Universidade Federal
de Mato Grosso
Campus Rondonópolis

Sistemas Operacionais

Aula 3 – Processos (1)

Prof. Msc. Cleyton Slaviero

cslaviero@gmail.com

O que é um processo?

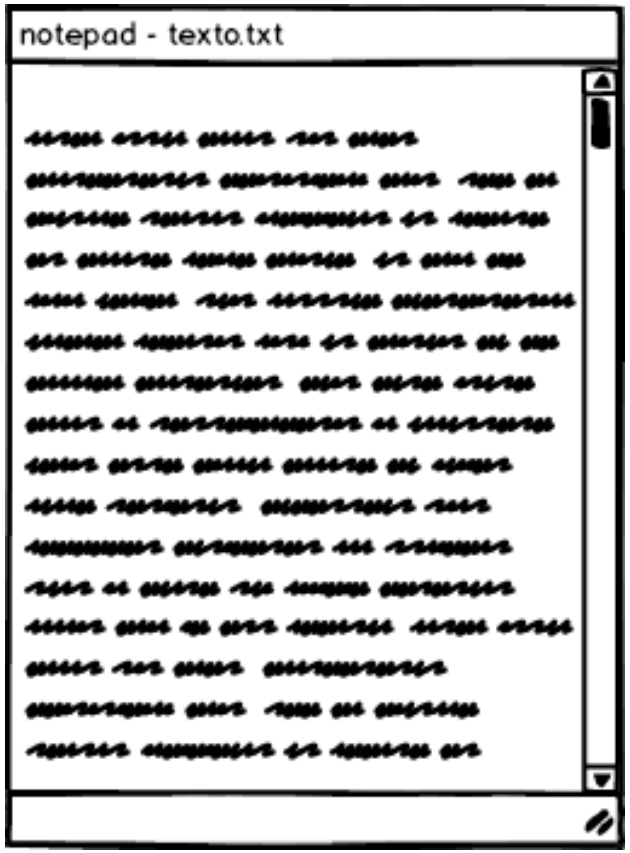
- Um SO gerencia hardware em favor dos aplicativos

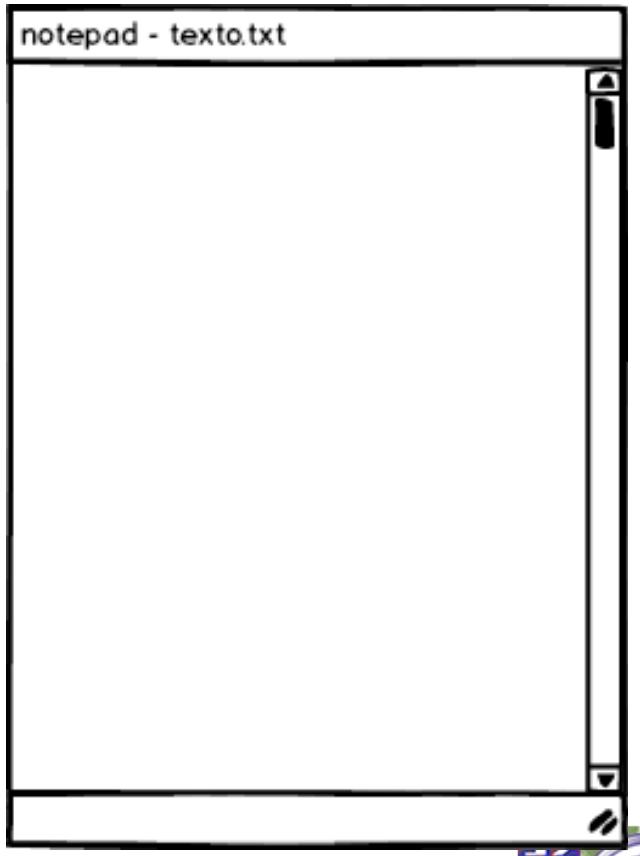
- Aplicativos
  - Programa armazenado
  - Entidade estática

- Processo
  - Estado de um programa enquanto executa
  - Entidade ativa
  - Um mesmo programa pode ter vários processos
    -

Universidade Federal
de Mato Grosso
Campus Rondonópolis

Exemplo





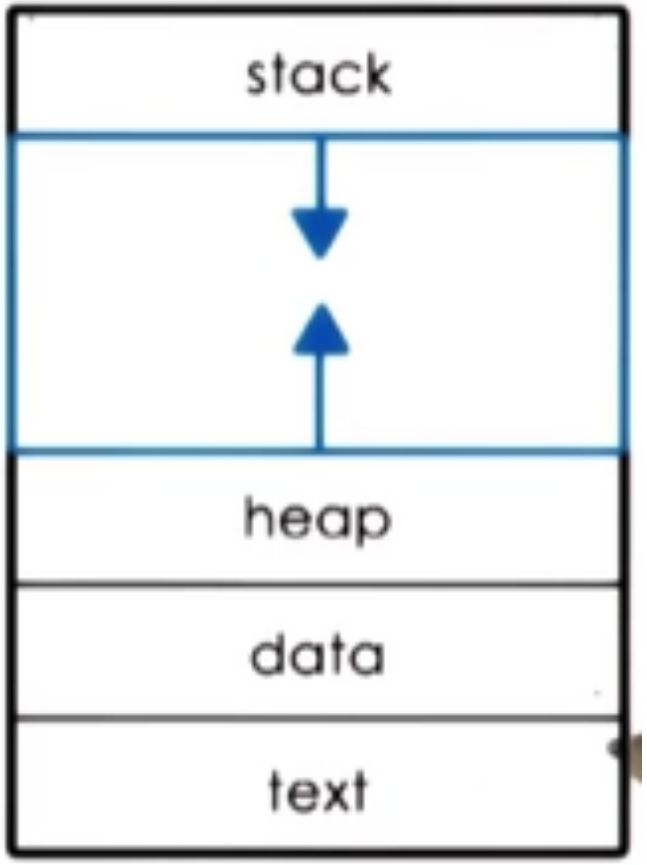
O que é um processo?

- Uma instância de um programa em execução
  - *task* (tarefa)
  - *job*
- Partes de um processo
  - Estado de execução
    - Contador de Programa (Program Counter – PC)
    - Pilha de execução (stack)
  - Partes e área temporária
    - Dados
    - Estados do registrador
- Pode requerer hardware especial
  - Dispositivos de E/S

Universidade Federal
de Mato Grosso
Campus Rondonópolis

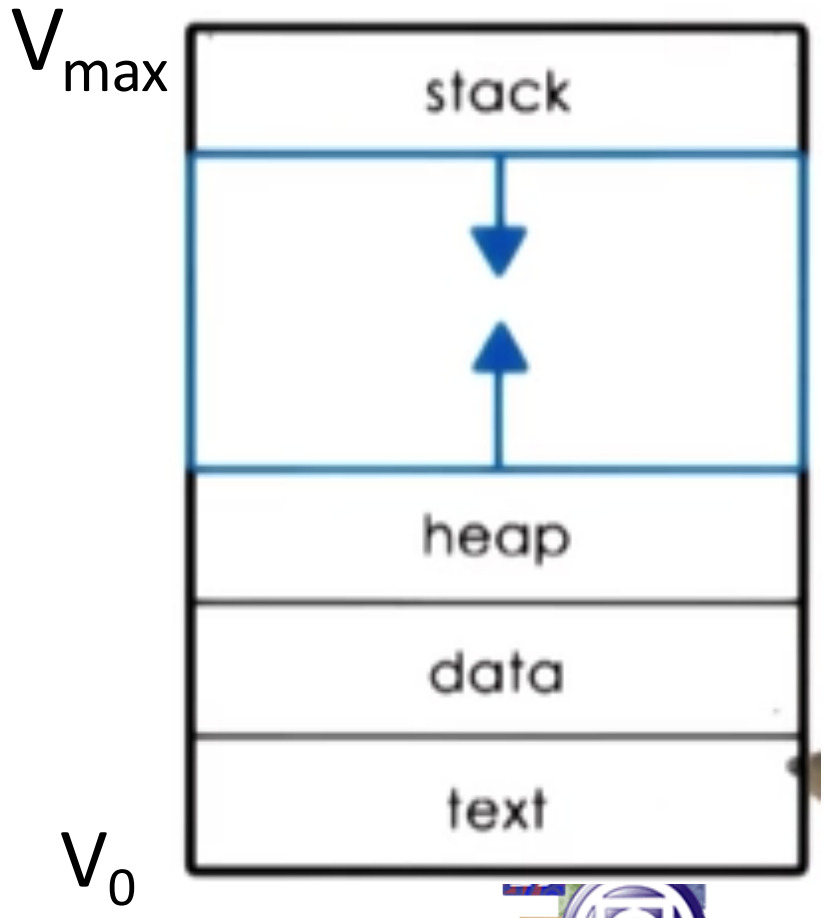
Como um processo se parece?

