DCA-0125 Sistemas de Tempo Real

Luiz Affonso Guedes
www.dca.ufrn.br/~affonso
affonso@dca.ufrn.br

Conceitos de Prioridade e Alocação de CPU de Processos

Conteúdo

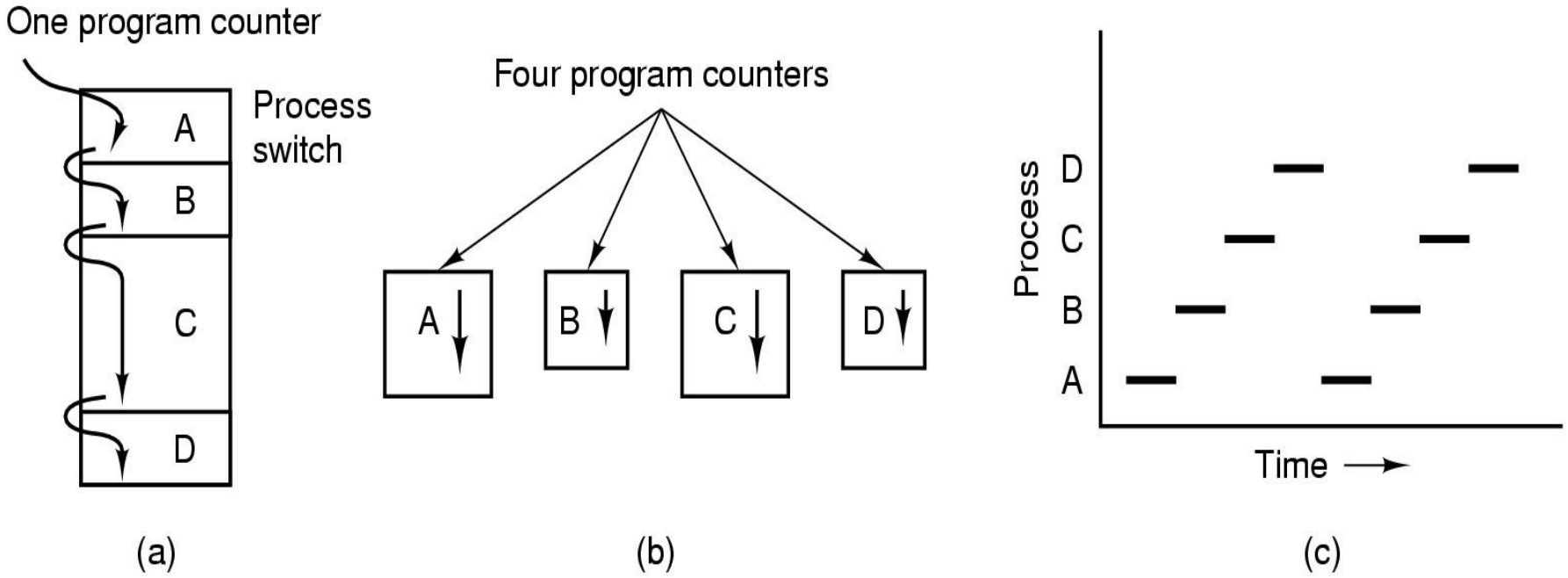
- Definição de prioridade de processos no Sistema Operacional.
- Exemplos de programas em C\C++ para atribuição de prioridade de processos.
- Alocação de CPUs para processos.
- Exemplos de programas em C/C++ para alocação de CPU para processos.

Conceitos Fundamentais

- Processos
- Interrupção
- Sincronização, comunicação, controle e proteção entre processos

