Laboratório de Pesquisa em Redes e Multimídia

Sistemas Operacionais

Processos - conceitos básicos

Universidade Federal do Espírito Santo
Departamento de Informática

Processo (1)

- **Abstração** usada pelo S.O. para designar a execução de um programa.
  - 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.
- Cada processo é representado no SO por **estruturas de controle** (ex: bloco de controle de processo).

Processo (3)

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

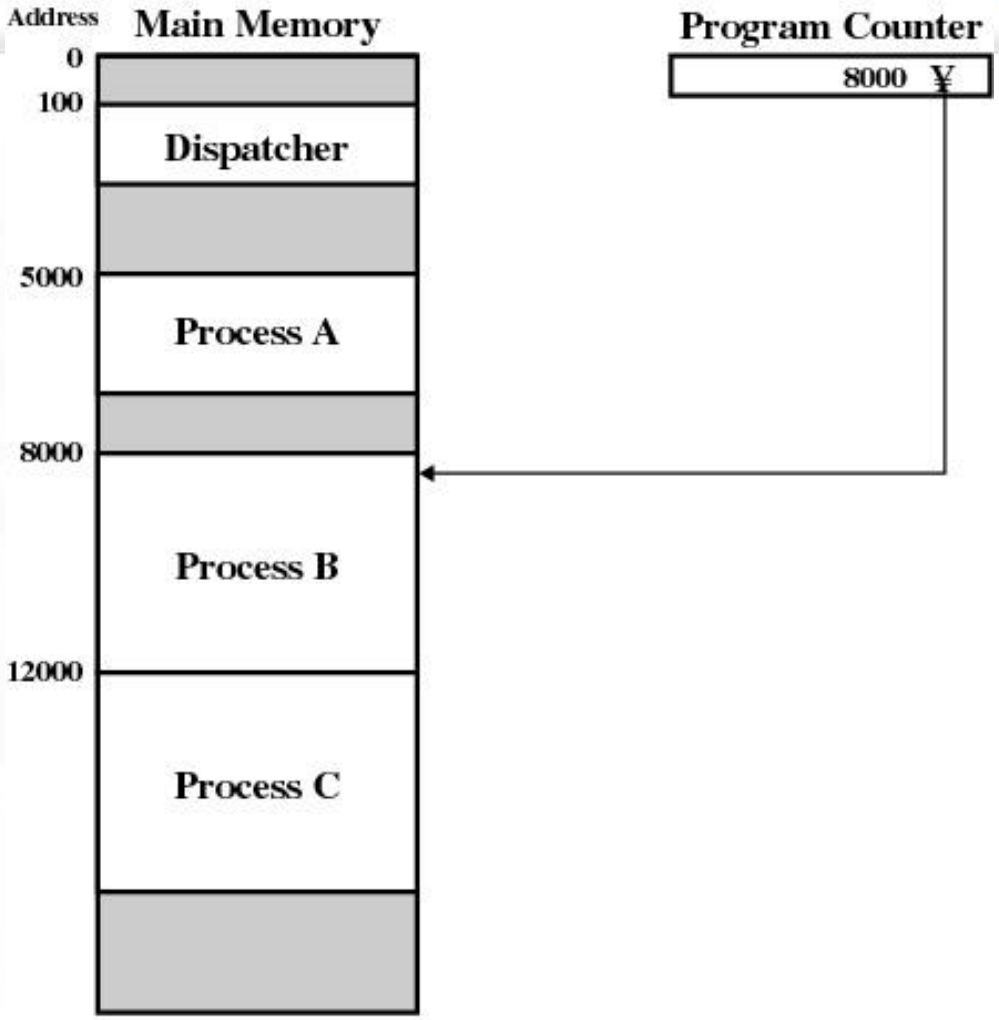


Figure 3.1 Snapshot of Example Execution (Figure 3.3) at Instruction Cycle 13

Processo (4)

- 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 (5)

- A multiprogramação pressupõe a existência de vários processos disputando o processador.
- Necessidade de algoritmos de escalonamento de processos.
- Nas linhas em "cinza" temos código do SO sendo executado, para fazer o escalonamento e outras tarefas de gerência
  - Isso é OVERHEAD!

1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
----------	Time out
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
----------	I/O request
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003
27	12004
28	12005
----------	Time out
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
----------	Time out
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
----------	T...ut

overhead

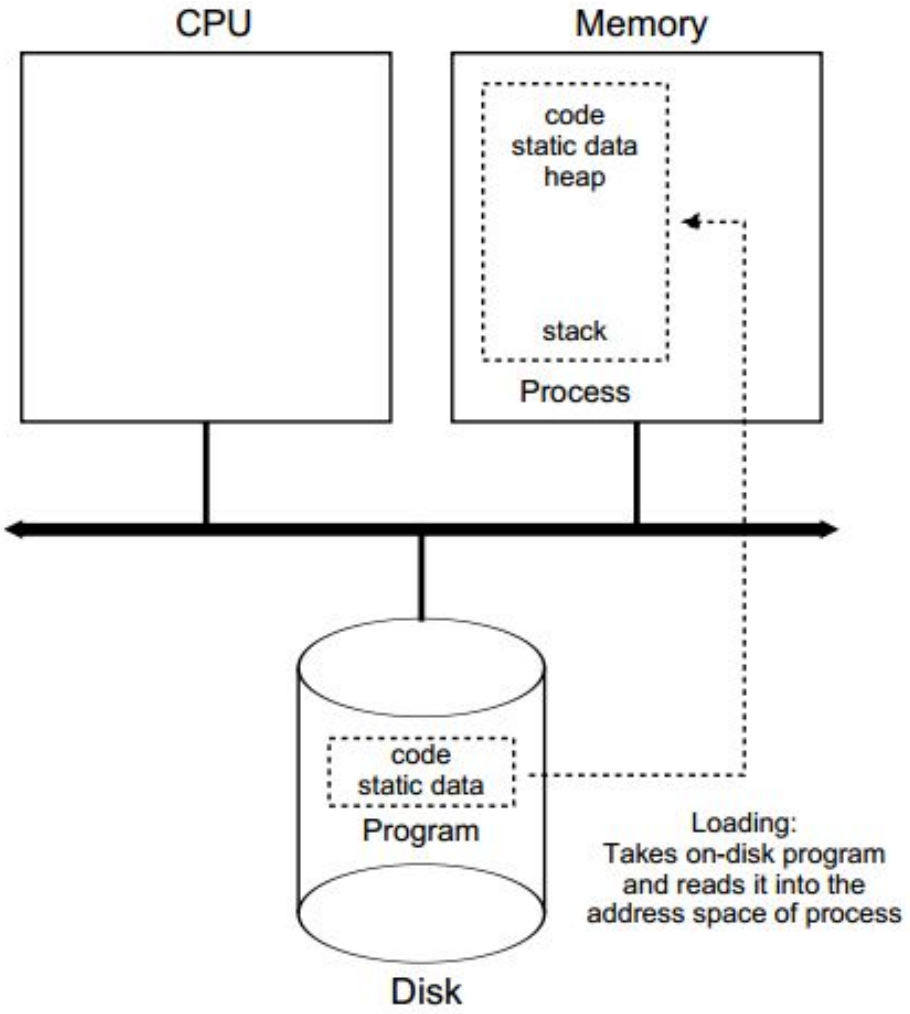
100 = Starting address of dispatcher program

shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

Figure 3.3 Combined Trace of Processes of Figure 3.1

Criação de processo

- Um processo é criado, geralmente "a pedido" de outro processo.
  - Esse processo "pai" faz uma solicitação ao kernel para que ele crie um processo "filho"
- Após sua criação , um processo pode começar a executar



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)

