Sistemas Operacionais
Capítulo 4

Gerência do Processador

Gerência do Processador

Sumário

- **Implementação do conceito de processos e threads**
- **Escalonamento**
  - Escalonadores não-preemptivos
- **Escalonamento**
  - Escalonamento preemptivos

Introdução

- **Multiprogramação pressupõe a existência simultânea de vários processos disputando o processador**
- **Necessidade de "intermediar" esta disputa de forma justa**
  - Gerência do processador
    - » Algoritmos de escalonamento
- **Necessidade de "representar" um processo**
  - Implementação de processos
    - » Estruturas de dados

Representação de processo (1)

- **Processo é um programa em execução**
  - Áreas na memória para código, dados e pilha
- **Possui uma serie de estados (apto, executando, bloqueado, etc) para representar sua evolução no tempo, implica em:**
  - Organizar os processos nos diferentes estados
  - Determinar eventos que realizam a transição entre os estados
  - Determinar quando um processo tem direito a "utilizar' o processador

Representação de processo (2)

- **Necessário manter informações a respeito do processo**
  - e.g.: prioridades, localização em memória, estado atual, direitos de acesso, recursos que emprega, etc.

Bloco descritor de processo (1)

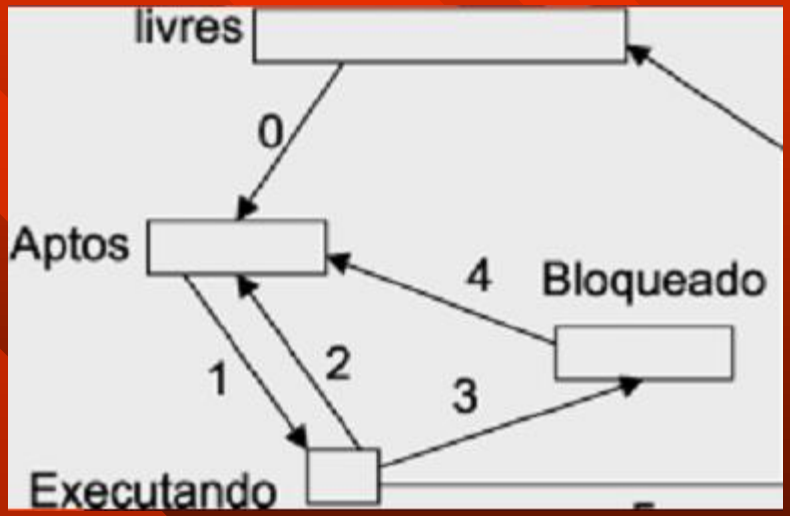
- **Abstração de processo é implementado através de uma estrutura de dados**
  - Bloco descritor de processos (Process Control Block - PCB)

Bloco descritor de processo (2)

- **Informações normalmente presentes em um descritor de processo (PCB):**
  - Prioridade
  - Localização e tamanho na memória principal
  - Identificação de arquivos abertos
  - Informações de contabilidade (tempo CPU, espaço de memória, etc)
  - Estado do processador (apto, executando, bloqueando, etc)
  - Contexto de execução
  - Apontadores para encadeamento dos próprios descritores de processo

Os processos e as filas

- **Um processo sempre faz parte de alguma fila**
- **Geralmente a própria estrutura de descritores de processos são empregadas como elementos dessas filas:**
  - Fila de livres
    - » Numero fixo (máximo) de processos
    - » Alocação dinâmica
  - Fila de aptos
  - Fila de bloqueados



- **Eventos realizam transição de uma fila a outra**

Exemplo de bloco descritor de processos (1)

- **Estrutura de dados representando bloco descritor de processo**

```
struct desc pros{
        char estado atual;
        int prioridade;
        unsigned inicio_memória;
        unsigned tamanho_mem;
        struct arquivos arquivos_abertos[20];
        unsigned tempo_cpu;
        unsigned proc_pc;
        unsigned proc_sp;
        unsigned proc_acc;
        unsigned proc_rx;
        struct desc_proc *proximo;
}

struct desc_proc tab_desc[MAX PROCESS];
```

Exemplo de bloco descritor de processos (2)

- **Estruturas de filas e inicialização**

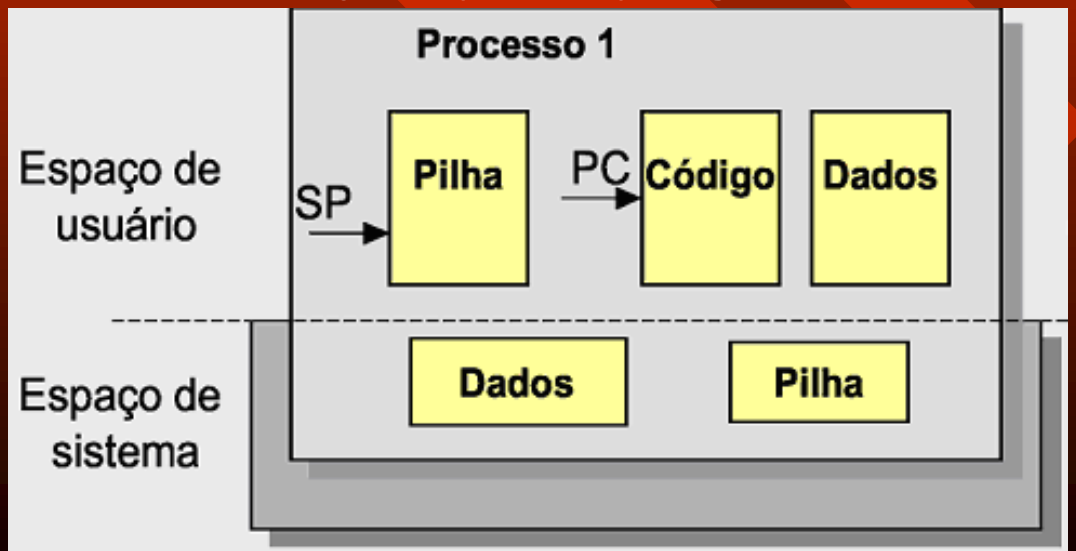
```
struct   desc_proc *desc livre;
struct   desc_proc *espera cpu;
struct   desc_proc *usando cpu;
struct   desc_proc *bloqueados;

/* Inicialização das estruturas de controle */
for (i=0; 1 < MAX_PROCESS; i++)
    tab_desc[ i].prox = &tab_desc[ i+1] ;

tab_desc[ 1].prox = NULL;
desc_livre = &tab_desc[ 0] ;
espera_cpu= NULL;
usando_cpu= NULL;
bloqueado = NULL;
```