- Texto e dados
  - Estado estático quando o processo carrega
- Heap
  - Criado dinamicamente durante a execução
  - Não contíguo
- Pilha
  - Cresce e diminui durante a execução
  - Posso armazenar o estado de execução



Como um processo se parece?

- Espaço de endereçamento
  - Representação em memória do um processo
- Endereços virtuais
  - Não correspondem a espaços reais
- Compartilham memória física com outros processos



Como um processo se parece?

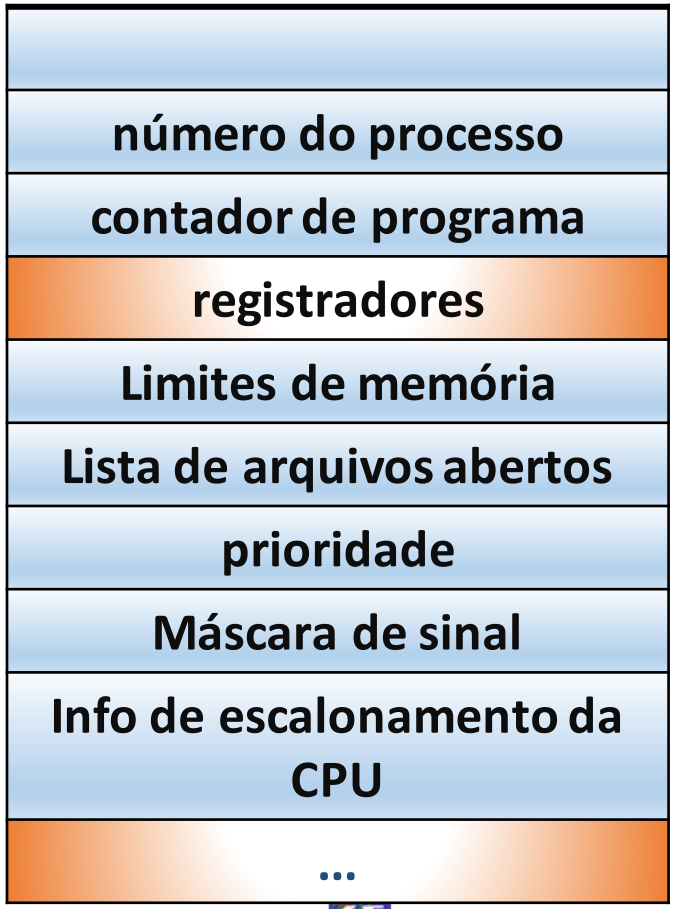
- Gerenciamento de memória faz um mapeamento
  - E.g.: tabelas de página
- E se não temos espaço suficiente?
  - SO é responsável por decidir quem vai (pra memória física) e quem fica

Como o SO sabe o que um um processo está fazendo?

- Um programa precisa ser compilado
  - O código assembly contém basicamente instruções de manipulação de registradores e ”saltos”
- O contador de programa (PC) diz em qual ponto do programa eu estou
  - Mantido na CPU durante a execução
- Além disso…
  - Registradores da CPU
  - Ponteiro de pilha
  - PCB (Process Control Block)

PCB – Process Control Block

- É uma estrutura mantida para cada processo que o SO gerencia
- Criado quando o processo é criado
- Campos são alteradoes de acordo com a mudança de estado



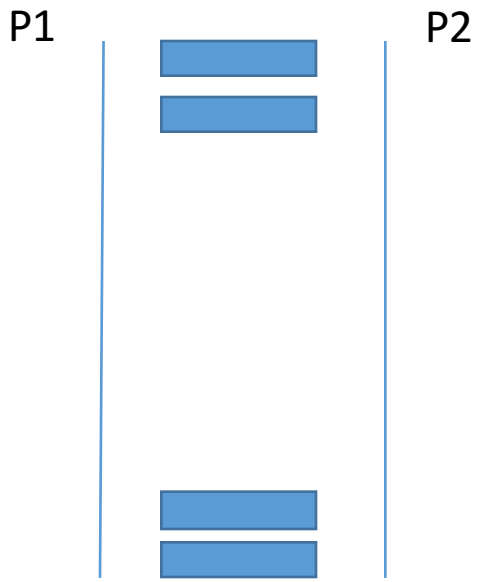
PCB – Process Control Block

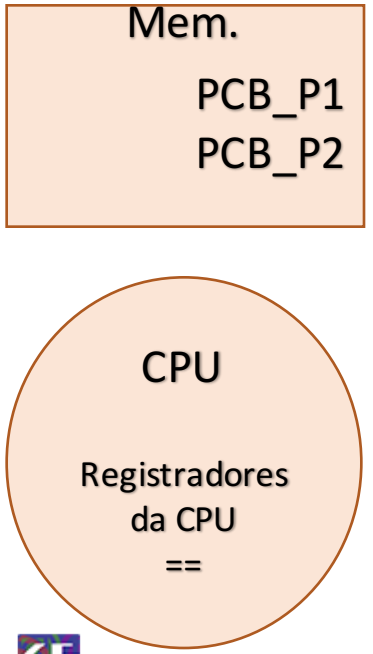
- Como um PCB é usado?
  - Quando o processo entra em espera, o SO armazena os valores de P1 e restaura P2 (e vice-versa)

Troca de contexto

Troca de contexto

- Troca-se o **contexto** de um processo para o **contexto** de outro

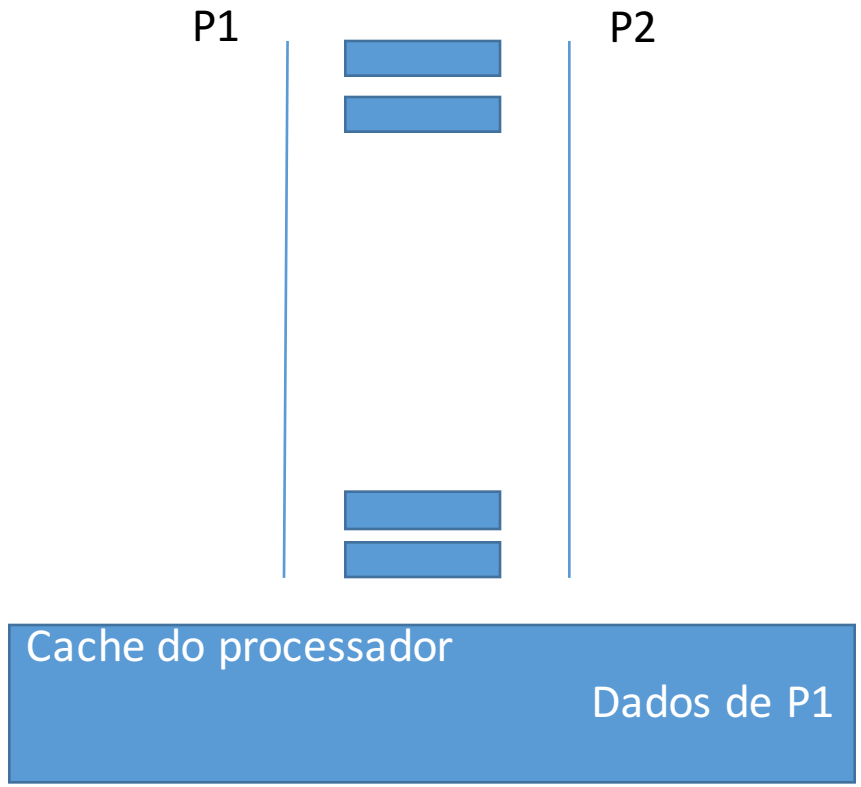


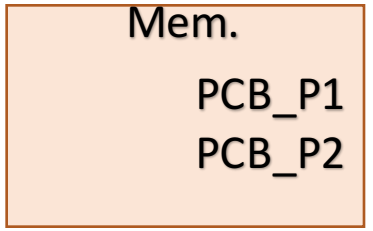


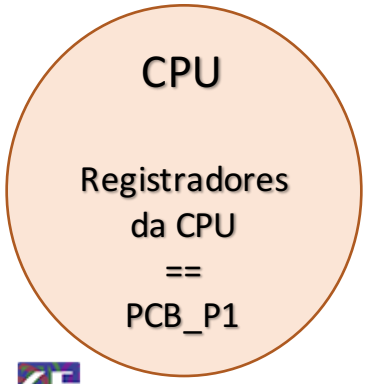
Universidade Federal de Mato Grosso
Campus Rondonópolis

