Laboratório de Pesquisa em Redes e Multimídia

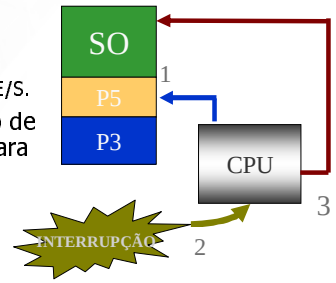
Processos
(Aula 4)

Conceitos Básicos

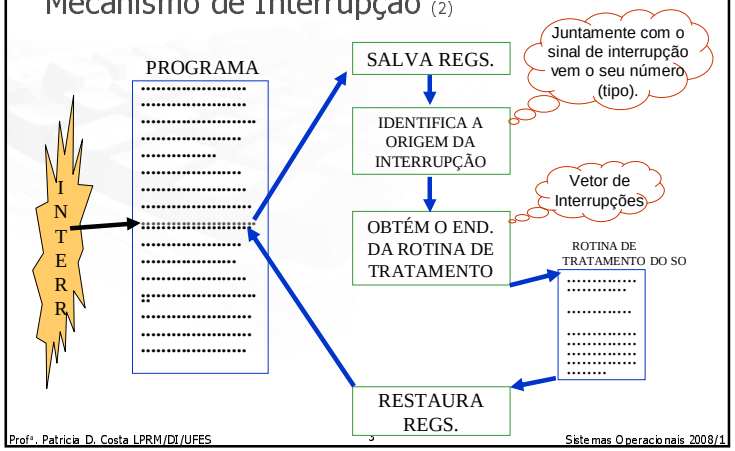
Universidade Federal do Espírito Santo
Departamento de Informática

Mecanismo de Interrupção (1)

- Constitui a base de operação de um sistema de multiprogramação.
- É um sinal de hardware que informa a ocorrência de um evento no sistema.
  - Ex: término de uma operação de E/S.
- Provoca uma mudança no fluxo de controle, o qual é transferido para a **rotina de tratamento de interrupção** correspondente.



Mecanismo de Interrupção (2)

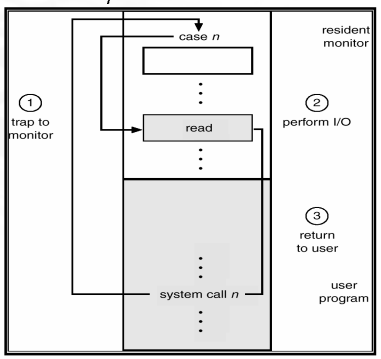


Interrupção de Software (1)

- **Chamadas ao Sistema**
  - Assim como as interrupções de hardware, as chamadas ao sistema (SVCs – **Supervisor Calls**) também provocam uma mudança no fluxo de controle da CPU.
  - Por conta deste comportamento semelhante à manipulação das interrupções de hardware, as SVCs são também referenciadas como "Interrupções de Software".
  - As exceções são ditas "interrupções internas".

Interrupção de Software (2)

- Exemplo 1: SVC de E/S



Processo (1)

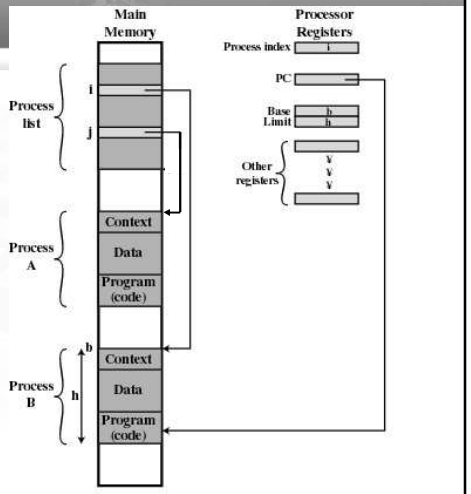
- Abstração usada pelo S.O. para designar a execução de um programa.
- Cada processo é representado no SO por um *Bloco de Controle do Processo*, que consiste de:
  - Um programa executável ("*text section*");
  - Os dados associados ("*data section*"), variaveis globais;
  - Contexto de execução do programa, como: ***program counter***, pilha ("*stack*"), registradores de cpu, arquivos abertos, informações de E/S, etc.

