# *Curso de Engenharia de Computação*
# *Sistemas Operacionais*

INSTITUTO MAUÁ DE TECNOLOGIA
MAUÁ

## Processos e Threads – Parte IV

## *Agendamento de processos e threads*

Slides da disciplina Sistemas Operacionais
Curso de Engenharia de Computação
Instituto Mauá de Tecnologia – Escola de Engenharia Mauá
Prof. Marco Antonio Furlan de Souza

# *Agendamento*

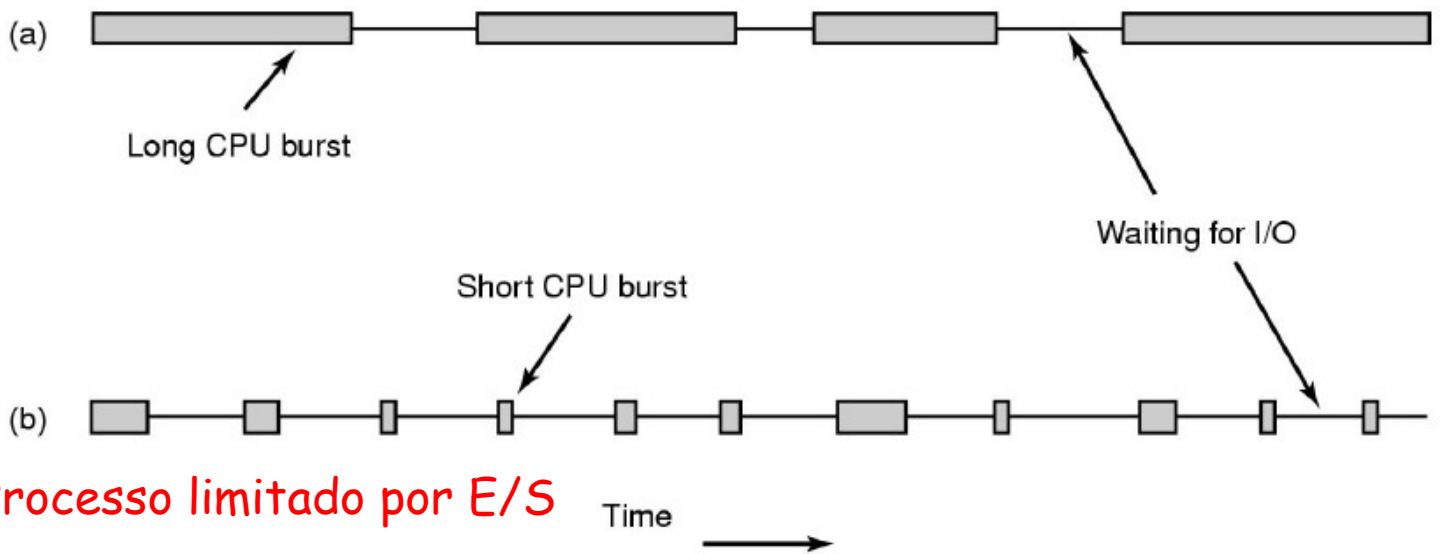
INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - **Para que agendar um processo?**
    - Porque sistemas operacionais precisam **executar diversos processos**. O **agendador** escolhe o **próximo processo** a ser executado, de **acordo** com **alguma política**.
  - **Agendador**
    - É o **componente** do **sistema operacional** (também um processo!) que se encarrega de **escolher** o próximo **processo** a ser **executado**.
  - **Algoritmo de agendamento**
    - É a **algoritmo implementado** no **agendador** que implementa a **política** de **agendamento**. **Depende** das **necessidades** do **sistema operacional** (**lote**, **interativo** ou **tempo real**).

