Gerência de Processos

Algoritmos

Jiyan Yari

Escalonamento de Processos

Quando um processo estiver ativo, ou seja, pronto para executar, o sistema operacional deve decidir qual irá ser executada da CPU.

A parte do sistema operacional que toma esta decisão é chamada escalonador e o algoritmo utilizado é o algoritmo de escalonamento.

Escalonamento de Processos

Critérios observados pelo algoritmo de escalonamento do SO:

1. progresso: garantir que cada processo tenha acesso à CPU;

2. eficiência: manter a CPU ocupada praticamente 100% do tempo;

3. resposta: minimizar o tempo de resposta na execução dos processos, principalmente os interativos (editores, planilhas, etc);

Escalonamento de Processos

4. espera: minimizar o tempo de espera de serviços não interativos (compilação, impressão, etc);

5. vazão: maximizar o número de processos executados por unidade de tempo.

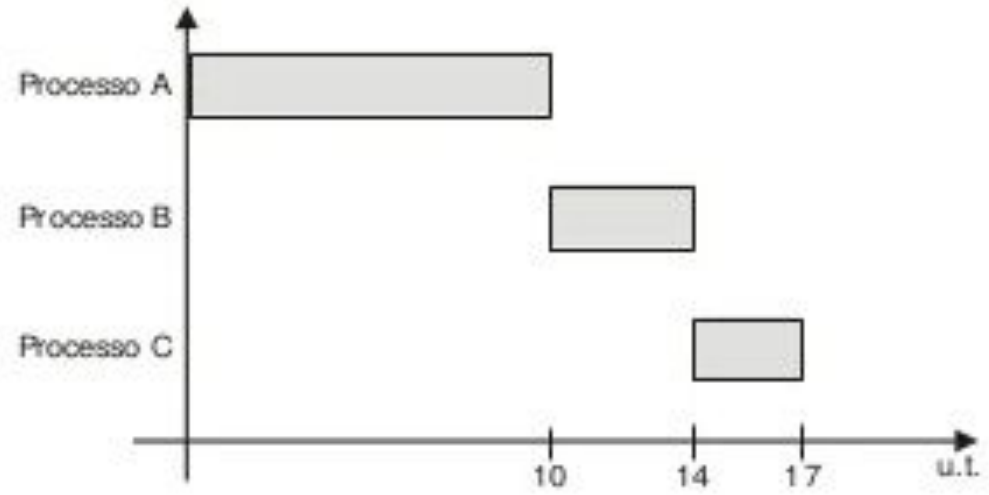
First In First Out - Primeiro que Entra Rrimeiro que Sai - FIFO

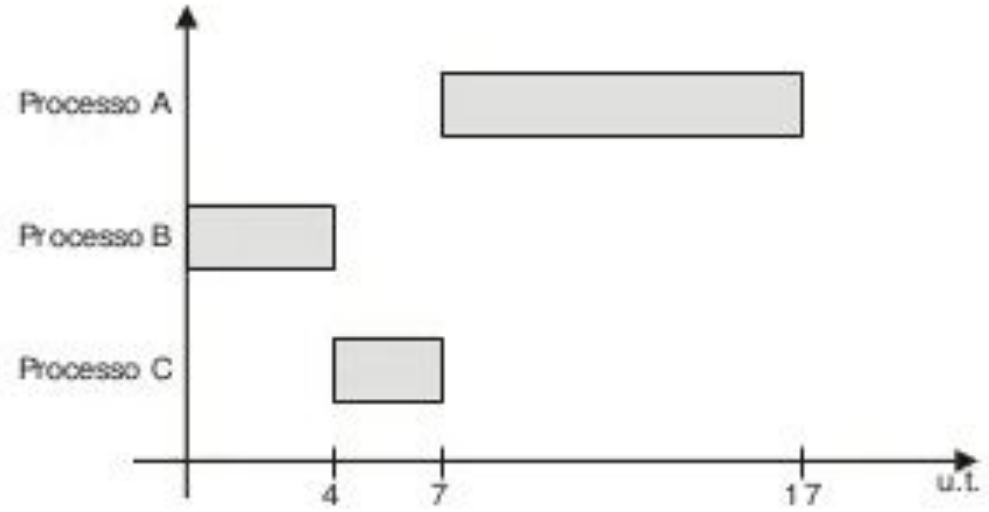
Este é o mais antigo e simples algoritmo de escalonamento.