Processo (2)

- É caracterizado por uma ***thread* de execução**, um estado corrente e um conjunto associado de recursos do sistema.
- Um processo é um programa individual em execução (uma instância de um programa rodando em um computador). É também referenciado como "tarefa" (*task*) ou mesmo "*job*".
- O processo é uma entidade ativa (i.e., é um conceito dinâmico), ao contrário do programa.

Processo (3)

- Uma possível implementação de processos



Processo (4)

- Do ponto de vista da UCP, um processo executa instruções do seu repertório em alguma seqüência ditada pelos valores do registrador PC (*program counter*).

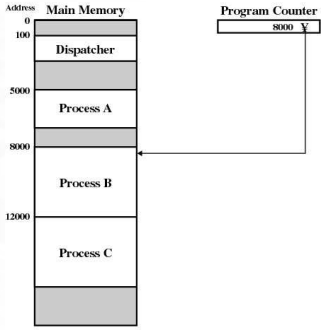


Figure 3.1 Snapshot of Example Execution (Figure 3.3) at Instruction Cycle 13

Processo (5)

- O comportamento de um processo pode ser caracterizado pela seqüência de instruções executadas (*trace*).

(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Figure 3.2 Traces of Processes of Figure 3.1

Processo (6)

- A multiprogramação pressupõe a existência de vários processos disputando o processador.
- Necessidade de algoritmos de escalonamento de processos.

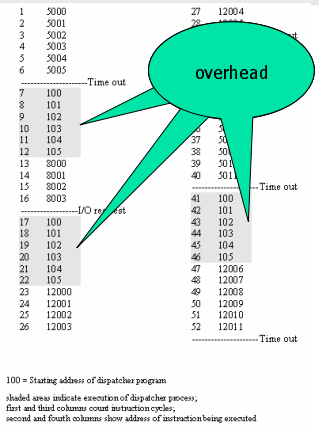


Figure 3.3 Combined Trace of Processes of Figure 3.1

Criação de processo (exemplo)

```
#include <stdio.h>
void main (int argc, char *argv[])
{   int pid;
    pid = fork();
    if (pid < 0)
    { fprintf (stderr, "Fork falhou");
      exit (-1);}
    else if (pid == 0)
    { printf ("processo filho");}
    else
    { printf ("processo pai ");
      wait (NULL);
      printf ("filho concluiu ");
      exit(0);}
}
```

Estados de um Processo

- Durante a sua execução, um processo passa por diversos estados, refletindo o seu comportamento dinâmico, isso é, a sua evolução no tempo.
- Exemplos de estados:
  - *New*: recém criado.
  - *Ready*: pronto para execução.
  - *Running*: em execução.
  - *Blocked*: esperando por um evento.
  - *Exit*: processo terminado.
- Apenas um único processo pode estar no estado "*running*" num dado processador, num dado instante.

Modelo de 5 Estados (1)

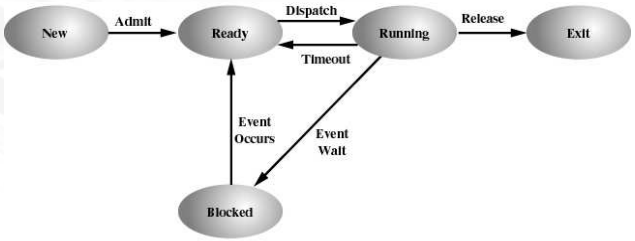


Figure 3.5 Five-State Process Model

Modelo de 5 Estados (2)

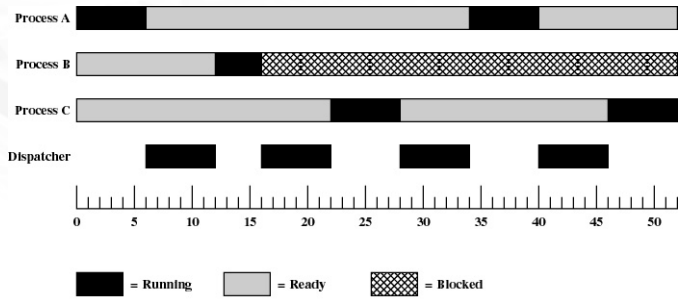


Figure 3.6 Process States for Trace of Figure 3.3

Transições de Estados (1)

