Sistemas Operacionais

Processos

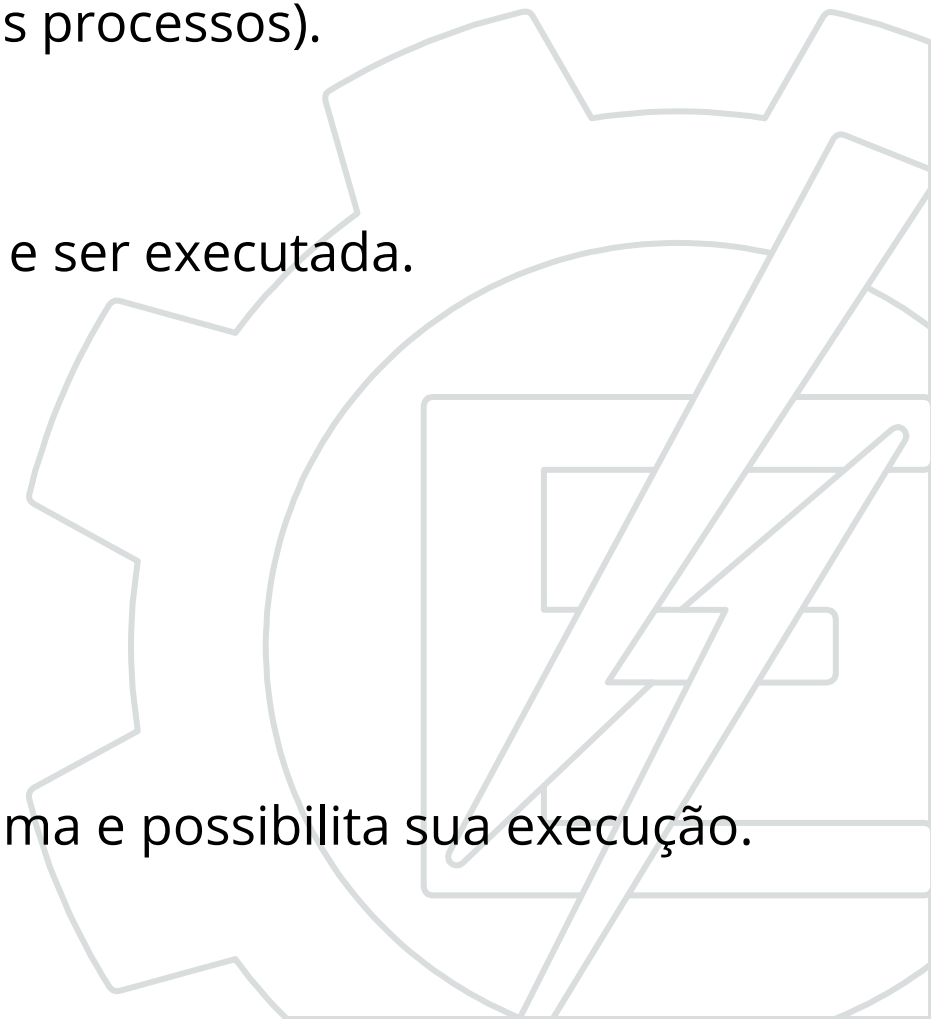
Prof. Otávio Gomes

otavio.gomes@unifei.edu.br

Processos
versus Programas

- **Programa**:
  - Pode ter várias instâncias em execução (em diferentes processos).
  - É um algoritmo codificado.
  - Representa a forma como o programador vê a tarefa e ser executada.

- **Processo**:
  - É único.
  - Código acompanhado de dados e estado.
  - Forma pela qual o Sistema Operacional vê um programa e possibilita sua execução.



Processos

- Processo em Primeiro Plano (*foreground*): Interage diretamente com o usuário.
- Processo em Segundo Plano (*background* - ***daemon***): possuem funções específicas que independem do usuário.

- Cada processo possui:
  - Conjunto de instruções (código executável);
  - Espaço de endereçamento (espaço reservado/exclusivo para que o processo possa ler e escrever);
  - Contexto de *hardware* (valores nos registradores: PC, Pilha, etc);
  - Contexto de *software* – descritores de S.O.(atributos como lista de arquivos abertos, variáveis, segurança, estado do processo, etc.;

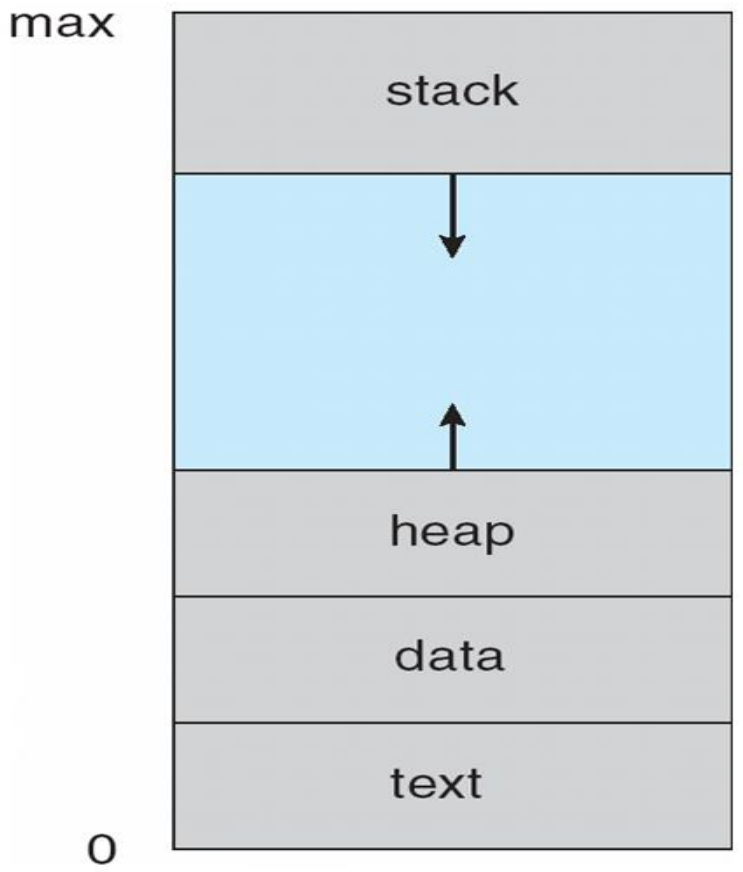
Processos

- Um sistema operacional executa uma variedade de programas:
  - Sistemas *Batch* (lote) – *Jobs* (tarefas)
  - Sistemas de Tempo Compartilhado – programas do usuário ou tarefas

- **Processo – um programa em execução**

  sua execução deve progredir de forma sequencial

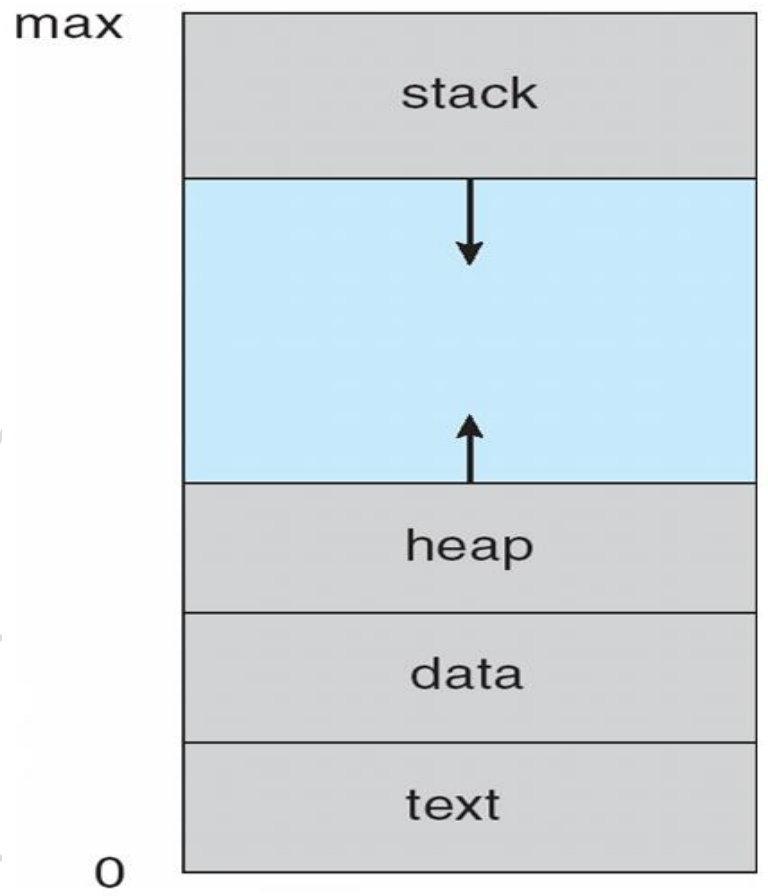
- Um processo possui:
  - Contador de programa
  - Pilha (*stack*)
  - Seção de dados



Processos

Espaço de endereçamento

- Texto: código executável do(s) programa(s);
- Dados: as variáveis;
- Pilha de Execução:
  - Controla a execução do processo
  - Empilha as chamadas a procedimentos, seus parâmetros e variáveis locais, etc.



