SISTEMAS OPERACIONAIS

Escalonamento de Processos

Andreza Leite
andreza.leite@univasf.edu.br

Plano da Aula

- Componentes básicos
  - Algoritmos de Escalonamento
    - Conceito escalonamento
    - Tipos de escalonadores
    - Critérios de rendimento
    - Algoritmos de escalonamento

Escalonamento

- **Definição:**
  - O escalonamento consiste em distribuir o acesso aos recursos do sistema entre os processos que o solicitam.
- **Objetivo:**
  - Otimizar o rendimento dos recursos.
  - Priorizar o acesso aos recursos disponíveis.
- **Recursos que necessitam escalonamento:**
  - Dispositivos E/S (discos)
  - Processador
  - Memória

Escalonamento

- **Multiprogramação:**
  - O S.O. gerencia múltiplos processos na memória principal de forma simultânea.
  - Os p acesso ao proces
- **Escalon**
  - Decidi
    - Que trabalhos serão admitidos pelo sistema
    - Que processos serão mantidos na memória principal
    - Que processo utilizará a CPU quando ela estiver livre

O escalonador de processos é o responsável de tomar estas decisões, repartindo o uso da memória e do processador entre os processos ativos do sistema.

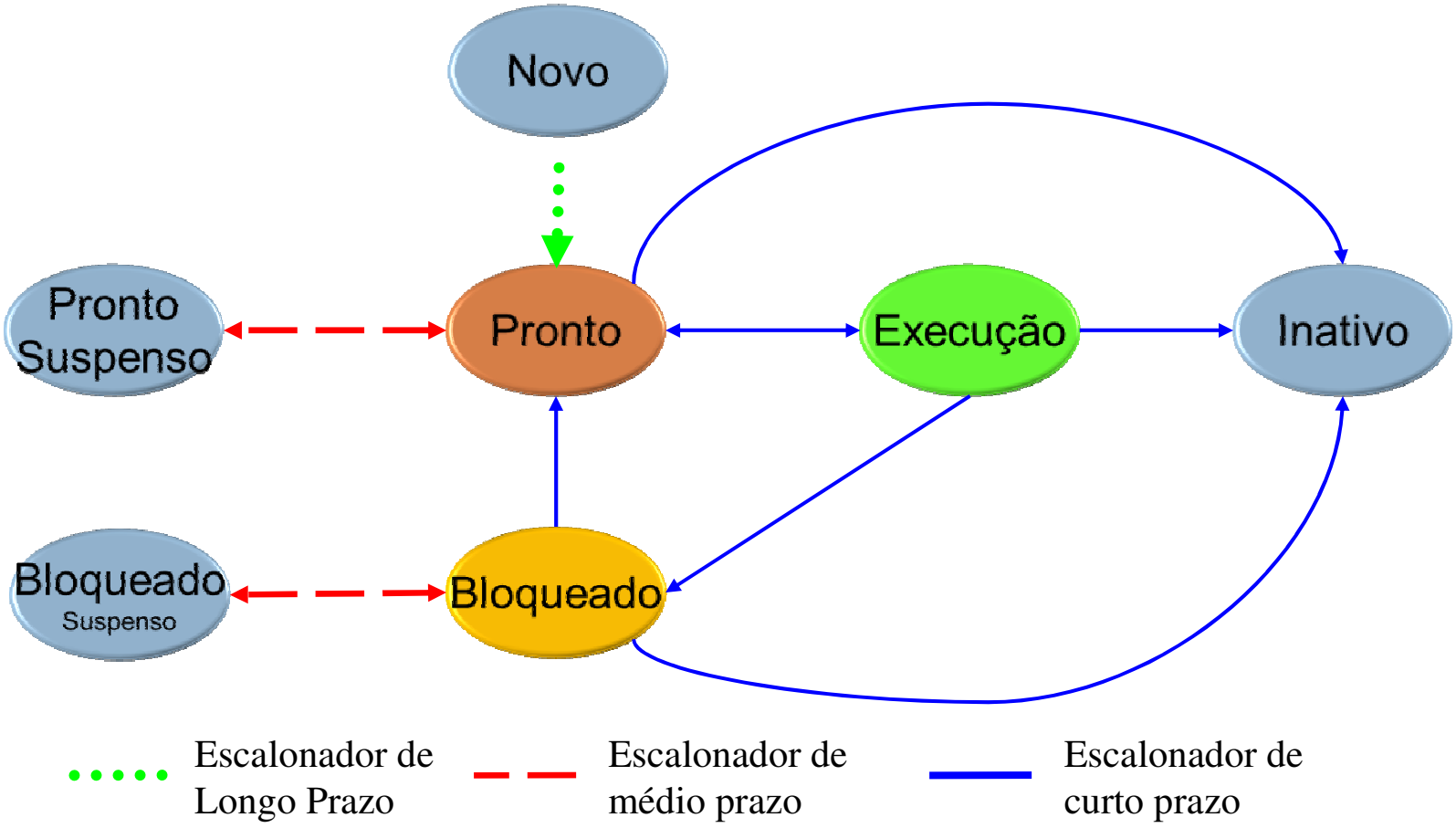
Escalonamento

- **Tipos de Escalonadores:**
  - **Escalonador de longo prazo**
    - É o responsável de controlar o grau de multiprogramação do sistema
    - Número de processos que serão executados “ao mesmo tempo”.
    - Admite novos trabalhos no sistema, convertendo estes em processos

Escalonamento

- **Tipos de Escalonadores:**
  - **Escalonador de médio prazo**
    - É o responsável de escolher os processos que serão removidos total ou parcialmente da memória para serem levados ao disco (suspensos)
    - Manter rendimento do sistema
  - **Escalonador de curto prazo**
    - Responsável por alocar à CPU os processos alocados em memória