- **Null → New:**
  - Um novo processo é criado para executar o programa.
    - Novo *batch job*.
    - *Logon* interativo (usuário se conecta ao sistema).
    - S.O. cria processo para prover um serviço (ex: impressão).
    - Processo cria um outro processo ("*process spawning*").
- **New → Ready:**
  - No estado **New**, recursos foram alocados pelo S.O. mas não existe um compromisso de que o processo será executado.
    - Número de processos já existentes;
    - Quantidade de memória virtual requerida, etc.
  - Manter um bom desempenho do sistema é o fator limitante da criação de novos processos.

Transições de Estados (2)

- **Ready → Running:**
  - Definido pela política de escalonamento de processos adotada pelo S.O.
- **Running → Exit:**
  - Processo terminou as suas atividades ou foi abortado.
    - Término normal;
    - Término do processo pai (em alguns sistemas)
    - Excedeu o limite de tempo;
    - Memória não disponível;
    - Erro aritmético ou de proteção;
    - Execução de instrução inválida ou de instrução privilegiada no modo usuário;
    - Intervenção do S.O.(ex: ocorrência de *deadlock*);

Transições de Estados (3)

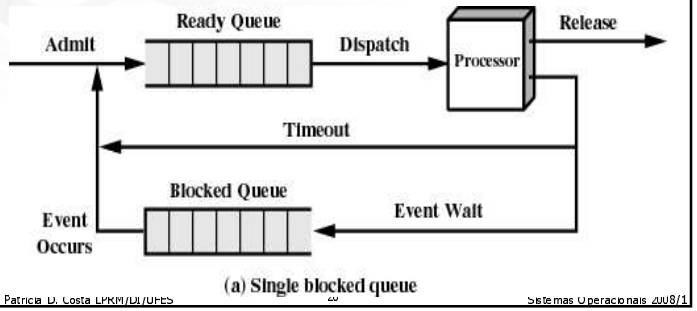
- **Running → Ready :**
  - Tempo máximo de execução sem interrupção foi atingida;
  - Processo é "preemptado" pelo S.O.
- **Running → Blocked:**
  - Processo requisitou alguma coisa pela qual deve esperar
- **Blocked → Ready:**
  - Evento pelo qual o processo espera aconteceu.
- **Ready → Exit:**
  - Processo pai termina um processo filho.
  - Processo pai é terminado, e os processos filhos associados são também finalizados.
- **Blocked → Exit:**
  - Idem anterior.

Transições de Estados (4)

Processo A	Processo B	Processo C
Running	Ready	Ready
Ready	Ready	Ready
Ready	**Running**	Ready
Ready	Blocked	Ready
Ready	Blocked	**Running**
Ready	Blocked	Ready
Running	Blocked	Ready
Ready	Blocked	Ready
Ready	Blocked	**Running**

Filas do Sistema (1)

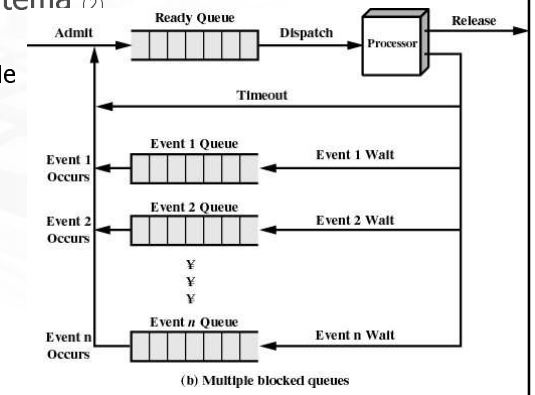
- Um processo sempre faz parte de alguma fila.
- Eventos realizam a transição de uma fila para outra.
- Fila de prontos e uma ou mais filas de bloqueados.