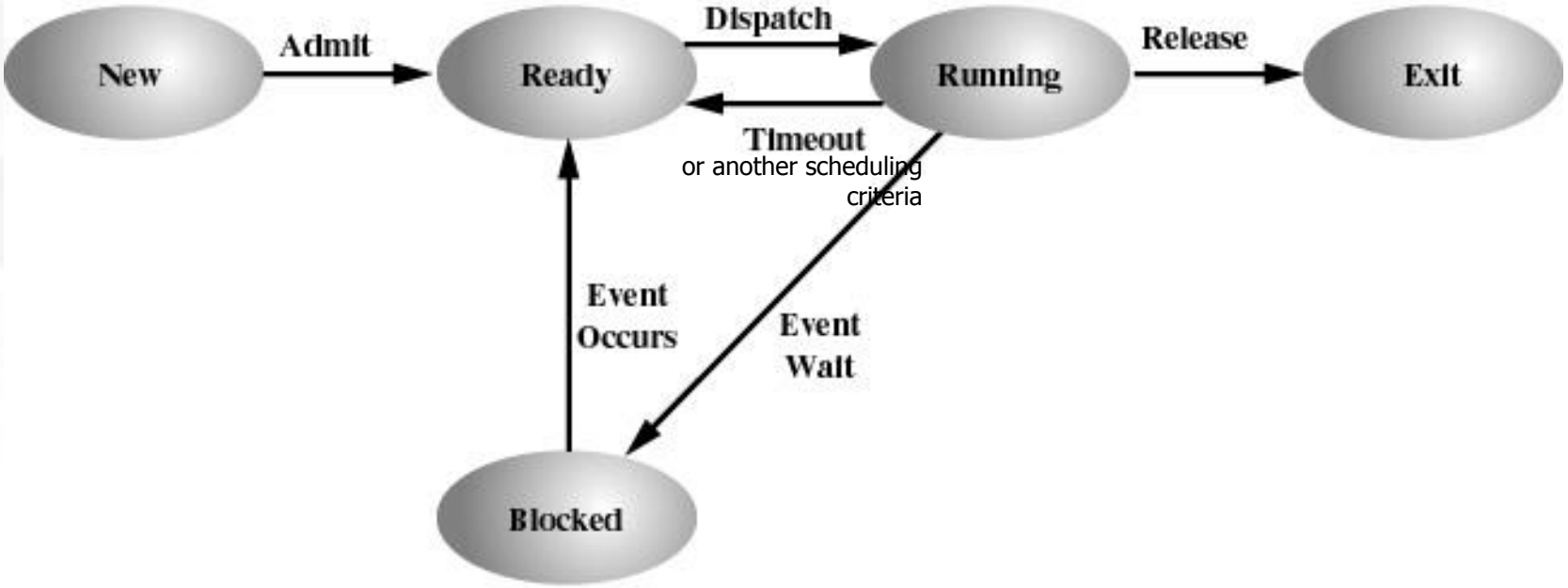


Figure 3.5 Five-State Process Model

Modelo de 5 Estados (2)

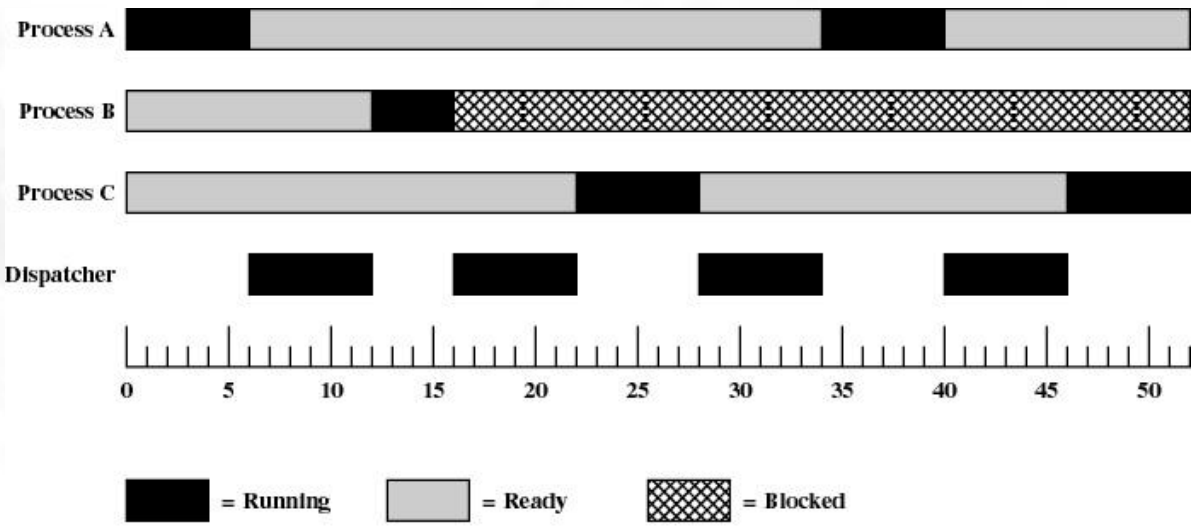


Figure 3.6 Process States for Trace of Figure 3.3

1	5000	27	12004
2	5001	28	12005
3	5002	----------	Time out
4	5003	29	100
5	5004	30	101
6	5005	31	102
----------	Time out	32	103
7	100	33	104
8	101	34	105
9	102	35	5006
10	103	36	5007
11	104	37	5008
12	105	38	5009
13	8000	39	5010
14	8001	40	5011
15	8002	----------	Time out
16	8003	41	100
----------	I/O request	42	101
17	100	43	102
18	101	44	103
19	102	45	104
20	103	46	105
21	104	47	12006
22	105	48	12007
23	12000	49	12008
24	12001	50	12009
25	12002	51	12010
26	12003	52	12011
		----------	Time out

100 = Starting address of dispatcher program

shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

Figure 3.3 Combined Trace of Processes of Figure 3.1

https://www.youtube.com/watch?v=bta48Ix3t_Q&gl=BR&hl=pt

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.

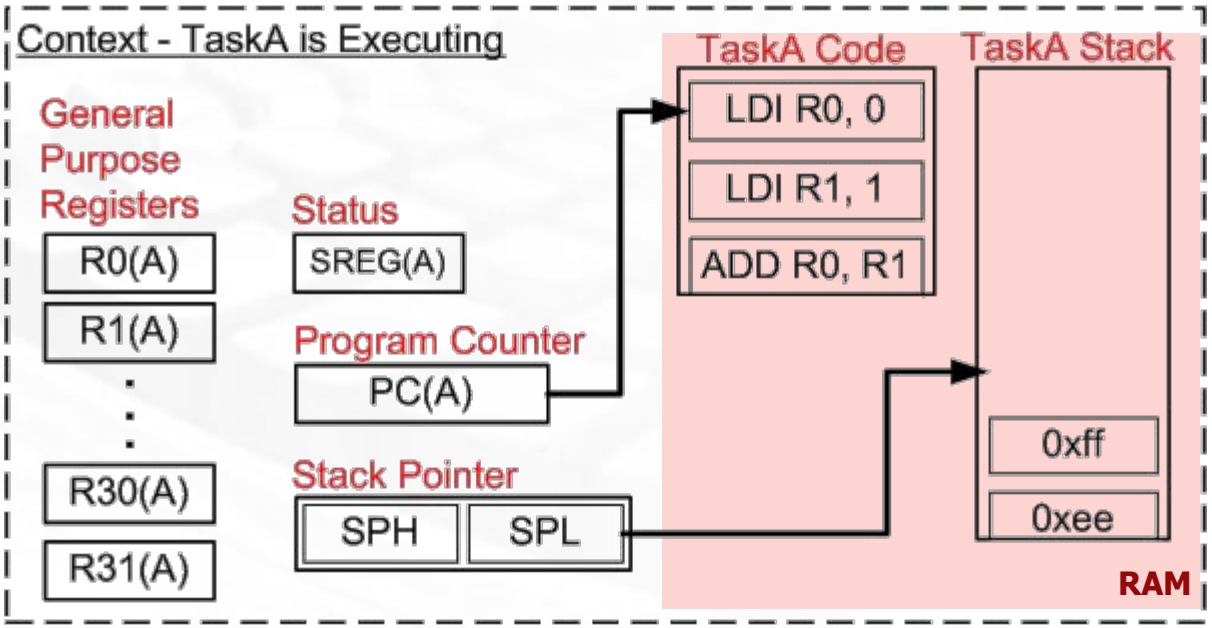
Transições de Estados (2)

- **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 :**
  - Processo é "**preemptado**" pelo S.O
  - Por exemplo:
    - o tempo máximo de execução na CPU foi atingido
    - outros critérios de escalonamento podem ser usados
- **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 (isso pode ocorrer em alguns SOs)
- **Blocked → Exit:**
  - Idem anterior.
  - Na realidade existe uma transição para o estado Exit **a partir de todos os outros estados!**

Contexto (de execução) (1)



Essas informações são essencialmente dados armazenados nos registradores da CPU e na cache no momento em que o processo está em execução