O primeiro processo que entra na fila de prontos, fila dos processos a serem executados, é o primeiro a ser executado.

O problema desse algoritmo é que quando entrava um processo muito grande, os outros que vinham depois esperavam muito tempo (starvation - inanição) sem poder usar a CPU.

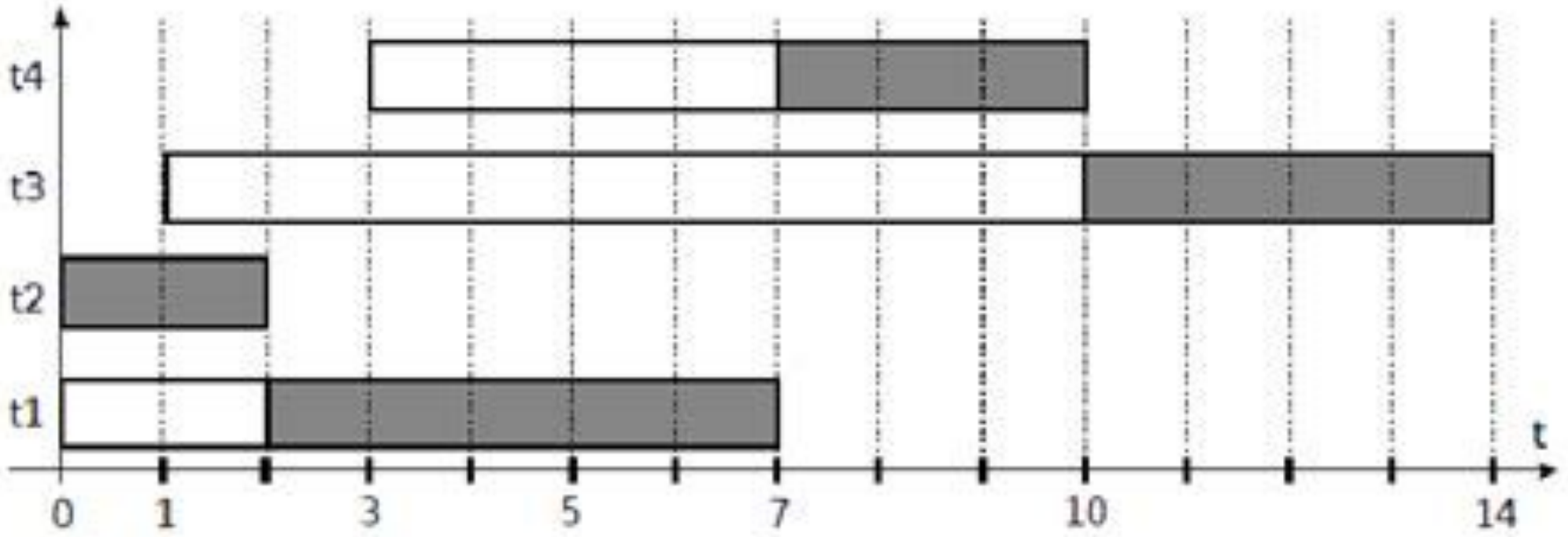
First In First Out - Primeiro que Entra Rrimeiro que Sai - FIFO





Processo	Tempo de processador (u.t.)
A	10
B	4
C	3

First In First Out - Primeiro que Entra Rrimeiro que Sai - FIFO



Round Robin

Um dos algoritmos de escalonamento mais simples, mas resolve o problema do FIFO.

Cada processo é executado por um intervalo de tempo (quantum).