Troca de contexto

- Troca-se o **contexto** de um processo para o **contexto** de outro





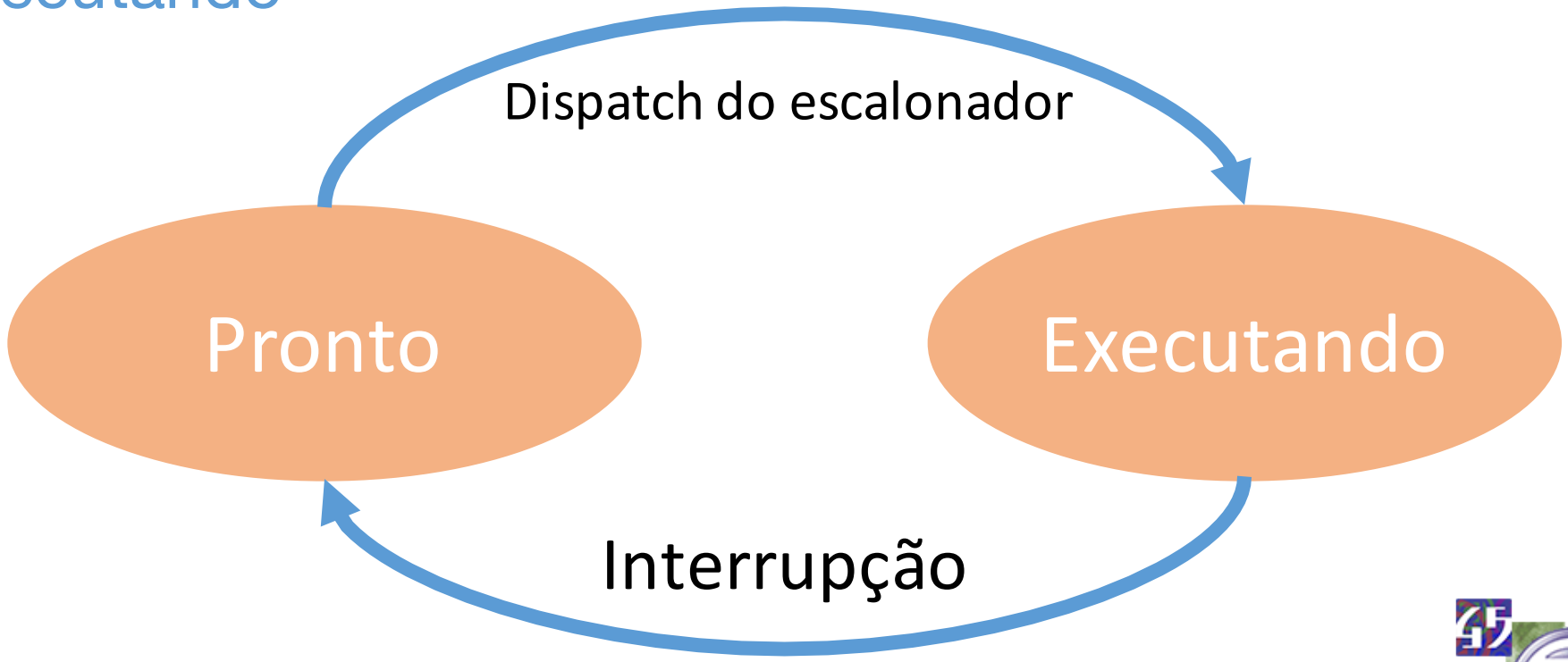


Troca de contexto

- Troca de contexto tem custos!
  - Custos diretos
    - Número de ciclos para carregar e armazenar instruções
  - Custos indiretos
    - Hot cache – dados na cache, acesso rápido (bom!)
    - Cold cache – é preciso buscar os dados em outra memória (cache miss)
- Boa ideia pensar em limitar a troca de contexto