Escalonamento



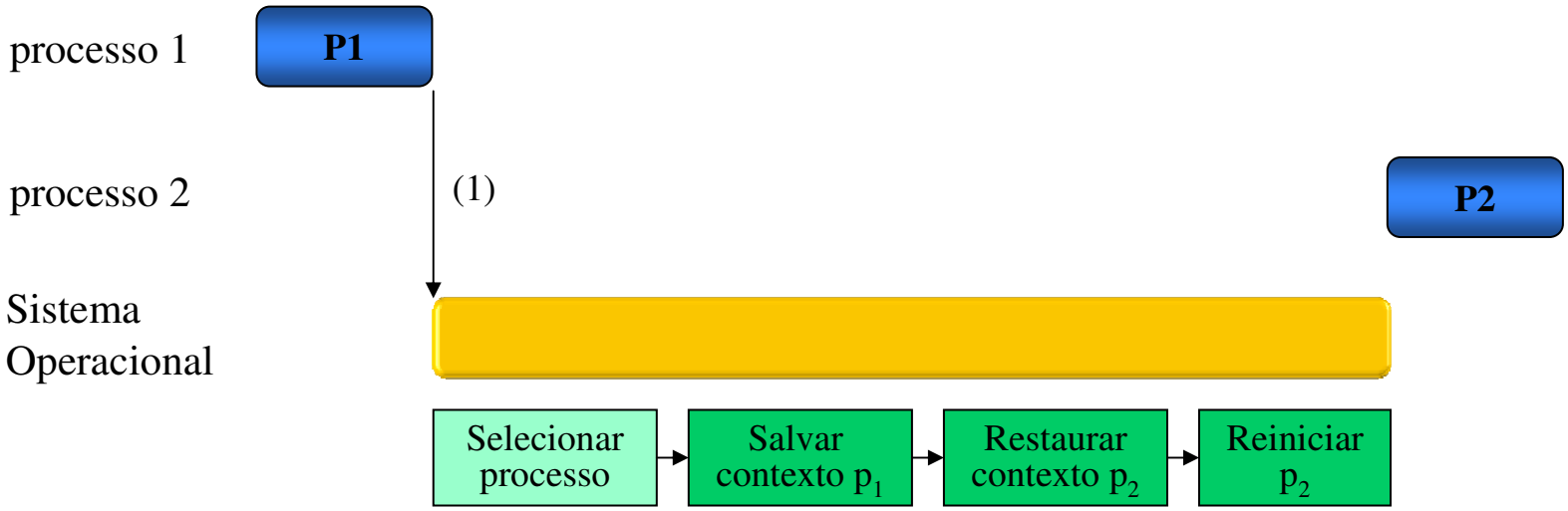
Escalonador:Curto Prazo

- **Escalonador**
  - Seleciona o processo para sua execução, atendendo a um determinado critério.
- ***Dispacher* (despachador)**

  É o módulo que dá controle da CPU para o processo selecionado pelo escalonador de curto prazo.
  - Salvar contexto do processo que sai da cpu
  - Restaurar contexto do processo que entra na cpu
  - Reiniciar a execução de processos
    - Alterar para estado pronto.
    - Configurar para o ponto apropiado do programa

Troca de contexto

Troca de processos



(1) Finalização do tempo de execução ou o processo se bloqueia à espera de um recurso que necessita

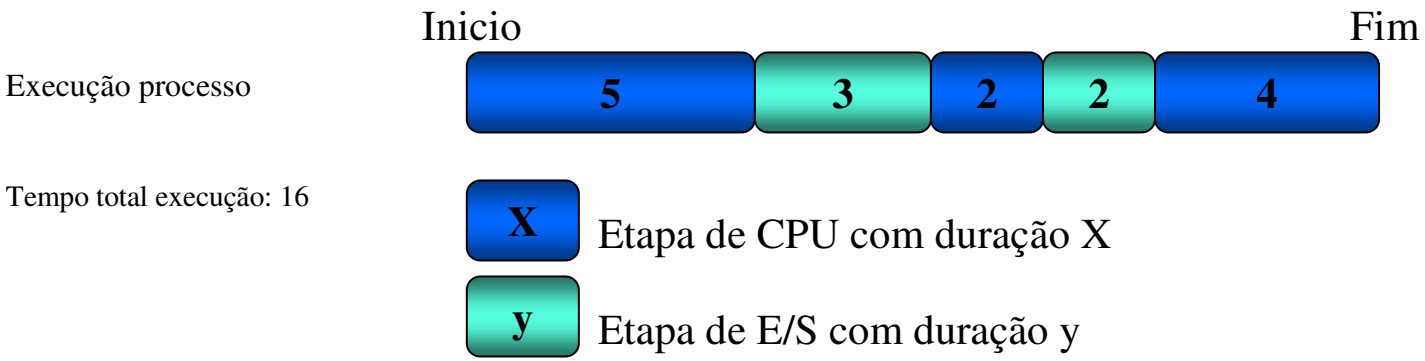
Escalonador Despachador

Tipos processos

- Do ponto de vista do escalonador os processos podem ser representados como uma sucessão de etapas:
  - Etapas de CPU.

    Processo esta executando instruções
  - Etapas de E/S.

    Processo utiliza ou espera por E/S.



Tipos de processos

- processos intensivos em CPU

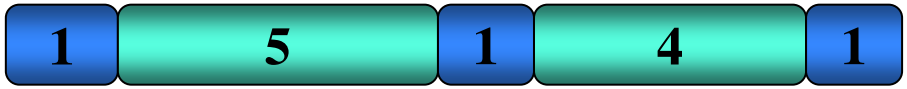
  as etapas de CPU são maiores que as de E/S.

processo CPU intensivo:

- processos intensivos en E/S

  as etapas de E/S são maiores que as de CPU.

processo E/S intensivo:



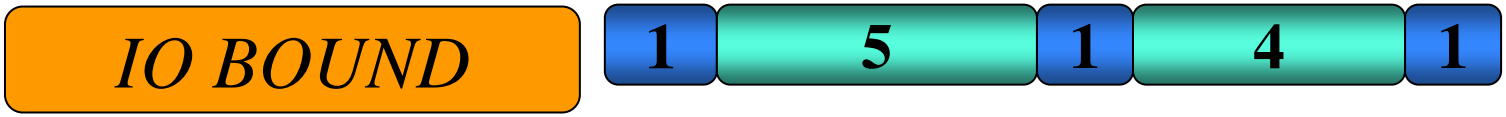
Tipos de processos

- processos intensivos em CPU

  as etapas de CPU são maiores que as de E/S.