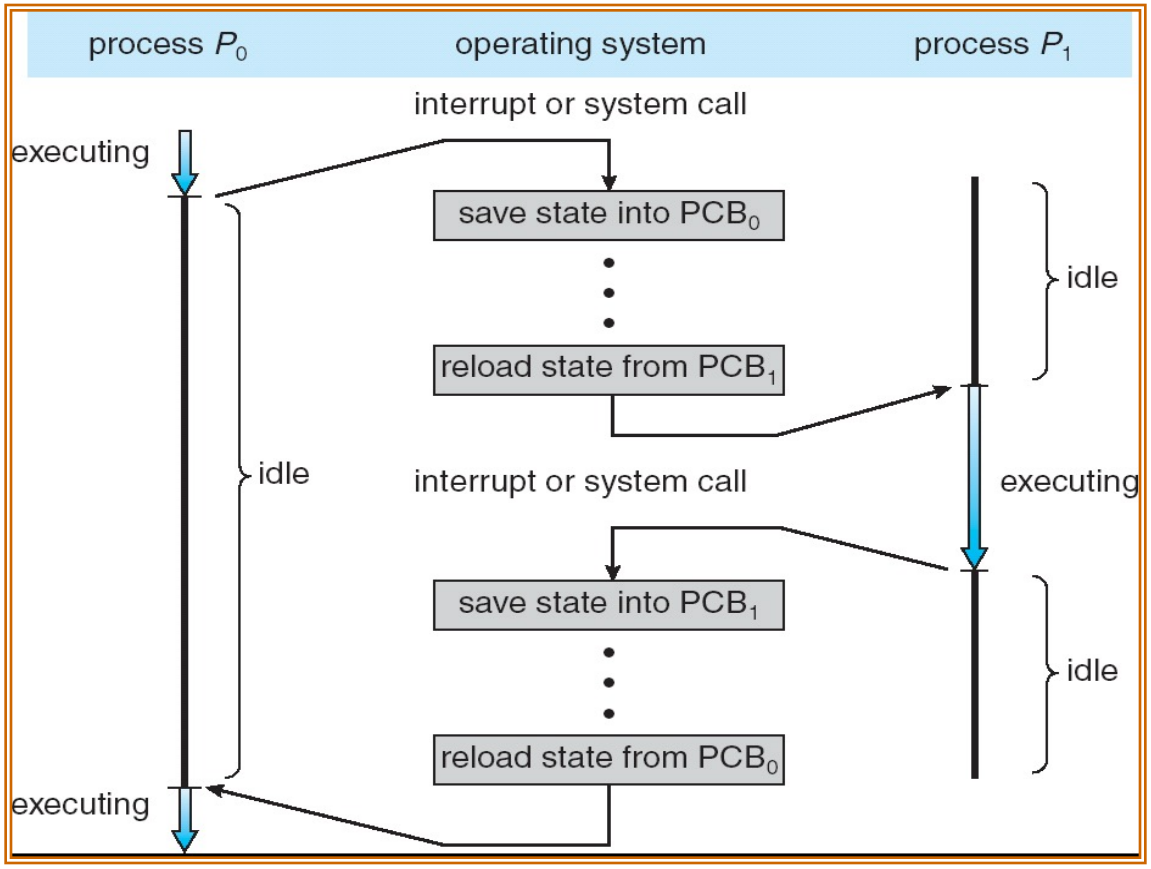
Questão Básica

- Como há mais processos que processadores, como intercalar o uso dos processadores entre os diversos processos?



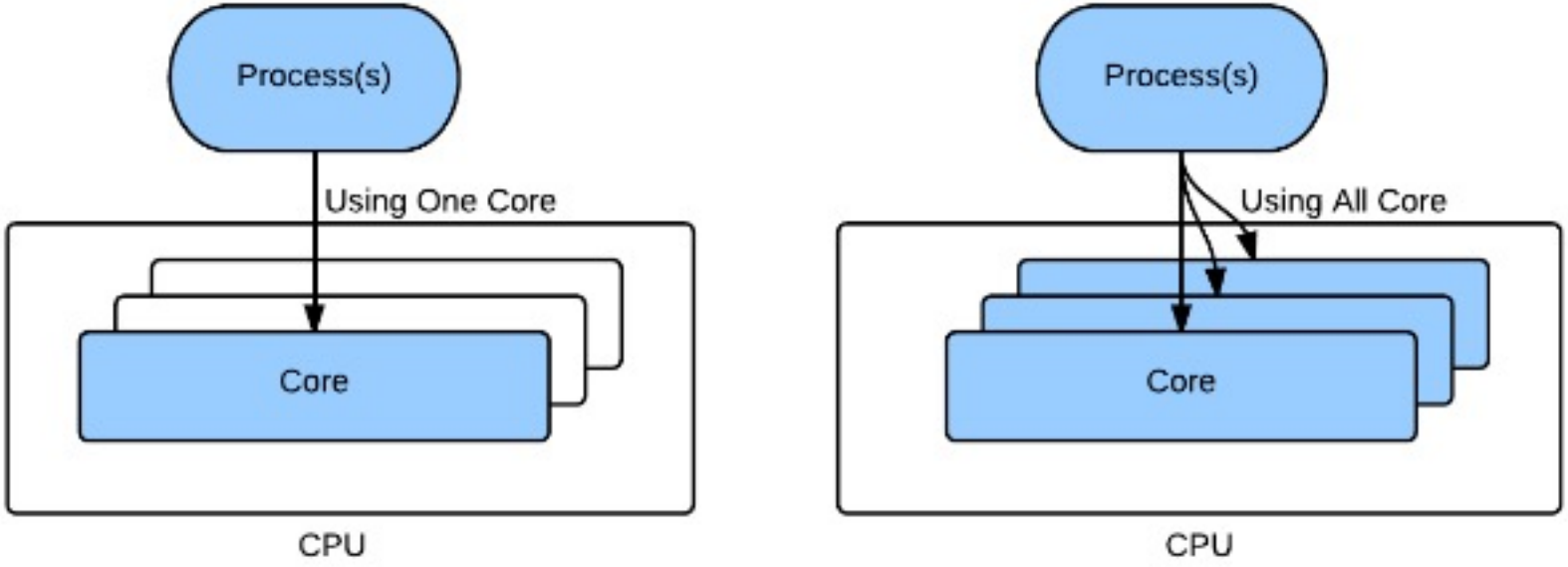
Questão Básica

- Necessidade de mudança de contexto entre processos



Processos - Alocação de CPUs

- Contexto de multi-core
  - A afinidade do processador, ou afixação da CPU ou "afinidade do cache", permite a ligação e a desatamento de um processo a uma CPU ou a uma variedade de CPUs, de modo que o processo seja executado apenas na CPU ou CPUs designadas



Alocação de CPUs

- Exercício:
  - Compile e execute o programa alocar_cpu.cpp
    - g++ alocar_cpu.cpp –o alocar_cpu
    - ./alocar_cpu
  - Em outro terminal, execute o programa Htop (ou top) e verifique em qual CPU o programa está executando.
  - Analise o resultado e o código do programa.