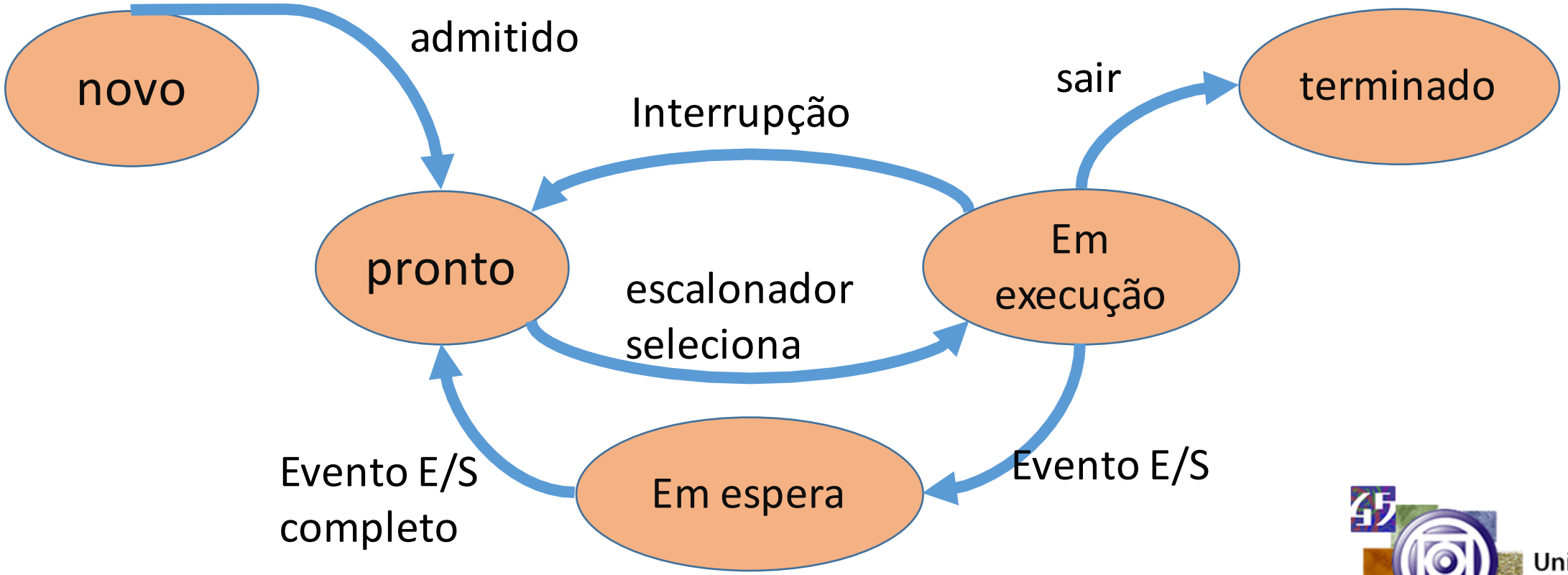
Ciclo de vida do processo

- Processos são entidades ativas, mas nem sempre estão executando



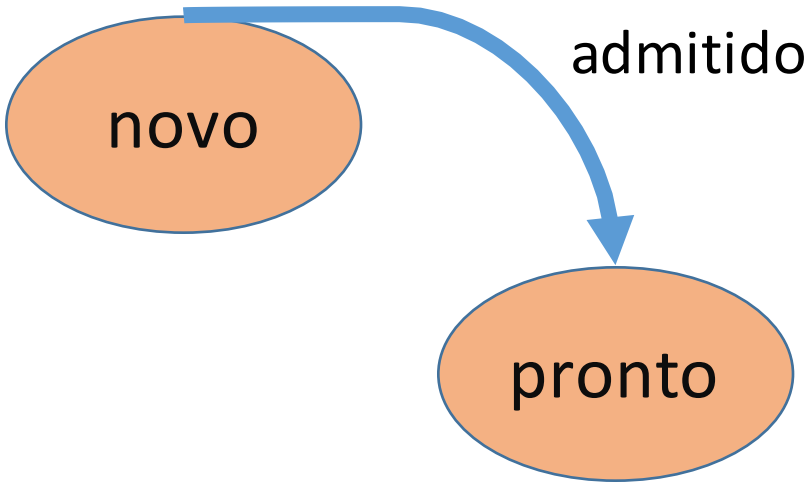
Ciclo de vida de um processo

- Versão extendida



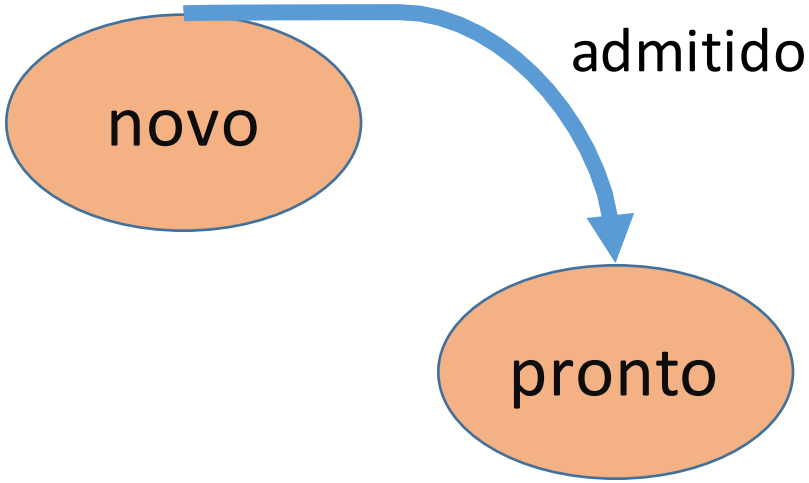
Criação de processos

- Nem sempre conseguimos criar todos os processos ao iniciar o sistema
- Quatro eventos gatilhos para criação de processos
  1. Início do sistema
  2. Execução de uma chamada de sistema de criação de processo por um processo em execução
  3. Uma requisição do usuário para criar um novo processo
  4. Início de uma tarefa em lote



Criação de processos

- Processo em primeiro-plano (foreground)
  - Processos que interagem com os usuários
- Processos em segundo-plano (background)
  - Processos que realizam tarefas específicoas
  - Exemplos
    - Processos para esperar e aceitar novas conexões
    - Processos para esperar por mensagens novas
  - Chamados de *daemons*



Criação de processos

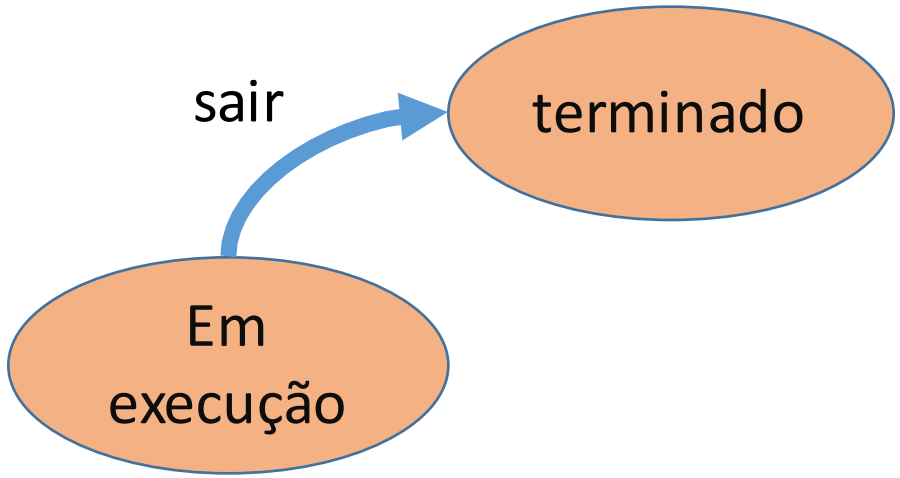
- Um processo cria processos-filhos
- Processo filhos possuem relações entre si
- Alguns são processos root
- Dois mecanismos comuns para criar procesos
  - Fork – copia o pcb do pai para um filho novo
    - Dali, continua a execução de onde parou
  - Exec – troca a imagem de núcleo do filho e carrega um novo programa
- No Windows
  - CreateProcess
    - Cria processo pai e Filho

Criação de processos

- UNIX
  - Processo init: gera vários processos filhos para atender
  - Outros processos são gerados nos terminais

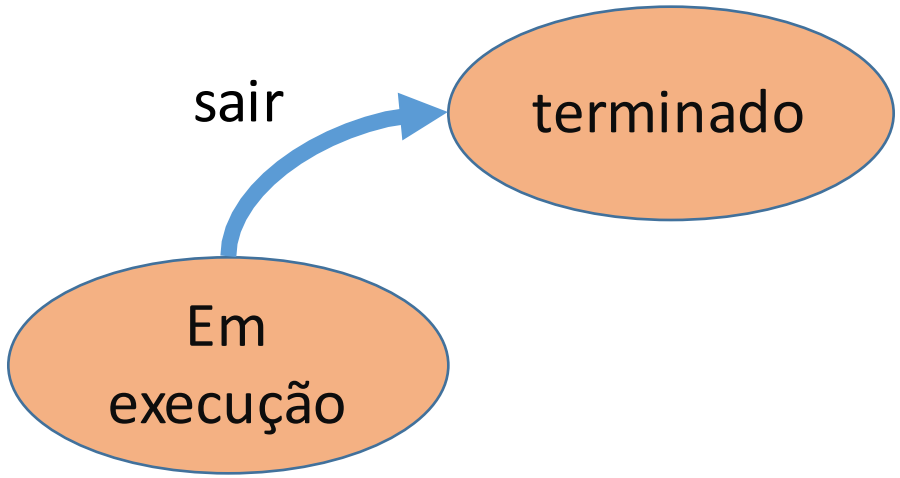
Término de processos

- Voluntário
  1. Saída normal – exit (UNIX) e ExitProcess (Win)
  2. Saída por erro – e.g.: arquivo não existe



Término de processos

- Involuntário
  1. Erro fatal
     - Divisão por zero
     - Referência a memória não existente
  2. Cancelamento por outro processo
     - Kill (UNIX)



Escalonamento de processos

- Problema: vários processos para executar
  - Qual executar? Quando executar? Quando trocar? Quando parar?
- Escalonador de processos (CPU scheduler)
  - Determina qual dos processos (prontos) serão enviados para a CPU para iniciar a execução, e por quanto tempo deve rodar
- Preempção
  - Interrupção do processo atual e salvamento do contexto

Escalonamento de processos

- Escalonamento
  - Rodar o escalonador para escolher o próximo passo
- Lembrando: para escolher, o processo deve estar no estado **Pronto**