Tarefas típicas no PCB durante o ciclo de vida

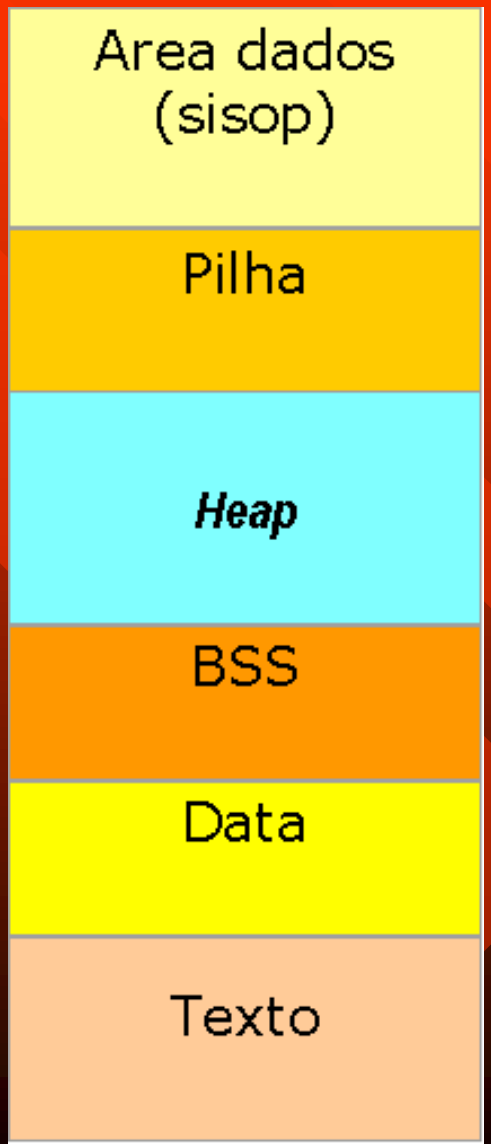
- **Criação**
  - Alocação de áreas de memória para código, dados e pilha e de estruturas de dados do sistema operacional
  - Inicialização do descritor de processo e inserção em filas do sistema
- **Execução**
  - Realizam das instruções da área de código
    - » Interação com sistema operacional via chamadas de sistema
  - Atualização do bloco descritor de processo
    - » Retratar estados e recursos que evoluem dinamicamente com a execução
  - Suscetível ao acionamento do escalonador/dispatcher em resposta a eventos
- **Término**
  - Liberação de recursos e estruturas de dados utilizadas

O modelo de processo

- **Processo é representado por:**
  - Espaço de endereçamento: área p/ armazenamento da imagem do processo
  - Estruturas internos do sistema (tabelas internas, áreas de memória, etc)
    - » Mantidos no descritor de processos
  - Contexto de execução (pilha, programa, dados, etc...)

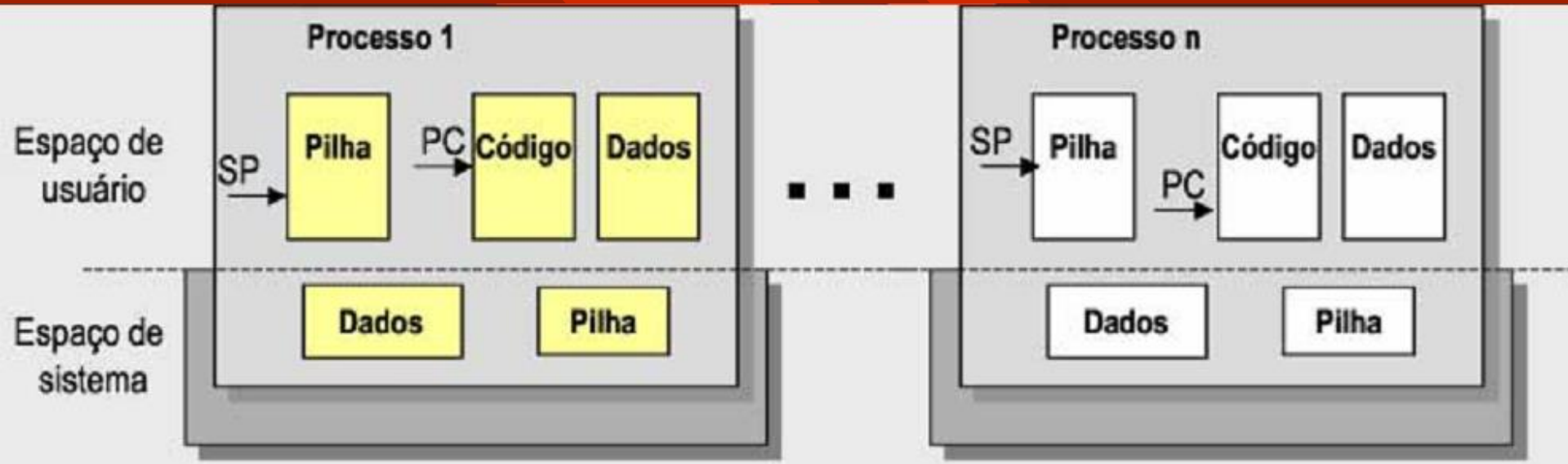


Exemplo: modelo de processo Unix (linux)



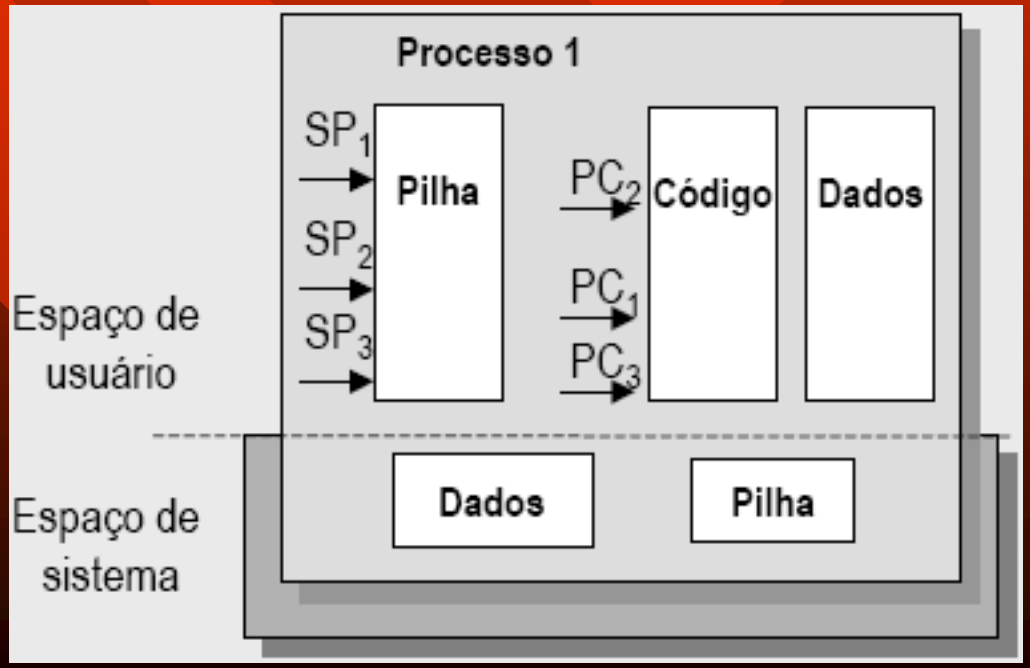
Vários processos

- **Um fluxo de controle por processo (thread)**
- **Troca de processo implica em atualizar estruturas de dados internas do sistema operacional**
  - e.g.; contexto, espaço de endereçamento, etc...



Vários fluxos em um Único processo

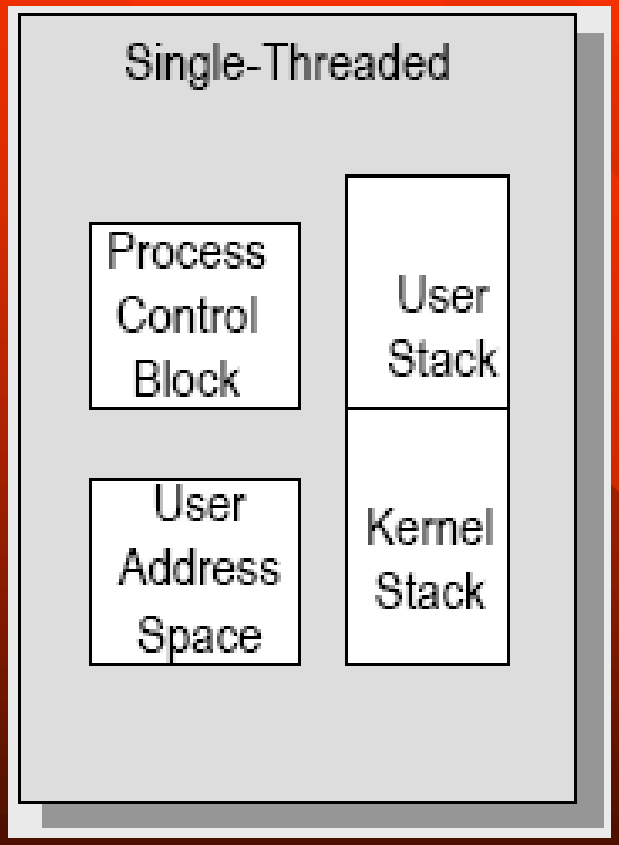
- **Um fluxo de instrução é implementado através do contador de programa (PC) e de uma pilha (SP)**
- **Estruturas comuns compartilhadas**
  - Código
  - Dados
  - Descritor de processo
- **Conceito de thread**