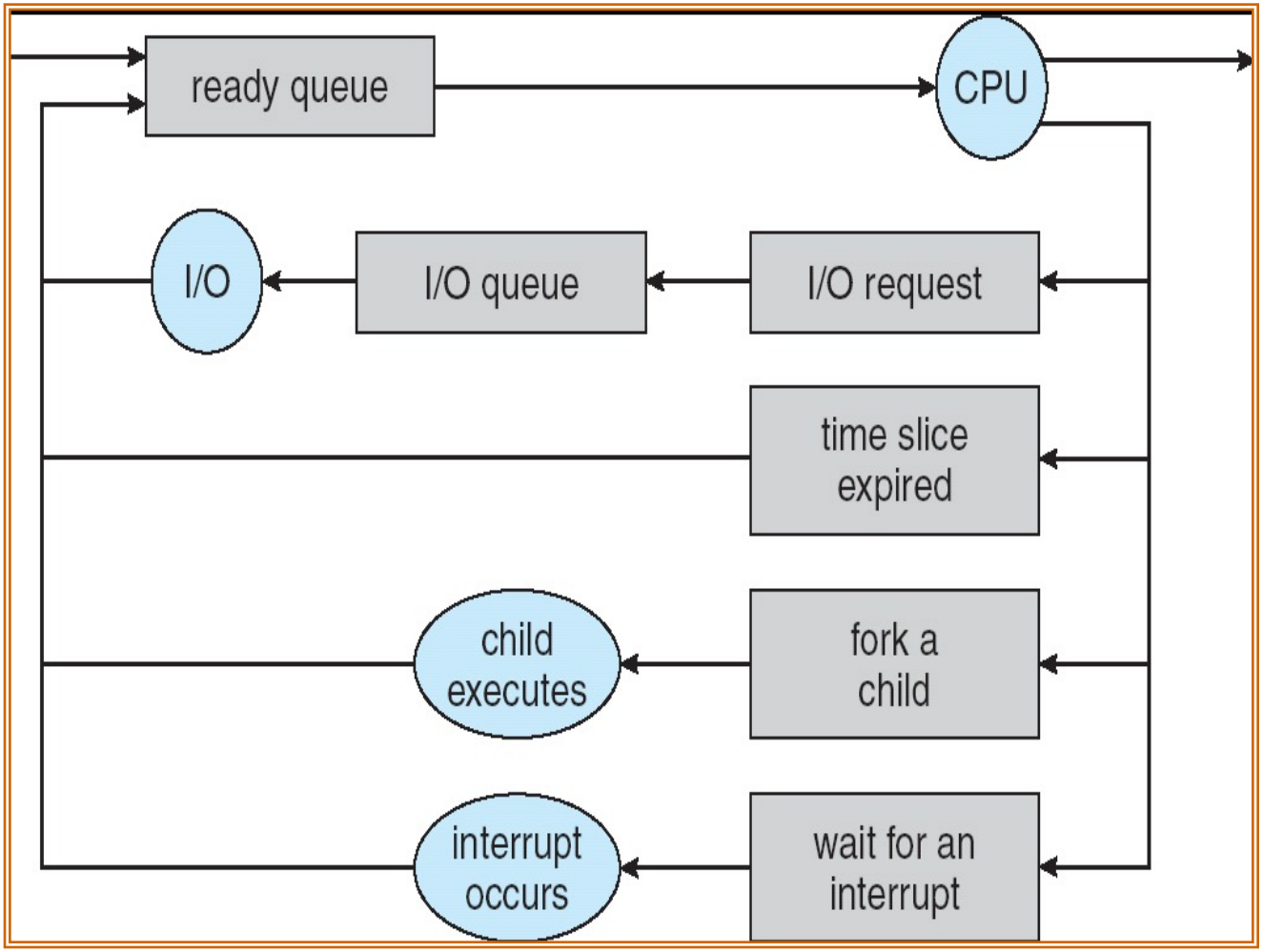
- Modifique o programa alocar_cpu.cpp para que ele possa executar nas CPUs 0 e 2.

Alocação de CPUs

- Exercício:
  - Execute novamente o programa alocar_cpu.
  - Num outro terminal execute o comando
  - $ taskset -p <PID do alocar_cpu>
  - $ taskset -pc 0-2 <PID do alocar_cpu>

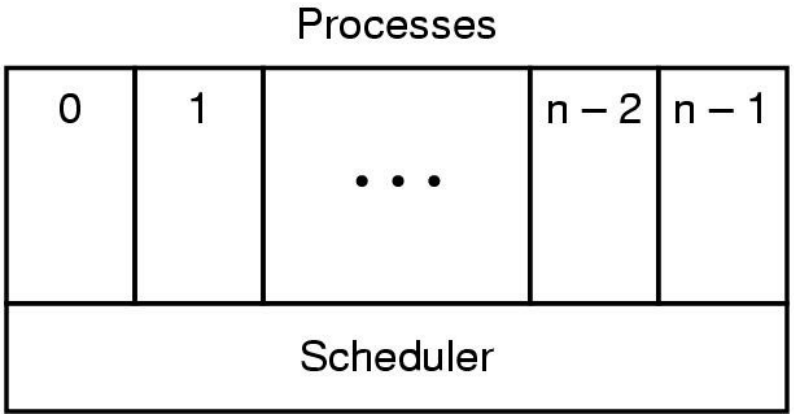
- Exercício:
  - Comente as linhas associadas com alocação de CPU do programa alocar_cpu.cpp. Salve e compile novamente.
    - g++ alocar_cpu.cpp -o alocar_cpu
    - $ taskset -c 2-3 ./alocar_cpu
  - Analise o resultado