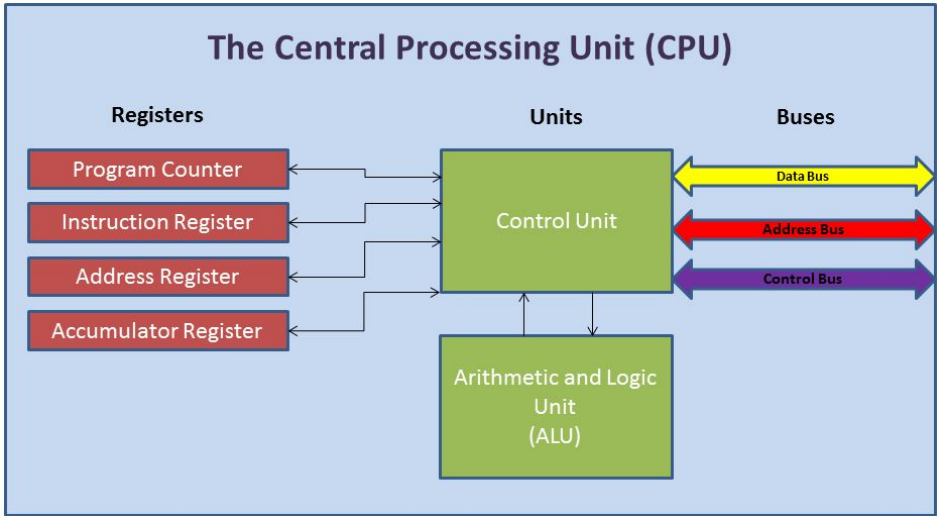
- *Contexto* de um processo são todas as **informações** necessárias para que o S.O. possa restaurar a execução do processo **a partir do ponto em que ele foi "interrompido".**
  - **Interrompido** aqui significa o momento em que ele parou de rodar (sai do estado *Running*, i.e., deixa de executar na CPU)

Troca de Contexto (1)

- A troca de contexto ocorre **sempre que um novo processo é selecionado para execução** (isso é, quando a UCP é chaveada para um outro processo).
  - A este "chaveamento" damos o nome de Escalonamento de Processos
- O tempo de troca de contexto é puro ***overhead*** e é dependente de suporte de hardware (ex: salvamento automático do *Program Counter- PC*).

Troca de Contexto (2)

- Execução do **Escalonamento de Processos**
  - Escalonamento de processos é a tarefa de alternar a CPU entre dois processos
- Sempre que for ocorrer um escalonamento, ocorrerá a **troca de contexto**
  - O tempo (duração) da troca de contexto depende muito do hardware
    - O tempo (duração) da troca de contexto depende muito do hardware:
      - Velocidade da memória, no. de registradores, instruções especiais de carga de registradores.
      - 1 a 1000 microseg.

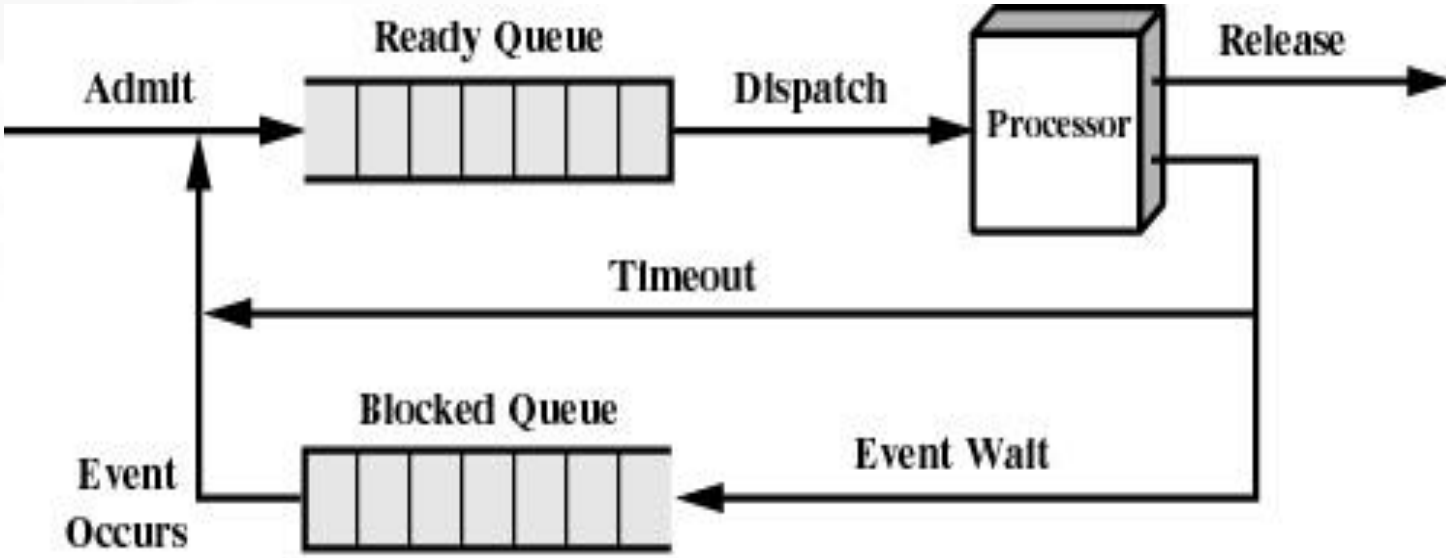


Troca de Contexto (3): quando ocorre?

- Chamada de Sistema (SVC)
- Interrupção do relógio
  - Fatia de tempo de "posse da UCP" expirou;
- Interrupção de E/S (hardware)
- Falta de memória (qdo houver mem. virtual)
  - Endereço de memória está na memória virtual (disco); logo deve ser trazido para a memória principal.
- Trap
  - Ocorrência de erro.
  - Pode fazer com que o processo seja movido para o estado *Exit*.

Filas do Sistema (1)

- Um processo sempre faz parte de alguma fila
  - A **exceção** é quando ele está em execução!
- Eventos ocasionam a transição de uma fila para outra.
- Exemplo **clássico (e simplista!!):** Uma fila de processos prontos e uma fila de processos bloqueados.