- processos intensivos en E/S

  as etapas de E/S são maiores que as de CPU.



Escalonamento

- **Escalonar...**
  - Divisão equitativa do processador
  - Otimizar alguns critérios:
    - Grau de utilização da CPU.
    - Produtividade (***throughput***).
      - Número de processos terminados por unidade de tempo
    - **Tempo de retorno (*Turnaround time*).**
      - Tempo transcorrido desde que se lança um processo (entra na fila de prontos) até que finalize sua execução.
      - É a soma do tempo de espera para ir para a memória, tempo de espera na fila dos prontos, tempo em execução na CPU e o tempo de espera por recursos.

Escalonamento

- **Escalonar...**
  - Divisão equitativa do processador
  - Otimizar alguns critérios:
    - **Tempo de espera.**
      - Tempo que o processo permanece na fila de prontos.
      - É a soma dos períodos utilizados pelo processo no estado de Pronto.
    - **Tempo médio de espera.**
      - Tempo médio que todos os processos devem esperar...

Escalonamento

- **Escalonar...**
  - Divisão equitativa do procesador
  - Otimizar alguns critérios:
    - **Tempo de resposta.**
      - Tempo que transcorre desde que o processo é lançado até que exista uma resposta. (Sistemas Interativos)
    - **Tempo de serviço.**
      - Tempo esperado para a finalização do processo (CPU+E/S)
    - **Tempo de retorno normalizado.**
      - Razão entre o tempo de retorno e o tempo de serviço
      - Indica a demora de um processo em relação à duração do mesmo.

Escalonamento

- **O escalonador ideal**
  - É aquele que consegue deixar a CPU 100% ocupada.
  - Objetivo
    - Maximizar a produtividade
    - Minimizar o tempo de retorno, resposta e espera.
  - Não existe nenhuma política de escalonamento ótima:
    - Cumprir com todos critérios anteriores
  - A política de escalonamento conveniente depende:
    - Tipo de processo.
    - Critério de otimizarão desejado.

Escalonamento

- **Algoritmos de Escalonamento:**
  - Algumas políticas de escalonamento podem funcionar em modo **não preemptivo** ou em modo **preemptivo**.
    - Modo não preemptivo:
      - O processo que possui a CPU somente a libera quando quer (quando acaba sua execução)
      - Não necessita suporte de hardware adicional
      - Um processo pode monopolizar a CPU
      - Não são convenientes para ambientes de tempo compartilhado.
        - Exemplo: Windows 3.1 e Apple Macintosh OS

Escalonamento

- **Algoritmos de Escalonamento:**
  - Algumas políticas de escalonamento podem funcionar em modo **não preemptivo** ou em modo **preemptivo**.
    - Modo preemptivo:
      - O escalonador pode desalocar um processo da CPU em qualquer instante de tempo.
      - Maior custo, porém evita-se que um processo tenha 100% da CPU

Escalonamento

- **Algoritmos de Escalonamento:**
  - **Não Preemptivos**
    - *Fisrt-Came, Fisrt-Served* – FCFS (FIFO)
    - *Shortest-Job-First* – SJF
  - **Preemptivos**
    - Por prioridades
    - Turno rotativo ou Circular (*Round-Robin*)
    - Filas multi-nivel
    - Tempo Real

Escalonamento
Exemplo (I)

- Para os exemplos dos algoritmos de escalonamento vamos supor a existência de 3 processos com as seguintes características:

Processo	Tempo de chegada	Etapas do proceso
Processo A	0	7_{CPU}
Proceso B	2	4_{CPU}
Processo C	3	2_{CPU}

- OBS: Considere os *delays* dos tempos de chegada de cada processo.

Escalonamento

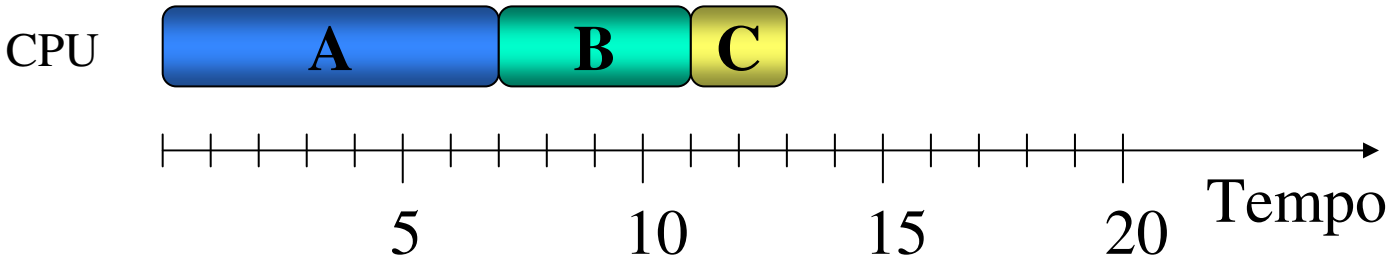
Não Preemptivos

Algoritmo FCFS

Algoritmo FCFS (*First-Come First-Served*)

- **Funcionamento:**
  - O procesador é alocado seguindo a ordem de chegada dos processos à fila de processos prontos.
  - O processo que tem a CPU não a libera até que acabe sua execução ou até que fique bloqueado por uma operação de E/S.
- **Implementação:**
  - A fila de processos prontos é implementada mediante uma fila FIFO (*First-In First-Out*).