Mecanismo de Escalonamento de Processos



Escalonamento de Processos

- O escalonador é a entidade do sistema operacional responsável por selecionar um processo apto para executar no processador.
  - Algoritmo que determinar qual processo irá ocupar a CPU.
  - Esse algoritmo deve seguir uma política justa.

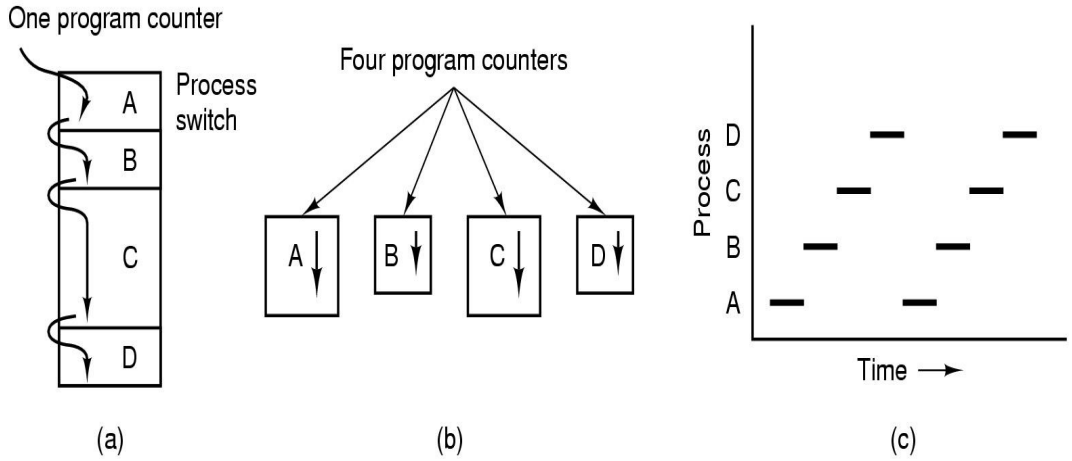


Objetivo do Escalonador

- Maximizar o uso do processador
- Maximizar o throughput
  - Número de processos executados por unidade de tempo
- Minimizar o turnaround
  - Tempo total para executar um determinado processo
- Minimizar o tempo de espera
  - Tempo que um processo permanece na fila de pronto
- Minimizar o tempo de resposta
  - Tempo transcorrido entre sua requisição e a sua realização

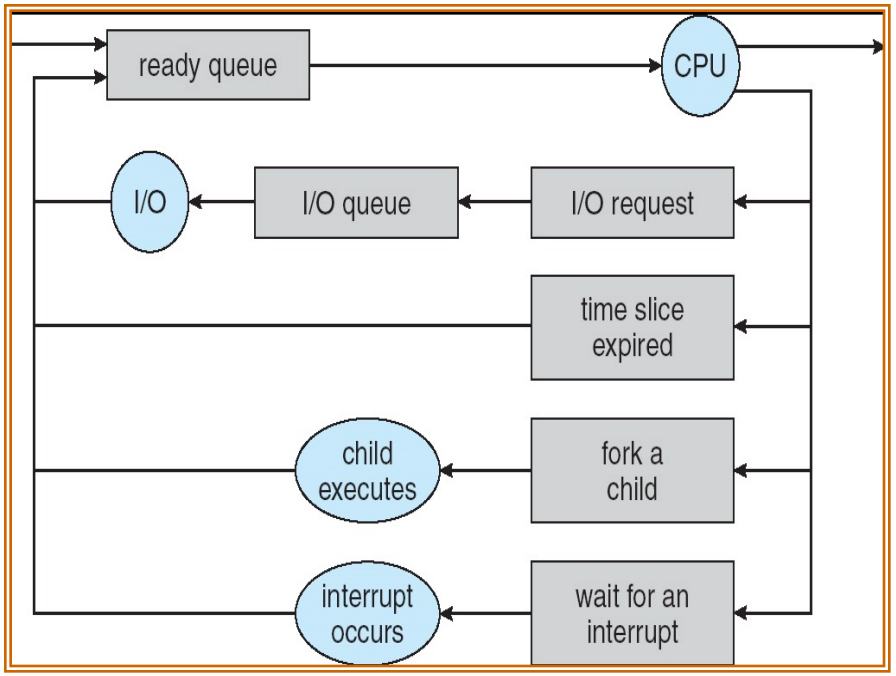
Tipos de Escalonadores

- Batch
- Interativos
  - Tempo de resposta
- De Tempo-real
  - Garantir deadlines