# Agendamento

- **Conceitos**
  - **Comportamento dos processos**
    - Praticamente todos os processos alternam picos de computação com requisições de E/S:

Como operações de E/S são muito mais lentas que na CPU, o agendador deve sempre que possível escolher o processo limitado por E/S para manter os dispositivos em operação



# *Agendamento*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - **Metas dos algoritmos de agendamento**
    - **Para todos os sistemas**
      - **Equidade** (*fairness*): dar a todos os processos uma fatia justa de tempo da CPU;
      - **Aplicação de política**: garantir que a política definida está de fato sendo aplicada;
      - **Equilíbrio**: manter todas as partes do sistema ocupadas.
    - **Sistemas em lote (*batch*)**
      - **Taxa de execução**: maximizar a execução de *jobs* por hora;
      - **Tempo de virada** (*turnaround time*): minimizar o tempo entre a submissão de um *job* e sua terminação;
      - **Utilização de CPU**: manter a CPU ocupada o tempo todo.

# *Agendamento*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - **Metas dos algoritmos de agendamento**
    - **Sistemas interativos**
      - **Tempo de resposta**: responder às requisições rapidamente;
      - **Proporcionalidade**: atender às expectativas de execução dos usuários.
    - **Sistemas em tempo real**
      - **Atender a prazos de computação**: evitar perdas de dados por não conseguir atendar às restrições de tempo;
      - **Previsibilidade**: evitar degradação da qualidade em alguns tipos de computação (exemplo: processamento de multimídia).

# *Agendamento*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Algoritmos de agendamento**
  - **Sistemas em lote**
    - Primeiro a chegar, Primeiro a ser servido
    - Primeiro o *job* mais curto
    - Tempo restante mais curto
  - **Sistemas interativos**
    - Agendamento *Round-Robin*
    - Agendamento com prioridade
    - Filas múltiplas
    - O próximo é o processo mais curto
    - Agendamento garantido
    - Agendamento por loteria
    - Agendamento por fatia justa

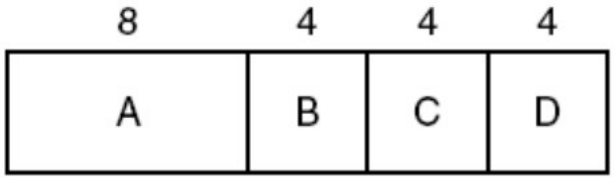
# *Agendamento*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Agendamento em sistemas em lote**
  - **Primeiro a chegar, Primeiro a ser servido**
    - Os **processos** são **atribuídos** à CPU na **ordem** que eles a **requisitam**;
    - Um **fila única** de **processos** é **utilizada**;
    - Quando um **processo** está em **execução** na **CPU**, os **demais aguardam** na **fila**;
    - Se um **processo** em **execução** é **interrompido** por **operações** de **E/S**, o **próximo** da fila é **escolhido** para executar e quando o processo interrompido se torna ativo novamente, é **transferido** para o **fim** da **fila**;
    - Quando um **processo termina** sua execução na CPU, **o processo** que está no **início** da **fila** é **selecionado** para executar na CPU;
    - **Vantagem**:  **algoritmo** de **simples** entendimento e implementação.
    - **Desvantagem**: o processos limitados por E/S são prejudicados em desempenho em relação aos processos limitados por CPU.

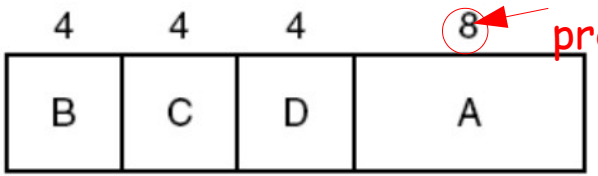
# Agendamento

- **Agendamento em sistemas em lote**
  - **Primeiro o *job* mais curto**
    - O agendador escolhe o *job* mais curto (em tempo) de uma fila de processos a executar:




Ordem original

Neste caso, tem-se os seguintes tempos de virada: A (8), B (8+4=12), C (8+4+4=16) e D (8+4+4 +4 = 20). A média de tempo de virada é (8+12+16+20)/4 = 14




Tempo de processamento

Ordem do job mais curto

Neste caso, tem-se os seguintes tempos de virada: B (4), C (4+4=8), D (4+4+4=12) e A (4+4+4+8=20)
A média de tempo de virada é (4+8+12+20)/4 = 11. Portanto, neste caso é o **melhor**. Este **algoritmo é ótimo** quando **todos** os **processos** estão **disponíveis simultaneamente**.