Diagrama de Gant FCFS



$$Utilização_{cpu} = \frac{TCPU_{ocupada}}{Tempo} = \frac{13}{13} = 1 = 100\%$$

$$\text{Pr}odutividade = \frac{n^{o}\ processos}{tempo} = \frac{3}{13} = 0{,}23$$

$$TEspera_{medio} = \frac{TEspera_A + TEspera_B + TEspera_C}{n^{o}\ processos} = \frac{0 + (7-2) + (11-3)}{3} = \frac{0+5+8}{3} = 4{,}3$$

$$T\,\text{Re}torno_{medio} = \frac{T\,\text{Re}torno_A + T\,\text{Re}torno_B + T\,\text{Re}torno_C}{n^{o}\ processos} = \frac{7+11+13}{3} = 10{,}33$$

Características FCFS

- Simples de implementar
- Dependente da ordem de chegada dos processos.
- Altos tempos de espera
- Tende a favorecer aos processos com muita carga de CPU, prejudicando aos processos intensivos de E/S

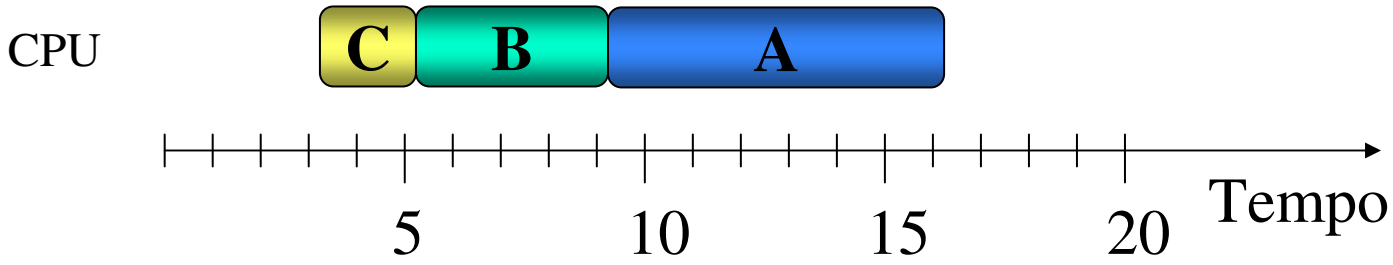
Algoritmo SJF

Algoritmo SJF (*Shortest Job First*)

- **Funcionamento:**
  - O processador é alocado ao processo com etapa de CPU mais breve.
  - Em caso de empate se aplica outro algoritmo (normalmente o FIFO).
  - Não preemptivo
    - O processo que possui a CPU somente a libera quando quando termina sua execução ou quando se bloqueia
  - Com preempção
    - Se um outro processo chegar pico de CPU menor do que o restante do processo atual, há preempção. Esse esquema é conhecido como "*Shortest Remaining Time First*" (SRTF).

- **Implementação:**
  - Ordena a fila de processos prontos em função do tempo das seguintes etapas de CPU dos processos.

Diagrama de Gant SJF



$$Util._{cpu} = \frac{TCPU_{ocupada}}{Tempo} = \frac{13}{16} = 0{,}8125 = 81{,}25\%$$

$$\text{Pr}odutividade = \frac{n^{o}\ processos}{tempo} = \frac{3}{16} = 0{,}17$$

$$TEspera_{medio} = \frac{TEspera_A + TEspera_B + TEspera_C}{n^{o}\ procesos} = \frac{(9-0)+(5-2)+(3-3)}{3} = \frac{9+3+0}{3} = 4$$

$$T\,\text{Re}torno_{medio} = \frac{T\,\text{Re}torno_A + T\,\text{Re}torno_B + T\,\text{Re}torno_C}{n^{o}\ procesos} = \frac{16+9+5}{3} = 10$$

Características SJF

- Reduz o tempo de espera médio
- Minimiza o efeito de priorizar processos do tipo *cpu-bound*
- É difícil determinar a priori qual será a duração da seguinte etapa de CPU dos processos.

Escalonamento

Preemptivos

Algoritmo Por Prioridades

Algoritmo por prioridades

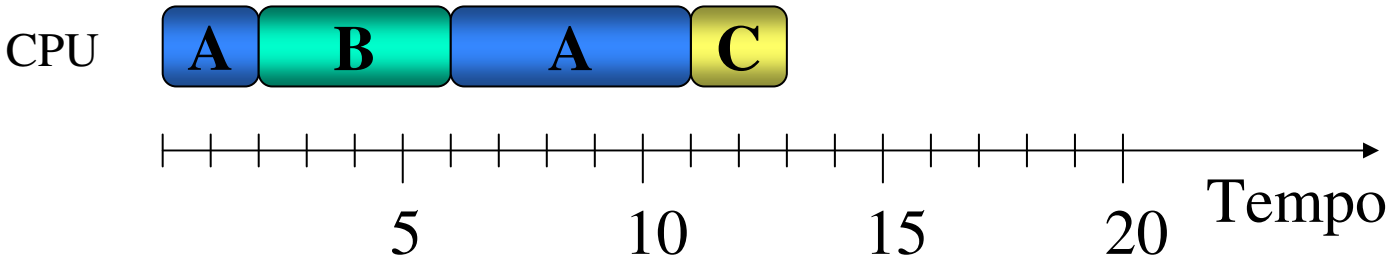
- **Funcionamento:**
  - Cada processo tem associado um valor inteiro que representa sua prioridade de execução
  - O escalonador escolhe o processo da fila de processos prontos que tenha a maior prioridade.

- **Implementação:**
  - A fila de processos prontos é ordenada pela prioridade dos processos.

- **Opcões:**
  - A Política pode ser preemptiva ou não.
  - As prioridades podem ser definidas de forma interna (pelo SO) ou de forma externa (pelo usuário).
  - Prioridades estáticas ou dinâmicas.