Características dos Escalonadores

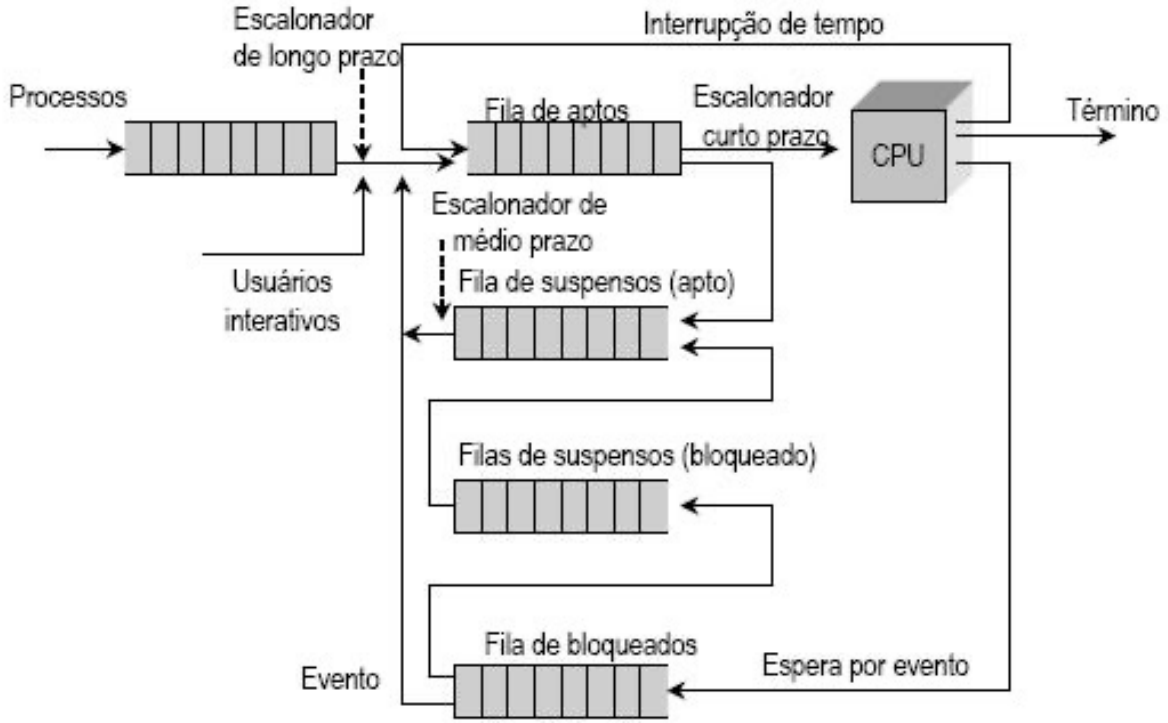
- Não-Preemptivos
  - Término do processo
  - I/O, Sincronização ou Erro
  - Liberação voluntária da CPU



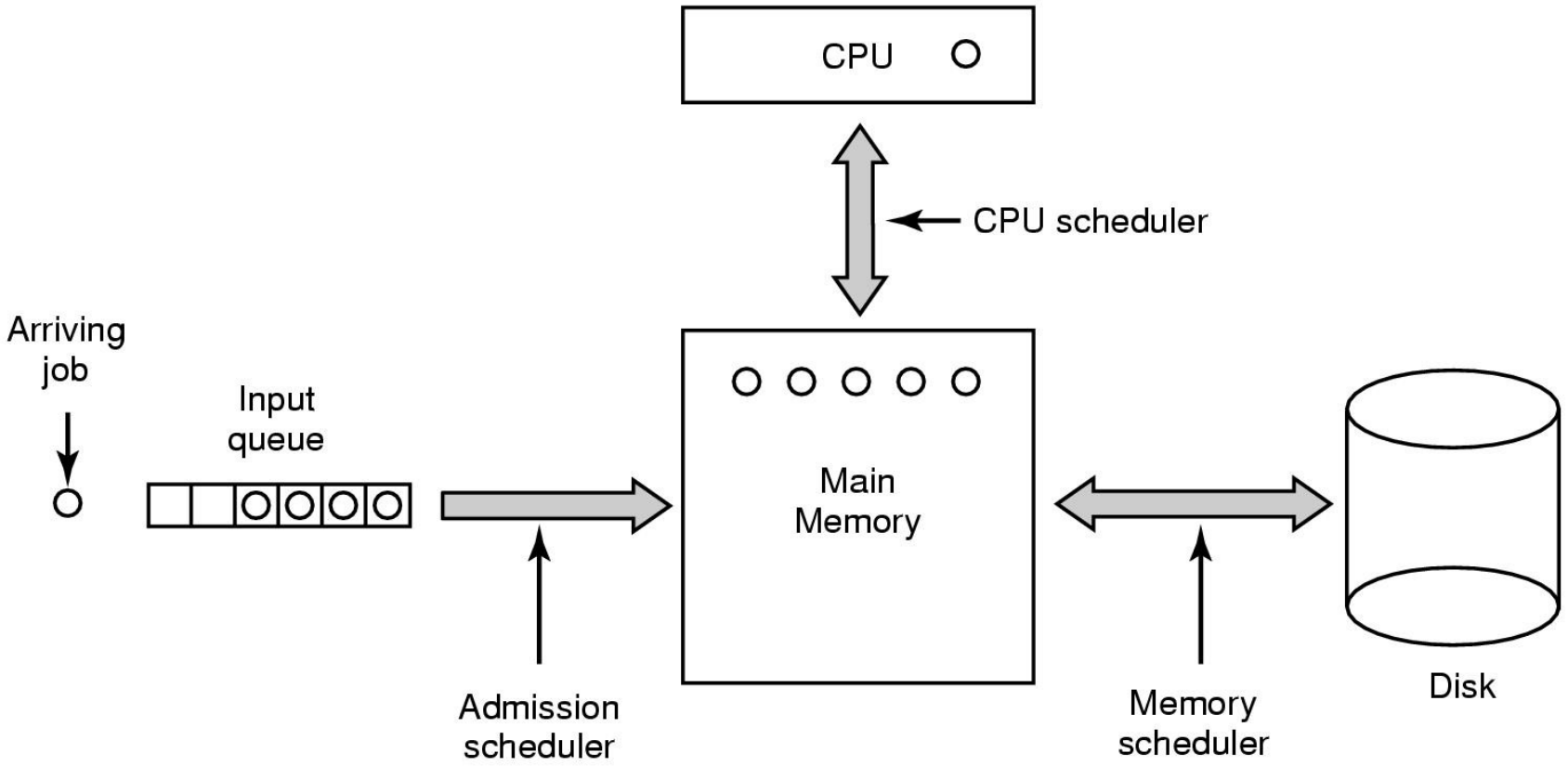
- Preemptivos (além das anteriores)
  - Término do processo
  - Interrupção de relógio (*slice time*)
  - Interrupção devido à existência de outro processo apto de maior Prioridade