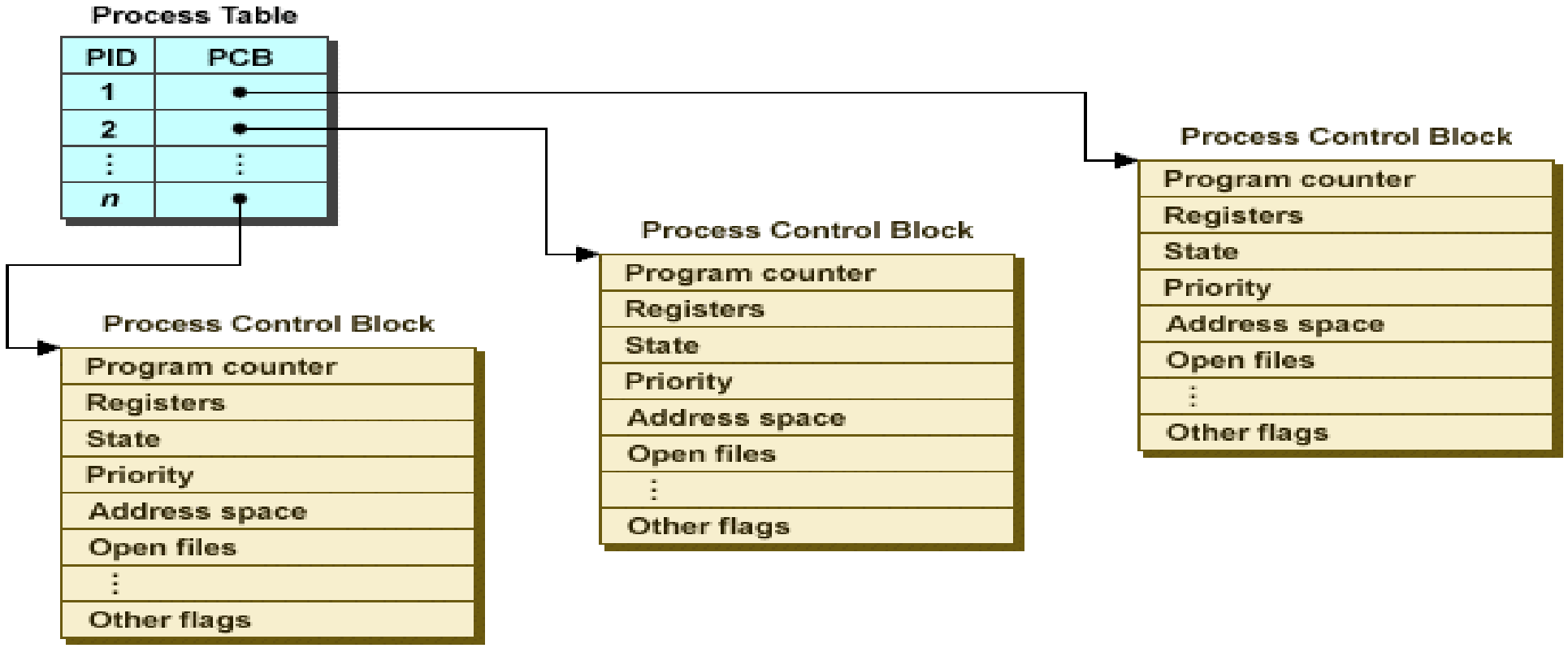
- **Gerenciador por admissão:** escolhe qual processo irá executar e quando, de acordo com sua prioridade. Geralmente utilizado em sistemas de tempo real;
- **Gerenciador por tempo médio;**
- **Gerenciador despachante:** define os processos a serem executados de acordo com os eventos ocorridos no sistema.

- Os gerenciadores possuem algoritmos para garantir a troca de processos de maneira correta e organizada. São exemplos de estratégias utilizadas: FIFO; *Pipes* unidirecionais (FIFO controlada pelo S.O.); Fila de mensagem; memória compartilhada.

Processos

Tabela de Controle de Processos (PCB)

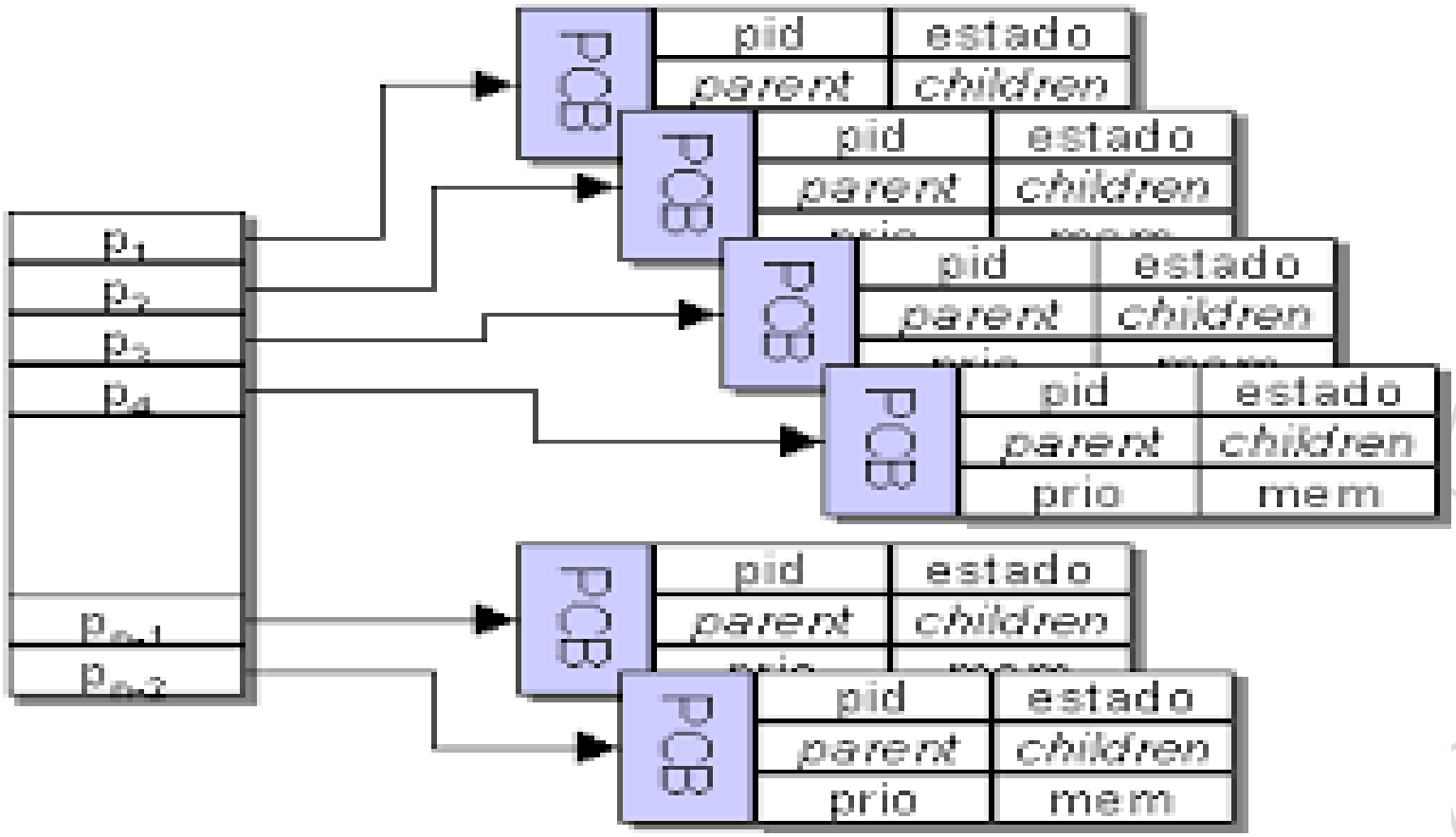
- A tabela não guarda o conteúdo do espaço de endereçamento do processo.



Processos

Tabela de Controle de Processos (PCB)

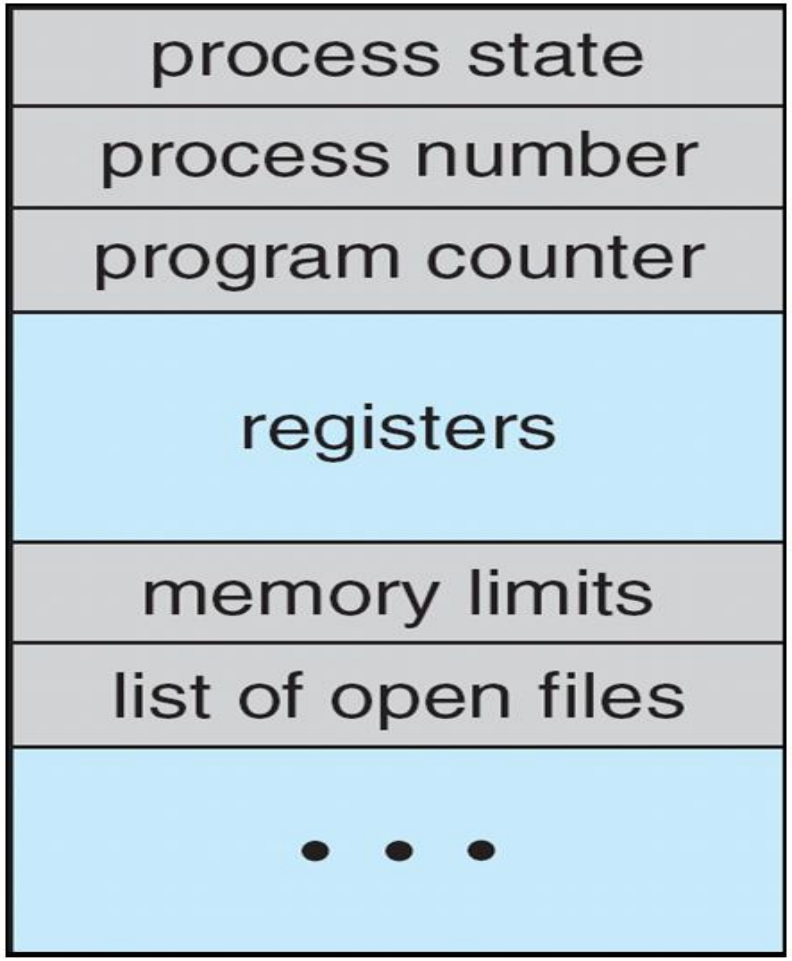
- A tabela <u>não guarda o conteúdo</u> do espaço de endereçamento do processo.



Processos

Bloco de Controle de Processos (PCB)

- Contém informações de contexto de cada processo (ex. ponteiros de arquivos abertos, posição do primeiro byte a ser lido em cada arquivo, etc);
- Contém informações necessárias para trazer o processo de volta, caso o S.O. tenha que tirá-lo de execução;
- Contém os estados do processo em um determinado tempo – Permite o rodízio de processos (*time-sharing*).



Processos

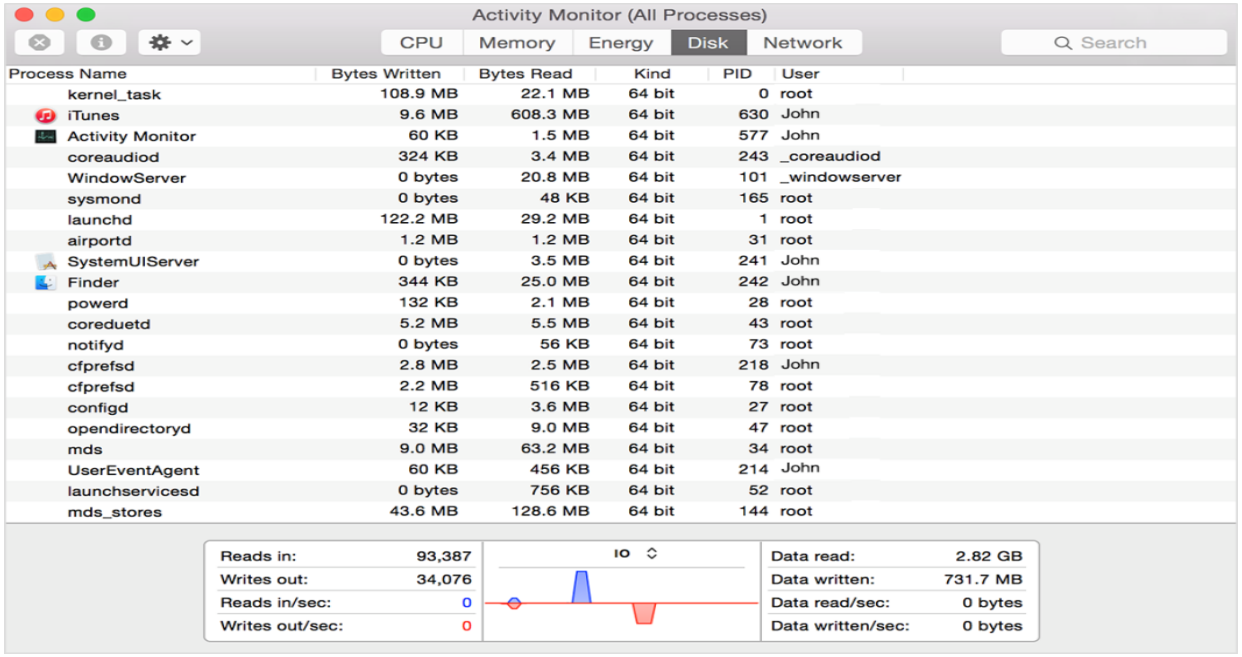
Bloco de Controle de Processos (PCB)