Algoritmo por prioridades

- Vamos supor que os processos possuem as seguintes prioridades **A=5, B=1** e **C=6** (Considerando que 1 é a prioridade mais alta e 9 a mais baixa).



$$TEspera_{medio} = \frac{TEspera_A + TEspera_B + TEspera_C}{n^{o}\ processos} = \frac{6+0+(11-3)}{3} = \frac{6+0+8}{3} = 4{,}66$$

$$T\,\mathrm{Re}torno_{medio} = \frac{T\,\mathrm{Re}torno_A + T\,\mathrm{Re}torno_B + T\,\mathrm{Re}torno_C}{n^{o}\ processos} = \frac{11+6+13}{3} = 10$$

Algoritmo *Round Robin*

Algoritmo *Round-Robin*(turno rotativo)

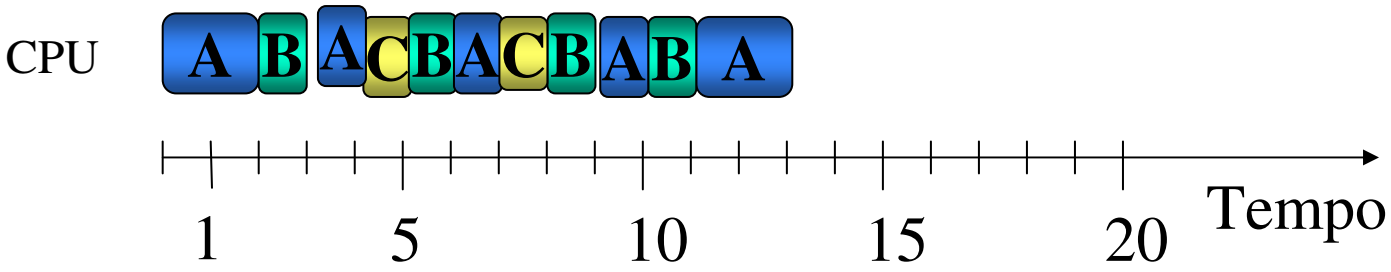
- Atribui-se a cada processo durante um intervalo de tempo um valor pré fixado de forma rotativa, denominado ***quantum***.
- Funcionamento:
  - Semelhante ao FCFS
  - Fila de prontos é uma fila FIFO circular
  - Escalonador percorre fila alocando, para cada processo, até 1 quantum
- Implementação:
  - Neste algorimo é requerido um valor temporal de troca de contexto.
- Características:
  - Permite esgotar ao máximo o tempo de resposta dos processos.
  - Algoritmo ideal para sistemas de tempo compartilhado.

Algoritmo *Round-Robin*

- Se processo não deixar a CPU dentro do quantum, é preemptado
- Se houverem n processos e o quantum for q, cada processo possui 1/n tempo de CPU, executado em porções de tempo de tamanho até q
- Nenhum processo espera mais do que (n-1)q para utilizar CPU
  - Não ocorre starvation (estagnação)
- Desempenho
  - Quantum muito grande: execução FCFS (FIFO)
  - Quantum muito pequeno: muitas trocas de contexto
    - Alto custo
  - Quantum deve ser pequeno suficiente para garantir o tempo compartilhado
  - Quantum deve ser grande bastante para compensar trocas de contexto
  - Bom desempenho: 80% dos picos de CPU devem ser menores que quantum

Diagrama de *Gant Round-Robin*

- Suponha que tenhamos um **quantum=1**



$$TEspera_{medio} = \frac{TEspera_A + TEspera_B + TEspera_C}{n^o\ processos} = \frac{6 + (7-2) + (6-3)}{3} = \frac{6+5+3}{3} = 4{,}6$$

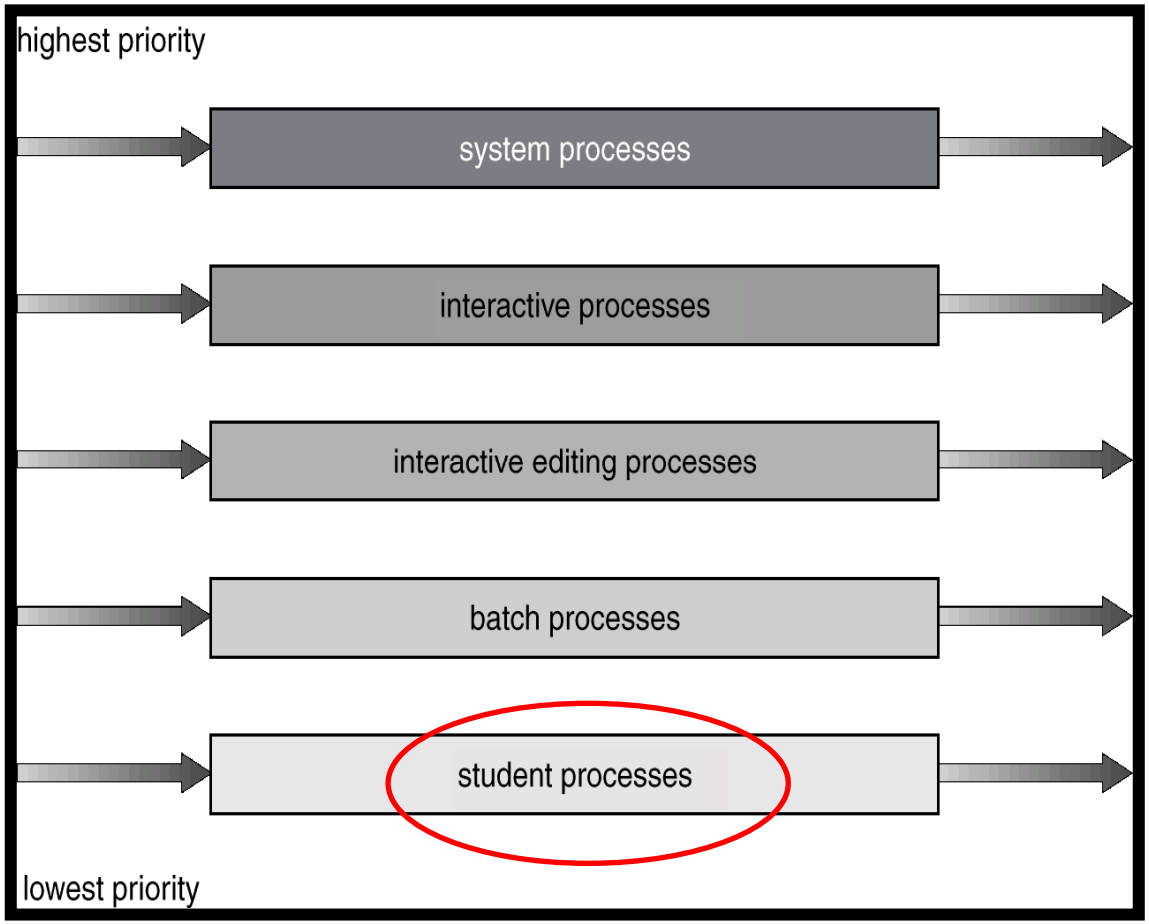
$$T\,\mathrm{Re}torno_{medio} = \frac{T\,\mathrm{Re}torno_A + T\,\mathrm{Re}torno_B + T\,\mathrm{Re}torno_C}{n^o\ processos} = \frac{13+11+8}{3} = 10{,}66$$