Níveis de Escalonamento

- Curto Prazo
- Médio Prazo
- Longo Prazo

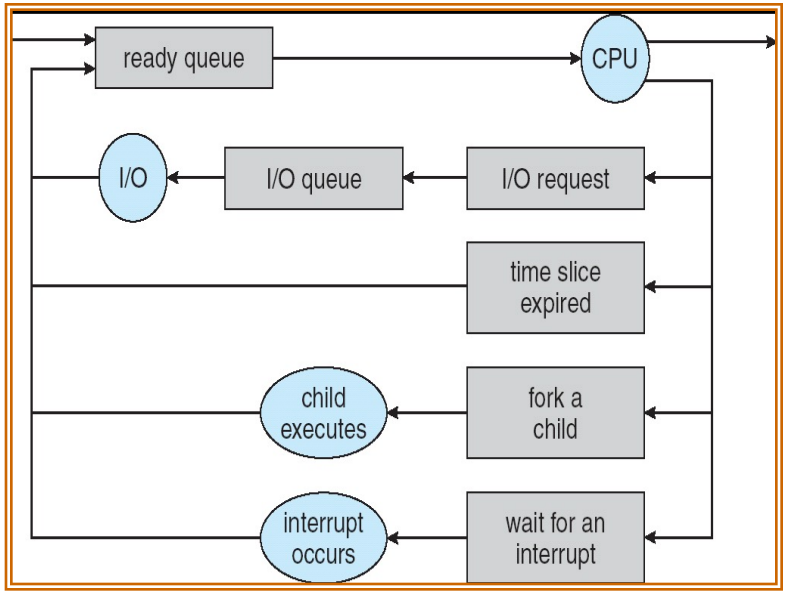


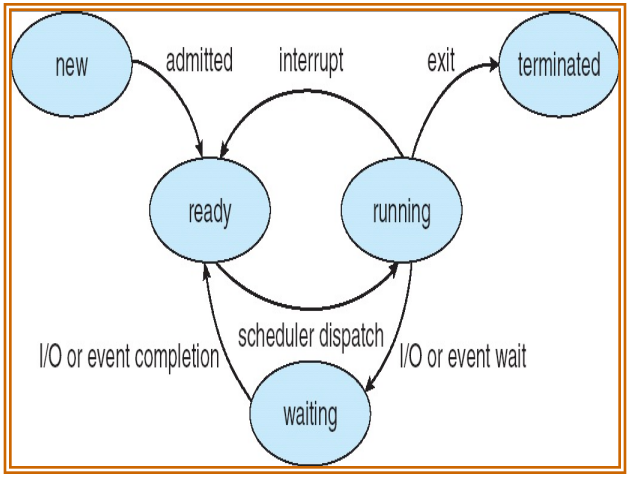
Os Três Níveis de Escalonamento



Escalonamento Round Robin (RR)