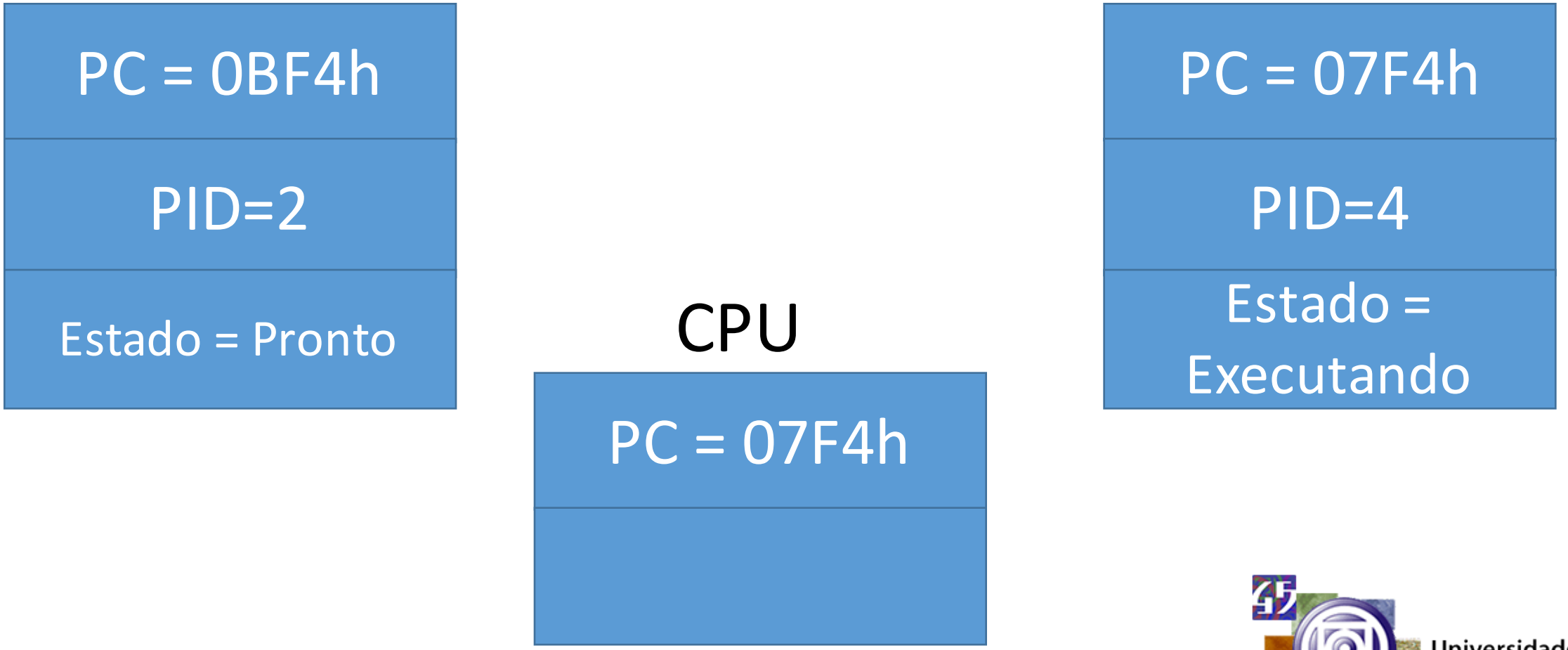
- Escalonador deve se preocupar com a eficiência!
  - Chaveamento é complexo e custoso
  - Afeta desempenho do sistema e satisfação do usuário;
- Escalonador de processo é um processo que deve ser executado quando há **mudança de contexto** (troca de processo);

Universidade Federal
de Mato Grosso
Campus Rondonópolis

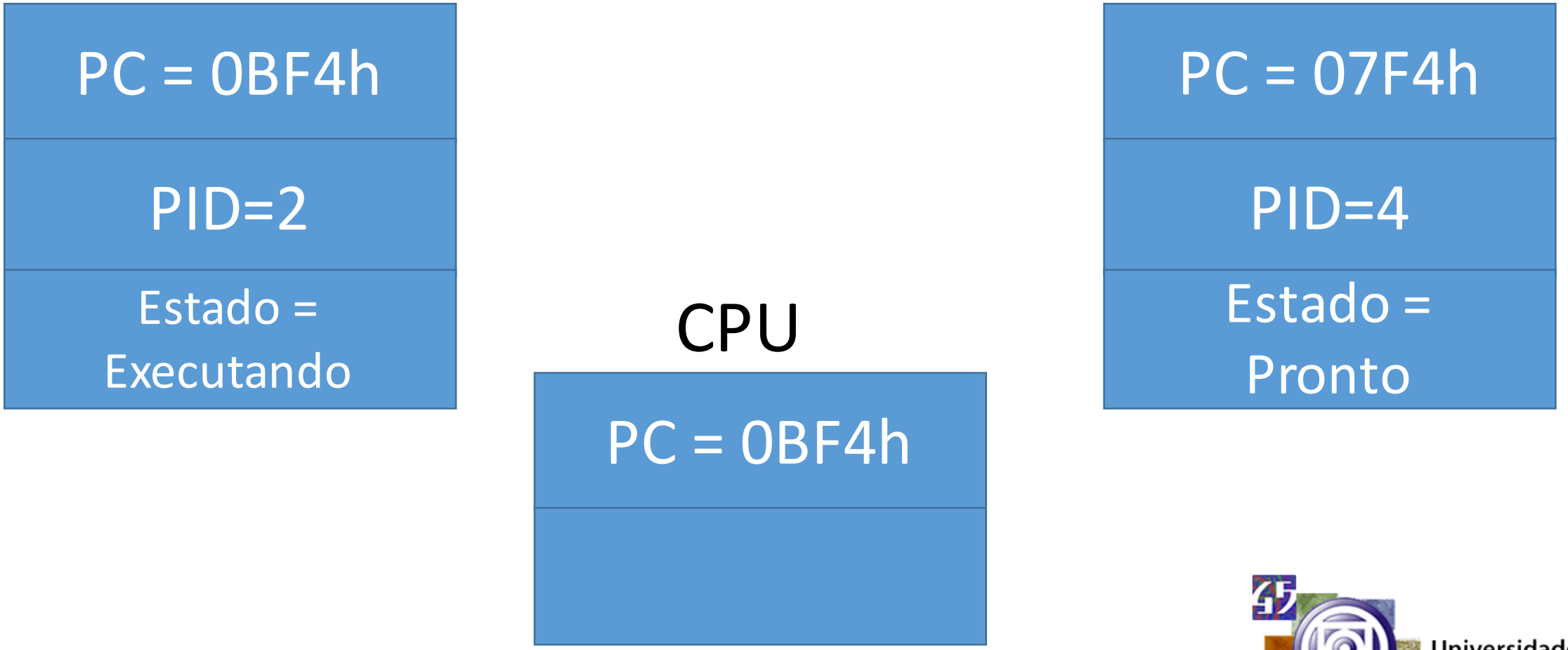
Escalonamento de processos

- Mudança de Contexto:
- Overhead de tempo;
- Tarefa cara:
  - Salvar as informações do **processo que está deixando a CPU** em seu BCP → conteúdo dos registradores;
  - Carregar as informações do **processo que será colocado na CPU** → copiar do BCP o conteúdo dos registradores;

Escalonamento de processos



Escalonamento de processos



Universidade Federal de Mato Grosso
Campus Rondonópolis

Escalonamento de processos

- Alguns processos passam a maior parte do tempo em atividades de E/S (I/O bound), enquanto outras passam tempo em atividades de CPU (CPU/bound)
- Conforme o tempo passa e processadores evoluem, temos mais processos I/O bound

Escalonamento de processos

- Minimizar o tempo ocioso da CPU é interessante!
- Escolher uma fatia de tempo (timeslice)
  - Templo alocado a um processo na na CPU
- Decisões de design
  - Quais valores apropriados ?
  - Métricas para escolher o próximo processo?

Escalonamento de processos

- Situações nas quais escalonamento é necessário:
  - Um novo processo é criado;
  - Um processo terminou sua execução e um processo pronto deve ser executado;
  - Quando um processo é bloqueado (semáforo, dependência de E/S), outro deve ser executado;
  - Quando uma interrupção de E/S ocorre o escalonador deve decidir por:
    - i) executar o processo que estava esperando esse evento;
    - ii) continuar executando o processo que já estava sendo executado ou;
    - iii) executar um terceiro processo que esteja pronto para ser executado.
- Quando CPU gera interrupções em intervalos entre 50 a 60 hz (ocorrências por segundo), é preciso tomar uma decisão de escalonamento

Escalonamento de processos

- Algoritmos de escalonamento podem ser divididos em duas categorias dependendo de como essas interrupções são tratadas:
  - **Preemptivo**: estratégia de suspender o processo sendo executado;
  - **Nãopreemptivo**: estratégia de permitir que o processo sendo executado continue sendo executado até ser bloqueado por alguma razão (semáforos, operações de E/S-interrupção);

Escalonamento de processos

- Categorias de Ambientes:
  - **Sistemas em Batch (lote)** : usuários não esperam por respostas rápidas; algoritmos preemptivos ou não preemptivos;
  - **Sistemas Interativos**: interação constante do usuário; algoritmos preemptivos; Processo interativo ☾ espera comando e executa comando;
  - **Sistemas em Tempo Real**: processos são executados mais rapidamente; tempo é crucial ☾ sistemas críticos; nem sempre preempção é interessante