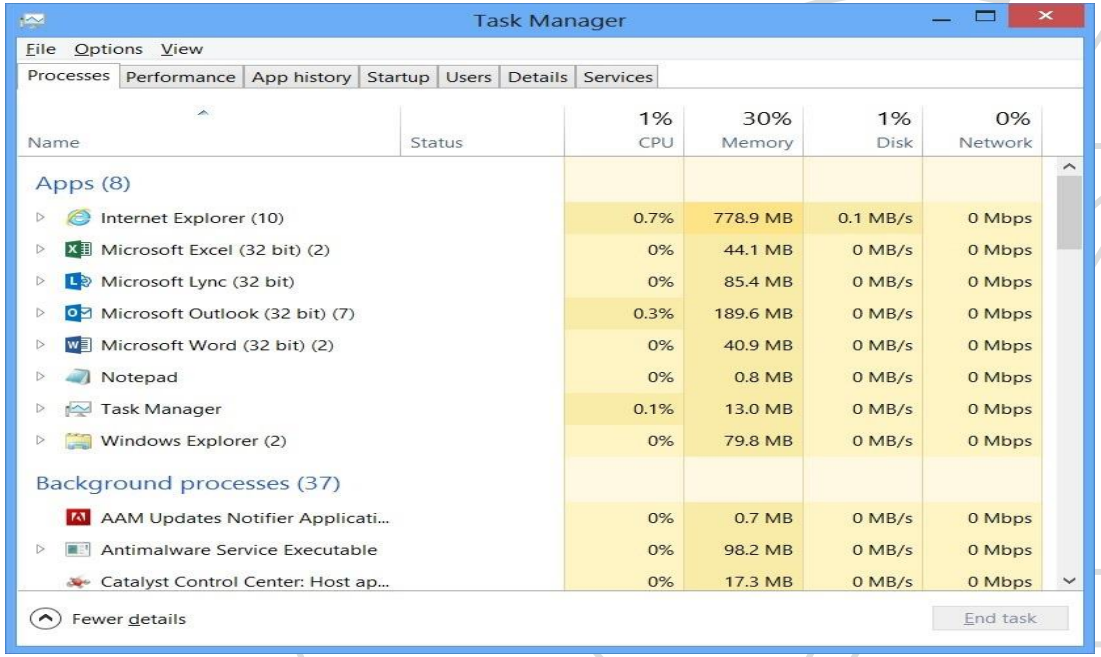
Gerenciamento de processos	**Gerenciamento de memória**	**Gerenciamento de arquivos**
Registradores Contador de programa Palavra de estado do programa Ponteiro de pilha Estado do processo Prioridade Parâmetros de escalonamento Identificador (ID) do processo Processo pai Grupo do processo Sinais Momento em que o processo iniciou Tempo usado da CPU Tempo de CPU do filho Momento do próximo alarme	Ponteiro para o segmento de código Ponteiro para o segmento de dados Ponteiro para o segmento de pilha	Diretório-raiz Diretório de trabalho Descritores de arquivos Identificador (ID) do usuário Identificador (ID) do grupo

Processos

Mac, Linux e Win

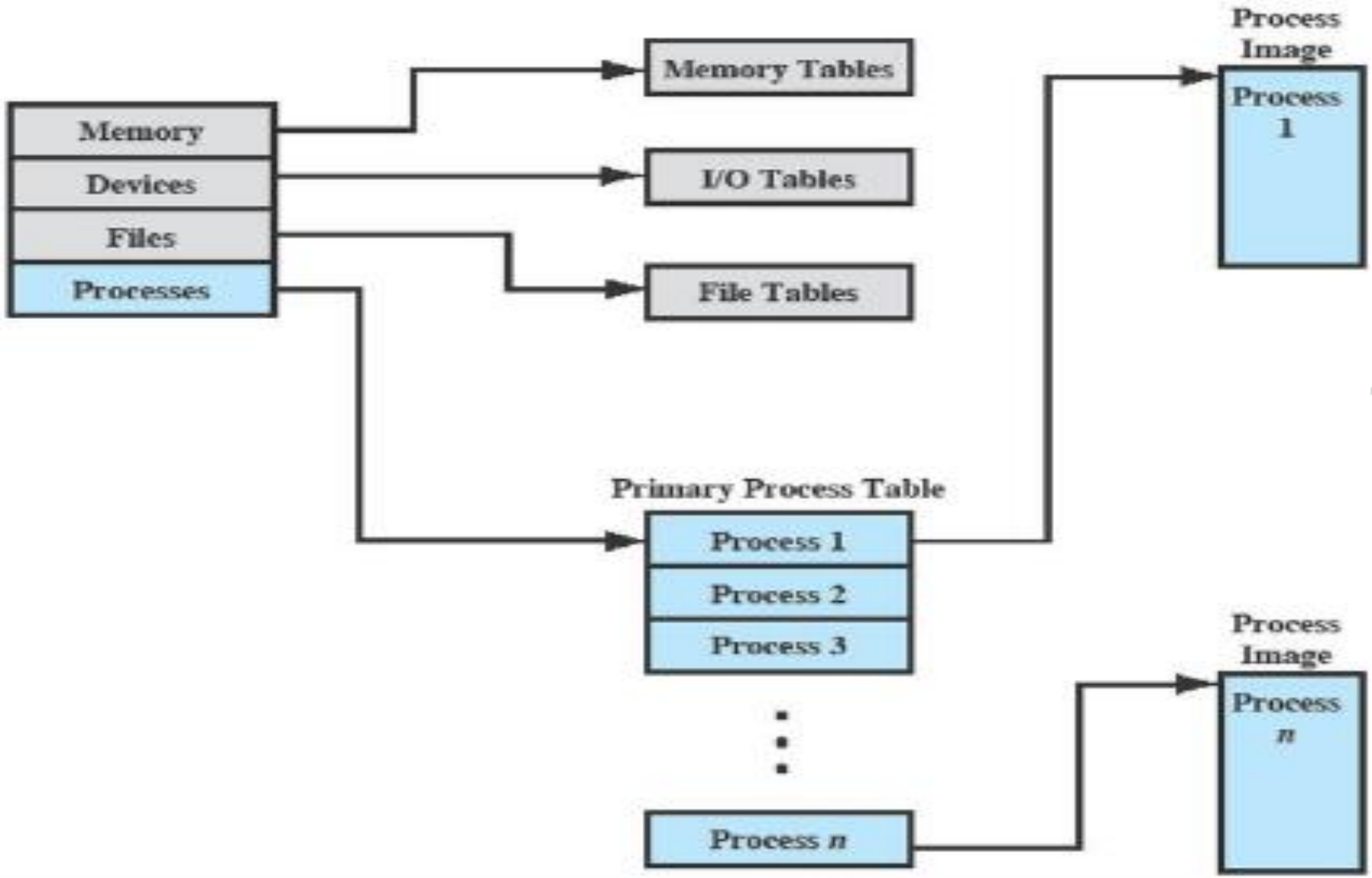
```
top - 19:00:06 up  7:47,  1 user,  load average: 0.65, 0.57, 0.51
Tasks: 198 total,   2 running, 196 sleeping,   0 stopped,   0 zombie
%Cpu(s): 12.6 us,  0.6 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  11894.0 total,   1511.5 free,   5763.0 used,   4619.4 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   5706.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2844 root      20   0 4169800   1.9g 152908 R  46.8  16.4 126:45.88 Web Content
 2758 root      20   0 2560304 462792 162424 S   5.6   3.8  49:01.37 firefox-esr
 1383 root      20   0  521120 113500  82664 S   0.3   0.9  12:35.23 Xorg
 2494 root      20   0 6347740   1.6g  37592 S   0.3  13.7  31:22.93 java
 3030 root      20   0  625936  50372  31888 S   0.3   0.4   0:34.02 gnome-terminal-
11209 root     -51   0   17868   3504   3028 R   0.3   0.0   0:00.03 top
    1 root      20   0  202592   8988   6760 S   0.0   0.1   0:17.10 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    5 root       0 -20       0      0      0 I   0.0   0.0   0:00.48 kworker/0:0H
    7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
    8 root      20   0       0      0      0 S   0.0   0.0   0:00.35 ksoftirqd/0
```





