COPPE UFRJ

GTA / UFRJ

Escalonamento

Pedro Cruz

EEL770 – Sistemas Operacionais

Avisos

GTA / UFRJ

- Propostas de trabalho já foram revisadas
  - Todos os grupos (que eu vi) foram comunicados
    - Se você não recebeu um comunicado, me envie um email

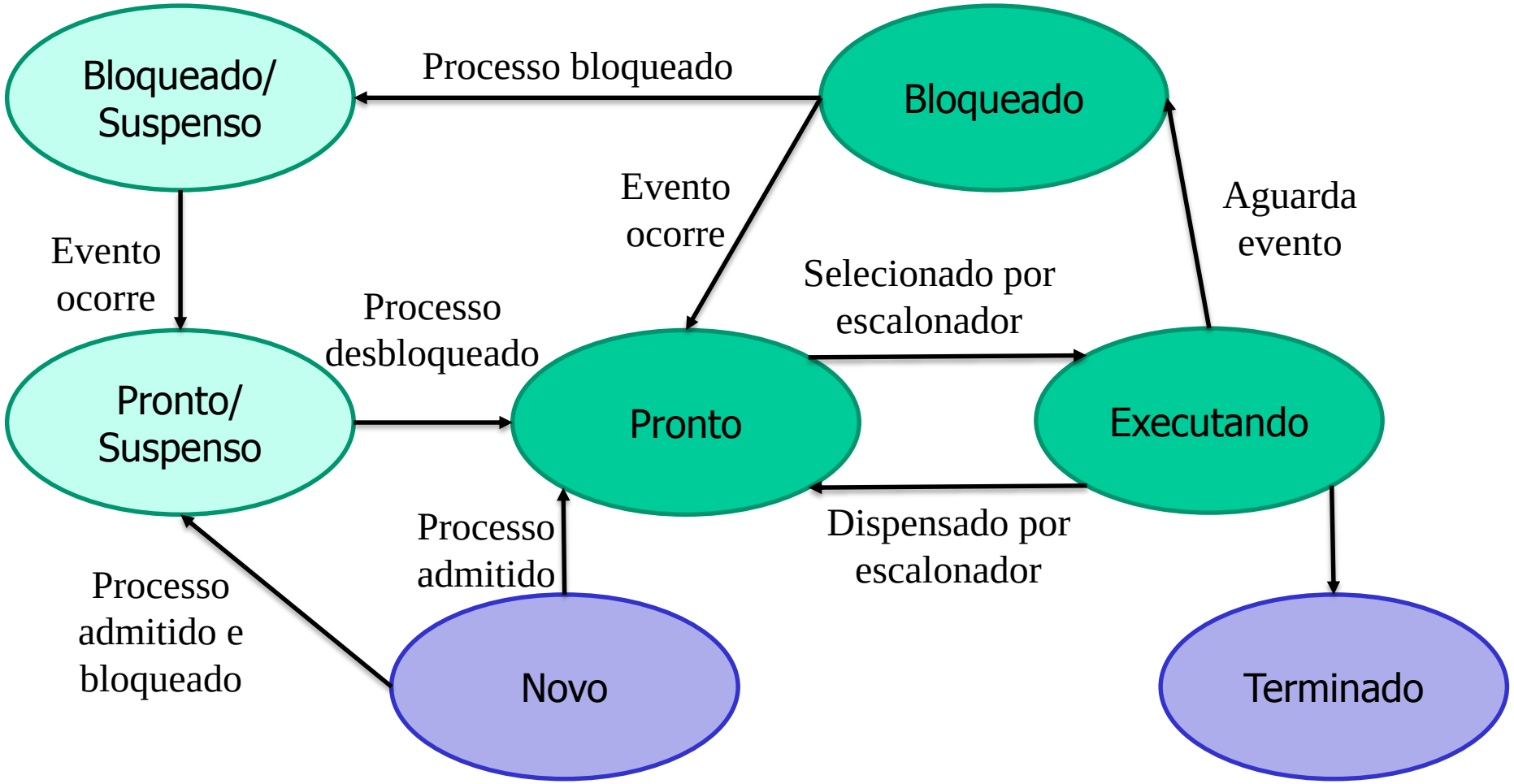
Última aula

GTA / UFRJ

- Impasses
  - Condições pra impasses
- Soluções para impasses
  - Aceitar
  - Detectar
  - Evitar
  - Prevenir