Se o processo ainda estiver executando ao final do quantum, ele é suspenso e a CPU é alocada a outro processo.

Se o processo acabar ou for bloqueado antes do final do quantum, a CPU também é passada a outro processo.

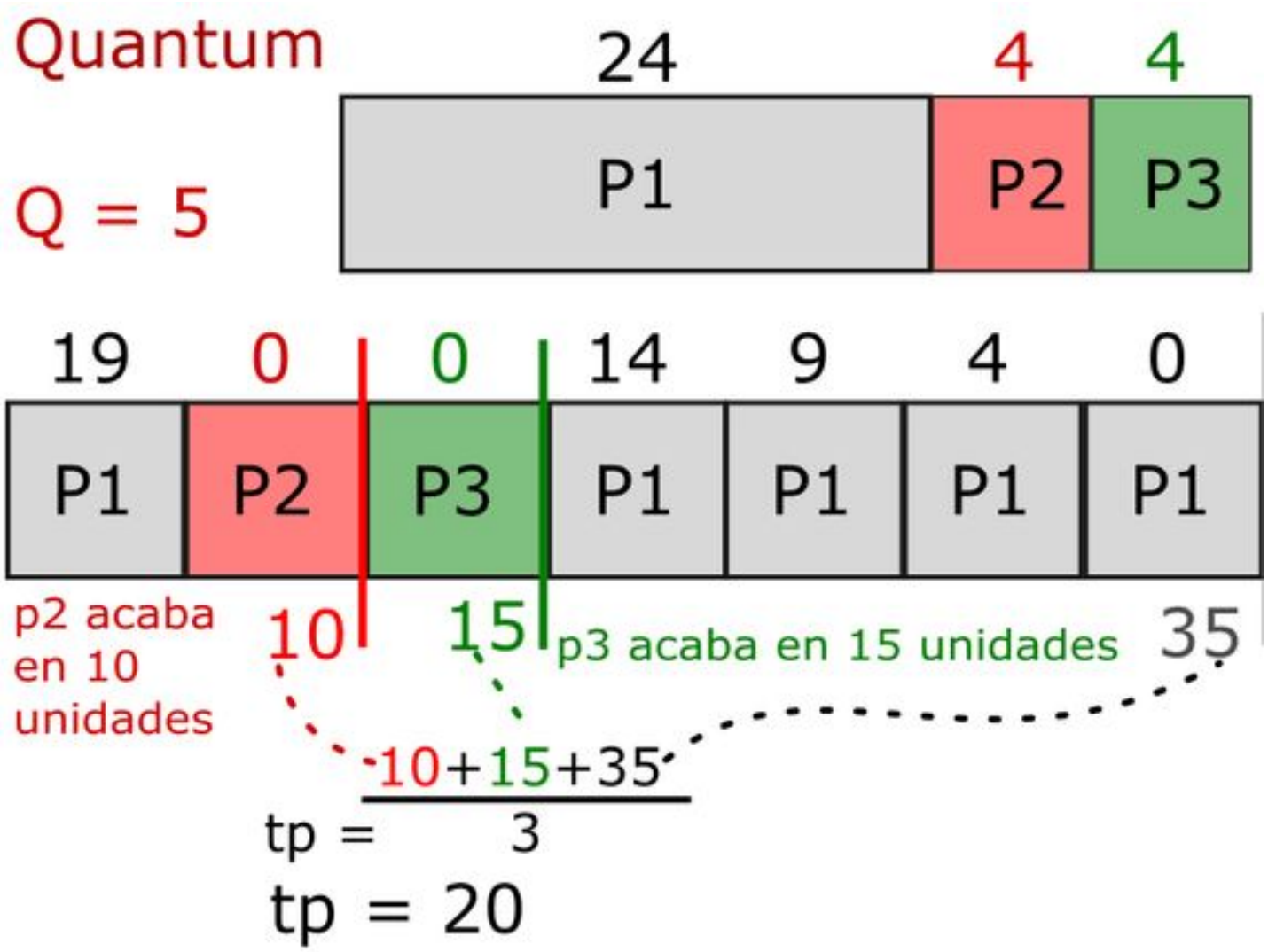
Round Robin

Deve ser analisada o tamanho do quantum.