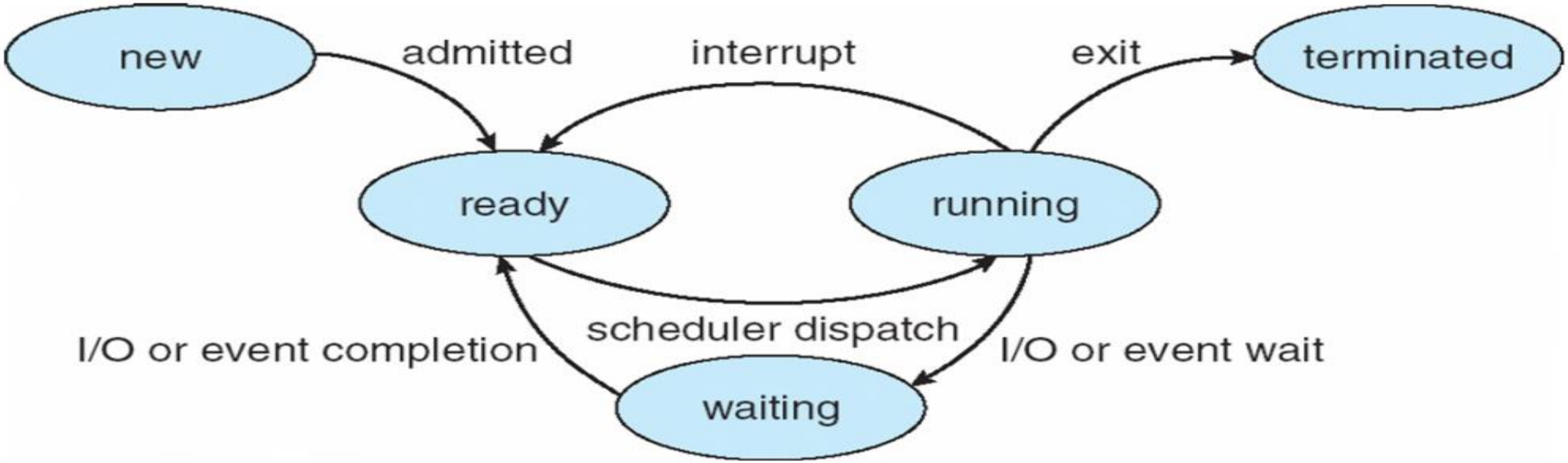
Processos

Tabelas de Controle do Sistema Operacional



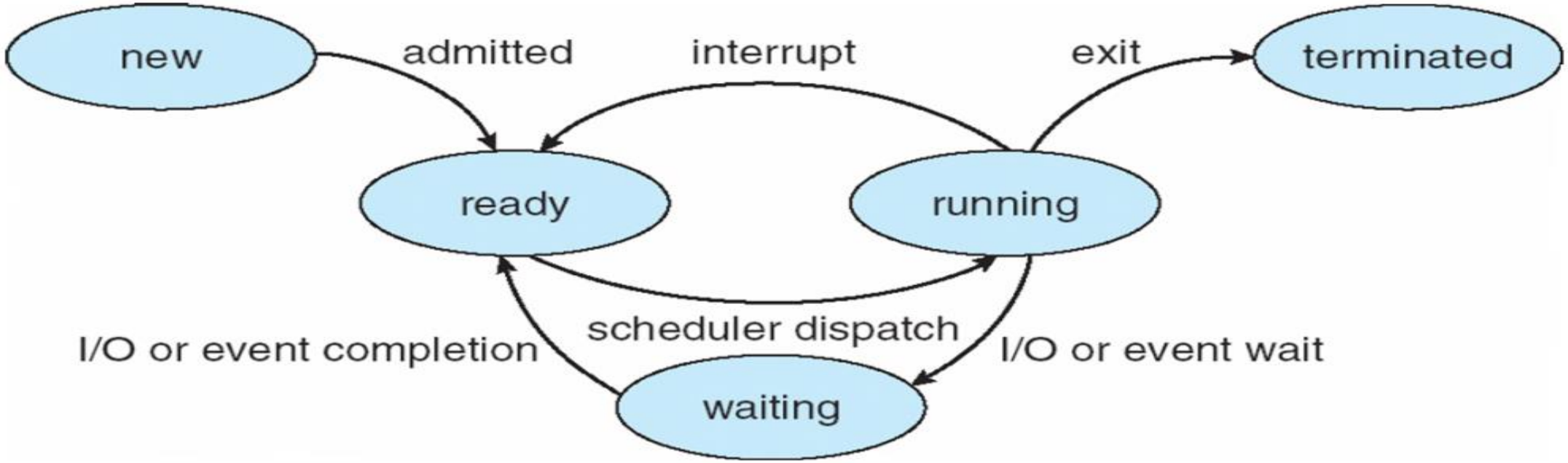
Operações em processos

- **Executando**: realmente utilizando a CPU naquele momento.
- **Bloqueado**: incapaz de executar enquanto um evento externo não ocorrer.
- **Pronto**: em memória, pronto para executar (ou para continuar sua execução), apenas aguardando a disponibilidade do processador.



Estados de um processo:

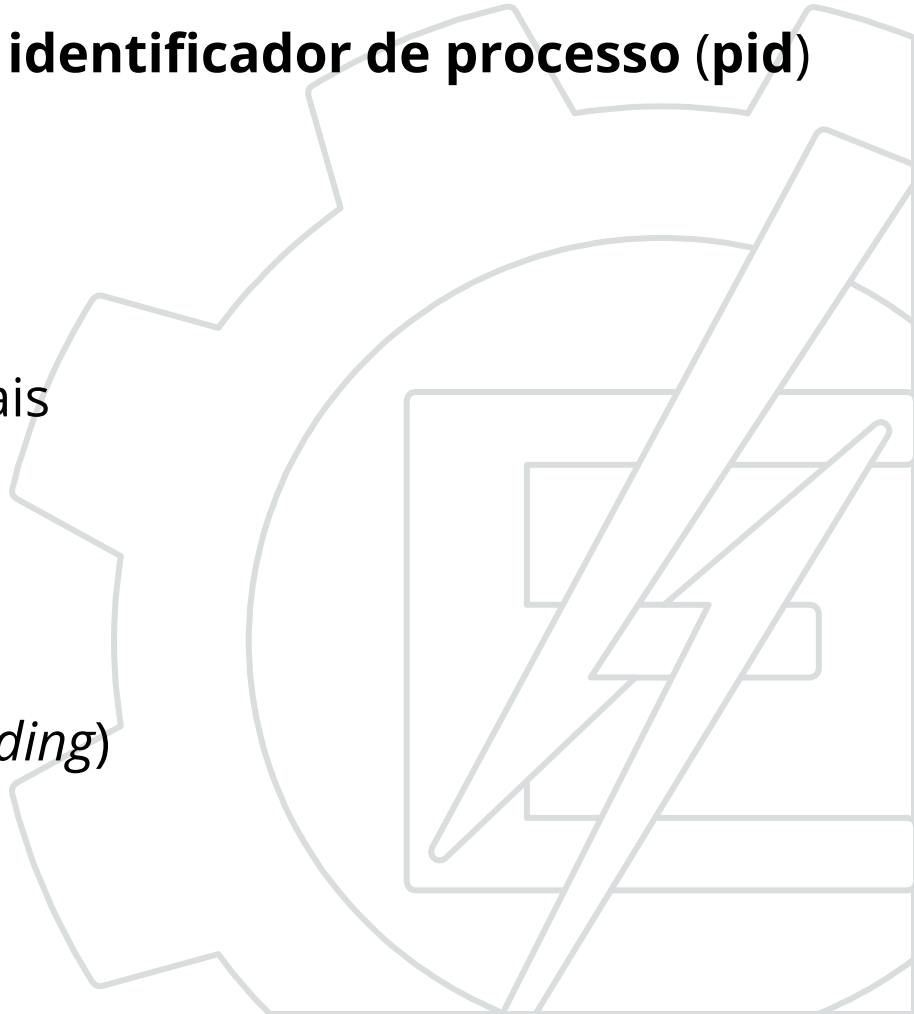
- Muda seu estado de “EM EXECUÇÃO” para “EM ESPERA”
- Muda seu estado de “EM EXECUÇÃO” para “PRONTO”
- Muda seu estado de “EM ESPERA” para “PRONTO”
- TERMINA



Processos

Criação

- **Processos Pai criam Processos Filhos, que por sua vez, criam outros processos, formando uma árvore de processos**
- Geralmente, um processo é identificado e gerenciado via um **identificador de processo** (**pid**)
- Opções de Compartilhamento de Recursos
  - Pais e Filhos compartilham todos os recursos
  - Filhos compartilham um subconjunto dos recursos dos pais
  - Pais e filhos não compartilham recursos
- Execução
  - Pais e filhos executam concorrentemente (ex.: *multi-threading*)
  - Pais aguardam o término de execução dos filhos



Processos

Árvore de processos no Linux

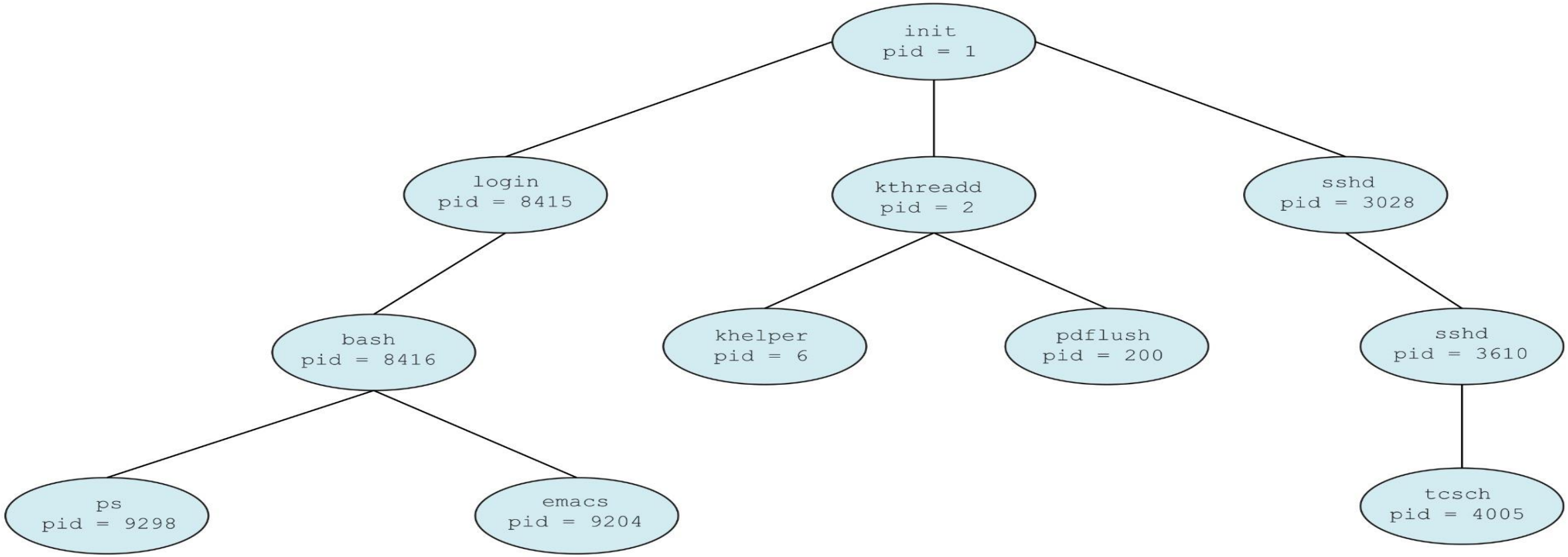
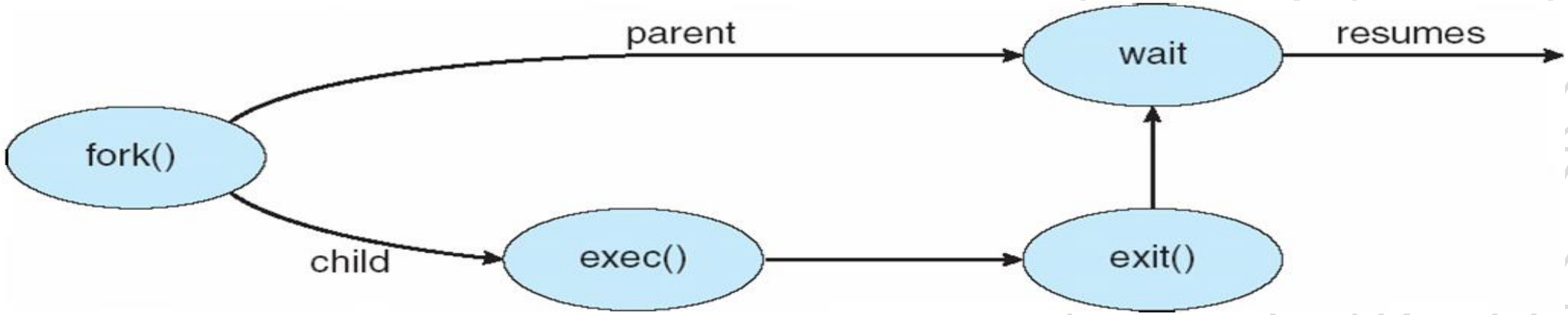


Figure 3.8 A tree of processes on a typical Linux system.