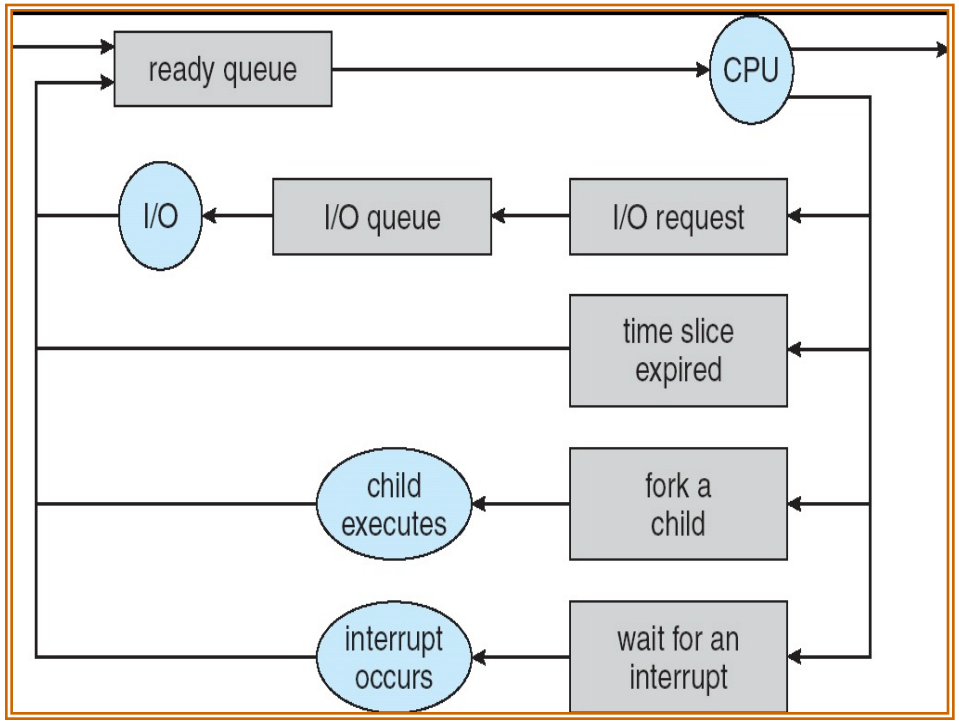
- Algoritmo preemptivo
- Similar ao FIFO, porém:
  - Cada processo recebe um tempo máximo (*time slice*, *quantum*) de CPU por vez.
    - Usualmente, quantum (q) entre 10-100ms
  - Fila de processos aptos é uma fila circular.
    - Interrupção de relógio

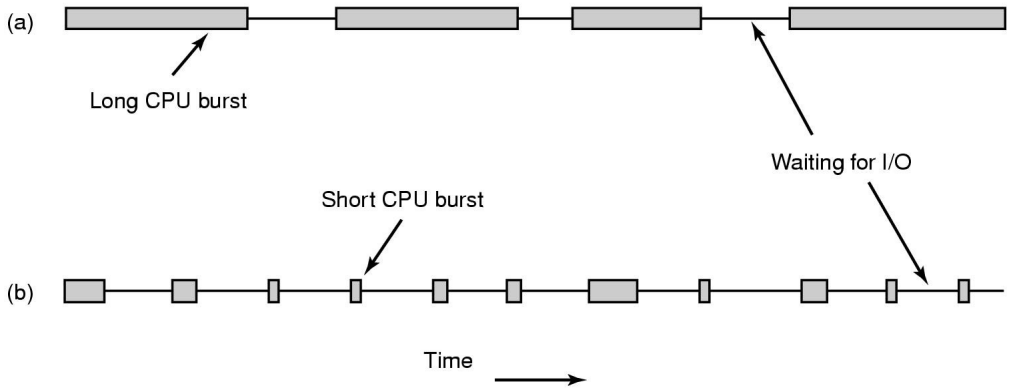




Escalonamento por Prioridade

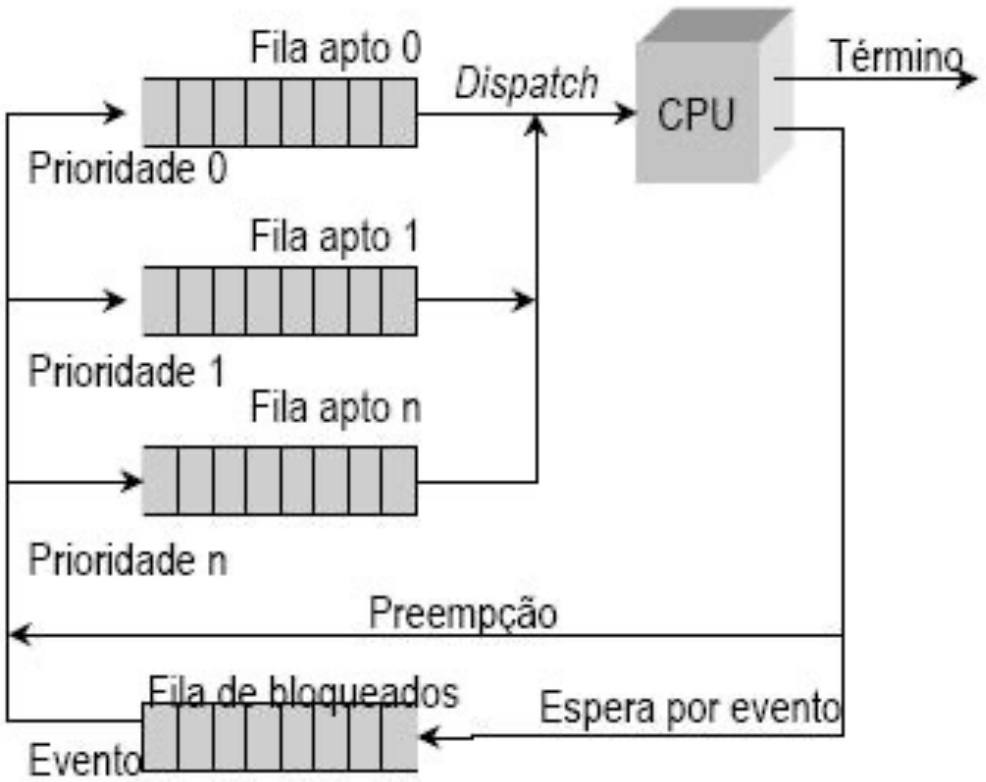
- O Round Robin tende a prejudicar os processos I/O bound, pois estes provavelmente não utilizam todo o seu quantum de tempo.