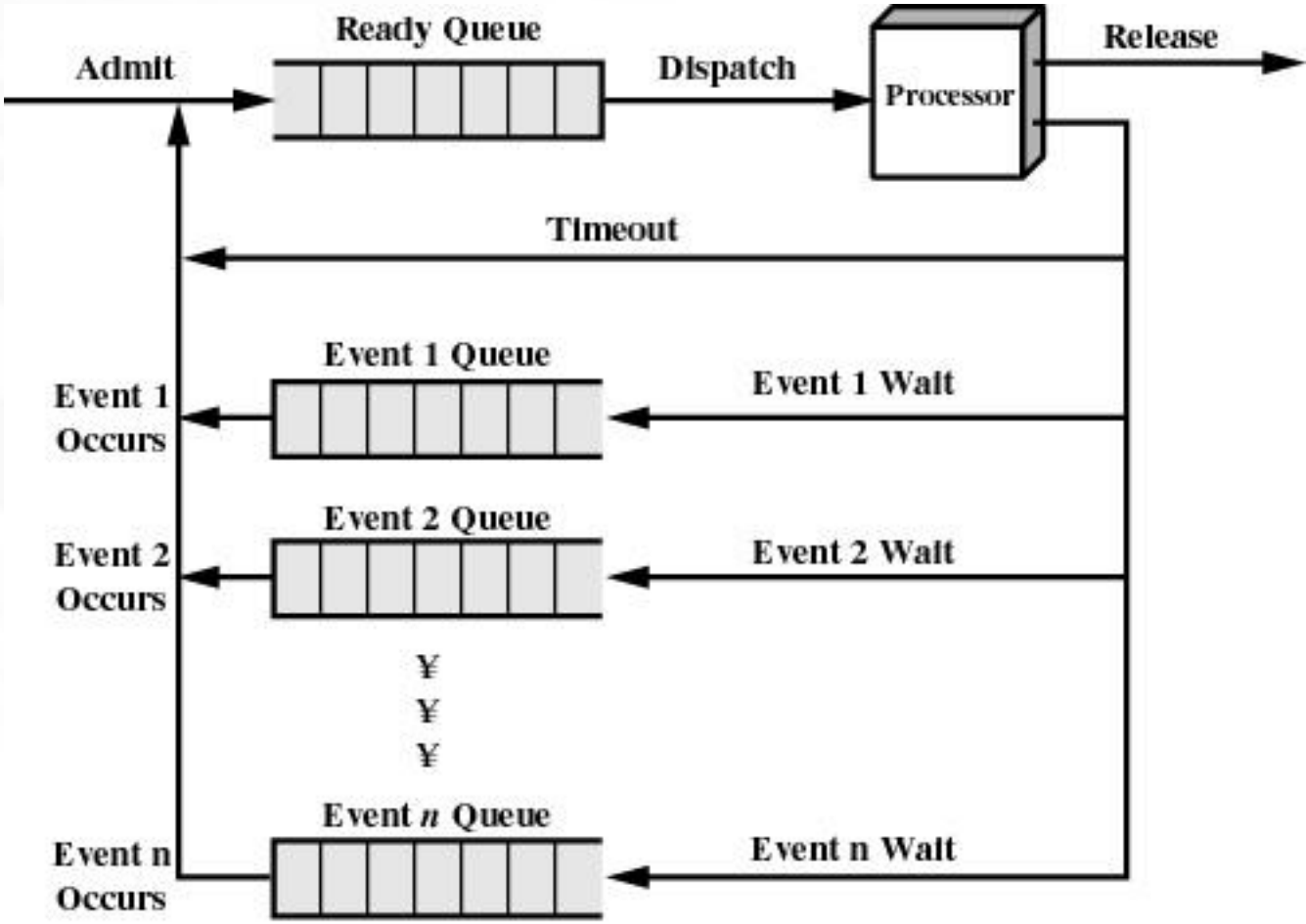


(a) Single blocked queue

Filas do Sistema (2)

- Múltiplas filas de bloqueados

... mas nos SOs mais utilizados... também teremos várias filas de prontos... veremos mais no curso!!

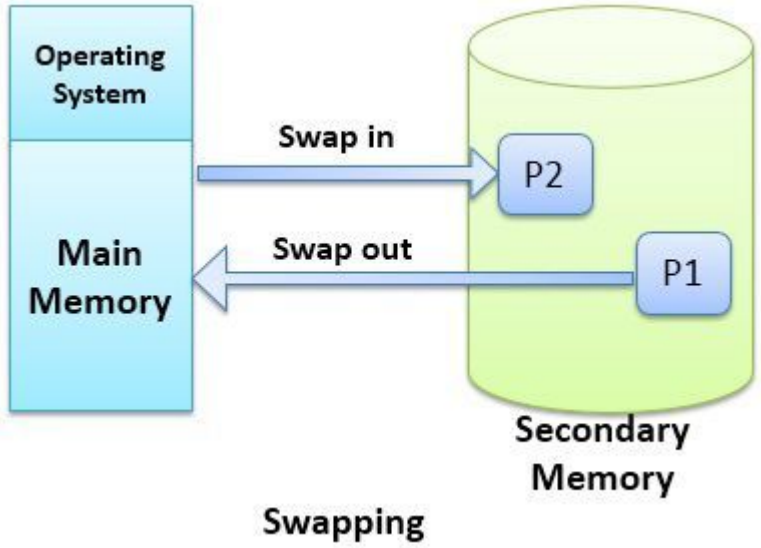


(b) Multiple blocked queues

Um novo estado!!!

Processos Suspensos (1)

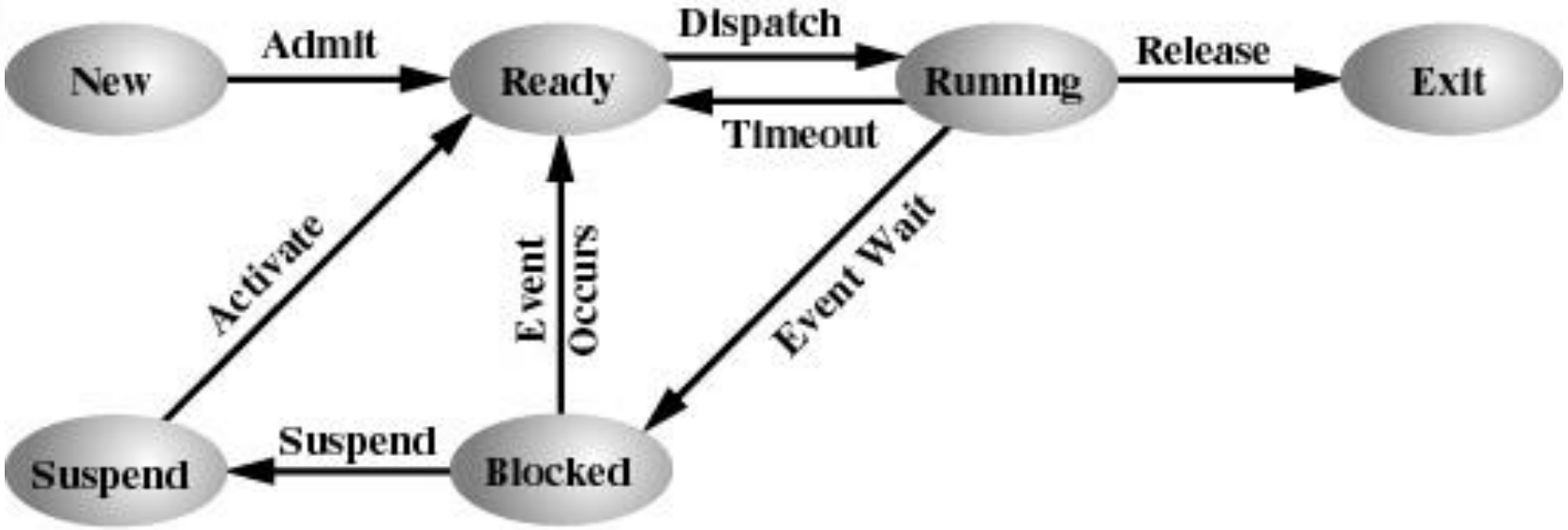
- Processador é tão mais rápido que os dispositivos de E/S que *todos* os processos em memória poderiam ficar em situação de espera (bloqueados).
  - Mesmo com multiprogramação, o processador poderia ficar a maior parte do tempo ocioso!
- Aumento de memória RAM para acomodar mais processos!?!
  - Aumento 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.



Swapping

Processos Suspensos (2)

- Na falta de memória RAM disponível, e havendo novos processos para serem criados (esperando memória... no estado New):
  - Se **nenhum** dos processos na memória principal está no estado ***Ready/Running***, a CPU está ociosa.
  - O SO manda um dos processos **bloqueados para o disco**, e o coloca numa *fila de processos suspensos*.