Se for muito pequeno, diminui a eficiência da CPU, pois a alocação da CPU para outro processo implica um certo overhead.

Se for muito grande, degrada a resposta para os processos interativos.

Round Robin


Quantum
Q = 5
24
4
4
P1
P2
P3
19
0
0
14
9
4
0
P1
P2
P3
P1
P1
P1
P1
p2 acaba en 10 unidades
10
15
p3 acaba en 15 unidades
35
10+15+35
tp =
3
tp = 20

Prioridade

Em contrapartida ao Round Robin, que considera todos os processos iguais, o algoritmo de prioridade atribui prioridade aos processos.

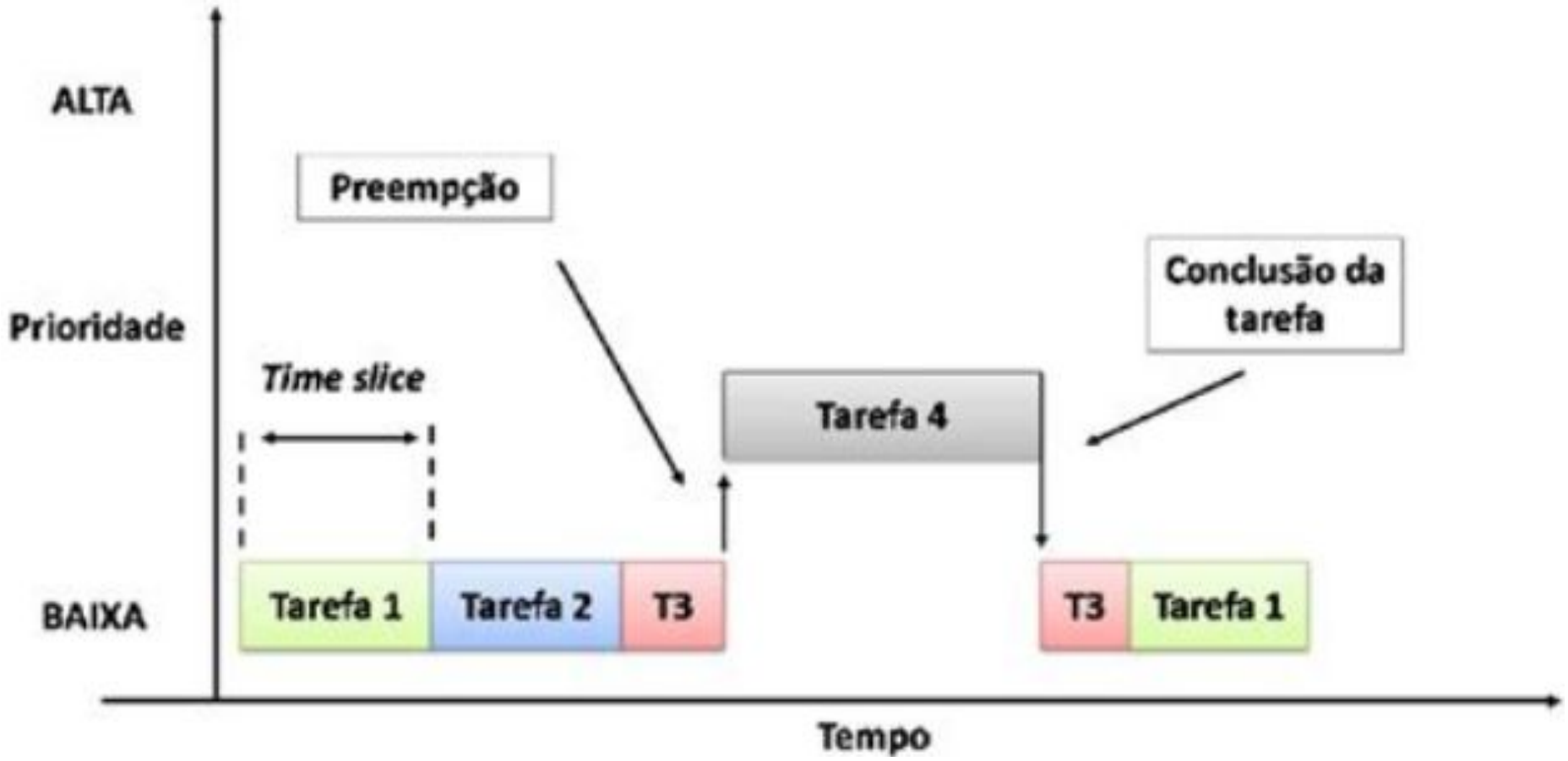
Aplicações urgentes e ou emergenciais, como um antivirus, por exemplo, precisam executar seus códigos com prioridades quando o SO os solicita.