Escalonamento de processos

- Características de algoritmos de escalonamento:
  - Qualquer sistema:
    - **Justiça** (*Fairness*)*:* cada processo deve receber uma parcela justa de tempo da CPU;
    - **Balanceamento**: diminuir a ociosidade do sistema;
    - **Reforço da política** – garantir o cumprimento da política

Escalonamento de processos

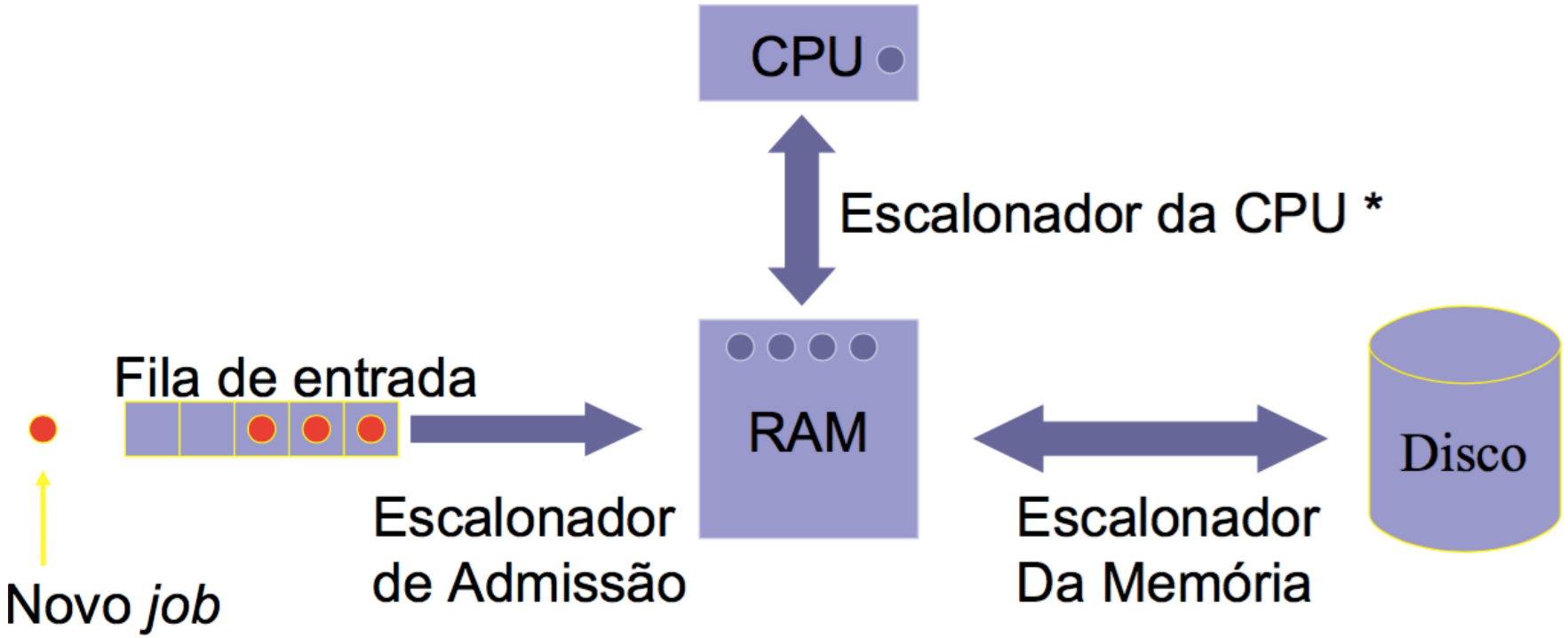
- Características de algoritmos de escalonamento:
- Sistemas em *Batch*:
  - **Taxa de execução** (*throughput*): máximo número de *jobs* executados por unidade de tempo (e.g. hora);
  - ***Turnaround time*** (tempo de retorno): minimizar tempo entre submissão e terminação (*Small is Beautiful* 🌸)
  - **Tempo de espera**: tempo gasto na fila de prontos;
  - **Eficiência**: Maximizar o tempo de CPU em uso;

Escalonamento de processos

- Características de algoritmos de escalonamento:
  - Sistemas Interativos:
    - **Tempo de resposta**: tempo esperando para iniciar execução;
    - Satisfazer **expectativas do usuários**
  - Sistemas em Tempo Real:
    - **Prevenir perda de dados**;
    - **Previsibilidade**: prevenir perda da qualidade dos serviços oferecidos;

Escalonamento de Processos
Sistemas em *Batch*

- Escalonamento *Three-Level*



Escalonamento de Processos
Sistemas em *Batch*

- Escalonamento *Three-Level*
  - **Escalonador de admissão**: processos menores primeiro; processos com menor tempo de acesso à CPU e maior tempo de interação com dispositivos de E/S;
  - **Escalonador da Memória**: decisões sobre quais processos vão para a MP:
    - A quanto tempo o processo está esperando?
    - Quanto tempo da CPU o processo já utilizou?
    - Qual o tamanho do processo?
    - Qual a importância do processo?
- **Escalonador da CPU**: seleciona qual o próximo processo a ser executado;

Universidade Federal de Mato Grosso
Campus Rondonópolis

Escalonamento de Processos
Sistemas em *Batch*

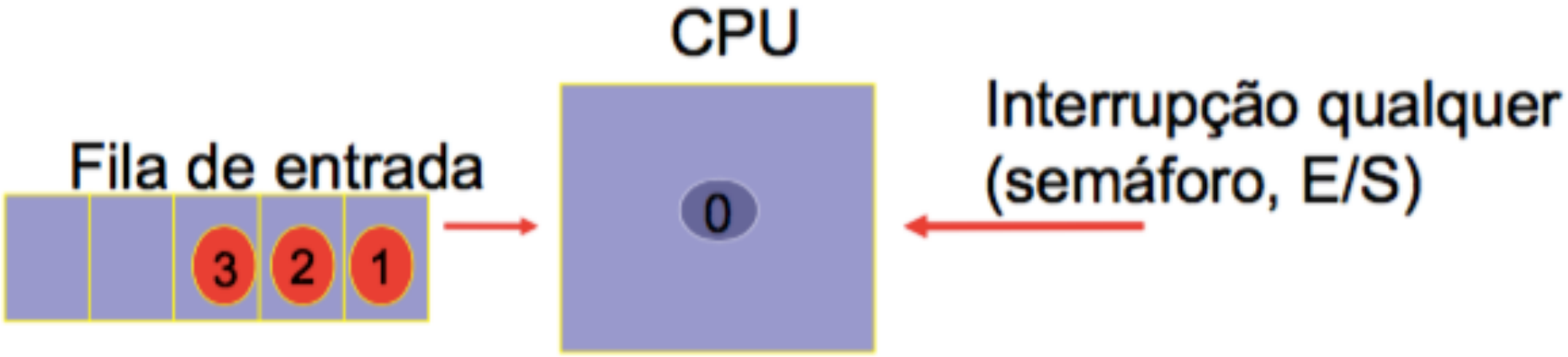
- Algoritmos para Sistemas em *Batch*:
  - *First-Come First-Served (ou FIFO)*;
  - *Shortest Job First (SJF)*;
  - *Shortest Remaining Time Next (SRTN)*;

Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo *First-Come First-Served*
  - Não-preemptivo;
  - Processos são executados na CPU seguindo a ordem de requisição;
  - Fácil de entender e programar;
  - Desvantagem:
    - Ineficiente quando se tem processos que demoram na sua execução;