- Espaço de endereçamento
  - Filho duplica o endereço do pai
  - Filho possui um programa carregado nele
- Exemplo no UNIX
  - Chamadas de sistema **fork** criam novos processos
  - Chamada de sistema **exec** usada após um **fork** para substituir o espaço de memória do processo com um novo programa



Processos

Programa em C (*fork/wait*)

```
int main() {
        pid_t  pid;
        pid = fork(); /* forquilha o processo, criando um processo filho */

        if (pid < 0) { /* erro encontrado */
                fprintf(stderr, "Falha na criação do processo filho");
                exit(-1);
        }/* OBS (pid >= 0) indica sucesso no fork  */

         /* OBS (pid == 0) indica o processo filho  */
        else if (pid == 0) { /* processo filho criado e alocado */
                execlp("/bin/ls", "ls", NULL);
        }
         /* OBS (pid > 0) indica o processo pai  */
        else { /*processo pai */
                /* pai aguardará o filho completar */
                wait(NULL);
                printf(“Filho completo");
                exit(0);
        }
}
```

Processos

Término

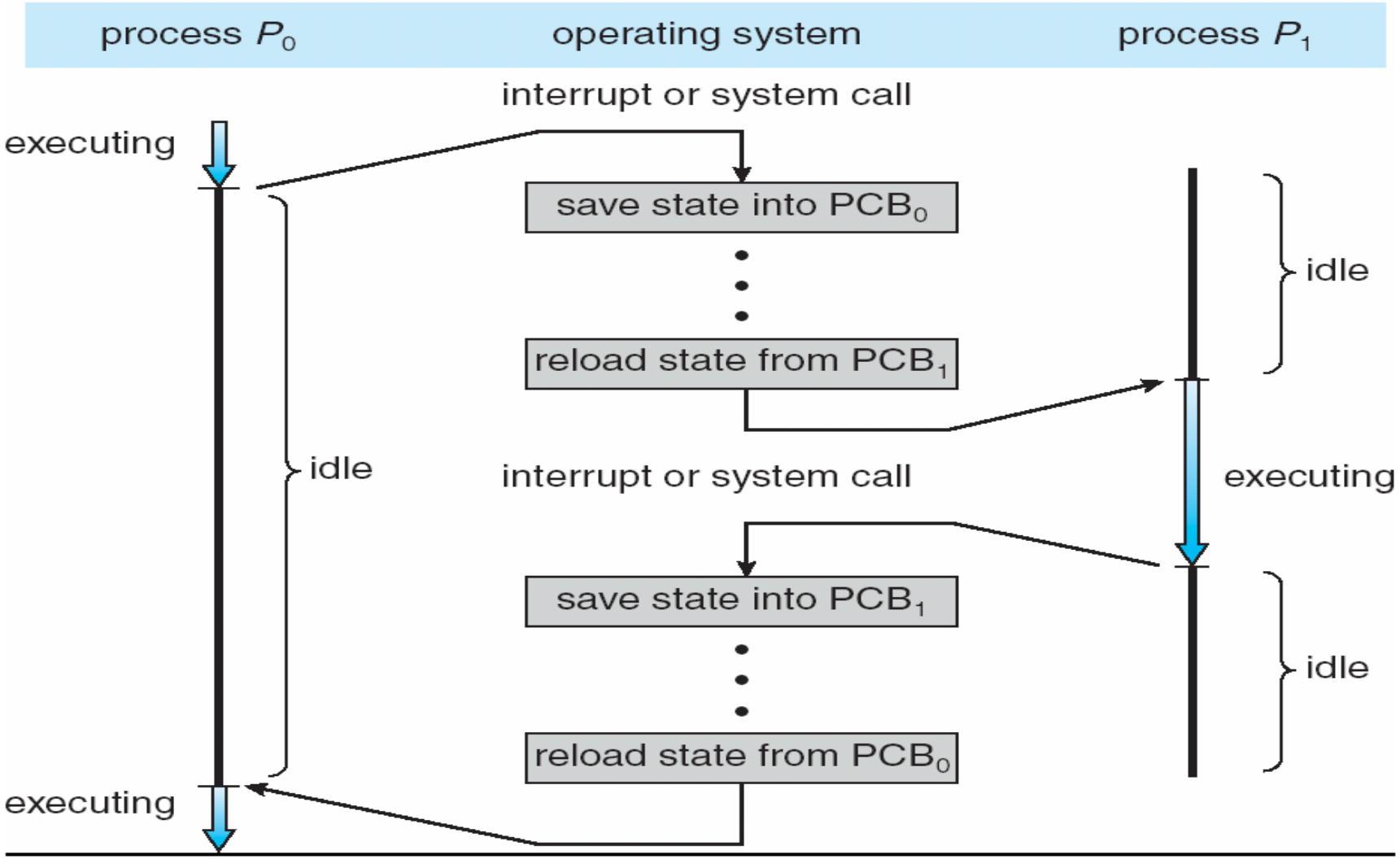
- Um processo executa sua última instrução e solicita ao sistema operacional a sua remoção (***exit***) da fila de processos
  - Mas antes coleta a saída de dados do filho para o pai (via ***wait***)
  - Os recursos do processo são desalocados pelo S.O.

- O processo pai pode terminar a execução dos processos filhos (***abort***)
  - Processo filho excedeu os recursos alocados
  - Tarefa associada ao filho não é mais necessária
  - Se o processo pai está terminando
    - Alguns sistemas operacionais não permitem o processo filho continuar se o processo pai terminar
      - Todos os filhos são terminados – **terminação em cascata**

Escalonamento de processos

Processos

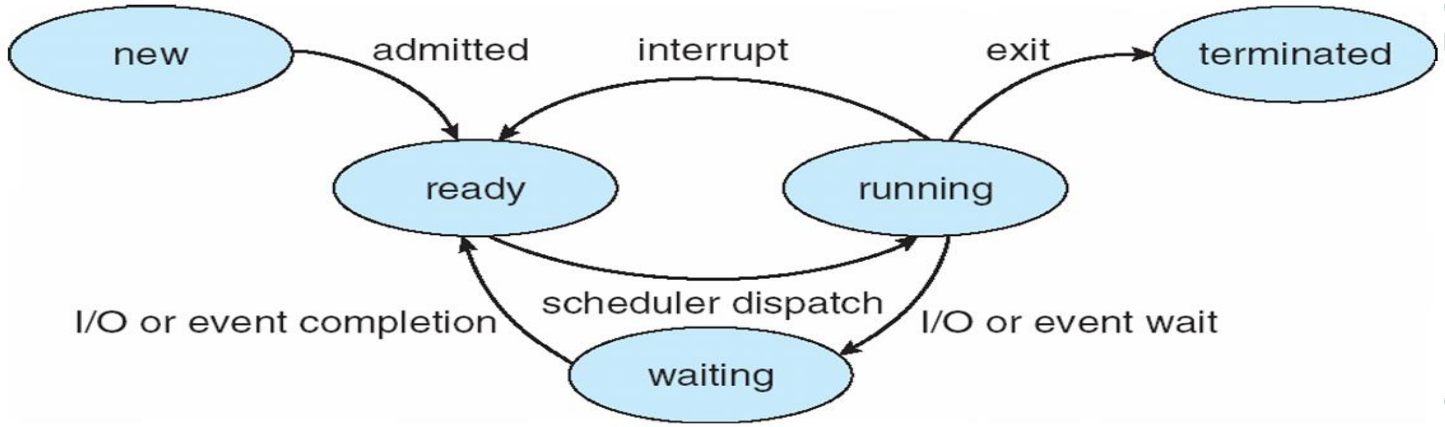
Troca de Processos pela CPU



Processos

Filas para escalonamento de processos

- **Fila de processos – conjunto de <u>todos</u> os processos do sistema**
- **Fila de prontos– conjunto dos processos residindo na memória principal, prontos e aguardando para execução (estado *ready*)**
- **Filas de dispositivos – conjunto de processos esperando por um dispositivo de E/S (estado *waiting*)**
- Observe que os processos migram entre as várias filas, de acordo com seu estado