# *Agendamento*

INSTITUTO MAUÁ DE TECNOLOGIA

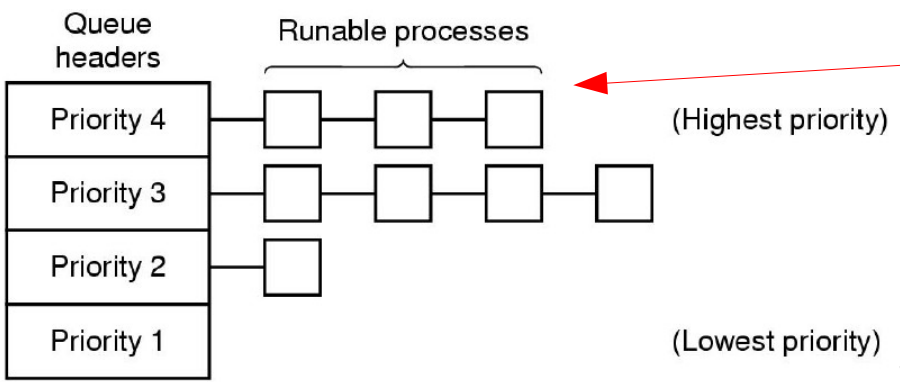
- **Agendamento em sistemas em lote**
  - **Tempo restante mais curto**
    - É uma versão **preemptiva do** algoritmo **primeiro o *job* mais curto**;
    - Se um **novo processo** está **disponível** e possui um **tempo execução menor** do **que** o **tempo restante** de **execução** de um **processo** que está atualmente **em execução**, o **processo** em **execução** é **suspenso** e este **novo processo** é **escolhido** para **executar**;
    - Assim novos *jobs* tem a chance de ser rapidamente executados.

# Agendamento

- **Agendamento em sistemas interativos**
  - **Agendamento *Round-Robin***
    - Para cada **processo** é **atribuído** um **intervalo** de **tempo** – ***quantum*** – no qual o **processo pode executar**;
    - **Se** o **processo** ainda está **executando** no **final** do *quantum*, a CPU se **antecipa** e **executa outro processo**, **passando** o **processo anterior** no **final** da **lista** de **processos executáveis**;
    - **Se** o **processo bloqueia** ou **termina** antes do *quantum* terminar, a **comutação** da CPU **termina** para esse **processo**;
    - Sobre a **escolha** do **quantum**: **muito pequeno**, **causa comutação excessiva** em processos e **reduz** a **eficiência** da **CPU**; **muito** grande, **causa baixa resposta** para **requisições interativas curtas**. Valores típicos: 20-50ms.

# Agendamento

- **Agendamento em sistemas interativos**
  - **Agendamento com prioridade**
    - Diferentemente do agendamento Round-Robin, **adiciona-se aqui** uma **prioridade** a **cada processo executável** e o **processo** com **maior prioridade** é **levado** à **execução** (na realidade podem haver vários processos com mesma prioridade);
    - Pra **prevenir** que um processo de **alta prioridade** execute **indefinitivamente**, o **agendador pode reduzir** sua **prioridade**, causando a **comutação** para outro processo, se sua prioridade for maior;
    - **Prioridades** podem ser **estáticas** ou **dinâmicas**.




