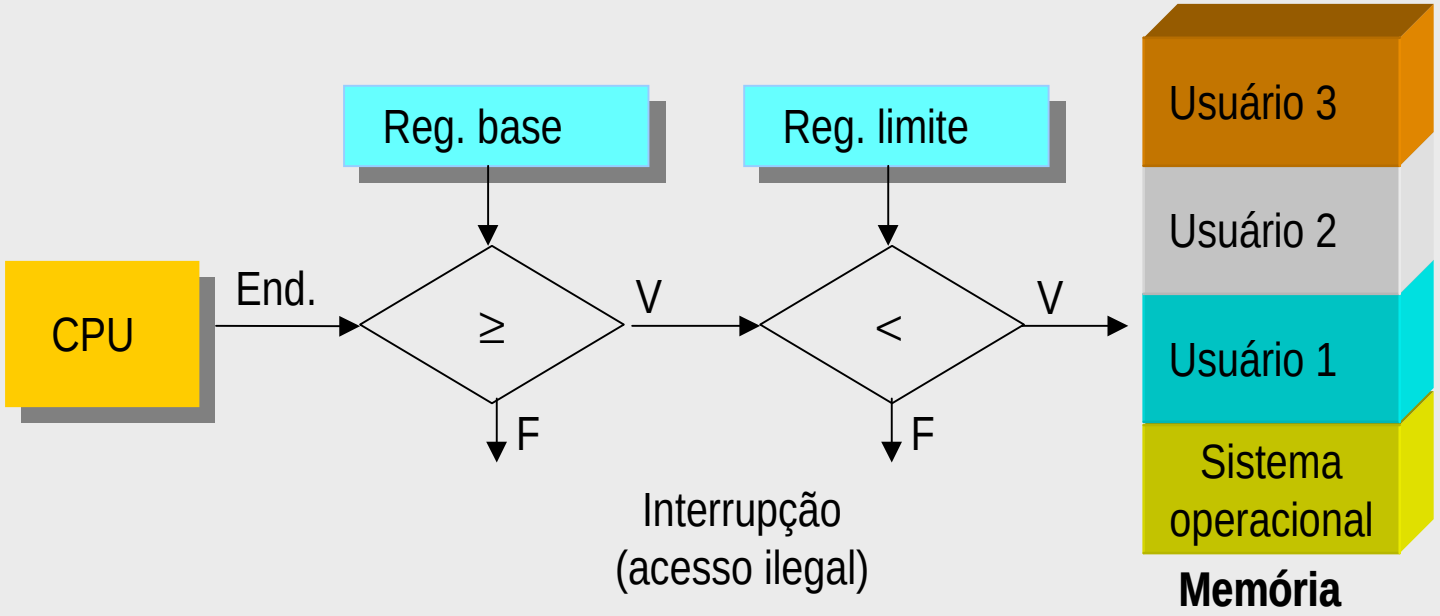
Princípio de chamada de sistema



Proteção de memória (1)

- Necessário para evitar que usuário corrompa espaços de memória não-pertencentes a seus processos
- Baseado em facilidades da arquitetura do processador:
  - Registrador de base (limite inferior)
  - Registrador de limite (limite superior)
- Faixa de endereçamento fora da área delimitada pelos registradores base e limite é protegida
- Possível proteger dispositivos de E/S quando a técnica E/S mapeada em memória é empregada

Proteção de memória (2)



Proteção do processador

- Para garantir a execução do sistema operacional uma interrupção de tempo (*timer*) ocorre periodicamente
- Interrupção de tempo:
  - Empregada para implementar multiprogramação
  - Mantém contabilização de tempo para o sistema operacional (relógio)
- Instruções relacionadas com a programação do tempo são privilegiadas

Leituras complementares

- R. Oliveira, A. Carissimi, S. Toscani; *Sistemas Operacionais*. Editora Sagra-Luzzato, 2001.
  - Capítulo 2
- A. Silberchatz, P. Galvin, G. Gagne; *Applied Operating System Concepts*. Addison-Wesley, 2000, (1 edição)
  - Capítulo 4 (4.1)
- W. Stallings; *Operating Systems*. (4th edition). Prentice Hall, 2001.
  - Capítulo 1, 3