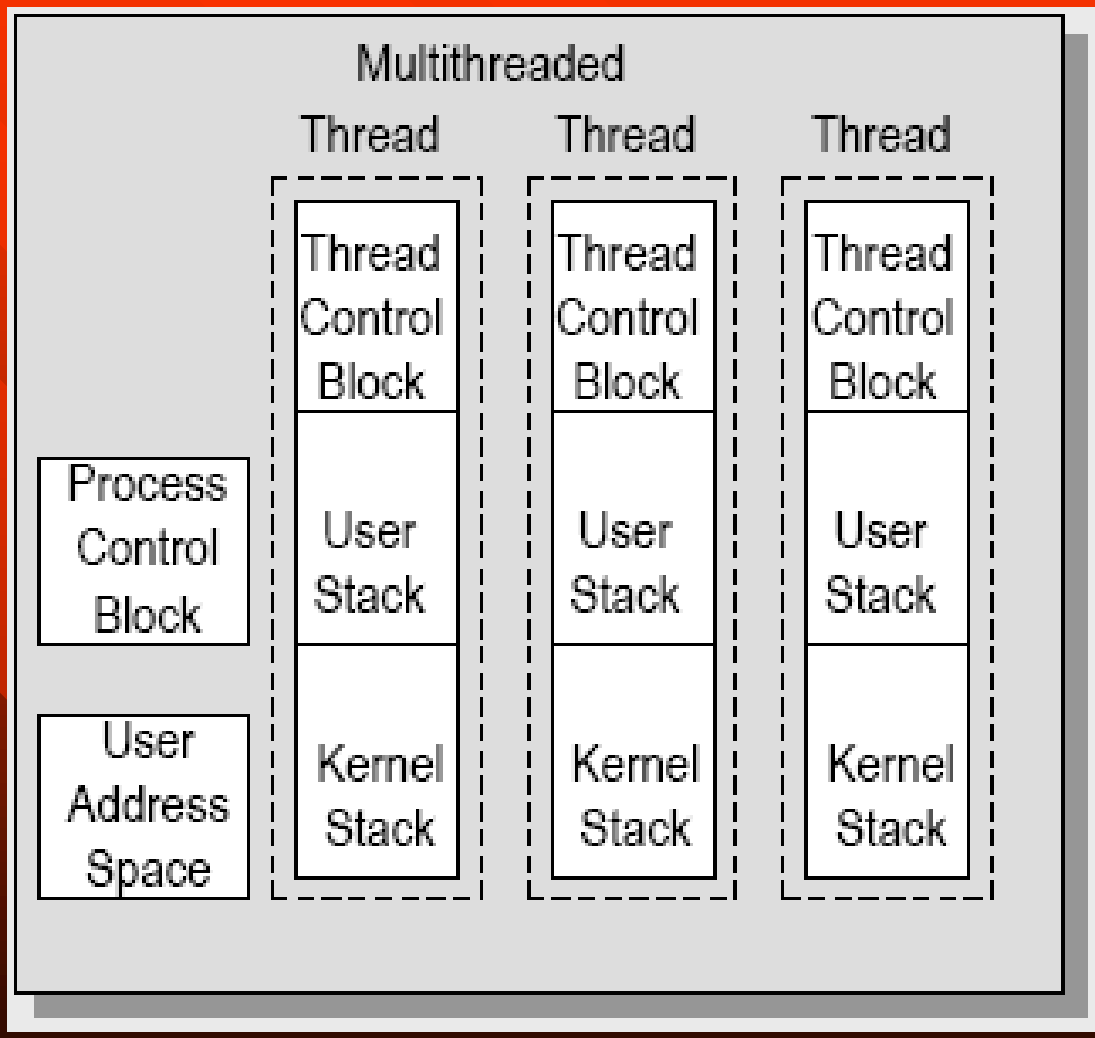


Implementação de threads

- **Threads são implementadas através de estruturas de dados similares ao descritor de processo**
  - Descritor de threads
  - Menos complexa (leve)
- **Podem ser implementadas em dois níveis diferentes:**
  - Espaço de usuário
  - Espaço de sistema

Modelos de processos single threaded e multithreaded





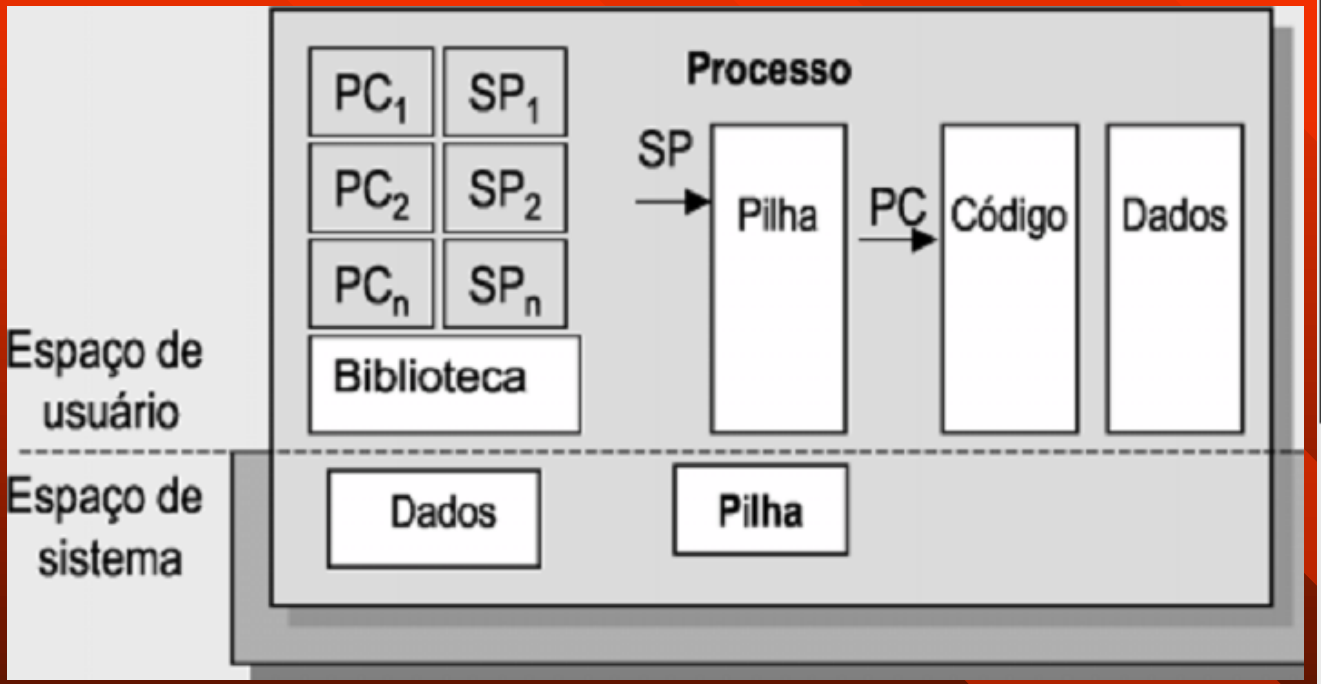
Modelo N:1 (1)