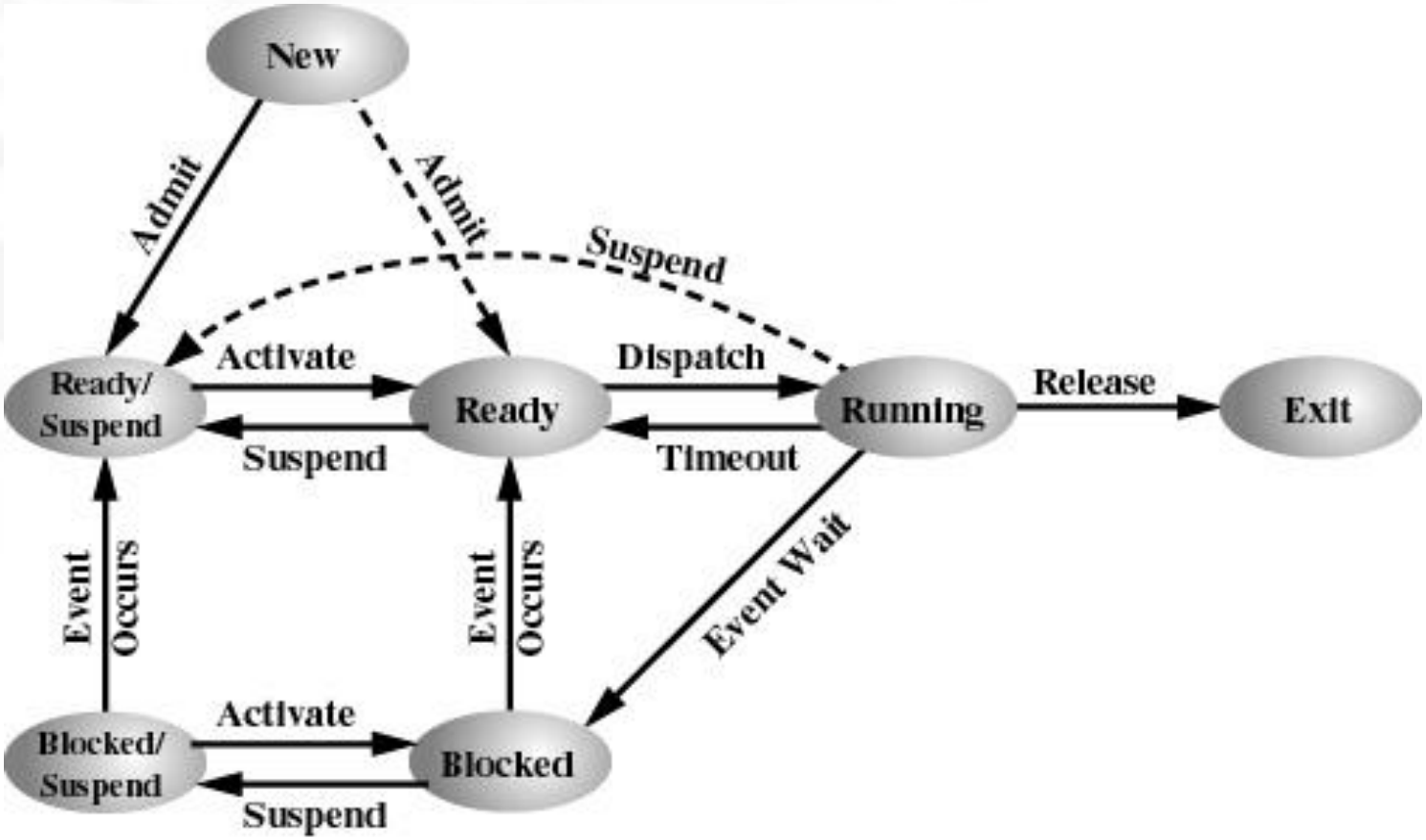


(a) With One Suspend State

Processos Suspensos (3)

- 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.
- **Modelo mais elaborado**: dois novos estados são então criados:
  - *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
  - OBS1: *Blocked*: processo está na MP (RAM) e aguardando por um evento
  - OBS2: Em sistemas mais modernos (mais memória!) os processos Suspended podem ser mantidos em memória

Processos Suspensos (4)



(b) With Two Suspend States

Processos:
Estruturas de Controle

Processos e Recursos (2)

- O S.O. gerencia recursos computacionais em benefício dos diversos processos que executam no sistema.
- A questão fundamental é:
  - Que **informações** o sistema operacional precisa manter para poder **controlar** os processos e **gerenciar** os recursos em benefícios deles?

Tabelas de Controle do S.O.

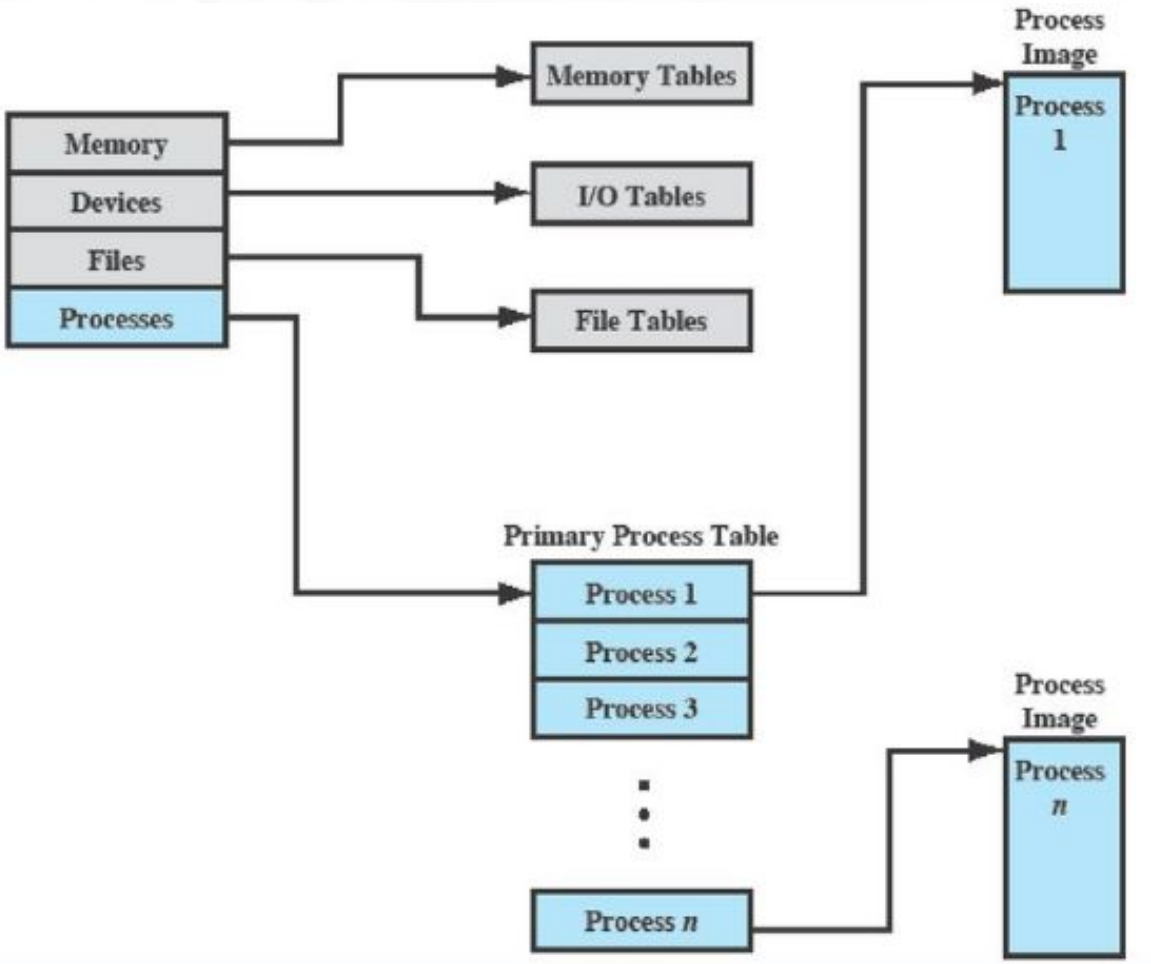


Imagem do Processo (1)

- Elementos típicos da Imagem do processo

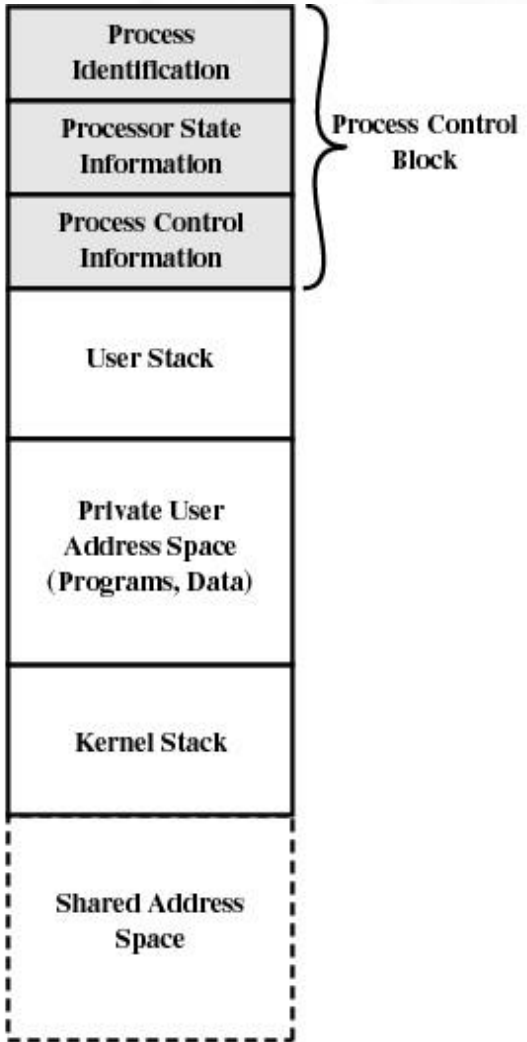
User Data
The modifiable part of the user space. May include program data, a user stack area, and programs that may be modified.

