INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA
Campus Salvador

Escalonamento

AULA 5
Flávia Maristela (flavia@flaviamaristela.com)
Romildo Martins (romildo@romildo.net)

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA
Campus Salvador

Retrospectiva da aula passada...

Na aula passada...

- *Race condition*
- Região Crítica
- Exclusão Mútua
- Problemas Clássicos

Comunicação entre processos (-- O jantar dos filósofos --)

Problemas Clássicos
(-- O jantar dos filósofos --)

- Quais os problemas que surgem com a questão do jantar dos filósofos?

Jantar dos filósofos
(-- 1ª solução --)

```
#define N 5

void philosopher (int i)
{
   while (TRUE)
   {
      think();
      take_fork (i);
      take_fork ((i+1) % N);
      eat();
      put_fork (i);
      put_fork ((i+1) % N);
   }
}
```

- O que acontece se todos os filósofos pegam o garfo da esquerda simultaneamente?
  - Nenhum filósofo consegue pegar o garfo da direita
  - ***DEADLOCK***

Jantar dos filósofos
(-- 2ª solução --)

```
#define N 5

void philosopher (int i)
{
   while (TRUE)
   {
      think();
      take_fork (i);
      if (fork((i+1) % N) is available)
      {
         take_fork ((i+1) % N);
         eat();
         put_fork (i);
         put_fork ((i+1) % N);
       }
      else
        put_fork (i);
   }
}
```

- O que acontece se todos os filósofos pegam o garfo da esquerda simultaneamente?
  - ***INANIÇÃO (starvation)***

Jantar dos filósofos
(-- 3ª solução --)

```
#define N 5

void philosopher (int i)
{
   while (TRUE)
   {
      think();
      down(mutex);
      take_fork (i);
      take_fork ((i+1) % N);
      eat();
      put_fork (i);
      put_fork ((i+1) % N);
      up(mutex);
   }
}
```

- O que acontece nesta solução?
  - Apenas um filósofo come por vez
  - Afeta o ***PARALELISMO***

Jantar dos filósofos
(-- 4ª solução --)

- Atribui 3 possíveis estados aos filósofos
  - PENSANDO
  - COMENDO
  - FAMINTO

- Idéia:
  - Um filósofo no estado “faminto” só pode pegar os garfos se os seus vizinhos (esquerda e direita) não estiverem “comendo”.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA
Campus Salvador

Deadlock
(-- Impasse --)

DEADLOCK
“Alguém tem que ceder”

Deadlock

- Problema de programação concorrente

- “Um conjunto de n processos está em *deadlock* quando cada um dos n processos está bloqueado a espera de um evento que somente pode ser causado por cada um dos n processos.”

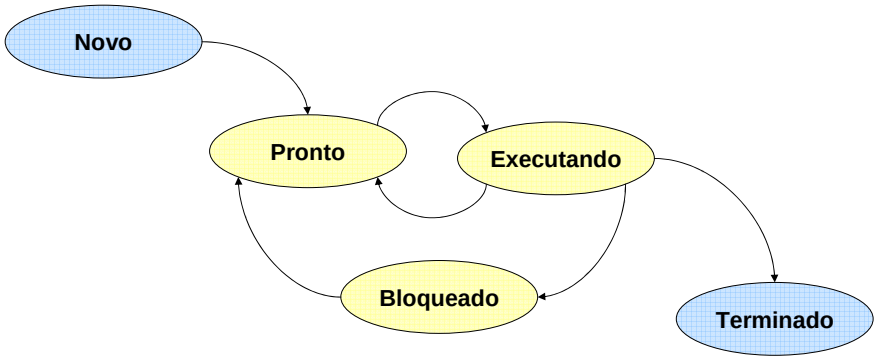
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA
Campus Salvador

Escalonamento de Processos

Porque é necessário escalonar?

- Processos precisam ser executados
- Processos concorrem a CPU
- Escalonador:
  - Componente (implementação) do sistema operacional
  - Determina a ordem de execução dos processos baseado num *algoritmo de escalonamento*
  - Lê a fila que contém os processos no estado “pronto” e os ordena para execução

O que provoca o escalonamento?



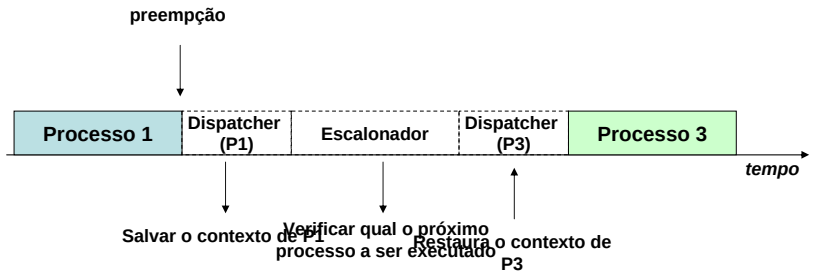
Tipos de algoritmo de escalonamento

- Preemptivo:
  - Execução de um processo dura um tempo pré-determinado
  - Quando o tempo acaba, o processo é interrompido.
- Não-preemptivo:
  - Processo fica em execução até que:
    - Termine
    - Libere a CPU VOLUNTARIAMENTE
    - Seja bloqueado por falta de recurso

O que afeta a performance de um algoritmo de escalonamento?