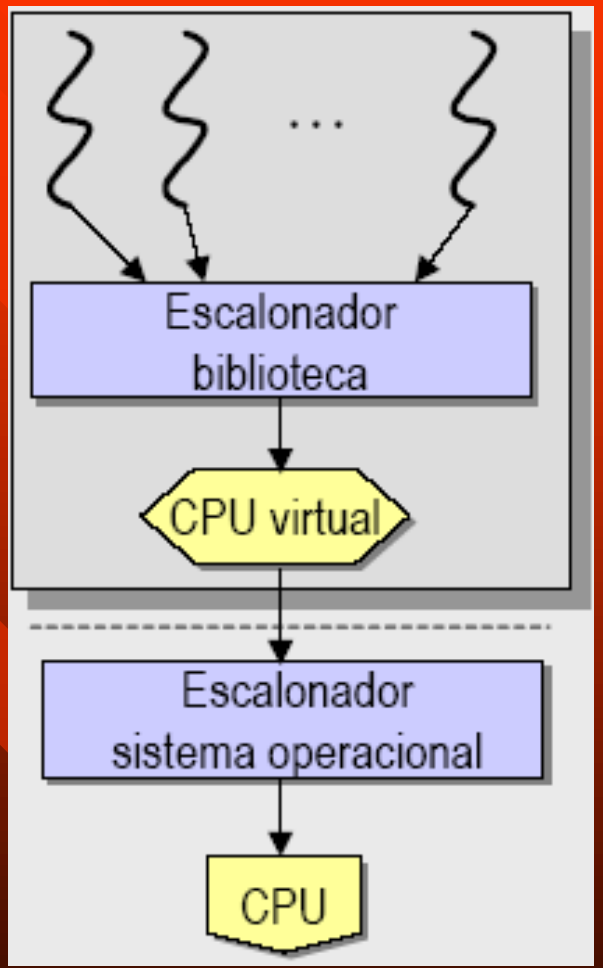
- **Threads a nível de usuário**
  - User level threads ou ainda process scope
- **Todas as tarefas de gerenciamento de threads é feito a nível da aplicação**
  - Threads são implementadas por uma biblioteca que é ligada ao programa
  - Interface de programação (API) para funções relacionadas com threads
    - » e.g; criação, sincronismo, termino, etc

Modelo N:1 (2)

- **O sistema operacional não "enxerga" a presença das threads**
- **A troca de contexto entre threads é feita em modo usuário pelo escalonador embutido na biblioteca**
  - Não necessita privilégios especiais
  - Escalonamento depende da implementação

Implementação modelo N:1





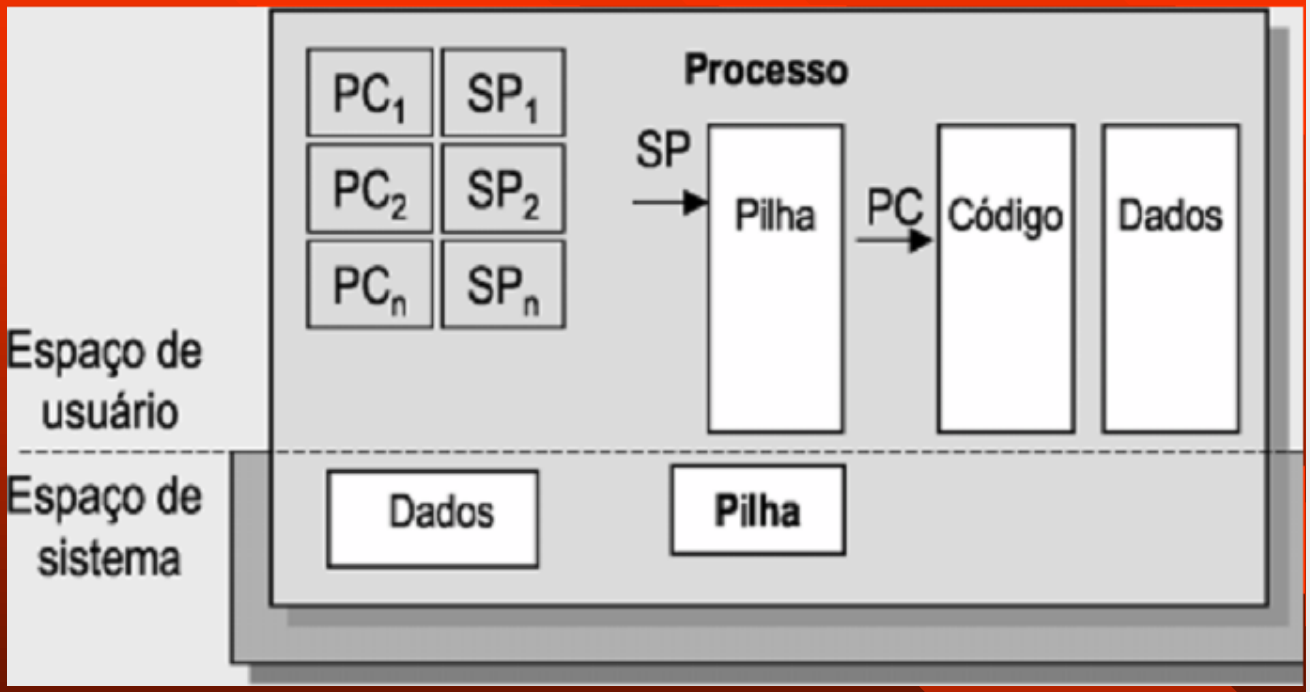
Vantagens e desvantagens

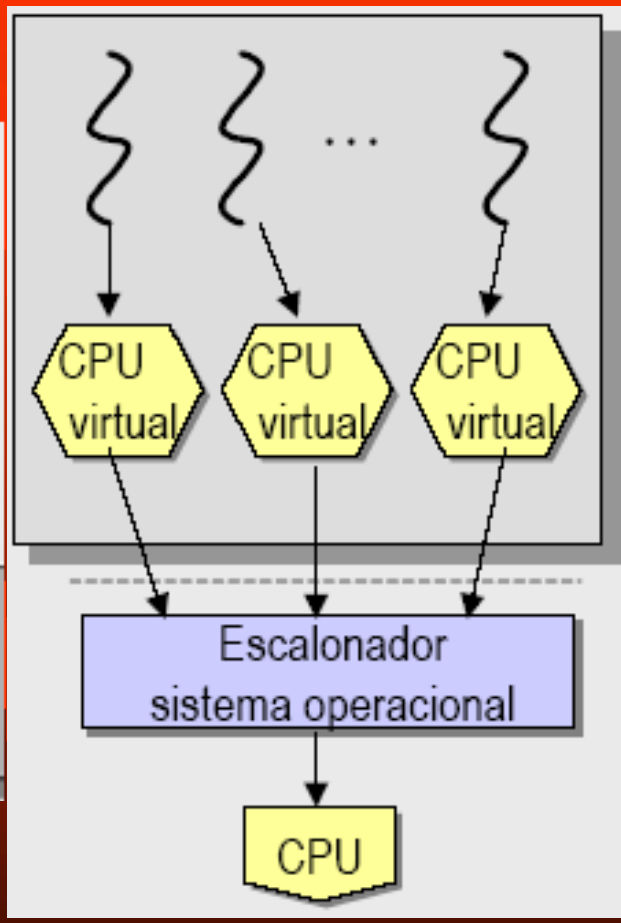
- **Vantagens:**
  - Sistema operacional divide o tempo do processador entre os processos «pesados» e, a biblioteca de threads divide o tempo do processo entre as threads
  - Leve: sem interação/intervenção do sistema operacional
- **Desvantagens:**
  - Uma thread que realiza uma chamada de sistema bloqueante leve ao bloqueio de todo o processo
  - e.g.; operações de entrada/saída
  - Não explora paralelismo em maquinas multiprocessadoras