Prioridade

A ideia básica é que cada processo tem uma prioridade e processos com prioridades superiores devem ser executados primeiro.

Para prevenir que processos de alta prioridade executem indefinidamente, o escalonador, via de regra, diminui a prioridade dos processos com o aumento de seu respectivo tempo de execução.

Prioridade


ALTA
Prioridade
BAIXA
Preempção
Conclusão da tarefa
Time slice
Tarefa 4
Tarefa 1
Tarefa 2
T3
T3
Tarefa 1
Tempo

Processos pequenos (Shortest Job First)

Algoritmo indicado para aplicações não interativas, onde o tempo médio de execução é conhecido.

O algoritmo define que as tarefas menores devem ser executadas primeiro.

Prova-se que esta política minimiza o tempo médio de espera das tarefas.

Processos pequenos (Shortest Job First)

- Shortest-Job-First (SJF)

  – Menor ciclo → entram em execução.

Tempo de espera médio: (11 + 6 + 2 + 0) = 19 / 4 = **4,75 !!!**

Multiplas filas

Semelhante ao algoritmo de prioridades, porém com quotas de tempo de acordo com sua prioridade.

Processos na classe de menor prioridade são executados por 1 quantum de tempo.

Processos na classe seguinte, por 2 quantums.

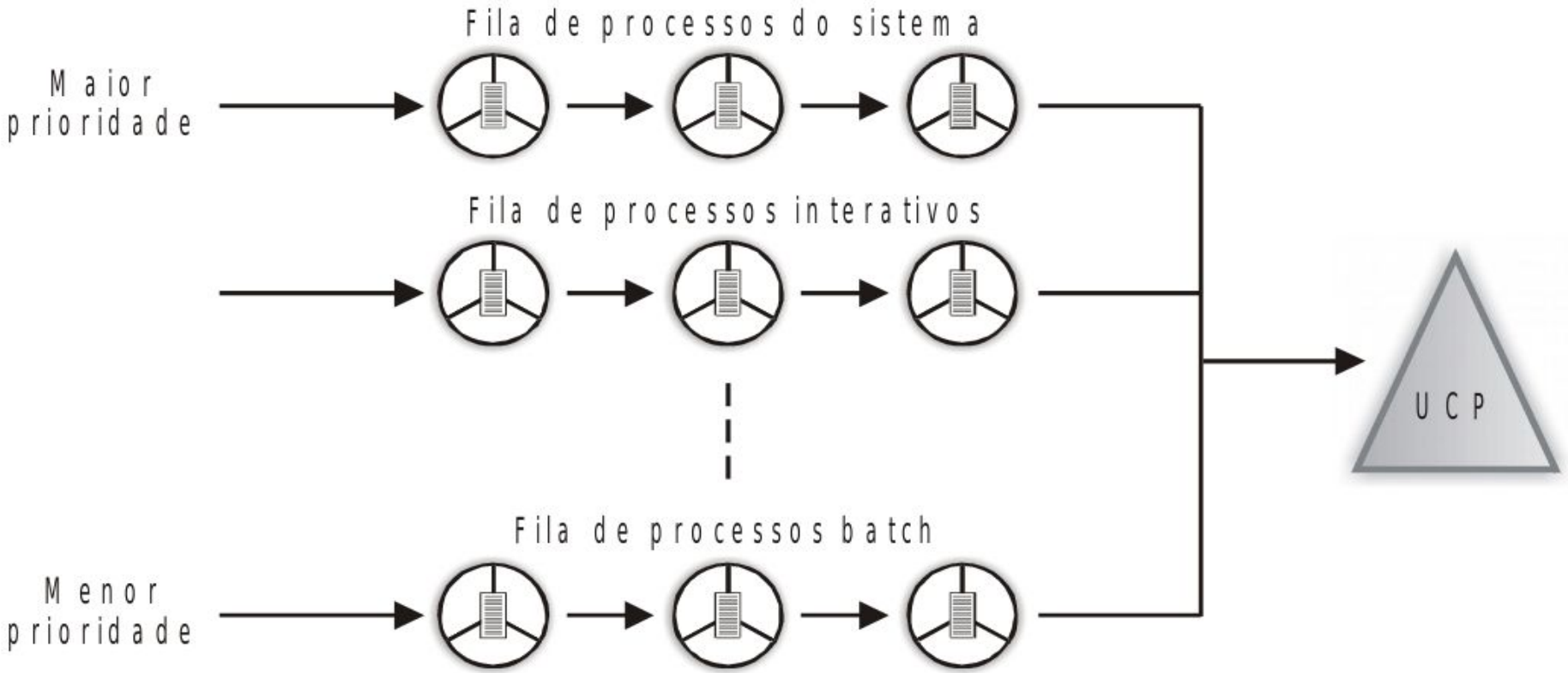
Na próxima classe por 4 quanta, e assim por diante.

Multiplas filas

Quando um processo utiliza todos os quantums a ele alocados, o mesmo é interrompido e sua classe tem a prioridade diminuída.

Este algoritmo diminui o número de comutações da CPU entre os processos ativos.

Multiplas filas



Descritores de processos

Os estados dos processos são descritos nos sistemas operacionais, de uma forma geral, com os seguintes elementos (Exemplo usando a estrutura do UNIX):

- a tabela de processos;

- e a área U.

Descritores de processos