Estados de processos - Lembrando

GTA / UFRJ



Escalonamento

GTA / UFRJ

- Pode haver mais de um processo em estado Pronto
  - Processos competem por tempo de CPU

- Escalonamento
  - Decisão e atuação sobre qual processo/*thread* deve usar a CPU

Escalonamento - motivação

- Interatividade com usuário
- Justiça
  - Entre processos/*threads*
  - Entre usuários
- Equilíbrio da carga
  - Garantir que não haja subutilização do sistema

Chaveamento entre processos

GTA / UFRJ

- Salvar contexto do processo que não será mais executado
  - Registradores e PSW
  - Memória principal
  - Memória secundária
- Carregar contexto do processo que será executado
  - Registradores e PSW
  - Memória principal
  - Memória secundária

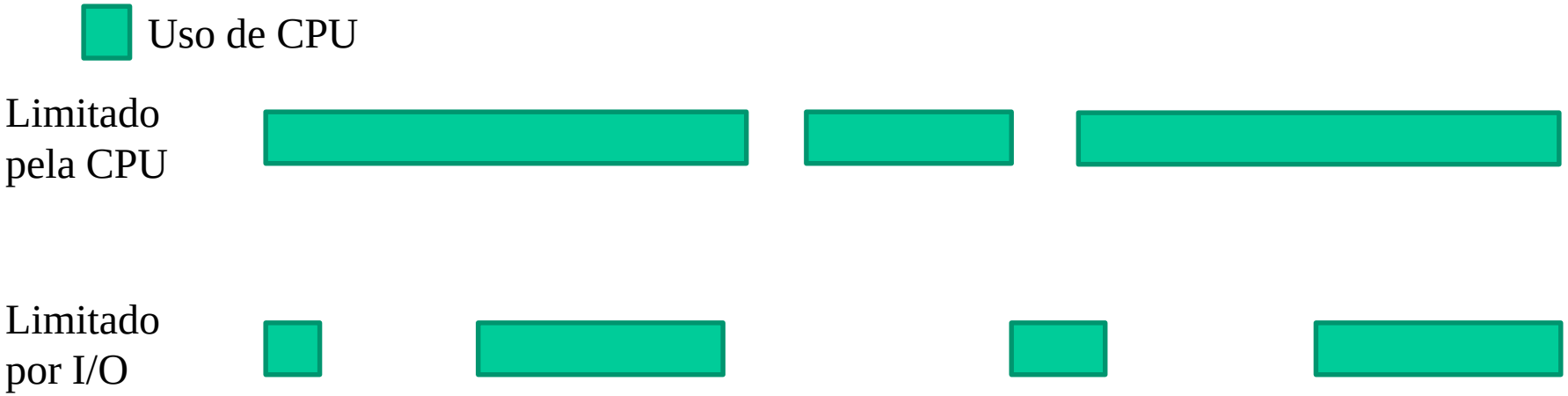
Tempo é gasto!

Caches são perdidos!

Limitações de processos

GTA / UFRJ

- Limitados pela CPU
- Limitados por I/O



Quando escalonar?

GTA / UFRJ

- Início de um processo
- Morte de um processo
- Bloqueio de um processo
  - Qual o motivo do bloqueio?
- Interrupção de I/O
- Interrupção de relógio
  - Escalonamento não-preemptivo
    - Processo não sai da CPU até que seja bloqueado
  - Escalonamento preemptivo
    - Processo é retirado da CPU após um tempo

Algoritmos de escalonamento

- De acordo com o sistema operacional onde executam
  - Lotes
  - Interativo
  - Tempo real

Requisitos de sistemas em lotes

- Vazão de tarefas
  - Maximizar o número de tarefas executadas por tempo
- Velocidade de tarefas
  - Minimizar o tempo de execução de cada tarefa
- Utilização da CPU
  - Não subutilizar a CPU

Escalonamento em sistemas em lote

- Primeiro a chegar, primeiro a ser servido
- Tarefa mais curta primeiro
- Tempo restante mais curto em seguida

Primeiro a chegar, primeiro a ser servido

- CPU é subutilizada por processos limitador por I/O
- Tarefas curtas precisam aguardar tarefas longas

Tarefa mais curta primeiro

GTA / UFRJ

- Garante máximo de tarefas
- Só vale quando tarefas estão disponíveis de forma simultânea
- Necessário saber o tamanho das tarefas