Modelo 1:1

- **Threads a nível do sistema**
  - kernel level threads ou ainda system scope
- **Resolver desvantagens do modelo N:1**
- **O sistema operacional "enxerga" as threads**
  - Sistema operacional mantém informações sobre processos e sobre threads
  - Troca de contexto necessita a intervenção do sistema operacional
- **O conceito de threads é considerado na implementação do sistema operacional**

Implementação modelo 1:1





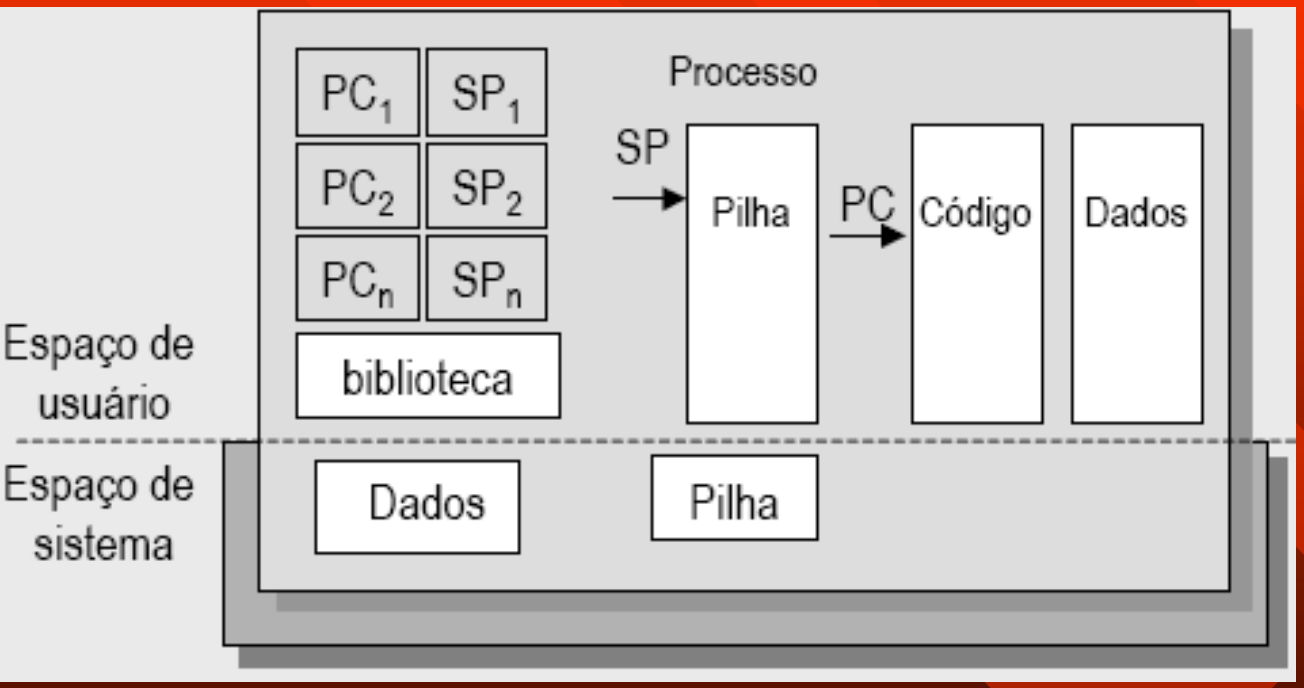
Vantagens e desvantagens

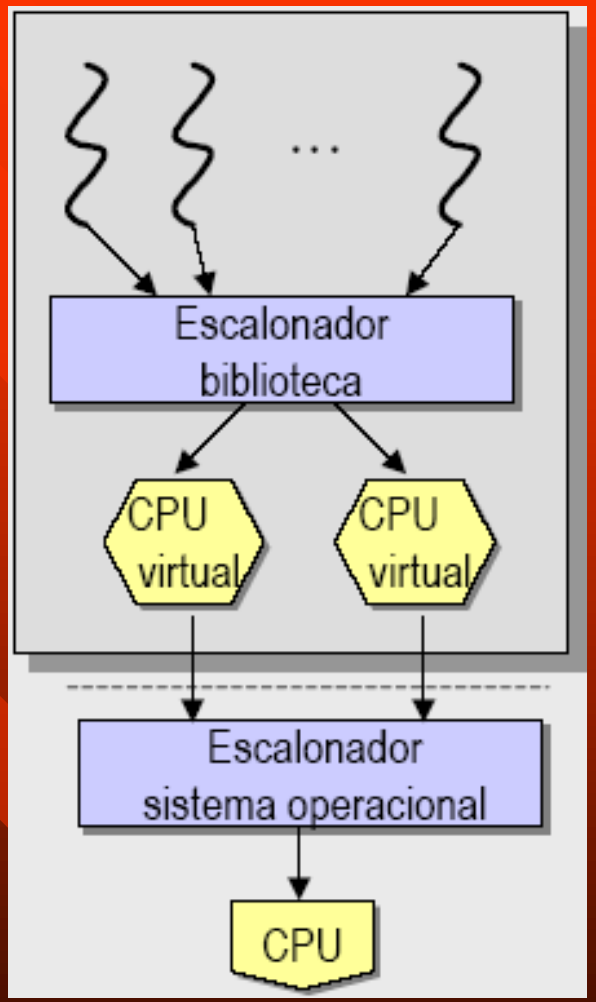
- **Vantagens:**
  - Explora o paralelismo de maquinas multiprocessadoras (SMP)
  - Facilita o recobrimento de operações de entrada/saída por cálculos
- **Desvantagens:**
  - Implementação "mais pesada" que o modelo N:1

Modelo M:N

- **Abordagem que combine os modelos N:1 e 1:1**
- **Oferece dois níveis de escalonamento**
  - Nivel usuário: threads sobre unidade de escalonamento
  - Nível sistema: unidades de escalonamento sobre processador
- **Dificuldade é parametrizar M e N**

Implementação modelo M:N





Porque utilizar threads ?

- **Permitir a exploração do paralelismo real oferecido por maquinas multiprocessadores (modelo M:N ou 1:1)**
- **Aumentar número de atividades executadas por unidade de tempo (throughput)**
- **Diminuir tempo de resposta**
  - Possibilidade de associar threads a dispositivos de entrada/saída
- **Sobrepor operações de calculo com operações de entrada e saída**

Vantagens de multithreading

- Tempo de criação/destruição de threads é inferior que tempo de criação/destruição de um processo
- Chaveamento de contexto entre threads é mais rápido que tempo de chaveamento entre processos
- Como threads compartilham o descritor do processo que as porta, elas dividem o mesmo espaço de endereçamento o que permite a comunicação por memória compartilhada sem interação com o núcleo

Escalonadores não-preemptivos

Escalonamento

- **O escalonador é a entidade do sistema operacional responsável por selecionar um processo apto para executar no processador**
- **O objetivo é dividir o tempo do processador de forma justa entre os processos aptos a executar**
- **Típico de sistemas multiprogramados: batch, time-sharing, multiprogramado ou tempo real**
  - Requisitos e restrições diferentes em relação a utilização da CPU
- **Duas partes:**
  - Escalonador: política de seleção
  - Dispatcher: efetua a troca de contexto