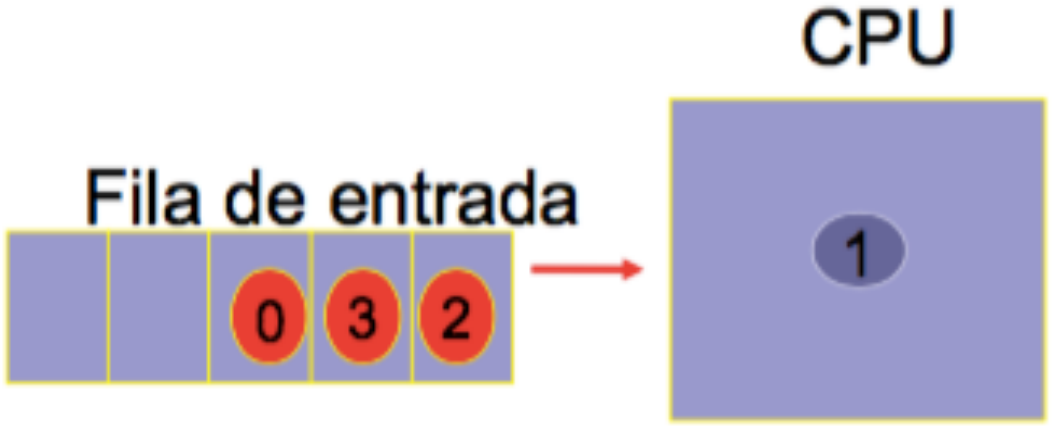
Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo First-Come First-Served



Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo First-Come First-Served



CPU não controla o tempo dos processos (não-preemptivo)

Escalonamento de Processos
Sistemas em *Batch*

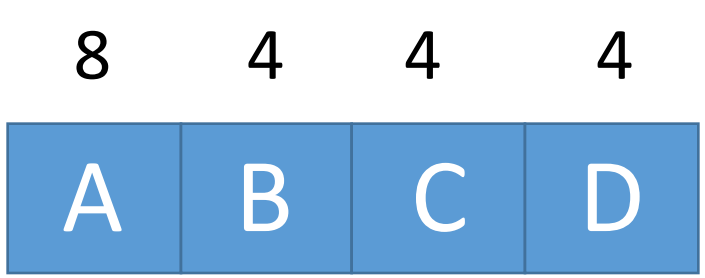
- Algoritmo *Shortest Job First*
  - Não-preemptivo;
  - Possível prever o tempo de execução do processo;
  - Menor processo é executado primeiro;
  - Menor *turnaround*;
  - Desvantagem:
    - Baixo aproveitamento quando se tem poucos processos prontos para serem executados;

Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo Shortest Job First
- A → a
- B → b+a
- C → c+b+a
- D → d+c+b+a
- --------
- Tempo médio-turnaround (4a+3b+2c+d)/4
- Contribuição → se a<b<c<d tem-se o mínimo tempo médio

Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo Shortest Job First



Em ordem:

Turnaround A = 8

Turnaround B = 12

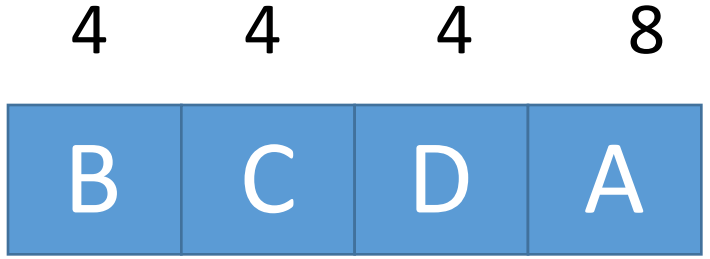
Turnaround C = 16

Turnaround D = 20

Média→56/4 = 14

Número de processos

(4a+3b+2c+d)/4



Menor *job* primeiro:

Turnaround B = 4

Turnaround C = 8

Turnaround D = 12

Turnaround A = 20

Média → 44/4 = 11

Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo *Shortest Remaining Time Next*
  - Versão preemptiva do *Shortest Job First*
  - Processos com menor tempo de execução são executados primeiro;
  - Se um processo novo chega e seu tempo de execução é menor do que do processo corrente na CPU, a CPU suspende o processo corrente e executa o processo que acabou de chegar;
- Desvantagem: processos que consomem mais tempo podem demorar muito para serem finalizados se muitos processos pequenos chegarem!

Escalonamento de Processos
Sistemas interativos

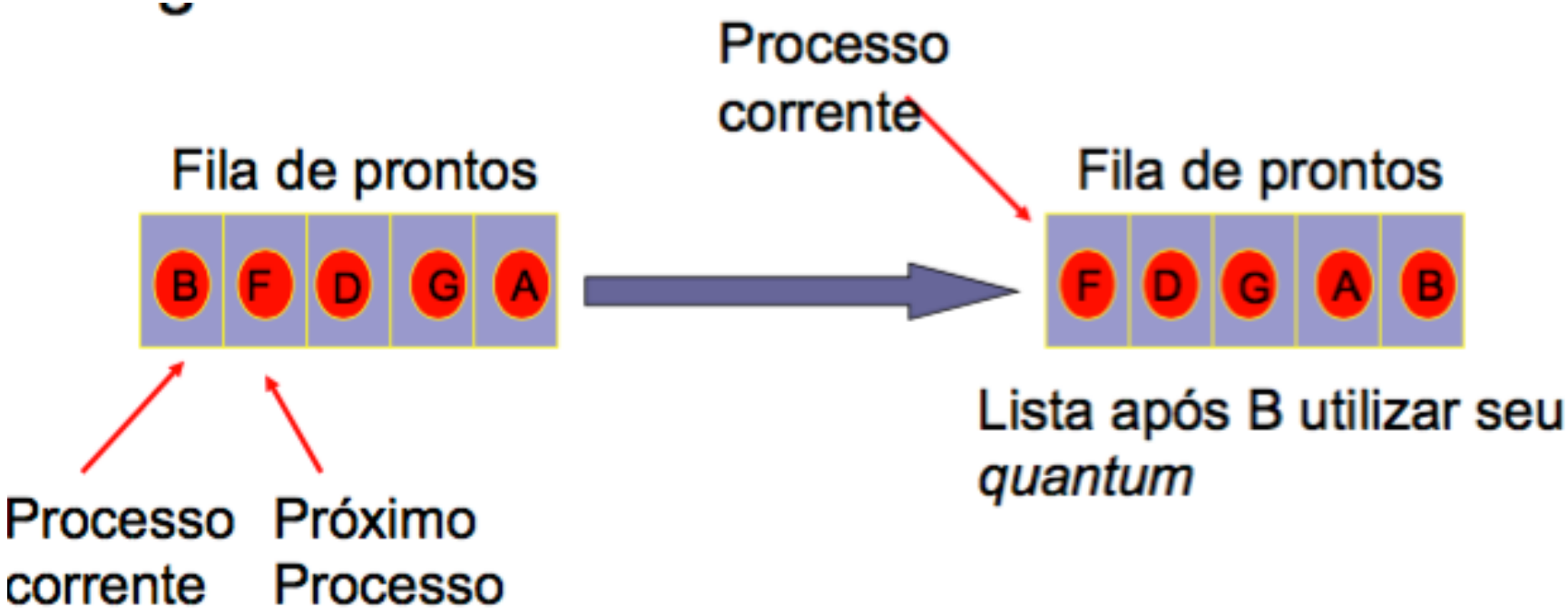
- Algoritmos para Sistemas Interativos:
  - *Round-Robin*;
  - Prioridade;
  - Múltiplas Filas;
  - *Shortest Process Next*;
  - Garantido;
  - *Lottery*;
  - *Fair-Share;*
- Utilizam escalonamento em dois níveis (escalonador da CPU e memória);

Escalonamento de Processos
Sistemas interativos

- Algoritmo Round-Robin
  - Antigo, mais simples e mais utilizado;
  - Preemptivo (quantum, I/O, system call pelo processo);
  - Cada processo recebe um tempo de execução chamado **quantum**; ao final desse tempo, o processo é suspenso e outro processo é colocado em execução;
  - Escalonador mantém uma lista de processos prontos;

Escalonamento de Processos
Sistemas interativos

- Algoritmo Round-Robin



Escalonamento de Processos
Sistemas interativos

- Algoritmo *Round-Robin*
  - Importante: tempo de chaveamento de processos;
  - ***quantum***: se for muito pequeno, ocorrem muitas trocas diminuindo, assim, a eficiência da CPU; se for muito longo o tempo de resposta é comprometido;

Escalonamento de Processos
Sistemas interativos

- Algoritmo *Round-Robin:*
- Exemplos:
  - Δt = 4 mseg
    x = 1mseg→25% de tempo de CPU é perdido→ menor eficiência
  - Δt = 100 mseg
  - x = 1mseg→ 1% de tempo de CPU é perdido→ Tempo de espera dos processos é maior