User Program
The program to be executed.

Stack
Each process has one or more last-in-first-out (LIFO) stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls.

Process Control Block
Data needed by the OS to control the process.

Imagem do Processo (2)

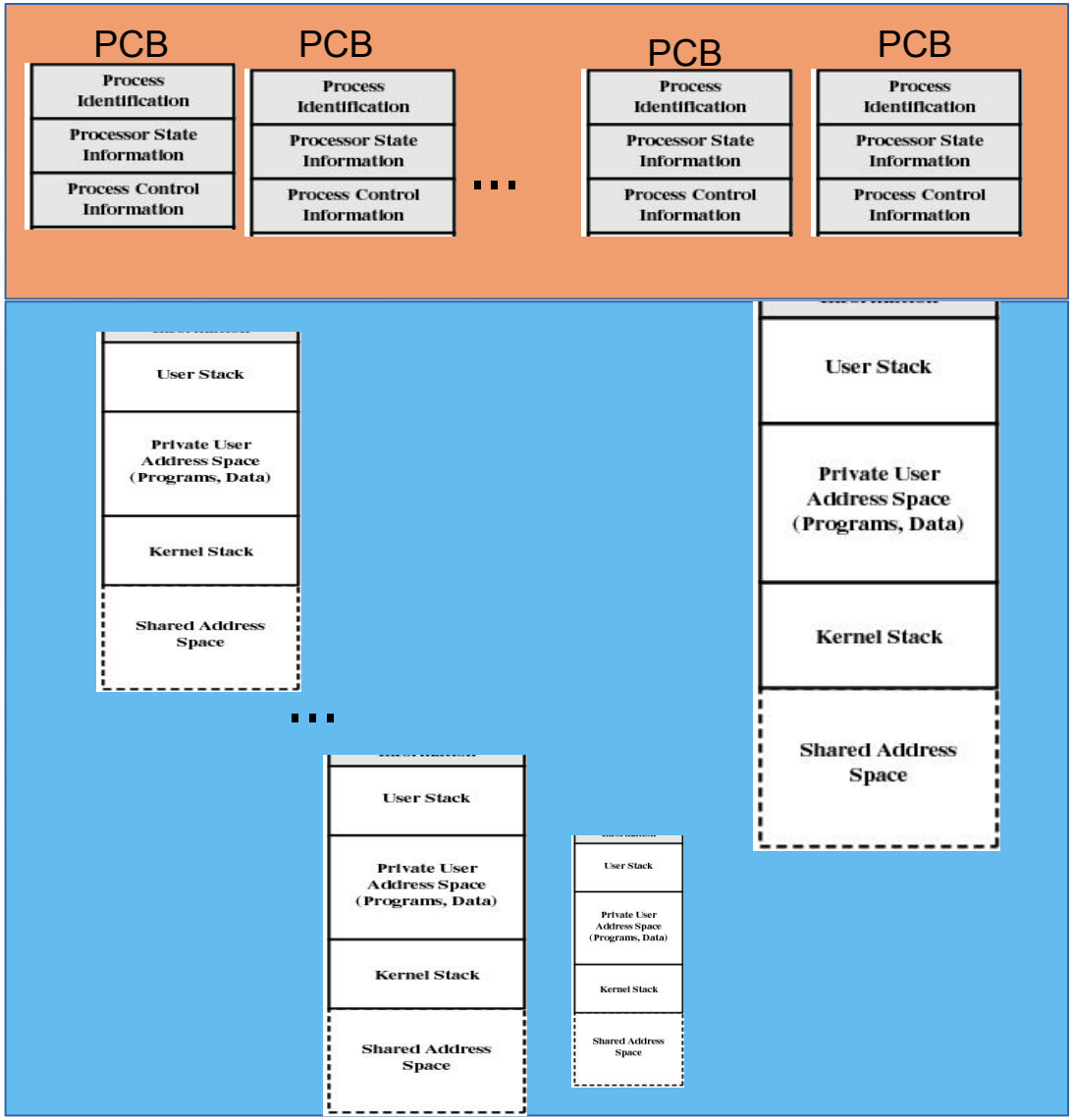


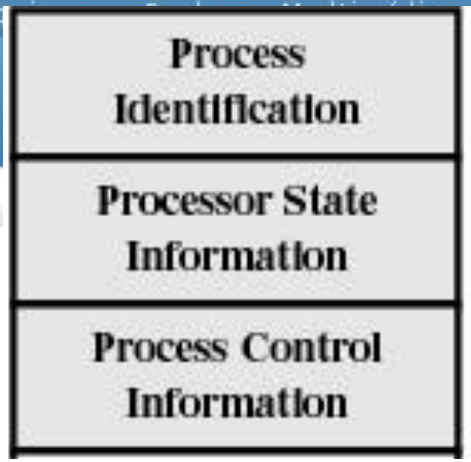
- **PCB - Process Control Block**
  - ou Bloco de Controle de Processo
  - Todas as informações que o S.O. precisa para poder controlar a execução do processo (atributos do processo)
    - **Identificação**
    - **Contexto de Execução**
    - **Outras infos de controle**
  - Número fixo ou variável de PCBs no SO
    - Alocação estática ou dinâmica

RAM

Memória do SO

Imagem do Processo (3)