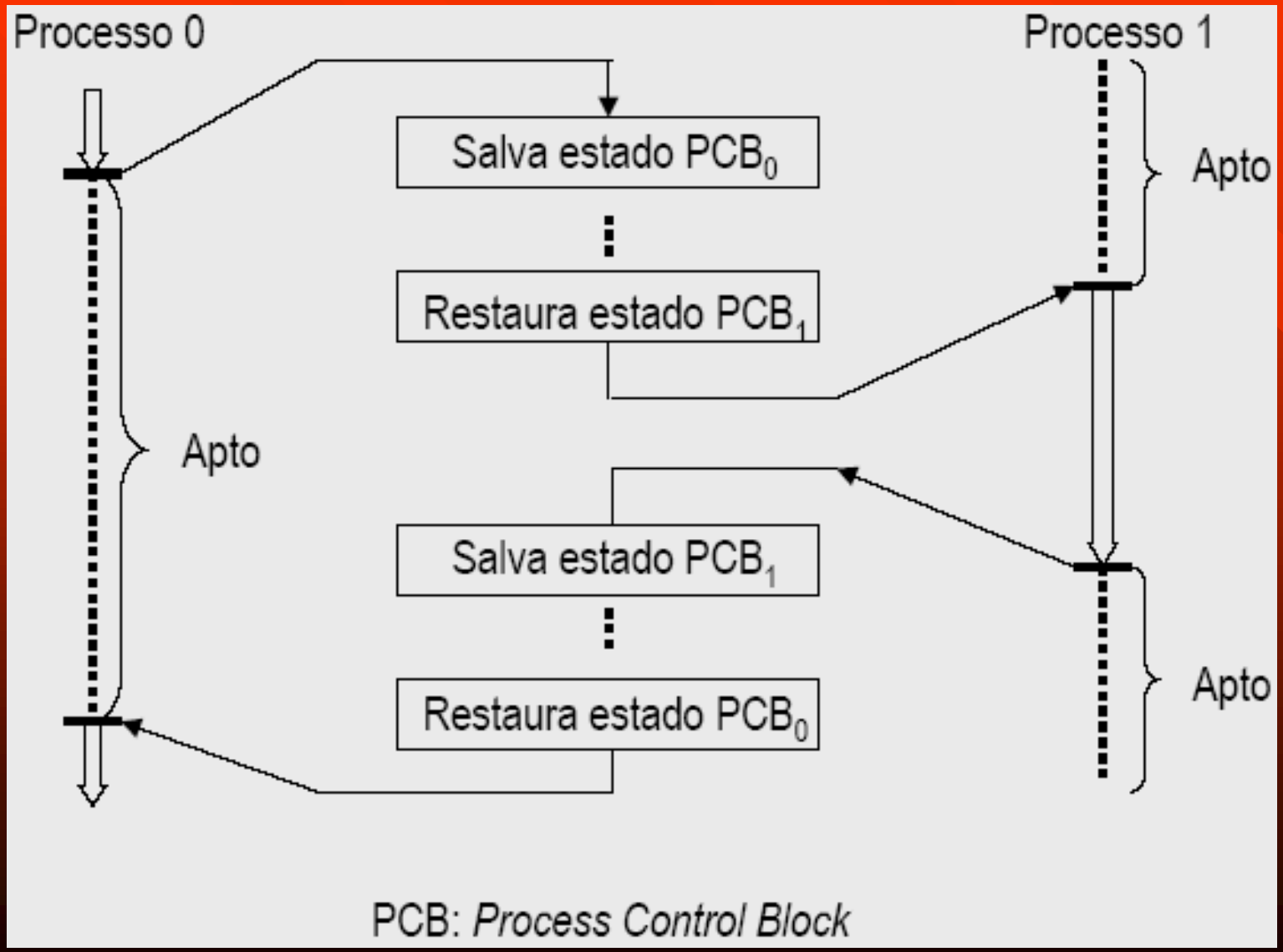
Objetivos do escalonamento

- **Maximizar a utilização do processador**
- **Maximizar a produção do sistema (throughput)**
  - Numero de processos executados por unidade de tempo
- **Minimizar o tempo de execução (turnaround)**
  - Tempo total para executar um determinado processo
- **Minimizar o tempo de espera**
  - Tempo que um processo permanece na lista de aptos
- **Minimizar o tempo de resposta**
  - Tempo decorrido entre uma requisição e a sua realização

Situações típicas para execução do escalonador

- **Dependem se o escalonador é preemptivo ou não, se considera prioridades ou não, etc...**
  - Sempre que a CPU estiver livre e houver processos aptos a executar
  - Criação e termino de processos
  - Um processo de mais alta prioridade ficar apto a executar
  - Interrupção de tempo
    - » Processo executou por um período de tempo máximo permitido
  - Interrupção de dispositivos de entrada e saída
  - Interrupção por falta de pagina (segmento) em memória
    - » Endereço acessado não está carregado na memória (memória virtual)
  - Interrupção por erros

Chaveamento de contexto (dispatcher)



Níveis de escalonamento

- Longo prazo
- Médio prazo
- Curto prazo

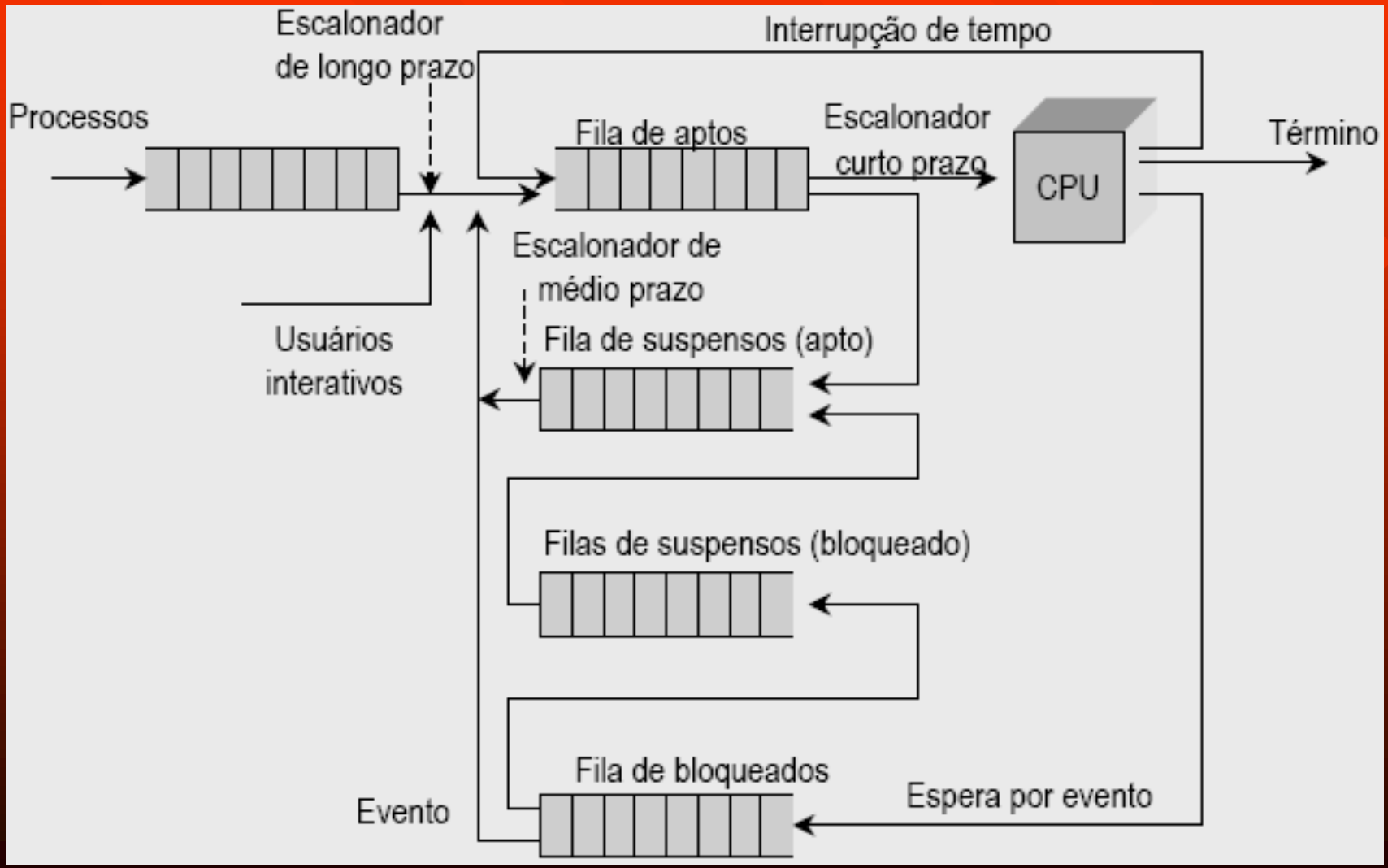
Escalonador longo prazo

- **Executado quando um novo processo é criado**
- **Determina quando um processo novo passa a ser considerado no sistema, isto é, quando apos sua criação ele passa a ser apto**
  - Controle de admissão
- **Controla o grau de multiprogramação do sistema**
  - Quanto major o numero de processos ativos, menor a porcentagem de tempo de use do processador por processo