A tabela de processos é uma estrutura mantida pelo núcleo com os seguintes elementos:

- o estado do processo;

- a localização da área U e do próprio processo, seja na memória primária ou secundária, bem como o tamanho do processo;

- o identificador do usuário (UID), isto é, o dono do processo;

Descritores de processos

- identificador do processo (PID), único durante toda a vida do processo;

- eventos aguardados pelo processo quando no estado bloqueado (dormindo);

- parâmetros de escalonamento, utilizados para decidir quais processos irão transitar entre estados de execução nos modos usuário e núcleo;

Descritores de processos

- sinais enviados ao processo, mas ainda não tratados;

- marcas de tempo como:
    - tempo total CPU;
    - despertadores armados pelo processo;
    - recursos consumidos do núcleo (estes parâmetros são utilizados no cômputo da prioridade de escalonamento do processo).

Descritores de processos

A área U de um processo armazena as seguintes informações:

- um ponteiro de volta ao respectivo índice na tabela de processos;

- privilégios que o processo dispõe, de acordo com o seu UID;

- tempos de execução nos modos usuário e núcleo;

Descritores de processos

- ponteiros para os gerenciadores de sinais denidos pelo processo;

- o terminal de login associado ao processo, caso exista;

- um campo de erro, armazenando os erros gerados por chamadas de sistema;

- parâmetros de entrada/saída, tais como endereço de dados a serem transferidos, tamanho destes dados, destino, etc;

Descritores de processos

- o diretório corrente e o diretório raiz;

- descritores de arquivos abertos pelo processo;

- limites (tamanho máximo de pilha, arquivos, etc);

- permissões (modos de acesso atribuídos durante a criação de arquivos).