Processos

Fila de prontos e Fila de Dispositivos de E/S

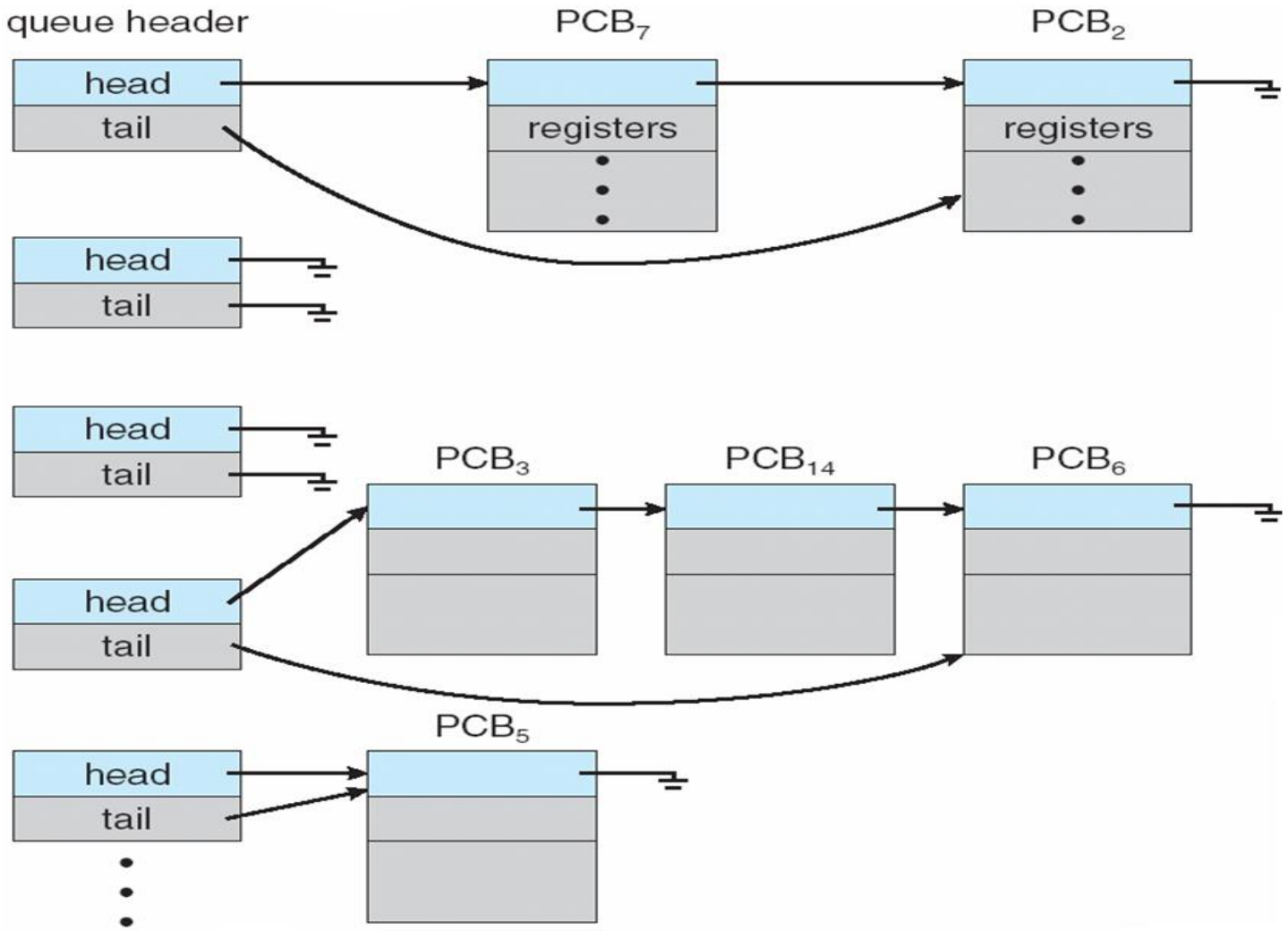
Fila de Prontos

Fila do drive de DVD

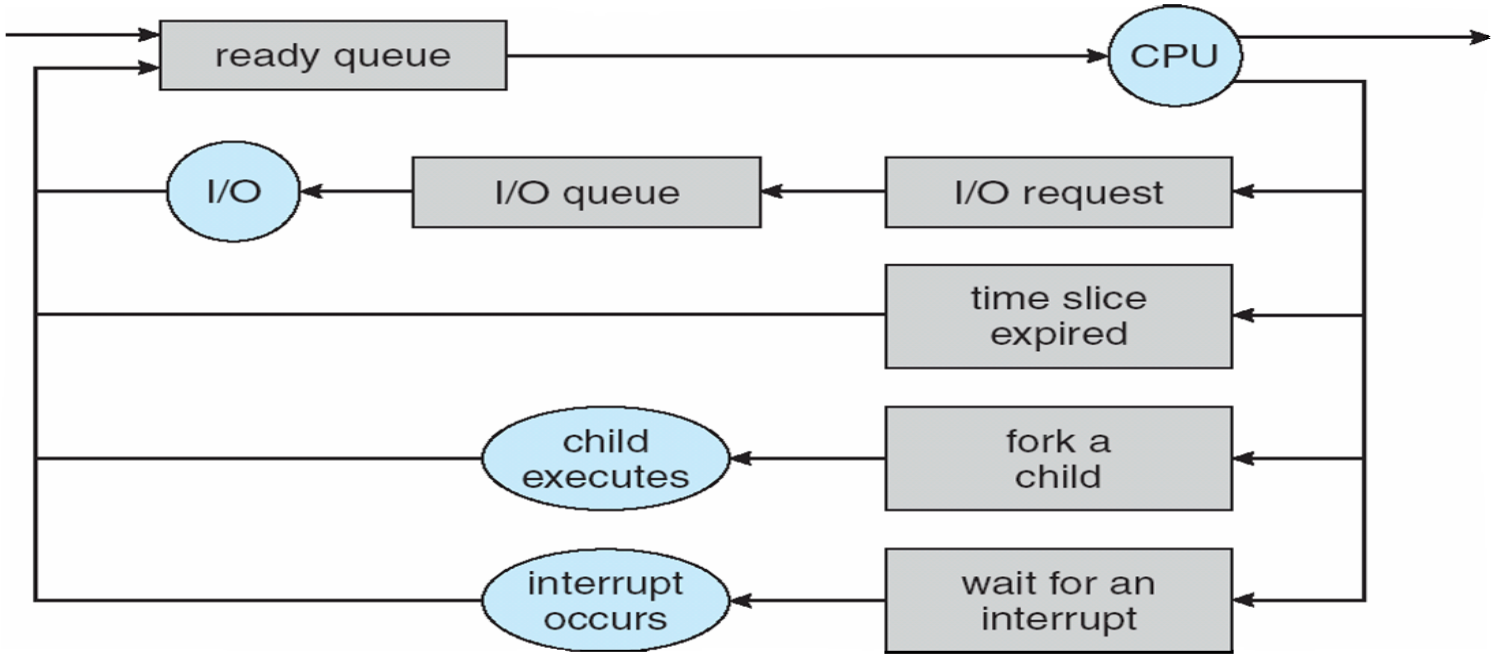
Fila do drive de SDD

Fila do drive de HDD

Fila do drive de Teclado

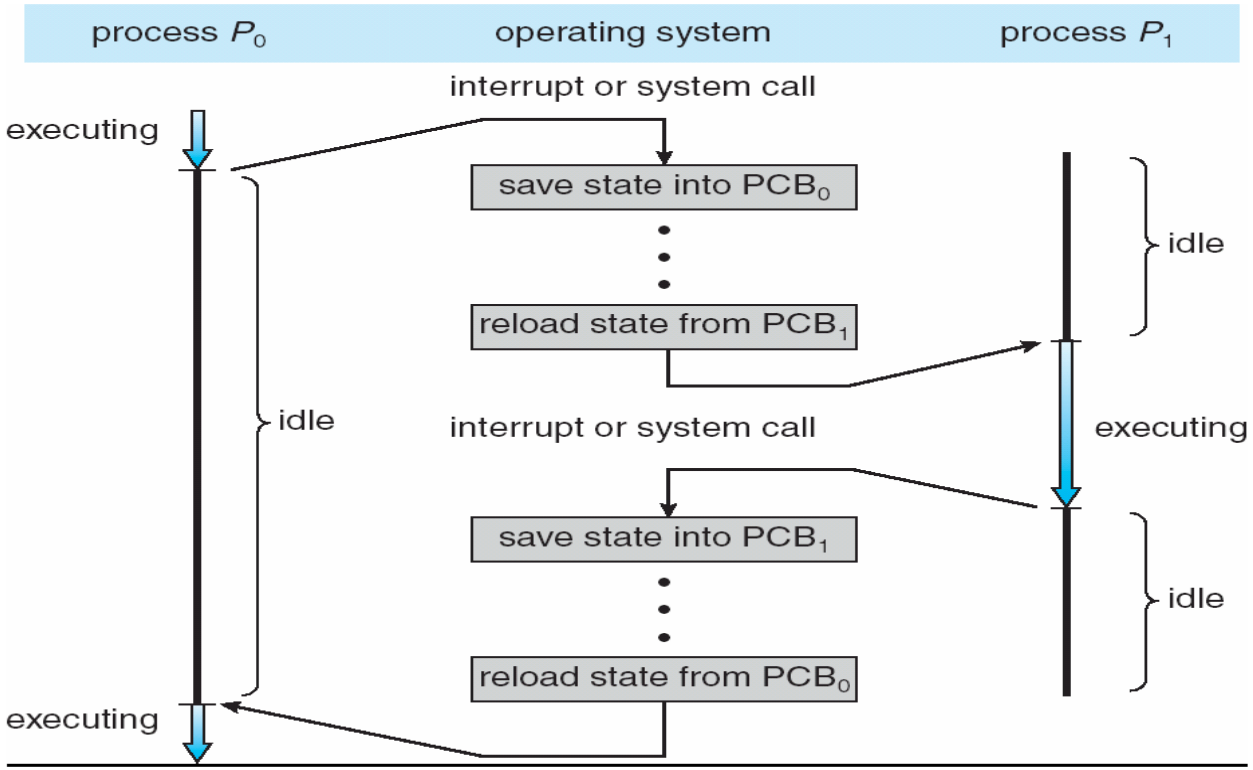


- Possui diversos processos que competem pelo uso da CPU.



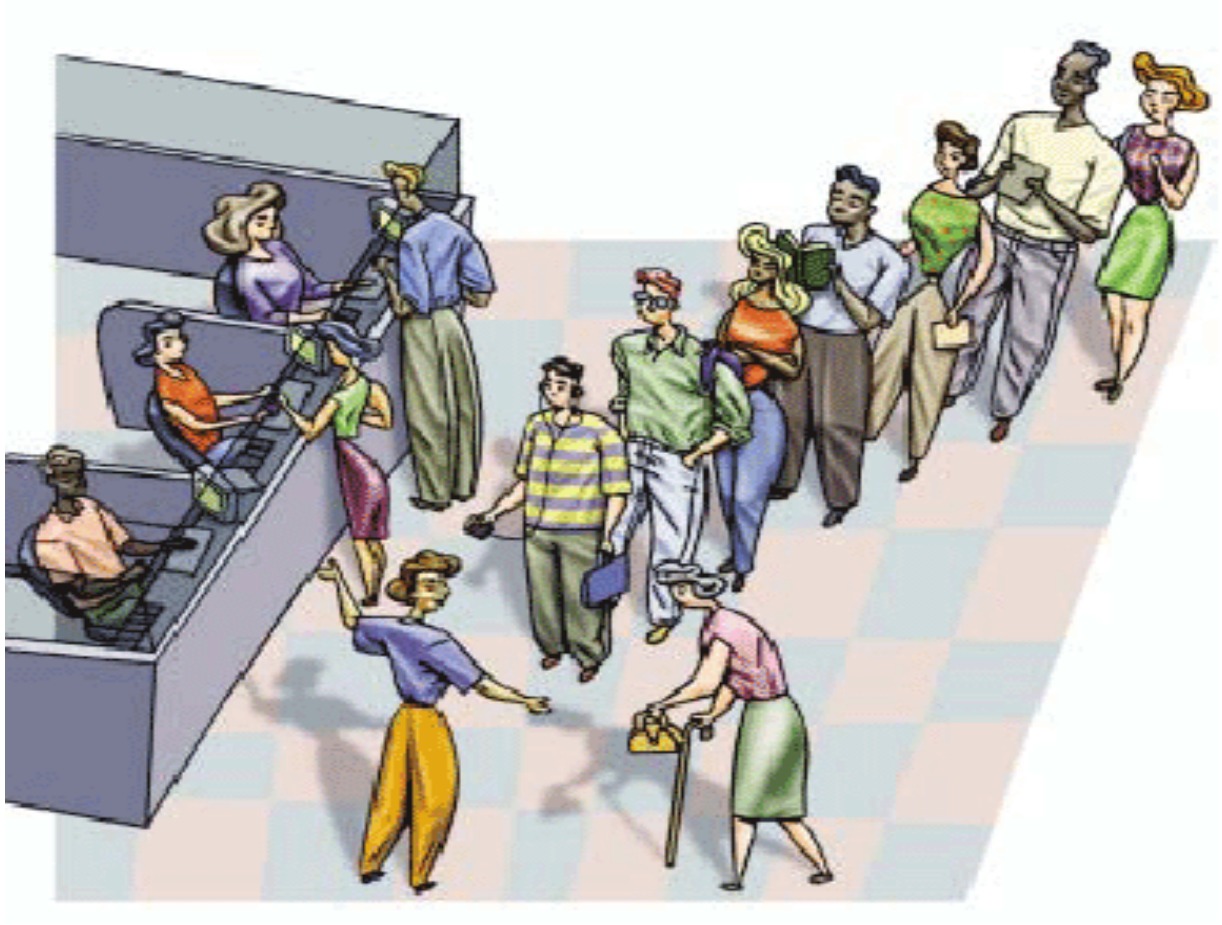
- **Despachante** (*Dispatcher*)**:** módulo que realiza o armazenamento e recuperação dos contextos dos processos e atualiza os PCBs.
- **Escalonador** *(Scheduler):* módulo que controla a mudança de estado dos processos, definindo o próximo processo a ser executado.