Escalonador de curto prazo

- **Mais importante**
- **Determina qual processo apto devera utilizar o processador**
- **Executado sempre que ocorre eventos importantes:**
  - Interrupção de relógio
  - Interrupção de entrada/saída
  - Chamadas de sistemas
  - Sinais (interrupção software)

Diagrama de escalonamento



Tipos de escalonador

- **Um vez escalonado, o processo utiliza o processador até que:**
  - **Não preemptivo:**
    - » Termino de execução do processo
    - » Execução de uma requisição de entrada/saída ou sincronização
    - » Liberação voluntária do processador a outro processo (yield)
  - **Preemptivo:**
    - » Termino de execução do processo
    - » Execução de uma requisição de entrada/saída ou sincronização
    - » Liberação voluntária do processador a outro processo (yield)
    - » Interrupção de relógio
    - » Processo de mais alta prioridade esteja pronto para executar

Algoritmos de Escalonamento (1)

- **Algoritmo de escalonamento seleciona qual processo deve executar em um determinado instante de tempo**
- **Existem vários algoritmos para atingir os objetivos do escalonamento**
- **Os algoritmos buscam:**
  - Obter bons tempos médios ao invés de maximizar ou minimizar um determinado critério
  - Privilegiar a variância em relação a tempos médios

Algoritmos de Escalonamento (2)

- **Algoritmos não-preemptivos (cooperativos)**
  - First-In First-Out (FIFO) ou First-Come First-Served (FCFS)
  - Shortest Job First (SJF) ou Shortest Process Next (SPN)
- **Algoritmos preemptivos**
  - Round robin (circular)
  - Baseado em prioridades
- **Existem outros algoritmos de escalonamento**
  - High Response Ratio Next (HRRN)
  - Shortest Remaining Time (SRT)
  - etc...

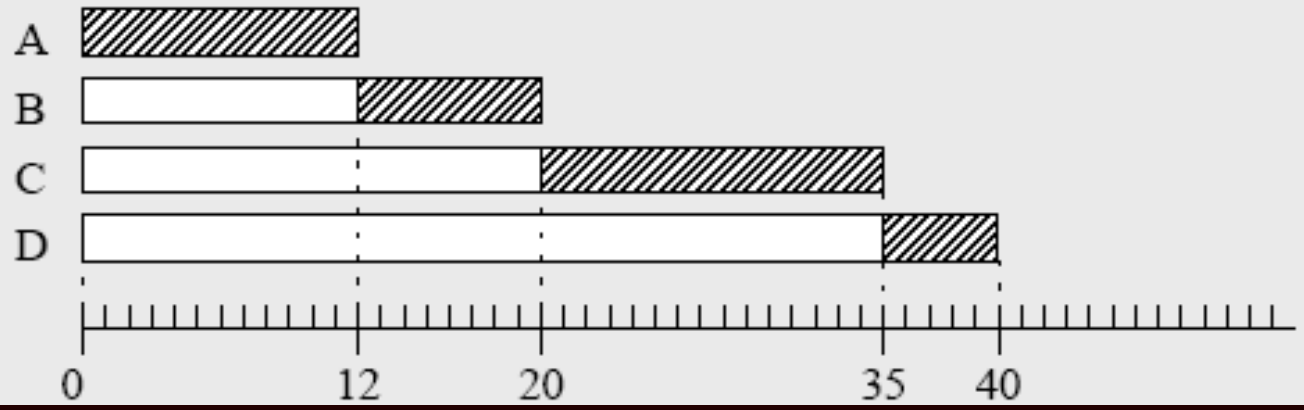
First In First Out

- **Simples de implementar**
  - Fila
- **Funcionamento:**
  - Processos que se tornam aptos são inseridos no final da fila
  - Processo que esta no inicio da fila e o próximo a executar
  - Processo executa ate que:
    - » Libere explicitamente o processador
    - » Realize uma chamada de sistema (bloqueado)
    - » Termine sua execução

First In First Out

- **Desvantagem:**
  - Prejudica processos I/O bound
- **Tempo médio de espera na fila de execução:**
  - Ordem A-B-C-D = (0 + 12 + 20 + 35) / 4 = 16.75 u.t.
  - Ordem D-A-B-C = (0 + 5 + 17 + 25) / 4 = 11.7u.t.

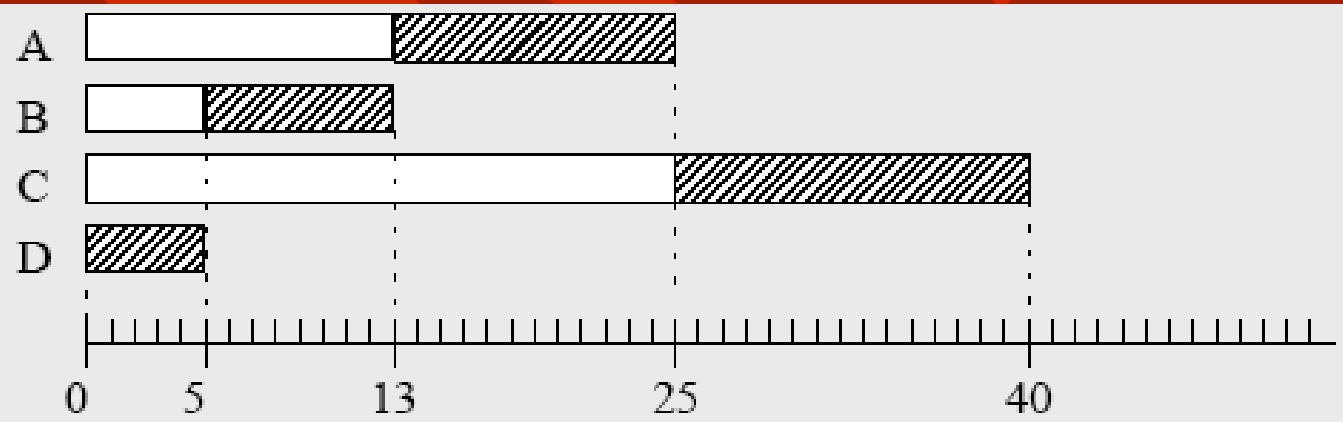
Processo	Tempo
A	12
B	8
C	15
D	5



SJF -Shortest Job First (1)

- Originário do fato que o menor tempo médio é obtido quando se executa primeiro os processos de menor ciclo de processador (I/0 bound)