Filas Multiníveis

Filas Multiníveis

- Fila de prontos é dividida em várias filas
  - Ex.: 2 filas
    - Processos em primeiro plano (interativos/foreground);
    - Processos em segundo plano (background/batch);
- Cada fila possui seu próprio algoritmo de escalonamento:
  - Ex.:
    - Processos em primeiro plano: RR (para manter tempo compartilhado);
    - Processos em background: FCFS;
- É necessário haver escalonamento entre as filas:
  - Para escolher o processo de qual fila será executado;
  - Se usar algoritmo de prioridade fixa de uma fila sobre outra: starvation;
  - Outra opção: dividir o tempo de execução entre as filas:
  - Foreground fica com 80% e background com 20% do tempo de CPU.

Filas Multiníveis e Prioridade fixa



Discriminação?

Filas Multiníveis com Retroalimentação

- Sem retroalimentação: processo nunca é trocado de fila;
- Com retroalimentação: processo pode ser trocado de fila;
  - Permite separar processos com características de picos de CPU semelhantes;
    - Um processo que usa muito tempo de CPU é movido para fila de mais baixa prioridade;
    - Dessa forma: processos IO-bound e interativos (dependem da interação do usuário) ficam nas filas com mais prioridade;
    - Processos que ficam aguardando muito tempo por CPU podem ser movidos para filas de mais alta prioridade: evita starvation (estagnação)

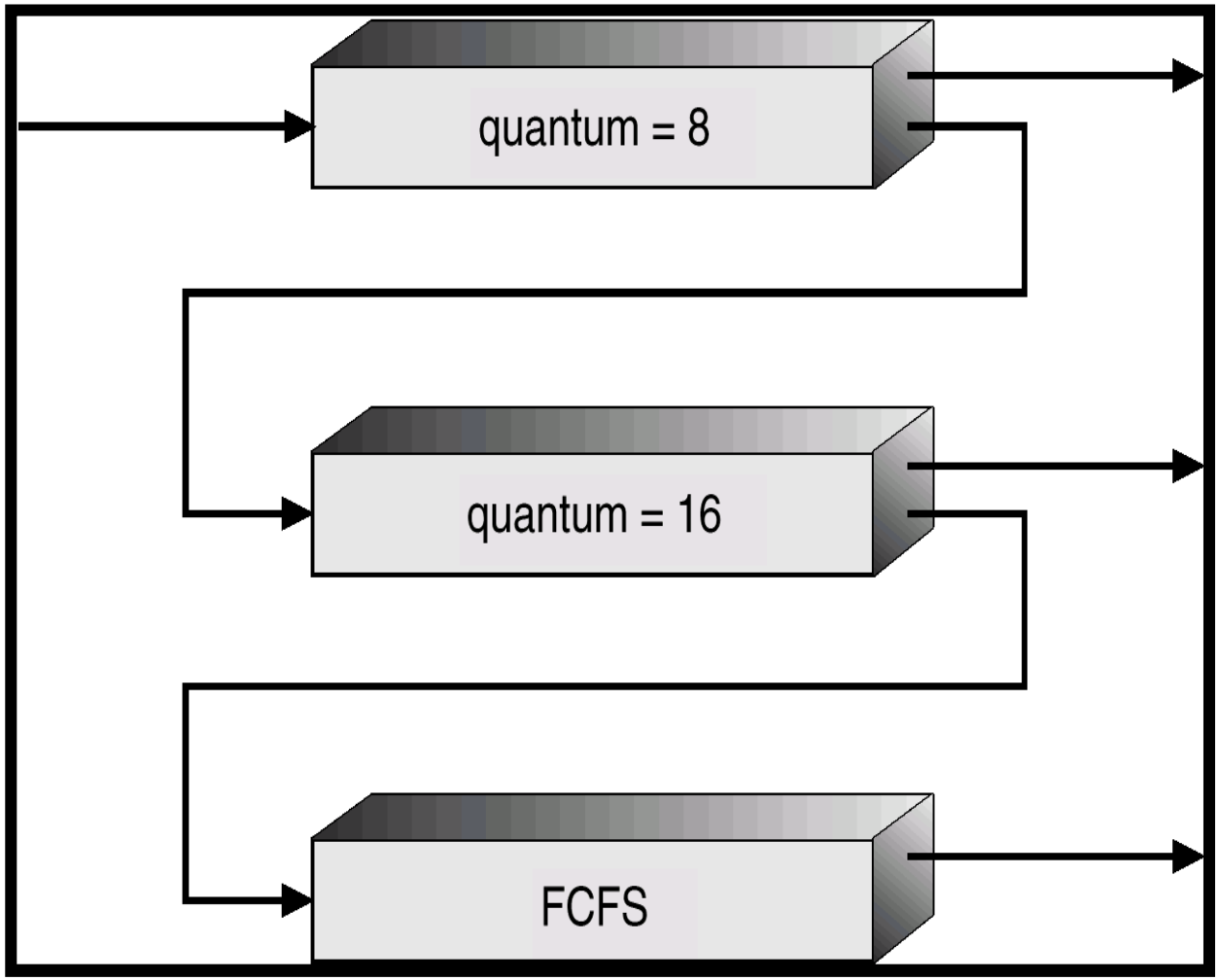
Filas Multiníveis com Retroalimentação

- Escalonador é definido pelos seguintes parâmetros:
  - número de filas;
  - algoritmos de escalonamento para cada fila;
  - método usado para determinar quando elevar um processo;
  - método usado para determinar quando rebaixar um processo;
  - método usado para determinar em que fila um processo entrará quando esse processo precisar de atendimento;