- Quando a CPU realiza a troca para um outro processo, o sistema deve fazer uma troca de contexto (mínima latência *versus quantum*), que consiste em:
  - Salvar o estado do processo antigo (atualizar o PCB na RAM)
  - Carregar (na CPU) o estado salvo (na RAM) do novo processo



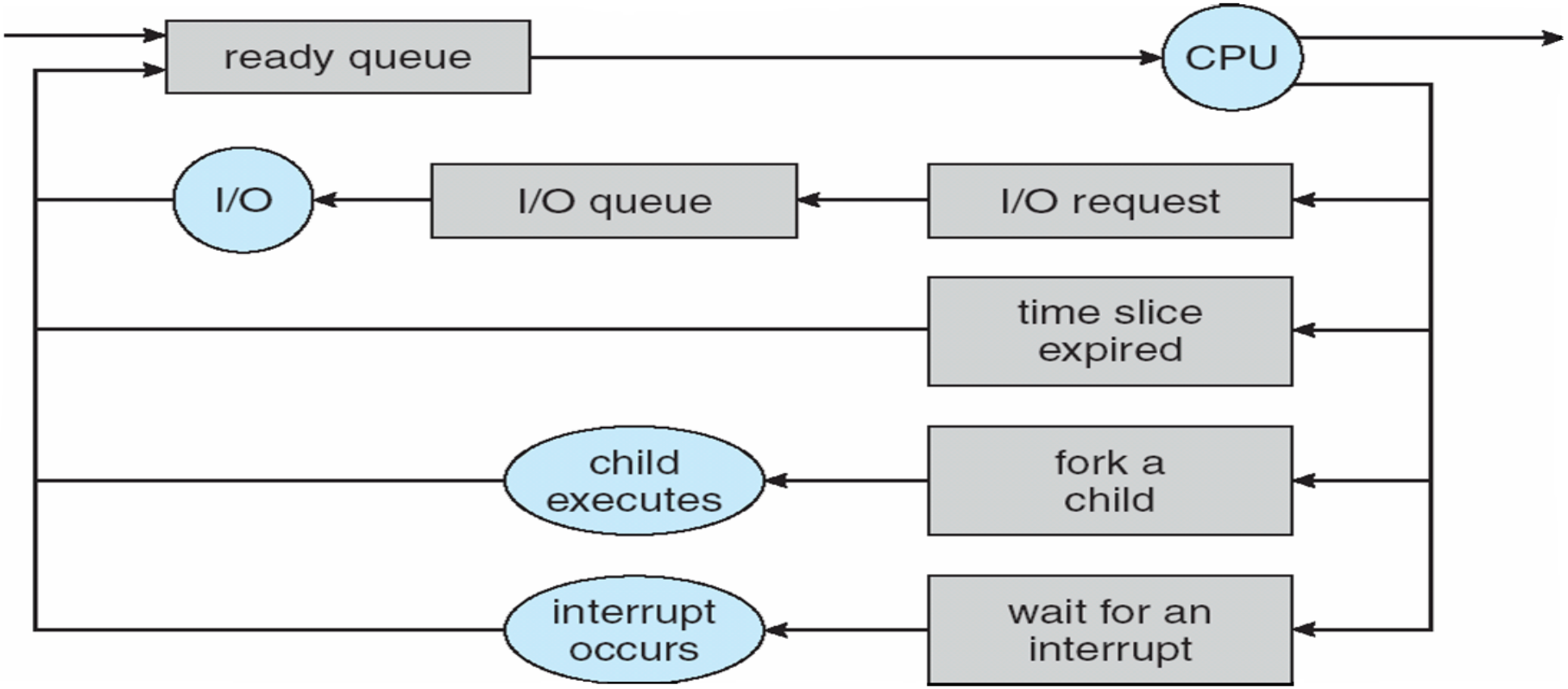
Processos

Escalonador de tarefas



- Pode ser classificado em:
  - **Escalonador de Longo Prazo** (ou **escalonador de *jobs***):
    - Seleciona quais processos devem ser carregados em memória e trazidos para a fila de prontos.
    - É chamado com menos frequência (segundos, minutos).
  - **Escalonador de Curto Prazo** (ou **escalonador da CPU**):
    - Seleciona qual processo (daqueles prontos) deve ser executado em sequência e faz a alocação da CPU.
    - É chamado frequentemente (~100ms) e deve ser rápido na escolha do processo.

- Pode ser classificado em:
  - **Escalonador de Longo Prazo** (ou escalonador de *jobs*);
  - **Escalonador de Curto Prazo** (ou escalonador da CPU).



- Com relação ao escalonamento da CPU, pode ser classificado em:
  - **Não-preemptivo:**
    - Implementação mais simples do escalonador.
    - O processo libera a CPU nas seguintes condições:
      - Término da execução; ou
      - Solicitação de operação de E/S (voluntário).
  - **Preemptivo:**
    - Escalonador mais complexo.
    - Compartilhamento da CPU é garantido.
    - Periodicamente o escalonador interrompe o processo em execução e muda-o para o estado “pronto”.

- Deve possuir um algoritmo que se preocupe com 5 regras:
  - **Justiça** – Todos processos devem ter acesso a CPU (tempo de espera)
  - **Eficiência** – buscar a máxima utilização da CPU
  - Minimizar o **Tempo de Resposta**
  - ***Turnaround*** – Minimiza os usuários *batch*. Tempo para conclusão do processo (alocação + fila + execução CPU + execução E/S)
  - ***Throughput*** – Maximizar o número de *jobs* processados

A partir da finalização da execução de um processo ou de sua parcela de tempo (*quantum*), qual será o novo processo a ser executado?

Um novo processo criado? Um processo que criou outro processo (filho)? Um processo que está pronto há mais tempo?

Como esta escolha pode ser feita?

- Podem ser descritos como:
  - ***I/O-bound*:** Gastam mais tempo fazendo E/S do que computação.
  - ***CPU-bound*:** Gastam mais tempo com computação.

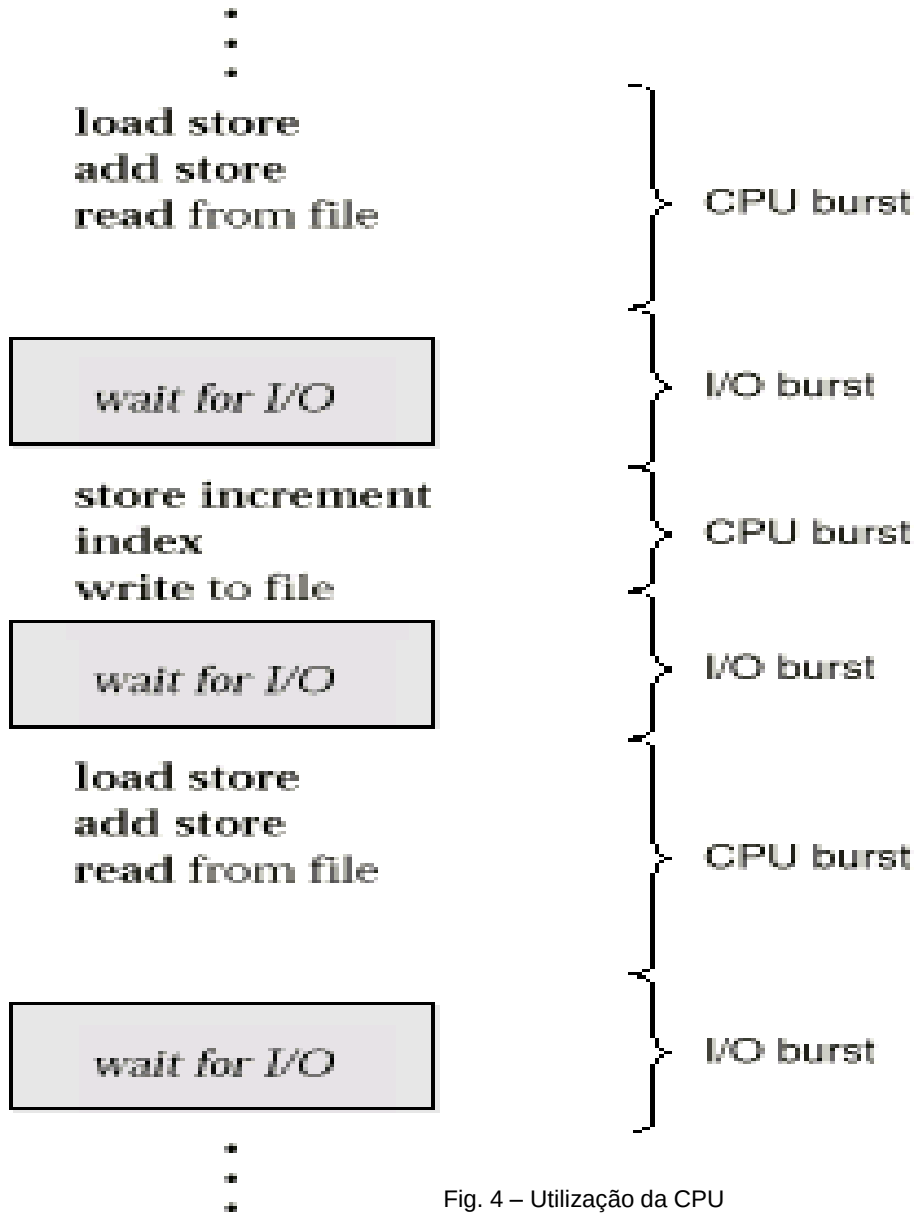
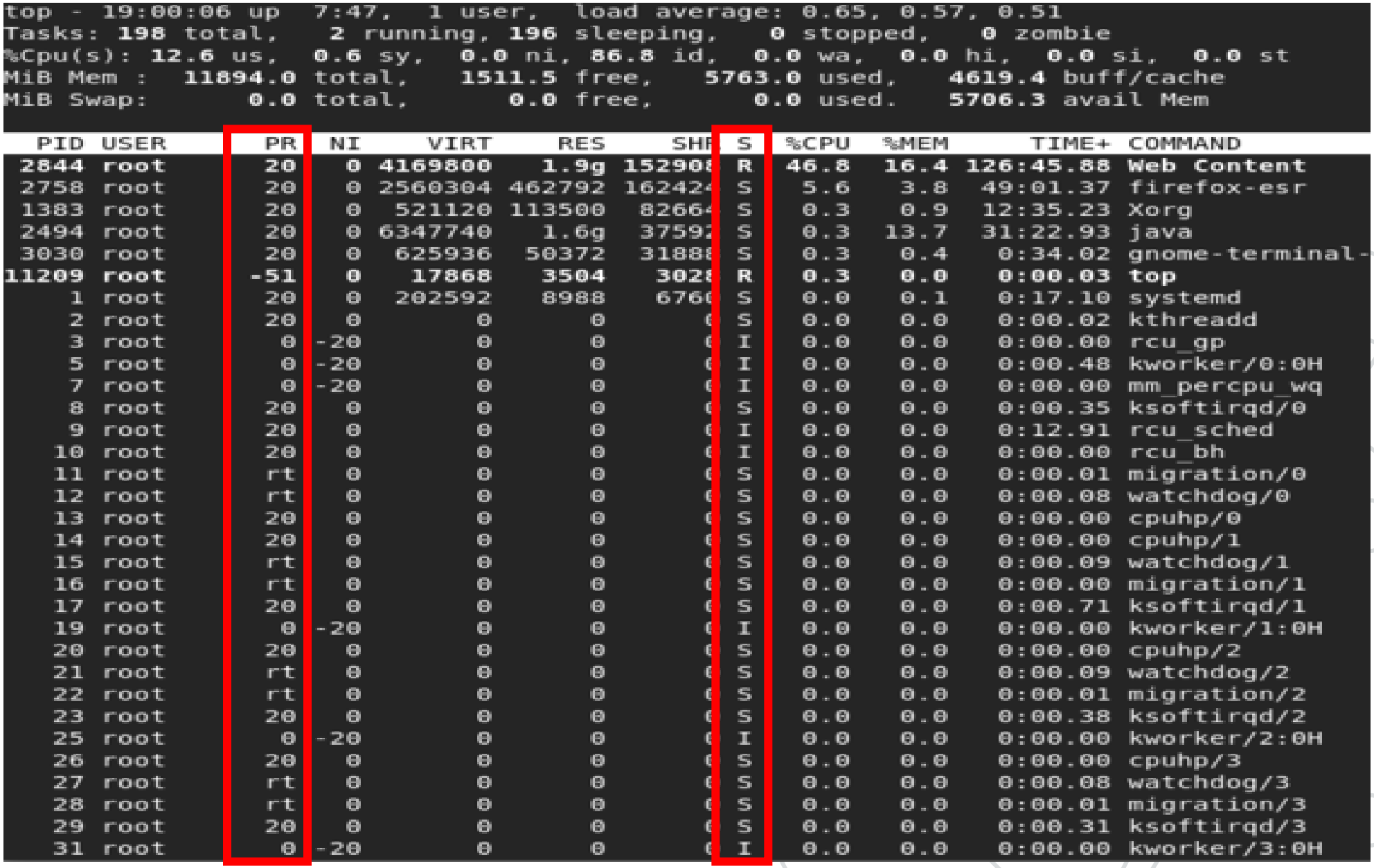