- Cada processo possui informações que permitem definir precisamente seu estado.
  - Tais informações definem o ***contexto*** do processo
- Troca de Contexto
  - Mecanismo que permite ao escalonador interromper uma tarefa, e executá-la posteriormente, sem corromper seu estado.
  - Separação do escalonamento
    - Escalonamento = Política + Mecanismo

Ilustração da troca de contexto



Qual o objetivo do escalonamento?

- DEPENDE do ***tipo*** de sistema operacional
  - Lote:
    - Não possui usuários aguardando ➔ pode ser preemptivo ou não
    - Não possui muita troca de contexto
    - OBJETIVOS:
      - melhorar o *throughput* (vazão)
      - melhorar o *turnaround* (tempo entre submissão e finalização)
      - manter a CPU ocupada

Qual o objetivo do escalonamento?

- Propósito Geral:
  - Possuem usuários interagindo
  - Precisam ser preemptivos
  - OBJETIVOS
    - melhorar o tempo médio de resposta
    - atender as expectativas dos usuários
- Tempo real:
  - Em geral são preemptivos
  - OBJETIVO:
    - cumprir requisitos lógicos
    - cumprir requisitos temporais

Qual o objetivo do escalonamento?

- Independente do *tipo* de sistema operacional, TODOS os algoritmos de escalonamento precisam atender a alguns critérios:
  - Justiça (fairness)
  - Aplicação da política de escalonamento
  - Equilíbrio (balance) entre as partes do sistema

Escalonamento para sistemas em lote

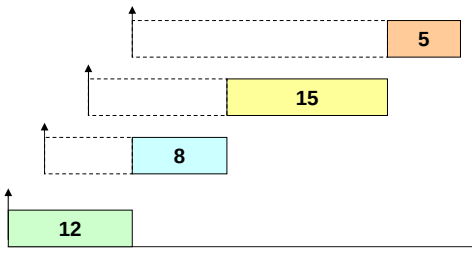
- FCFS (ou FIFO)
  - Primeiro processo da fila de pronto é o escolhido para executar.
  - Não-preemptivo
  - Fácil de entender
  - Fácil de programar
  - "Justo"
  - Processos de baixo custo de execução podem esperar muito tempo para ser executado

Escalonamento para sistemas em lote

- FCFS (ou FIFO)
  - Fazer o escalonamento para os seguintes processos:

Processo	Custo de execução	Instante de chegada
A	12	t = 0
B	8	t = 3
C	15	t = 5
D	5	t = 10

FCFS



Processo	Custo de execução	Instante de chegada
A	**12**	t = 0
B	**8**	t = 3
C	**15**	t = 5
D	**5**	t = 10

Escalonamento para sistemas em lote

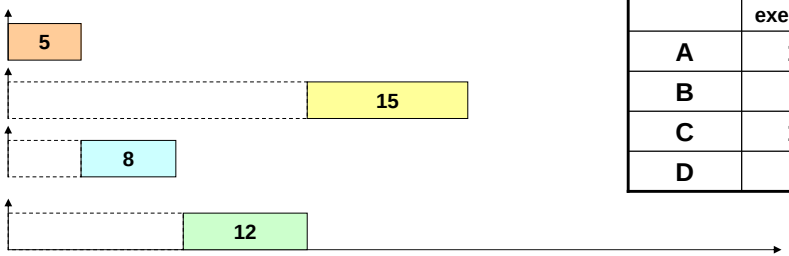
- Menor Job Primeiro
  - O *job* de menor custo de execução executa primeiro.
  - Não-preemptivo
  - Fácil de entender
  - Fácil de programar
  - "Justo"
  - Para ser adequado, requer que todos os jobs estejam disponíveis simultaneamente

Escalonamento para sistemas em lote

- Menor Job Primeiro
  - Fazer o escalonamento para os seguintes processos

Processo	Custo de execução
A	12
B	8
C	15
D	5

SJF – *Shortest Job First*



Processo	Custo de execução
A	12
B	8
C	15
D	5

Escalonamento em sistemas de propósito geral

- Alternância circular (*Round-Robin*)
  - Processos executam dentro de uma fatia de tempo predefinida (***quantum***)
  - Preemptivo
  - Simples
  - Justo
  - Amplamente utilizado
  - Tamanho do *quantum* pode ser um problema