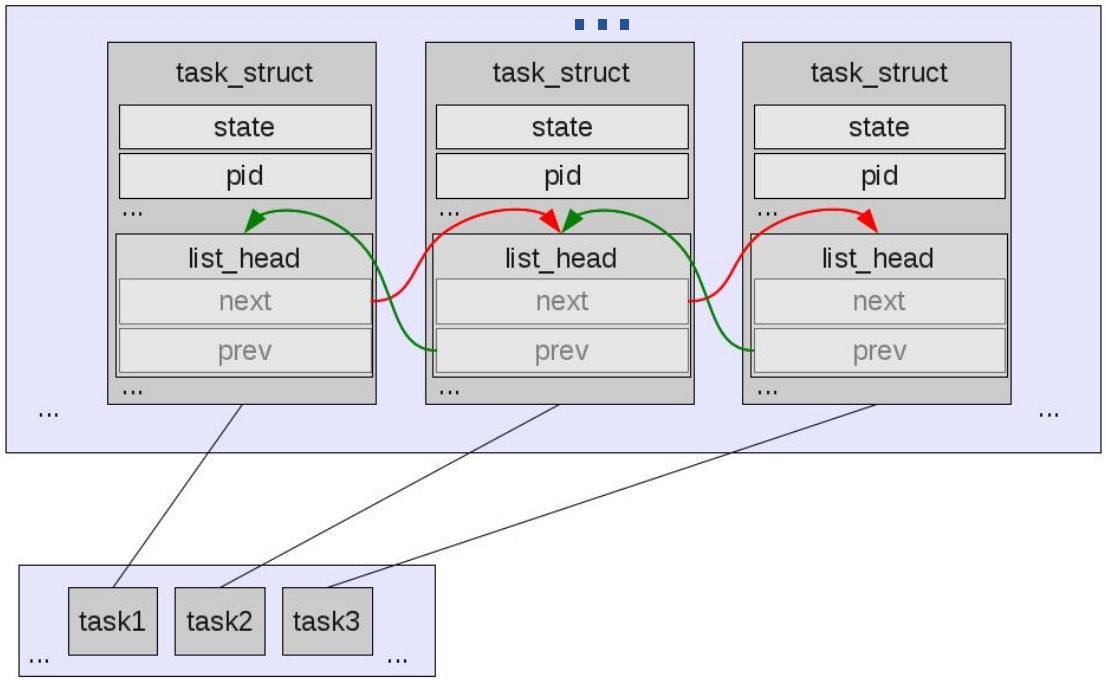


PCB - Process Control Block

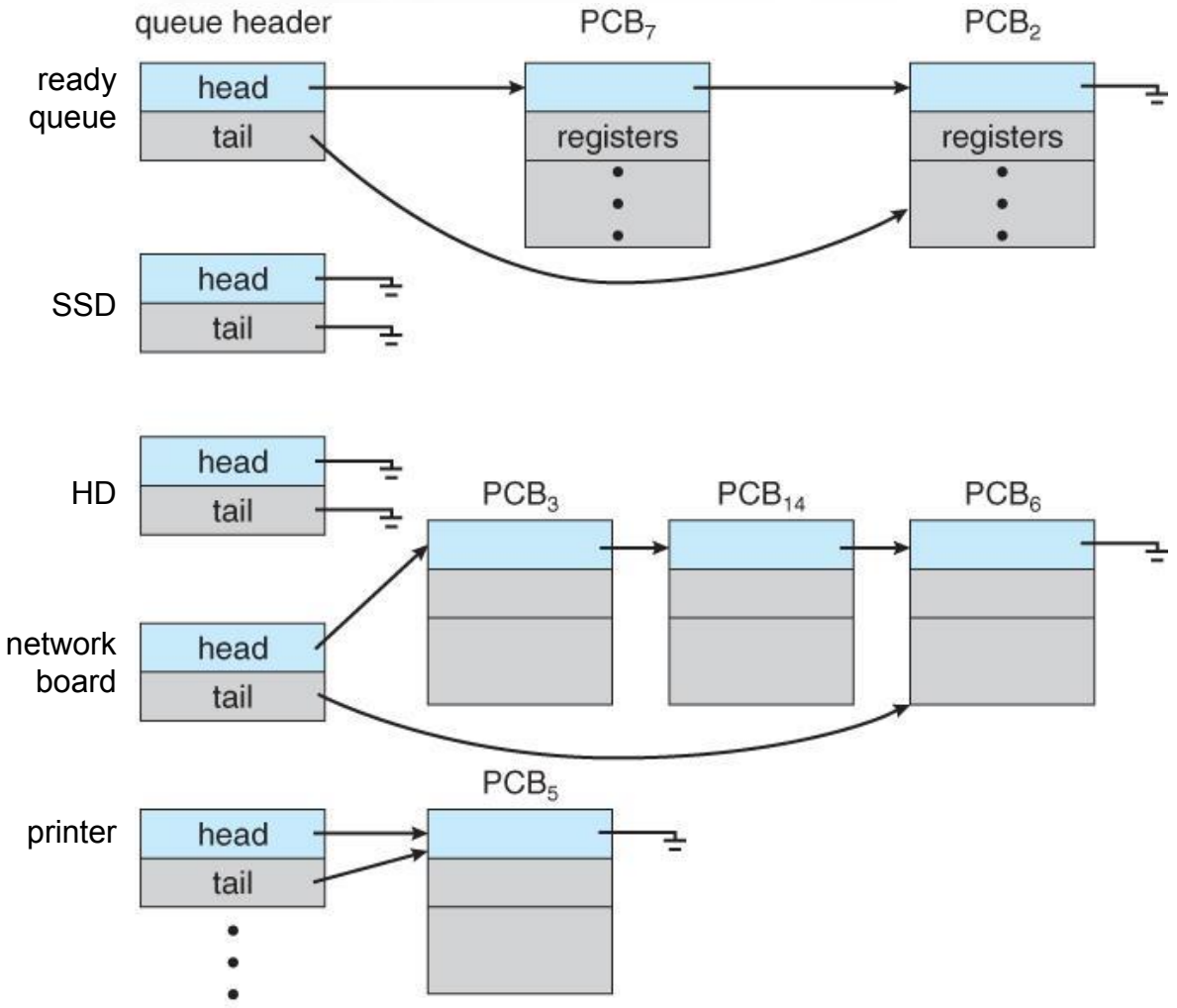
- **Identificação do processo**
  - ID do processo, do processo pai, do usuário...
- Informações de **Contexto de execução**
  - Registradores visíveis ao usuário
  - Reg. de controle/estado: PC, SP, Flags, *Status* (modo supervisor /usuário, interrupção habilitada /disabilitada)...
- Informações de **Controle do Processo**
  - Estado do processo (*ready*, *running*, *suspended*, etc.)
  - Prioridade (*default*, corrente, máxima)
  - Tempos de fila
  - Gerência de memória
  - Ownership (quem é o user...)
  - etc.

Exemplo... PCB no Linux: Estrutura C `task_struct`

```
pid_t pid; /* process identifier */
long state; /* state of the process */
unsigned int time_slice /* scheduling information */
struct task_struct *parent; /* this process's parent */
struct list_head children; /* this process's children */
struct files_struct *files; /* list of open files */
struct mm_struct *mm; /* address space of this process */
...
```