Quantum razoável: 20-50ms

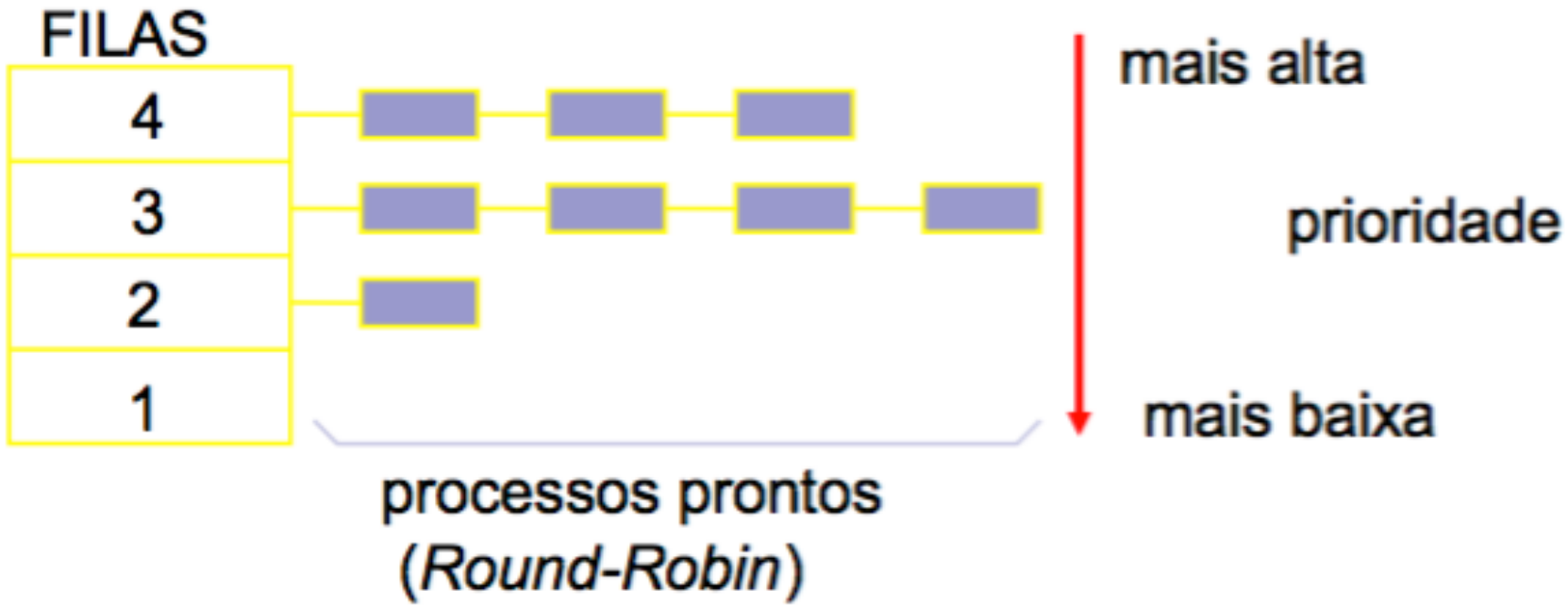
Universidade Federal de Mato Grosso
Campus Rondonópolis

Escalonamento de Processos
Sistemas interativos

- Algoritmo de escalonamento por prioridade (*Priority Scheduling)*
  - Round-robin considera todos os processos iguais
  - Cada processo possui uma prioridade → os processos prontos com maior prioridade são executados primeiro;
  - Prioridades são atribuídas dinâmica ou estaticamente;
  - Classes de processos com mesma prioridade;
  - Preemptivo;

Escalonamento de Processos
Sistemas interativos

- Algoritmos com Prioridades



Escalonamento de Processos
Sistemas interativos

- Algoritmos com Prioridades
  - Como evitar que os processos com maior prioridade sejam executados indefinidamente?
    - Diminuir a prioridade do processo corrente e troca-lo pelo próximo processo com maior prioridade (chaveamento);
    - Cada processo possui um *quantum*;

Universidade Federal de Mato Grosso
Campus Rondonópolis

Escalonamento de Processos
Sistemas interativos

- Múltiplas Filas:
  - CTSS (*Compatible Time Sharing System*);
    - IBM7094 só poderia ter um processo na memória
    - Solução 1: Mais quantum! RUIM
  - Solução 2: Classes de prioridades;
  - Cada classe de prioridades possui *quanta* diferentes;
  - Assim, a cada vez que um processo é executado e suspenso ele recebe mais tempo para execução;
  - Preemptivo;

Escalonamento de Processos
Sistemas interativos

- Múltiplas Filas:
  - Ex.: um processo precisa de 100 *quanta* para ser executado;
    - Inicialmente, ele recebe um *quantum* para execução;
    - Das próximas vezes ele recebe, respectivamente, 2, 4, 8, 16, 32 e 64 *quanta* (7 chaveamentos) para execução;
    - Quanto mais próximo de ser finalizado, menos frequente é o processo na CPU → eficiência

Escalonamento de Processos
Sistemas interativos

- Sistemas Linux e Windows
  - Multilevel feedback queue
    - Dá preferência a processos curtos
    - Dá prioridades para processos I/O bound
    - Estuda o processo e escalona de acordo com o estudo
    - Cada fila usa um escalonamento round-robin
    - Os I/Os promovem o processos para as filas com maior prioridade

Escalonamento de Processos
Sistemas interativos

- Algoritmo *Shortest Process Next*
  - Mesma idéia do *Shortest Job First*;
  - Processos Interativos: não se conhece o tempo necessário para execução;
  - Como empregar esse algoritmo: ESTIMATIVA de TEMPO!
  - Verficar o comportamento passado do processo e estimar o tempo.
    - Aging – estimativa do próximo valor em uma série

Escalonamento de Processos
Sistemas interativos

- Outros algoritmos:
  - Algoritmo de Escalonamento Garantido:
    - Garantias são dadas aos processos dos usuários:
      - **n** usuários → **1/n** do tempo de CPU para cada usuário;
  - *Lottery Scheduling*:
    - Cada processo recebe "*tickets*" que lhe dão direito de execução;
    - Processos podem compartilhar tickets
    - Novos processos podem ganhar tickets e "concorrer"

Escalonamento de Processos
Sistemas interativos

- Algoritmo *Fair-Share*:
  - O dono do processo é levado em conta;
  - Se um usuário A possui mais processos que um usuário B, o usuário A terá prioridade no uso da CPU;
- Usuário 1 → A, B, C, D
  - Usuário 2 → E
  - Garantia de 50%
- Circular → A, E, B, E, C, E, D, E
- 50% a mais para Usuário 1 → A, B, E, C, D, E

Escalonamento de Processos
Sistemas em Tempo Real

- Tempo é um fator crítico; possui um deadline
- Sistemas críticos:
  - Aviões;
  - Hospitais;
  - Usinas Nucleares;
  - Bancos;
  - Multimídia;
- Ponto importante: obter respostas em atraso é tão ruim quanto não obter respostas;

Escalonamento de Processos
Sistemas de Tempo Real

- Tipos de STR:
  - ***Hard Real Time***: atrasos não são tolerados;
    - Aviões, usinas nucleares, hospitais;
  - ***Soft Real Time***: atrasos são tolerados;
    - Bancos; Multimídia;
- Programas são divididos em vários processos;
- Eventos causam a execução de processos:
  - **Periódicos**: ocorrem em intervalos regulares de tempo;
  - **Aperiódicos**: ocorrem em intervalos irregulares de tempo;

Escalonamento de Processos
Sistemas de Tempo Real

- Algoritmos podem ser estáticos ou dinâmicos;
  - **Estáticos**: decisões de escalonamento antes do sistema começar;
    - Informação disponível previamente;
  - **Dinâmicos**: decisões de escalonamento em tempo de execução;

Política vs. Mecanismo

- Até então, todos os processos pertencem a diferentes usuários e competem pela CPU
- Processos podem ter filhos rodando
- Escalonadores não consideram o plano do processo-pai
- Solução: **separação** entre **mecanismo** e **política** de escalonamento
  - Parâmetros do algoritmo podem ser preenchidos pelos procesos