Pode-se aplicar Round-Robin com processos de mesma prioridade

# *Agendamento*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Agendamento em sistemas interativos**
  - **Filas múltiplas**
    - Utiliza **filas múltiplas** de **processos** que representam "**categorias**" de **processos**, organizados de **acordo** com "**quanta**" que o agendador atribui;
    - Por exemplo, na primeira fila estão processos que podem gastar no máximo 1 quantum; na segunda, 2 quanta; na terceira, 4 quanta; e assim por diante;
    - Um **processo** que **necessite** de **100 quanta** para executar começaria na primeira fila, após comutação depois de gastar 1 quantum, executaria na segunda fila até consumir 2 quanta e, assim por diante;
    - Então o número de filas que este processo entra resulta na quantidade de tempo necessário para sua execução: 1 + 2 + 4 + 8 + 16 + 32 + 37 (o processo precisa de 100 quanta, então não precisa consumir 64!).
    - Corresponde, portanto à 7 comutações – é um método que tende a reduzir o número de comutações.

# *Agendamento*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Agendamento em sistemas interativos**
  - **O próximo é o processo mais curto**
    - **Similar** ao **algoritmo** do *job* mais curto;
    - No entanto, em sistemas interativos, utilizam-se técnicas para ajustar o tempo de execução do processo para ajustar o agendador;
    - **Neste caso**, faz-se uma **estimativa baseada** no **passado** do tempo de execução dos processos;
    - Supor que o **tempo medido** de um **processo** é $T_0$ e que **após sua próxima execução resultou** em $T_1$. Pode-se definir uma estimativa para uma próxima execução como $aT_0 + (1-a)T_1$, $(0<=a<=1)$;
    - No mais, escolhe-se para execução o processo que possuir menor estimativa de tempo.

# *Agendamento*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Agendamento em sistemas interativos**
  - **Agendamento garantido**
    - **Promessa**: em um sistema com *n* **processos** em execução, cada **processo** deveria receber ***1/n*** do tempo da **CPU**;
    - Para **manter** esta **promessa**, o sistema deve manter **acompanhar quanto** da **CPU** cada **processo** teve **desde** sua **criação**;
    - **Computa-se** então a **quantidade** de **CPU** que cada **processo tem direito** – o **tempo desde** sua **criação dividido** por ***n***;
    - Já que a **quantidade** de **tempo** de **CPU** que cada **processo** já **consumiu** é **conhecido**, pode-se **calcular** a **relação** do **tempo atual** de **CPU consumida** para o **tempo** de **CPU** de **direito** ao **processo**;
    - Uma **relação de 0.5 significa** que ele **somente teve metade** do que **deveria ter** e uma **relação** de **2.0** significa que um **processo teve duas vezes mais** tempo do que ele teria direito;
    - O algoritmo **escolhe** para **executar** o **processo** que **possui** o valor **mais baixo** desta **relação**.

# *Agendamento*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Agendamento em sistemas interativos**
  - **Agendamento por loteria**
    - É uma **forma similar**, porém mais **simples**, de **agendamento garantido**;
    - A **ideia** é **distribuir** aos processos "**tickets** de **loteria**" que representam diversos **recursos** do **sistema**, como o tempo de CPU;
    - Sempre que uma **decisão** de **agendamento** tem que ser feita, um **ticket** de **loteria** é **escolhido aleatoriamente** e o **processo** que tem este **ticket recebe** o **recurso**;
    - Por **exemplo**, aplicado ao agendamento de CPU, o **sistema** poderia **realizar** um **sorteio 50 vezes** por **segundo** e **sortear** um ganhador com **20ms** de tempo de CPU como prêmio.

# Agendamento

- **Agendamento em sistemas interativos**
  - **Agendamento por fatia justa**
    - Trata-se de uma **técnica** que tenta **balancear** o **agendamento** existente (por exemplo, Round-Robin) de **acordo** com alguma **promessa** ou de algum **direito** de execução;
    - Por **exemplo**, considerar um **sistema** com **dois usuários**, cada um com a **promessa** de utilizar **50% da CPU**;
    - O **usuário 1** tem **quatro** processos. A, B, C, and D, e o **usuário 2** tem **somente um** processo, E. Um **agendamento justo** com **Round-Robin** poderia **ser**: A-E-B-E-C-E-D-E-A-E-B-E-C-E-D-E …;
    - Por outro lado, **se** o **usuário 1** tem **direito** a **duas vezes mais CPU** que o **usuário 2**, pode-se ter **este agendamento**: A-B-E-C-D-E-A-B-E-C-D-E …
    - Existem muitos outros esquemas que se pode utilizar.