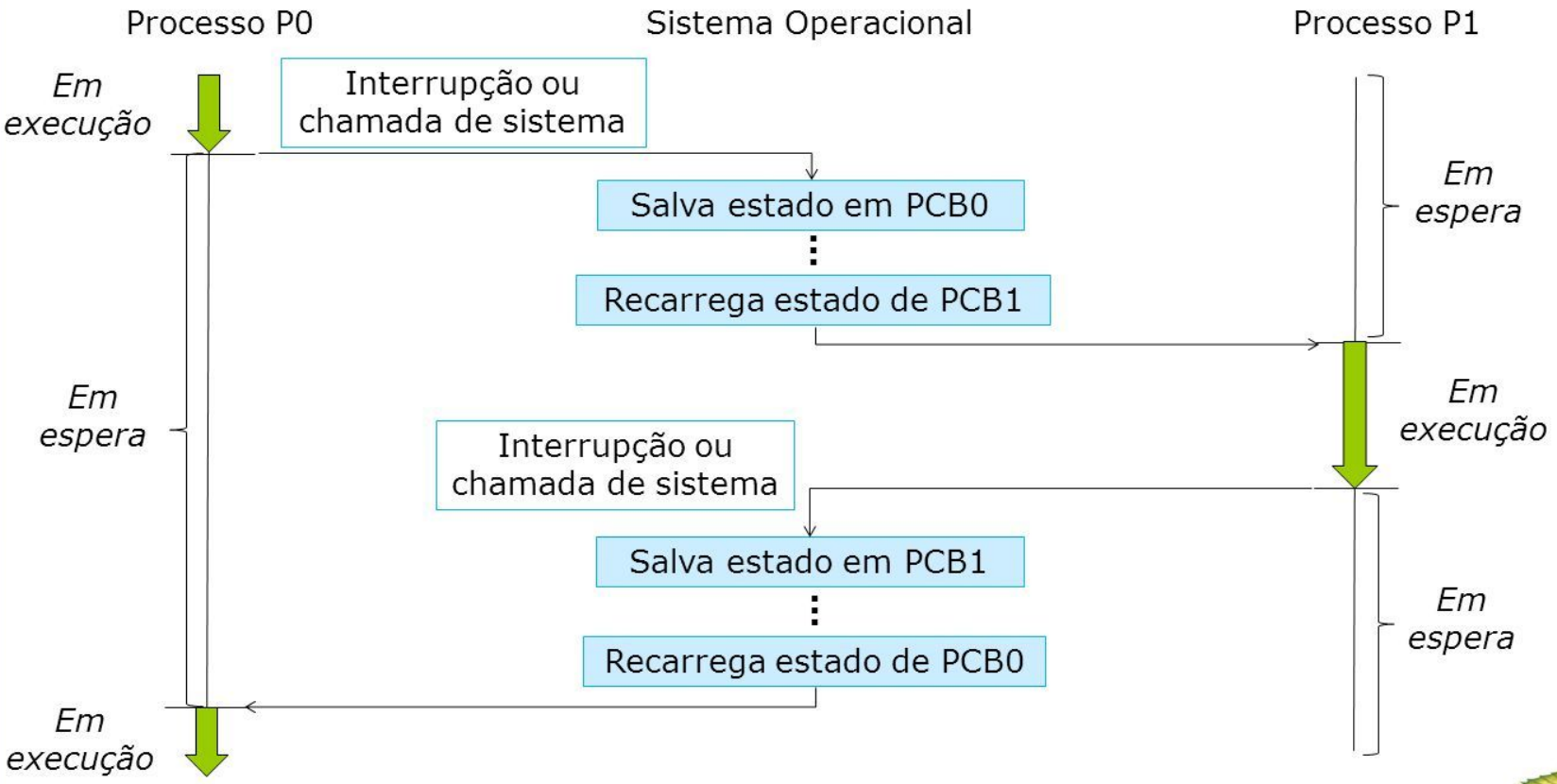
PCBs e as Filas do Sistema



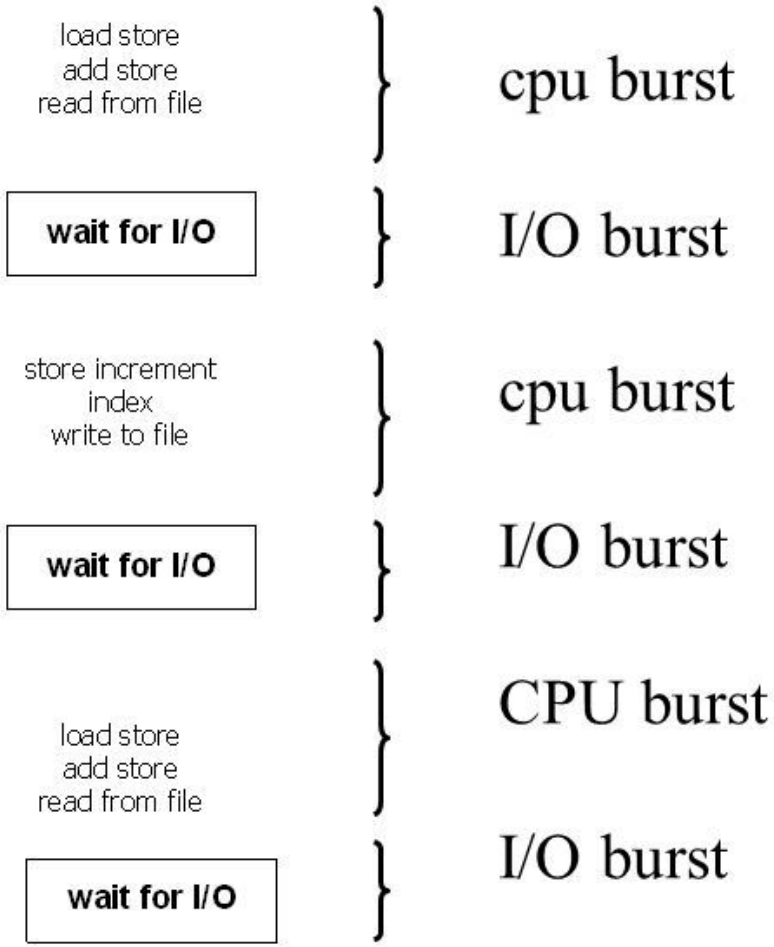
Troca de Contexto (1)

- **Contexto de execução:** estado do processador
  - Valores nos registradores
- Ações na troca de contexto
  1. **Salvar o contexto** do processador, incluindo o PC e outros registradores.
  2. **Alterar o PCB** do processo que estava no estado "em-execução" (*running*)
     1. - (Por exemplo, ele pode estar indo pra "Blocked")
  3. **Mover o PCB** para a fila apropriada.
  4. **Selecionar outro processo** para execução.
  5. **Alterar o PCB** do processo selecionado.
  6. **Alterar as tabelas** de gerência de **memória**.
  7. **Restaurar o contexto do processo selecionado.**

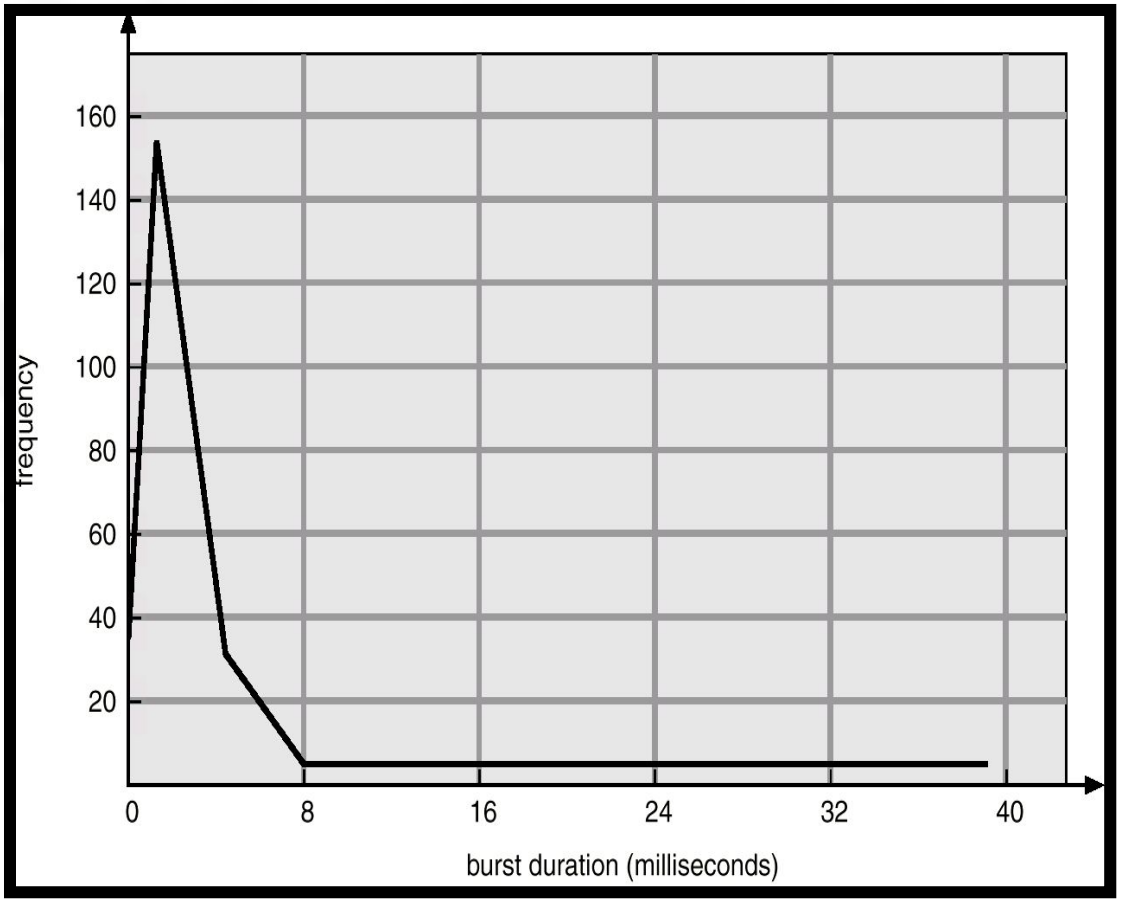
Troca de contexto



Ciclos de CPU e de I/O (1)



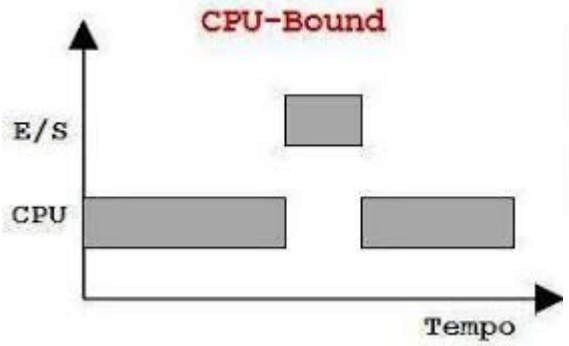
Ciclos de CPU e de I/O (2)

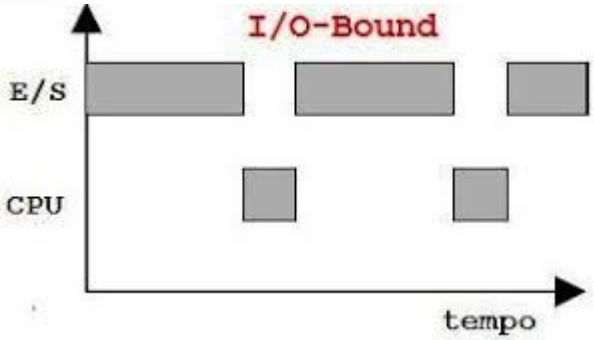


Estatisticamente, a maior parte consiste em CPU bursts pequenos !!

Tipos de Processos

- a) Processo **CPU Bound**:
  - Uso intensivo de CPU.
  - Realiza pouca operação de E/S.
  - Pode monopolizar a CPU, dependendo do algoritmo de escalonamento.
- b) Processo **I/O Bound**:
  - Orientado a I/O.
  - Normalmente, devolve deliberadamente o controle da CPU





Referências

- W. Stallings, Operating Systems, sixth edition, Chapter 3 (3.3), p.126-135, Process Description.

Complementares:

- A. Sylberschatz, Operating Systems, 6th edition, Chapter 4 (4.1 to 4.2), p.95-103, Process.
- C. Maziero, Sistemas Operacionais, Capítulo 5 (5.1 a 5.2), p.51-54, Implementação de Tarefas.