Filas Multiníveis com Retroalimentação

- EX:
- Três filas (com prioridade fixa):
  - Q0 – quantum de tempo 8 milissegundos
  - Q1 – quantum de tempo 16 milissegundos
  - Q2 – FCFS
- Escalonamento
  - Uma nova tarefa entra na fila Q0 , que é atendida com base no RR. Quando ganha a CPU, a tarefa recebe 8 milissegundos. Se não terminar nesse tempo, a tarefa é movida para a fila Q1.
  - Em Q1, a tarefa é atendida novamente com base no RR e recebe 16 milissegundos adicionais. Se ainda não estiver completa, a tarefa é apropriada e movida para a fila Q2.

Filas Multiníveis com Retroalimentação



Escalonamento em Tempo Real

Escalonamento

- Escalonador de TEMPO REAL
  - Tipos de aplicações
    - Industriais
    - Automóveis
    - Multimídia
  - Tipos de sistemas tempo real
    - Sistemas críticos (*Hard Real-Time*)
    - Sistemas não críticos (*Soft Real-Time*)

Escalonamento

- Escalonador de TEMPO REAL
  - Sistemas críticos (*Hard Real-Time*)
    - É necessário garantir que a(s) tarefa(s) consideradas críticas terminem antes de um determinado tempo (*deadline*), caso contrário o seu não cumprimento pode resultar em graves danos para o sistema.
    - Exemplos:
      - Aplicações aeroespaciais
      - ABS de um carro
      - Sistema de automação

Escalonamento

- Escalonador de TEMPO REAL
  - Sistemas não críticos (*Soft Real-Time*)
    - O funcionamento do sistema é apenas ligeiramente afetado caso não seja possível cumprir um determinado *deadline*.
    - Exemplos:
      - Aplicações multimídia
      - Jogos de computador

Escalonamento

- Escalonador de TEMPO REAL
  - Os métodos de escalonamento devem garantir à priori que o sistema cumpre as suas metas temporais
    - É normalmente necessário conhecer o tempo de execução das tarefas
      - Periódicas (por ex. para aquisição de dados)
      - Esporádicas (por ex. para o tratamento de alarmes)
    - O sistema é pouco dinâmico
      - Escalonamento por prioridades
      - Escalonamento utilizando o algoritmo *Earliest Deadline First* (EDF)

Escalonamento em Tempo Real

- Sistemas de tempo real rígido: tarefas devem ser completadas dentro do tempo exigido
  - Praticamente impossível em um sistema de propósito geral com memória secundária ou memória virtual
- Sistemas de tempo real flexível (maleável): tarefas críticas têm maior prioridade sobre as demais
  - Permite sistemas gráficos/mutimídia executarem de forma adequada (em detrimento de outros processos): aceitável
  - Latência de despacho deve ser pequena: diminui o tempo para um processo tempo real pronto entrar em execução
    - Problema: SOs não realizam preempção no meio de chamada de sistema ao bloco de E/S

Escalonamento em Tempo Real

- Para reduzir tempo de despacho
  - Algumas chamadas ao sistema (demoradas) tornam-se interceptáveis
    - Possuem pontos de interceptacao: locais que verificam se há processos de tempo real para serem executados
    - Chamadas ao sistema podem ser interrompidas nesses pontos e retomadas futuramente (após execução do processo em tempo real)
    - Pontos de interceptação devem estar em “áreas seguras” do kernel
      - Não pode haver possibilidade de inconsistências

Escalonamento em Tempo Real

- Inversão de prioridade
  - Processo de alta prioridade necessita aguardar recursos que estão sendo atualizados por processo de baixa prioridade: ineficiência
  - Protocolo de herança de prioridade: processos que estão acessando recursos necessários a um processo de alta prioridade herdam essa prioridade até a liberação do recurso
    - Melhor desempenho
    - Prioridades voltam ao original após liberar recurso

Escalonamento em Múltiplos Processadores

- Escalonamento da CPU é muito mais complexo quando várias CPUs estão disponíveis
- Processadores idêndicos: simplificação
- Compartilhamento de carga: várias CPUs cooperando
  - Idéia 1: usar fila de prontos separada para cada processador
    - Enquanto uma CPU fica sobrecarregada outra pode ficar ociosa
  - Idéia 2: usar fila de prontos comum
    - Faz um melhor balanceamento da carga
    - Problemas: CPUs diferentes podem escolher mesmo processo, pode haver outras inconsistências devido ao paralelismo
- Multiprocessamento assimétrico (mestre/escravo): mais simples
  - Só 1 CPU acessa as estruturas de dados do SO (mestre), demais executam