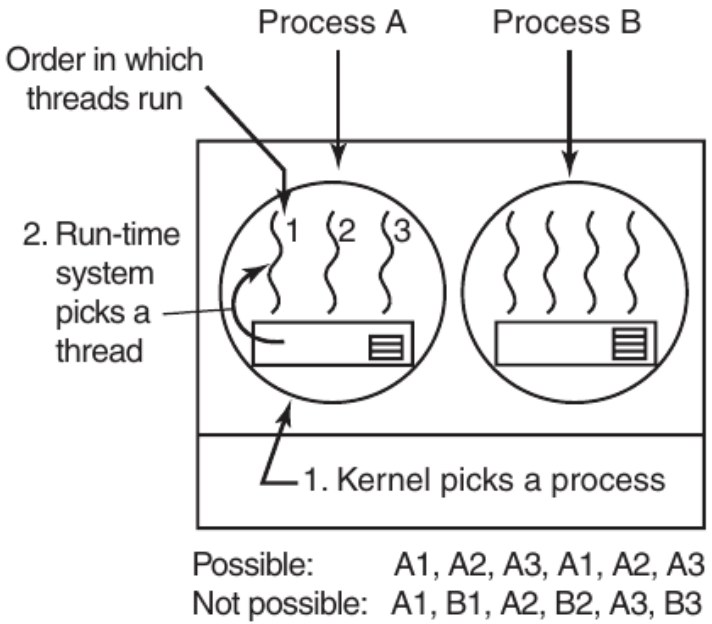
# *Agendamento*

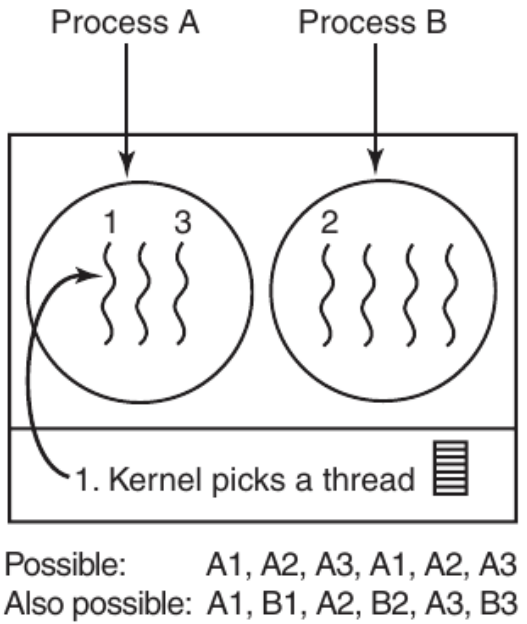
INSTITUTO MAUÁ DE TECNOLOGIA

- **Agendamento de threads**
  - O agendamento de threads depende do tipo de thread que se está considerando – threads de usuário e threads de kernel;
  - Nas t**hreads de usuário**, o kernel desconhece essas threads e agenda os processos como já visto. Internamente, o agendamento das threads podem utilizar qualquer um dos métodos apresentados – depende de uma biblioteca de threads;
  - Nas **threads de kernel**, o kernel seleciona uma thread de um dos processos para executar, independentemente se o processo desta thread está em execução ou não. Então o agendamento das threads é independente do agendamento dos processos.

# Agendamento

- **Agendamento de threads**
  - **Visão geral**



# *Referências bibliográficas*

TANENBAUM, Andrew S. **Sistemas operacionais modernos**. 3. ed. São Paulo: Pearson, 2013. 653 p.