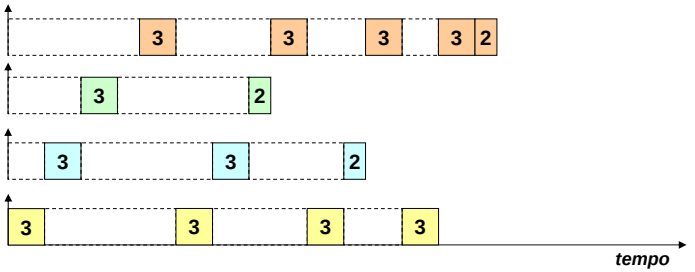
Escalonamento em sistemas de propósito geral

- *Round-Robin*
  - Fazer o escalonamento para os seguintes processos considerando um *quantum* = 3

Processo	Custo de execução	Instante de Chegada
A	12	t = 0
B	8	t = 0
C	15	t = 0
D	5	t = 0

Round-Robin

Processo	Custo de execução	Instante de chegada
A	**12**	t = 0
B	**8**	t = 0
C	**15**	t = 0
D	**5**	t = 0



Round-Robin

- Fazer o escalonamento *Round-Robin* para o seguinte conjunto de tarefas

Processo	Custo de execução	Prioridade	Instante de Chegada
A	12	3	t = 0
B	8	4	t = 0
C	15	2	t = 0
D	5	1	t = 0

Podemos implementar o algoritmo de *Round-Robin* usando prioridade?

Round-Robin

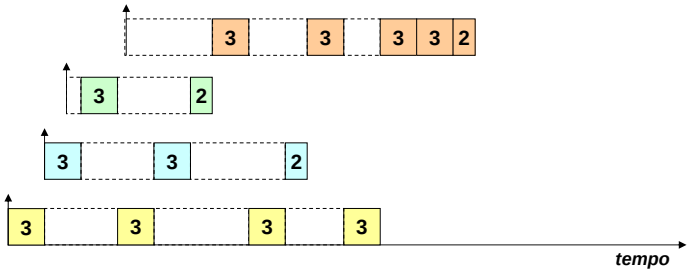
- Fazer o escalonamento *Round-Robin* para o seguinte conjunto de tarefas

Processo	Custo de execução	Instante de Chegada
A	12	t = 0
B	8	t = 3
C	15	t = 5
D	5	t = 10

Round-Robin

Processo	Custo de execução	Instante de chegada
A	**12**	t = 0
B	**8**	t = 3
C	**15**	t = 5
D	**5**	t = 10

Qual a diferença quando os processos são ativados em instantes diferentes?



Escalonamento em sistemas de propósito geral

- Prioridade
  - Processos tem diferentes prioridade de execução
  - Preemptivo
  - Baseado nos ciclos da CPU ou *quantum*
  - Prioridade pode ser atribuída estaticamente ou dinamicamente
  - Pode ser implementado considerando filas de prioridades
  - A implementação de filas pode representar um problema!

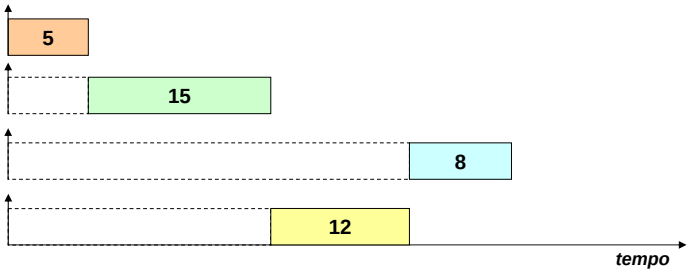
Escalonamento em sistemas de propósito geral

- Prioridade
  - Fazer o escalonamento para os seguintes processos

Processo	Custo de execução	Instante de Chegada	Prioridade
A	12	t = 0	3
B	8	t = 0	4
C	15	t = 0	2
D	5	t = 0	1

Prioridade

Processo	Custo de execução	Instante de Chegada	Prioridade
A	12	**t = 0**	3
B	8	**t = 0**	4
C	15	**t = 0**	2
D	5	**t = 0**	1



Escalonamento em sistemas de propósito geral

- Prioridade
  - Fazer o escalonamento para os seguintes processos

Processo	Custo de execução	Instante de Chegada	Prioridade
A	12	t = 0	3
B	8	t = 3	4
C	15	t = 5	2
D	5	t = 10	1

Escalonamento em sistemas de propósito geral

- Filas Múltiplas
  - Processos executam dentro de uma fatia de tempo predefinida (***quantum***)
  - Preemptivo
  - Justo
  - Tamanho do *quantum* variável ➔ trocas de contexto.
  - Adaptável para diferentes tamanhos de processo
  - Os processos são promovidos a medida que o tempo passa

Para a próxima aula

- Trazer todos os exercícios dos slides respondidos.
- Verificar as implementações de semáforo para o problema do produtor consumidor.
- Escalonamento com múltiplas filas.
- Descrever a diferença entre processos I/O- Bound e CPU-Bound.