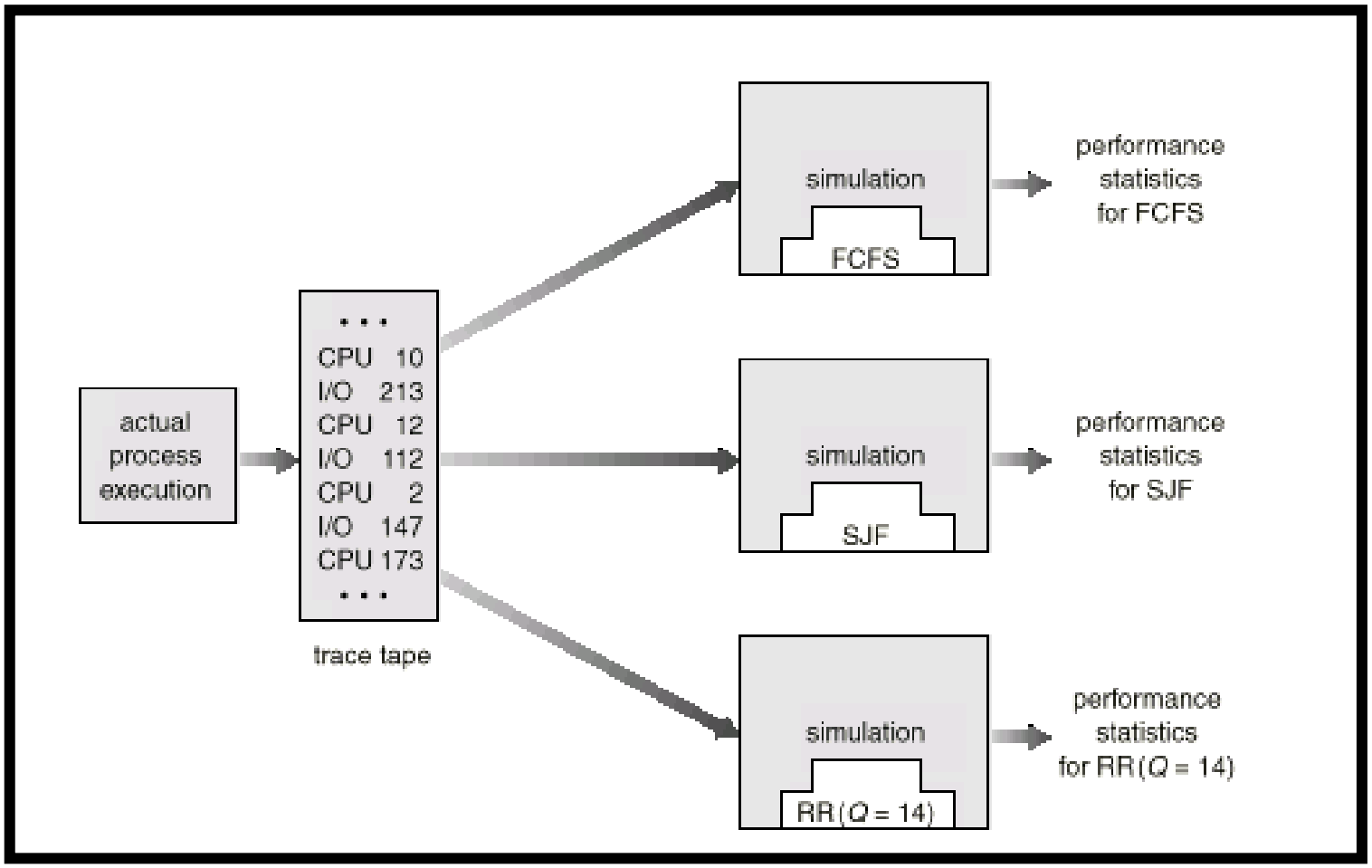
Avaliação de Algoritmos

- Modelagem determinística
  - Considera uma carga de trabalho particular (pré-determinada)
  - Define (calcula) o desempenho de cada algoritmo para a carga
    - Utilização de CPU,
    - Throughput,
    - Tempo de espera,
    - Tempo de turnarorund,
    - etc.

Avaliação de Algoritmos

- Avaliação por simulação
  - Método mais preciso
  - Utiliza um modelo de sistema de computação
  - Informações (processos, picos de CPU, chegadas, E/S, términos, etc.) podem ser geradas aleatoriamente
    - De acordo com distribuições probabilísticas
  - Resultados são usados para verificar o que ocorre na realidade e adota-se a distribuição adequada
- Avaliação por implementação
  - Mais realista
  - Alto custo: necessário implementar no kernel e testar sob as diversas situações reais

Avaliação por Simulação



Exercício Proposto em Sala

Utilize os Algoritmos de Escalonamento:

a) FCFS

b) SJF(não preemptivo)

c) Prioridades (A=1, B=2, C=3)

d) RR (quantum = 1)

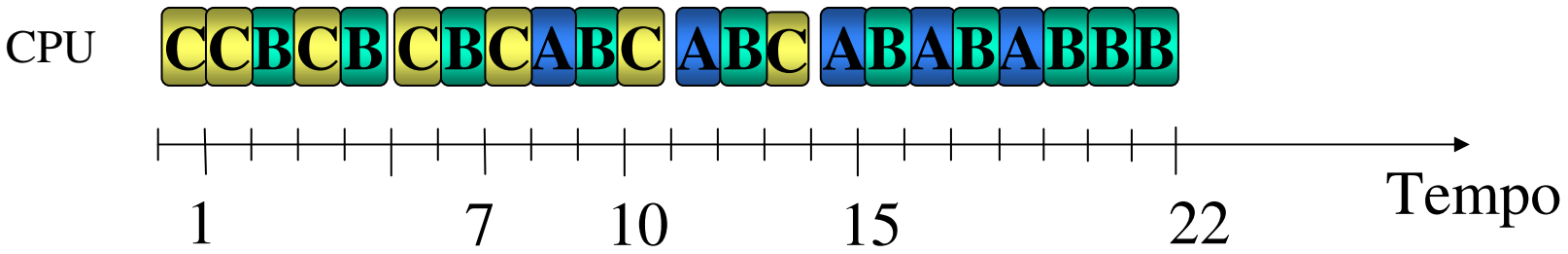
- Calcular os tempos de espera e retorno (*Turnaround*).

Proceso	Tempo de chegada	Consumo da CPU
Proceso A	7	5_{CPU}
Proceso B	3	10_{CPU}
Proceso C	1	7_{CPU}

- OBS: Considere os *delays* dos tempos de chegada de cada processo.

Exercício

- D) RR quantum=1



$$TEspera_{medio} = \frac{TEspera_A + TEspera_B + TEspera_C}{n^o\ processos} = \frac{(14-7)+(12-3)+(7-1)}{3} = \frac{7+9+6}{3} = 7{,}33$$

$$T\,\mathrm{Re}torno_{medio} = \frac{T\,\mathrm{Re}torno_A + T\,\mathrm{Re}torno_B + T\,\mathrm{Re}torno_C}{n^o\ processos} = \frac{19+14+22}{3} = 18{,}33$$