Tempo restante mais curto em seguida

GTA / UFRJ

- Necessário saber o tamanho das tarefas

Requisitos de sistemas interativos

- Tempo de resposta
  - Responder rápido aos usuários
- Proporcionalidade
  - Atender aos usuários de forma homogênea

Escalonamento em sistemas interativos

- Chaveamento circular (*round-robin*)
- Escalonamento por prioridades
- Escalonamento de múltiplas filas
- Processo mais curto em seguida
- Escalonamento garantido
- Escalonamento por loteria
- Escalonamento por fração justa

Chaveamento circular (*round-robin*)

GTA / UFRJ

- Cada processo recebe um *quantum* para executar
  - Bloqueado ao final do *quantum*
  - Bloqueado quando realiza chamada bloqueante
- Tamanho do *quantum* é importante
  - Por quê?
    - *Quantum* pequeno?
    - *Quantum* grande?

Escalonamento por prioridades

GTA / UFRJ

- Prioridades estáticas
  - *Quantum* varia de acordo com prioridade
- Prioridades dinâmicas
  - Processos já executados recebem prioridades menores
    - Prioridades crescem com o tempo

Problema: decidir prioridades

Múltiplas filas

GTA / UFRJ

- Filas representam classes
  - Classes inferiores recebem *quantum* maior
  - Classes superiores são executados mais frequentemente
- Processos executados até o final de seu *quantum* descem pra uma classe inferior
  - Reduz tempo de espera de processos limitados por CPU
  - Melhora interatividade

Processo mais curto em seguida

GTA / UFRJ

- Processo interativo
  - Espera comando -> executa comando
- Escalonador estima tempo de execução de comando
  - Deixa o comando ser executado até o fim
- Estimativa é feita com média móvel
  - Envelhecimento (*aging*)

Escalonamento garantido

GTA / UFRJ

- Promete tempo para processos
- Cumpre promessa
  - Calcula quanto da promessa foi cumprida para processos
  - Executa processo com promessa menos cumprida

Escalonamento por loteria

GTA / UFRJ

- Processos recebem bilhetes
  - Podem receber mais ou menos bilhetes
    - Implementação de prioridades
  - Podem trocar bilhetes
    - Implementação de cooperação
- Escalonador sorteia bilhete
  - Processos com mais bilhetes têm mais chances de ganhar
- Processo dono do bilhete recebe um *quantum*

Escalonamento por fração justa

- Usuários são tratados de forma justa por escalonador
  - Processos de um usuário não se sobrepõem aos processos de outro usuário

Requisitos de sistemas de tempo real

GTA / UFRJ

- Cumprir prazos
  - Evitar perda de dados
  - Evitar prejudicar a experiência do usuário
- Previsibilidade
  - Dar qualidade a serviços de multimídia
- Atender a eventos
  - Periódicos
  - Não-periódicos

Escalonamento em sistemas de tempo real

GTA / UFRJ

- Tolerância a atrasos
  - Crítico
    - Não aceita atrasos
  - Não-crítico
    - Aceita atrasos
- Possibilidade de escalonamento
  - Sistema escalonável
    - Tempo exigido pelos eventos é menor ou igual que tempo disponível de CPU

Tempo de escalonamento

- Estático
  - Eventos são conhecidos na inicialização
    - Possível estabelecer um plano de escalonamento
- Dinâmico
  - Eventos não são conhecidos na inicialização
    - Necessário tomar decisões em tempo de execução

Política vs mecanismo

GTA / UFRJ

- Processos podem querer cooperar ao invés de competir
  - Pais & filhos
- Separação de política e mecanismo pode facilitar isso
  - Algoritmo de escalonamento
    - Política num processo de usuário
      - Parâmetros ajustáveis pelos processos
    - Mecanismo no núcleo
      - Execução rígida executada pelo SO

Escalonamento de *threads*

GTA / UFRJ

- *Threads* no espaço de usuário
  - Usuário decide qual *thread* será executada
    - Escalonamento personalizado
    - *Threads* são chaveadas até escalonador parar o processo
    - *Threads* de processos diferentes não podem cooperar

- *Threads* no espaço do núcleo
  - Operacional decide qual *thread* será executada
    - Escalonamento não pode ser personalizado
    - *Threads* de processos diferentes podem cooperar
    - Trocar *threads* custa trocar contextos de processos

Fim dos processos

GTA / UFRJ

- Próxima matéria: memória e gerenciamento de memória

COPPE UFRJ

GTA / UFRJ

Escalonamento

Pedro Cruz

EEL770 – Sistemas Operacionais