Fig. 4 – Utilização da CPU

Processos

- Podem ser descritos como:
  - ***I/O-bound*:** gastam mais tempo fazendo E/S do que computação.
  - ***CPU-bound*:** Gastam mais tempo com computação.

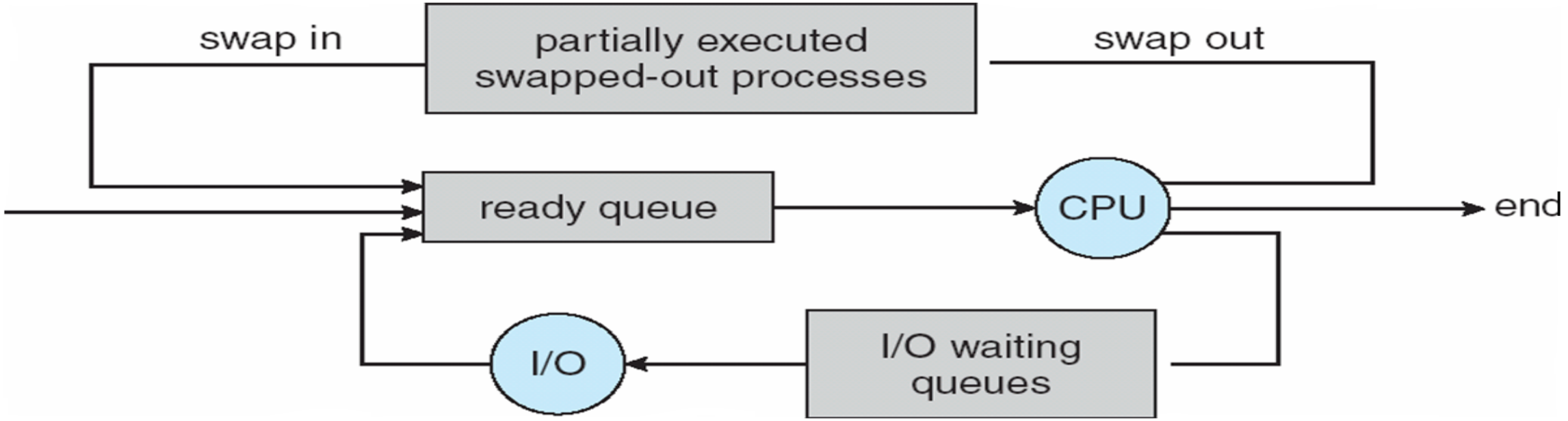
- Podemos classificá-los por:
  - **Uso de recursos:** temos os processos **convencionais** e os de **tempo real** (de sistema).
    - No Linux, os processos de tempo real recebem prioridade entre 1 e 99, enquanto os processos convencionais recebem prioridade entre 100 e 139 (padrão 120).
  - **Tipo de execução:** temos os **interativos**, **em série** ou **tempo real**.



S - Em série
I - Interativos
R - Tempo real

Processos

Escalonador de médio prazo (*Swapping*)



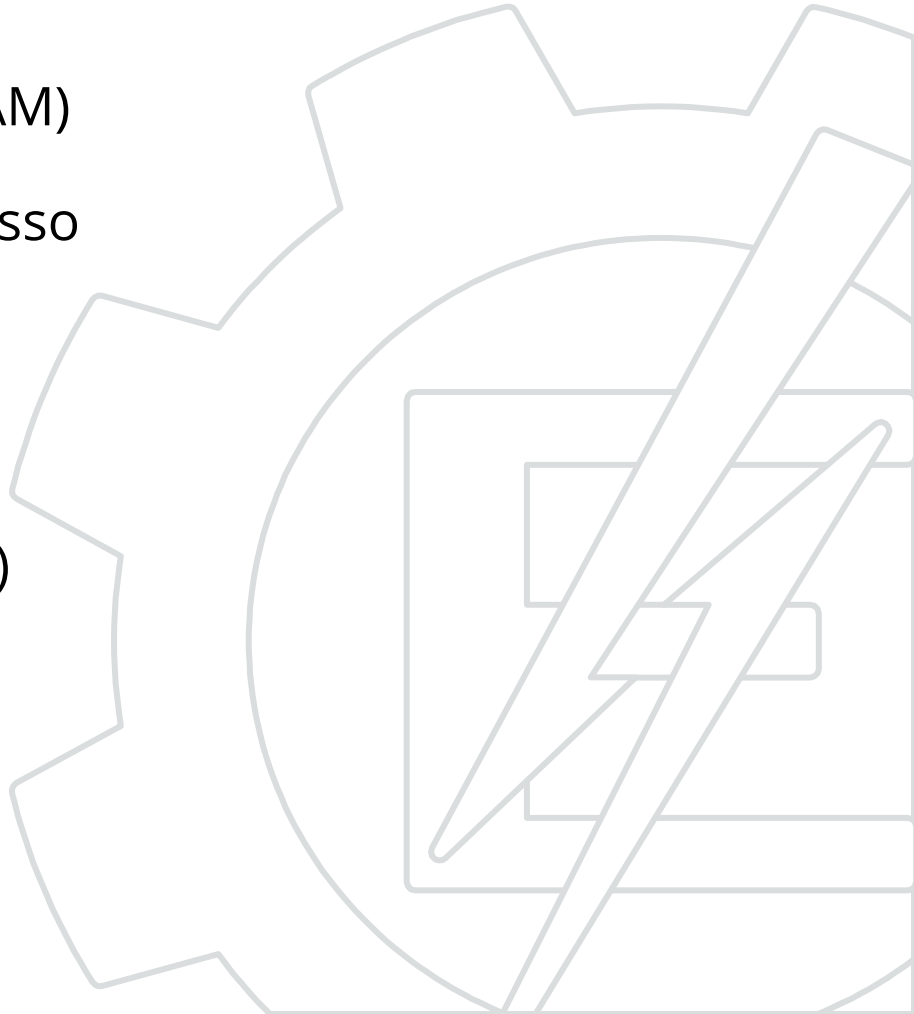
- OBS.: alguns sistemas de tempo compartilhado **não possuem** Escalonador de Longo Prazo, como o Unix e o MS Windows
  - Simplesmente coloca cada novo processo na memória para o Escalonador de Curto Prazo
  - A estabilidade depende de limitações físicas (terminais ou RAM) ou da adaptabilidade dos usuários humanos (Alt+F4)

Processos

Troca de contexto

- Quando a CPU realiza a troca para um outro processo, o sistema deve fazer uma troca de contexto:
  - salvar o estado do processo antigo (atualizar o PCB na RAM)
  - carregar (na CPU) o estado salvo (na RAM) do novo processo

- O contexto de um processo é representado no PCB
  - O tempo de troca de contexto gera *overhead* (sobrecarga)
  - O sistema não trabalha de forma útil durante a troca
  - Tempo dependente do suporte de *hardware*



Bibliografia

- TANENBAUM, Andrew S; BOS, Herbert. Sistemas operacionais modernos. 4a ed. São Paulo: Pearson Education do Brasil, 2016. **Capítulo 2.**

  https://plataforma.bvirtual.com.br/Acervo/Publicacao/1233

- DEITEL, H.M; DEITEL, P.J; CHOFFNES,D.R. Sistemas Operacionais. 3a ed. São Paulo: Pearson Prentice Hall, 2005. **Capítulo 3.**

  https://plataforma.bvirtual.com.br/Acervo/Publicacao/315

Sistemas Operacionais

Prof. Otávio Gomes

otavio.gomes@unifei.edu.br