Interrupção de processos

Processos podem ter sua execução alterada assincronamente por ação de um outro processo ou do usuário, via comando através do shell ou usando o gerenciador de processos.

Esta ação é realizada enviando um sinal ao SO.

Interrupção de processos

Sinais de processos podem ser enviados ao SO para:

- requisição de mudança de estado (ex: matar processo usando o kill);

- encerrar um processo-filho (subprocesso do processo principal, por exemplo, uma aba de navegador);

- informar ocorrência de exceções (ex: exceção de código);

Interrupção de processos

- informar situações de travamento;

- ocorrência de erros inesperados (ex: tela azul ou não execução de uma aplicação);

- do disparo de despertadores de tempo (ex: retorno da chamada sleep);

Interrupção de processos

- informar interrupções oriundas de terminais (ex: Ctrl-C);

- informar interrupções para fins de depuração (ex: ocorrência de um breakpoint).

Interrupção de processos

Sinais são explicitamente enviados a processos através da chamada de sistema kill.

O kill tem como parâmetros o PID do processo ao qual o sinal se destina e o tipo de sinal.

Interrupção de processos

O Unix/Linux possuem cerca de 30 tipos de sinais no UNIX System V, sendo que os principais são:

sinal	significado
SIGHUP	hang-up
SIGINT	interrupção
SIGILL	instrução ilegal (trap)
SIGFPE	exceção aritmética (trap)
SIGKILL	término forçado
SIGSEGV	violação de segmentação (trap)
SIGSYS	argumento inválido em chamada de sistema
SIGALRM	alarme de relógio
SIGSTOP	suspensão da execução
SIGCONT	continuação da execução
SIGCHLD	mudança de status de processo filho

Interrupção de processos

Os processos podem apresentar as seguintes respostas aos sinais:

- o processo é encerrado;

- o processo é bloqueado até a ocorrência de um sinal de desbloqueio;

- o processo ignora o sinal (travamento);

- o processo captura o sinal (o sinal não consegue agir no processo).

Interrupção de processos

Para cada sinal é definido um comportamento padrão para o processo que o recebe.

Geralmente a resposta é o encerramento do processo.

Um processo pode alterar o comportamento padrão frente a ocorrência de um dado sinal através da chamada de sistema signal.

Interrupção de processos

O sinal (ou signal) possui dois parâmetros:

- o número do sinal;

- e a ação a ser executada quando do recebimento deste sinal.

Interrupção de processos

A ação do segundo comportamento possui 3 valores:

0: o processo é terminado quando receber o sinal;

1: o sinal é ignorado;

Endereço válido de função: a função é chamada assincronamente quando da ocorrência do sinal. Esta função é denominada gerenciador do sinal para o processo.

Interrupção de processos

Exemplo:

Captura de interrupções de teclado:

- Quando o usuário executa Ctrl-C, o sinal SIGINT é enviado ao processo que está executando em foreground;

- O comportamento default para este sinal é o término do processo.

Comunicação e Sincronização Inter-processos

A comunicação entre processos pode ocorrer de 3 formas:

- pipes;

- sockets.

Comunicação e Sincronização Inter-processos

Pipes

O “pipe” (|) é utilizado quando ocorre comunicação entre 2 ou mais processos intra-host, ou seja, dentro do mesmo sistema computacional.

Ex:

$ ls | less

Comunicação e Sincronização Inter-processos

Socket

O “socket” (@) é utilizado quando ocorre comunicação entre 2 ou mais processos extra-host, ou seja, fora do mesmo sistema computacional, ou seja, entre sistemas computacionais separados.

Ex:

Comunicadores que utilizam protocolos de comunicação (whatsapp, Telegram e etc.).