(a) Single blocked queue

Filas do Sistema (2)

- Múltiplas filas de bloqueados



(b) Multiple blocked queues

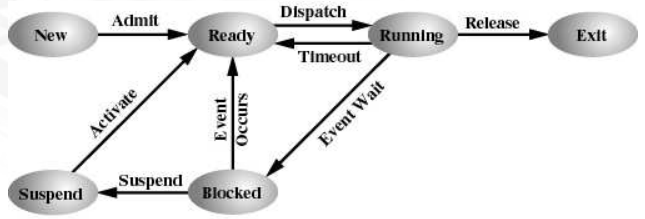
Processos Suspensos (1)

- Processador é tão mais rápido que os dispositivos de E/S que *todos* os processos em memória poderiam ficar em situação de espera.
  - Mesmo com multiprogramação, o processador poderia ficar a maior parte do tempo ocioso!
- Aumento de memória para acomodar mais processos:
  - Aumento do custo;
  - Disponibilidade de mais memória geralmente resulta em processos maiores e não em maior número de processos.
- *Swapping*: procedimento que consiste em mover todo ou parte de um processo da memória para o disco.

Processos Suspensos (2)

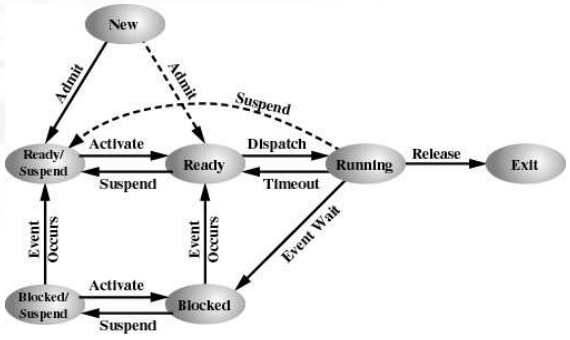
- Quando nenhum dos processos na memória principal está no estado *Ready* o sistema operacional manda um dos processos bloqueados para o disco, e o coloca numa *fila de processos suspensos*.
- O S.O traz então do disco algum outro processo da fila de suspensos ou atende a uma solicitação de criação de um novo processo.
- O *swap* é uma operação de E/S e, como tal, existe a possibilidade de tornar o problema ainda pior, caso o sistema de E/S não seja eficiente.
- Dois novos estados são então incluídos no modelo:
  - *Blocked, suspend*: o processo está em memória secundária e aguardando por um evento.
  - Ready, suspend: o processo está em memória secundária mas está disponível para execução, tão logo ele seja carregado na memória principal.
  - OBS: [*Blocked*: processo está na MP e aguardando por um evento]

Processos Suspensos (3)



(a) With One Suspend State

Processos Suspensos (4)



(b) With Two Suspend States

Máquina de Estados do Unix (1)

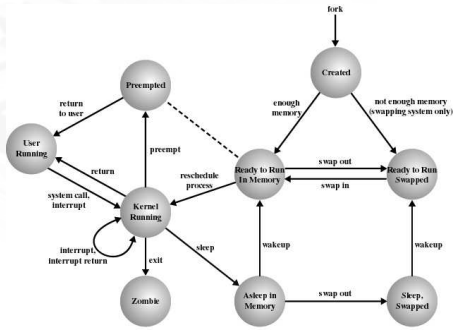


Figure 3.16 UNIX Process State Transition Diagram

Máquina de Estados do Unix (2)

User Running	Executing in user mode.
Kernel Running	Executing in kernel mode.
Ready to Run, in Memory	Ready to run as soon as the kernel schedules it.
Asleep in Memory	Unable to execute until an event occurs; process is in main memory (a blocked state).
Ready to Run, Swapped	Process is ready to run, but the swapper must swap the process into main memory before the kernel can schedule it to execute.
Sleeping, Swapped	The process is awaiting an event and has been swapped to secondary storage (a blocked state).
Preempted	Process is returning from kernel to user mode, but the kernel preempts it and does a process switch to schedule another process.
Created	Process is newly created and not yet ready to run.
Zombie	Process no longer exists, but it leaves a record for its parent process to collect.