Processo	Tempo
A	12
B	8
C	15
D	5



Tempo médio: (0 + 5 + 13 + 25)/4 = 10.75 u.t

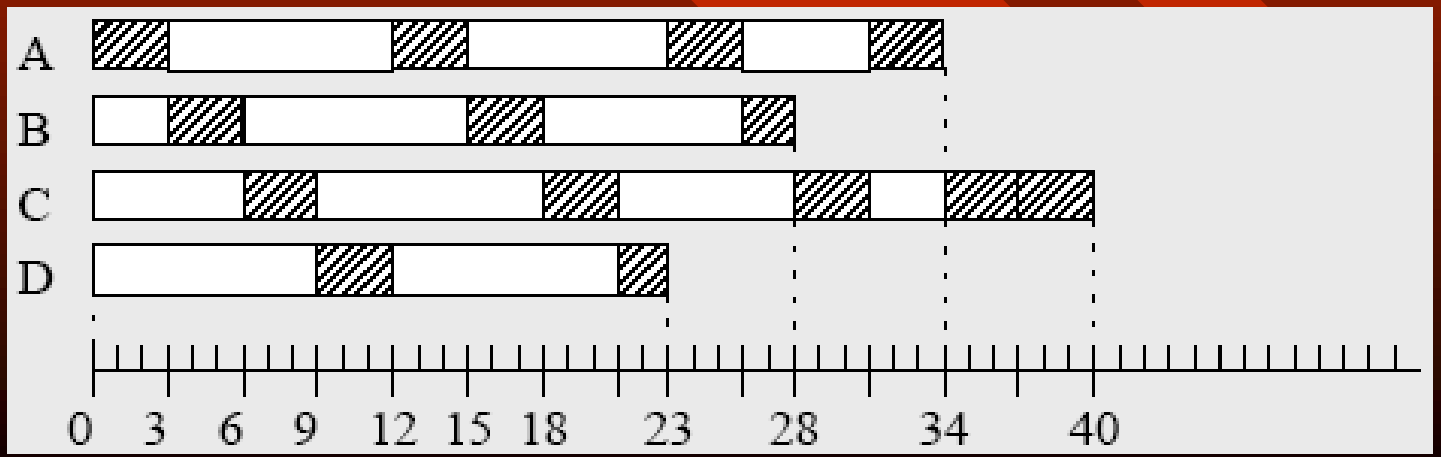
Escalonamento preemptivo

Escalonadores preemptivos

- **Por interrupção de tempo**
  - Round robin (circular)
- **Por prioridades**

RR - Round Robin (1)

- **Similar ao algoritmo FIFO, só que:**
  - Cada processo recebe um tempo limite máximo (time-slice, quantum) para executar um ciclo de processador
- **Fila de processos aptos é uma fila circular**
- **Necessidade de um relógio para delimitar as fatias de tempo**
  - Interrupção de tempo



RR - Round Robin (2)

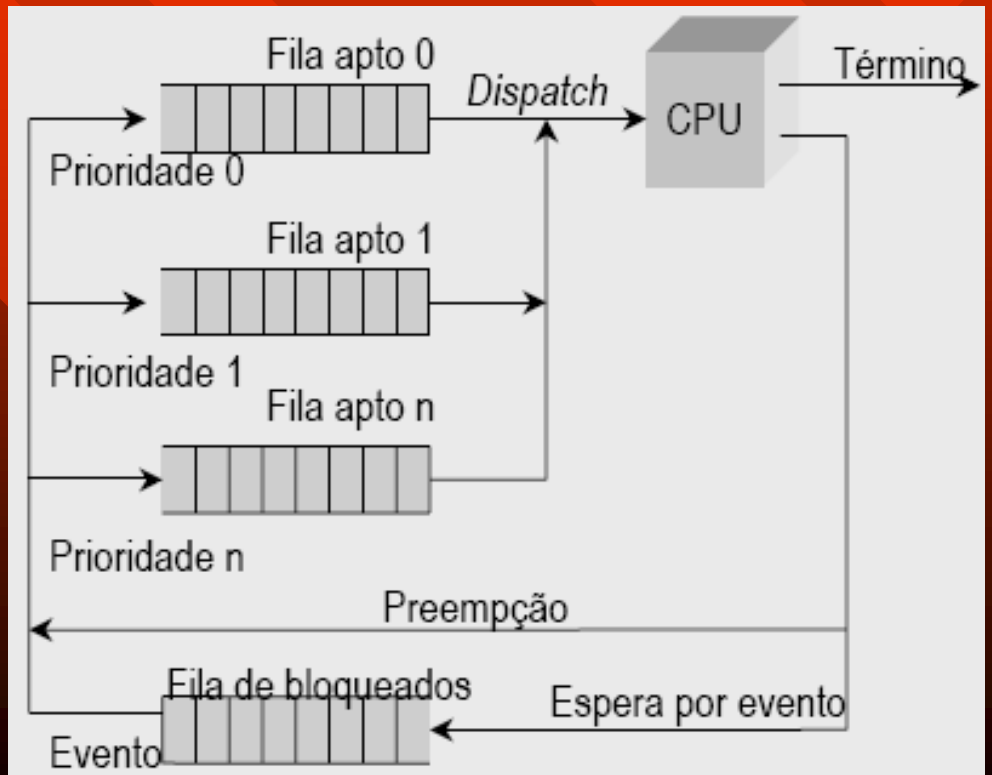
- **Por ser preemptivo, um processo perde o processador quando:**
  - Libera explicitamente o processador (yield)
  - Realize uma chamada de sistema (bloqueado)
  - Termina sua execução
  - Quando sua fatia de tempo é esgotada

Escalonamento com prioridades

- **Sempre que um processo de maior prioridade que o processo atualmente em execução entrar no estado apto deve ocorrer uma preempção**
  - A existência de prioridades pressupõem a preempção
  - E possível haver prioridade não-preemptiva
- **Escalonador deve sempre selecionar o processo de mais alta prioridade segundo uma política:**
  - Round-Robin
  - FIFO (FCFS)
  - SJF (SPN)

Implementação de escalonador com prioridades

- Múltiplas filas associadas ao estado apto
- Cada fila uma prioridade
  - Pode ter sua própria política de escalonamento (FIFO, SJF, RR)



Exemplo: pthreads

- **A política de escalonamento FIFO com prioridade considera:**
  - Quando um processo em execução é preemptado ele é inserido no inicio de sua fila de prioridade
  - Quando um processo bloqueado passa a apto ele é inserido no final da fila de sua prioridade
  - Quando um processo troca de prioridade ele é inserido no final da fila de sua nova prioridade
  - Quando um processo em execução "passa a vez" para um outro processo ele e inserido no final da fila de sua prioridade

Como definir a prioridade de um processo?

- **Prioridade estática:**
  - Um processo é criado com uma determinada prioridade e esta prioridade e mantida durante todo o tempo de vida do processo
- **Prioridade dinâmica:**
  - Prioridade do processo é ajustada de acordo com o estado de execução do processo e/ou do sistema
    - » e.g; ajustar a prioridade em função da fração do quantum que foi realmente utilizada pelo processo.

Problemas com prioridades

- **Um processo de baixa prioridade pode não ser executado**
  - Postergação indefinida (starvation)
- **Processo com prioridade estática pode ficar mal classificado e ser penalizado ou favorecido em relação aos demais**
  - Típico de processos que durante sua execução trocam de padrão de comportamento (CPU bound a I/O bound e vice-versa)
- **Solução:**
  - Múltiplas filas com realimentação

Escalonamento não-preemptivo com prioridades

- **SJF é um forma de priorizar processos**
  - A prioridade e o inverso do próximo tempo previsto para ciclo de CPU
- **Processos de igual prioridade são executados de acordo com uma política FIFO**
- **Problema de postergação indefinida (starvation)**
  - Processo de baixa prioridade não é alocado a CPU por sempre existir um processo de mais alta prioridade a ser executado
  - Solução:
    - » Envelhecimento
- **O conceito de prioridade é mais "consistente" com preempção**
  - Processo de maior prioridade interrompe a execução de um menos prioritário