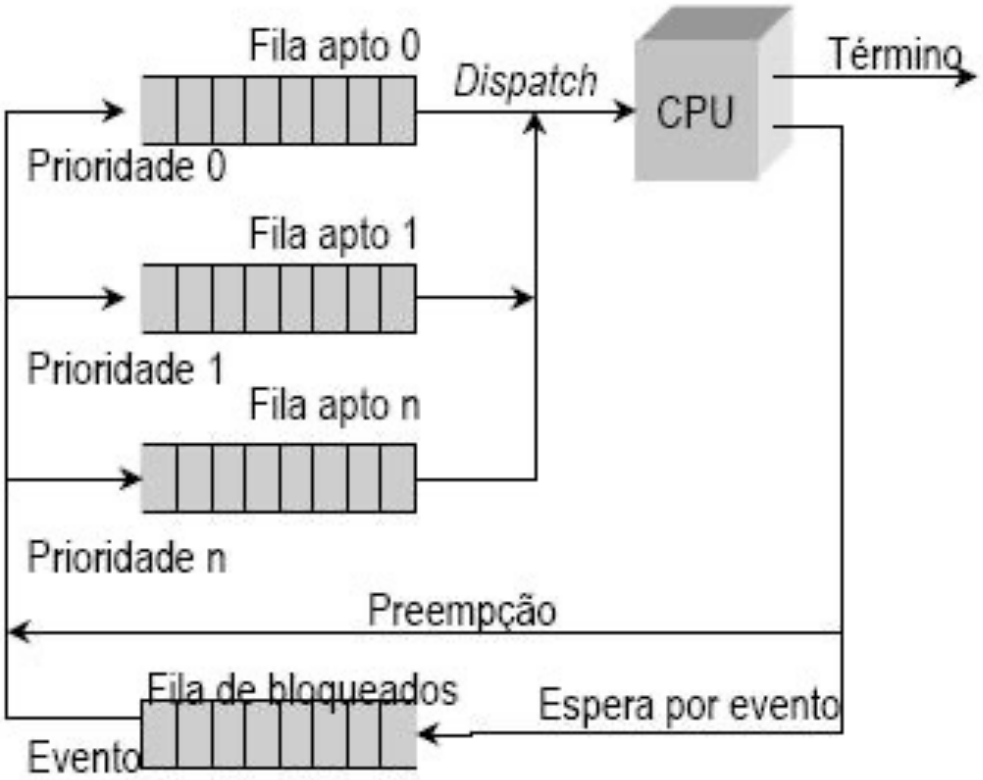
Escalonamento por Prioridade

- Como minimizar este problema?
  - Utilizar mecanismos de prioridade para os processos I/O bound.
  - Múltiplas filas de prioridade



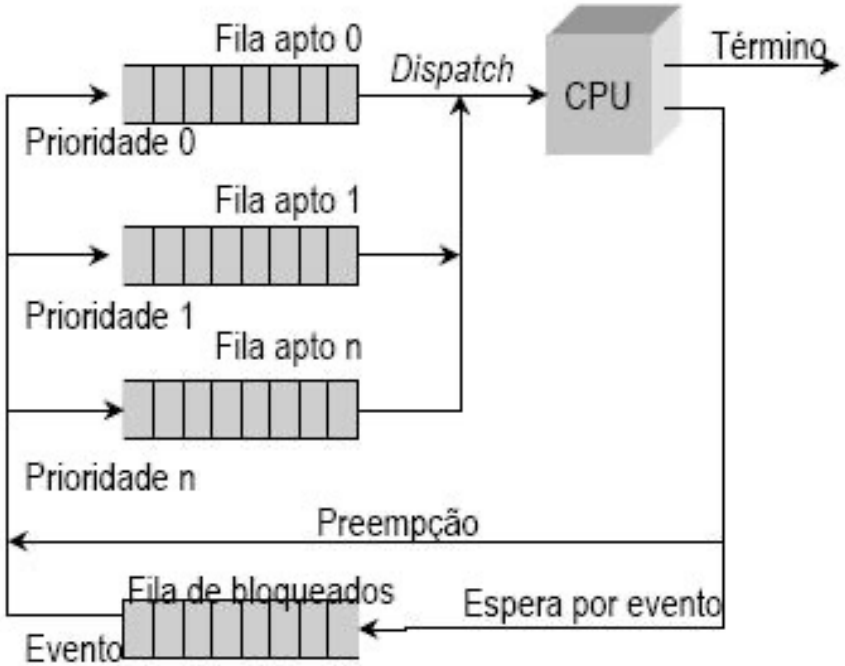
Implementação de Prioridade

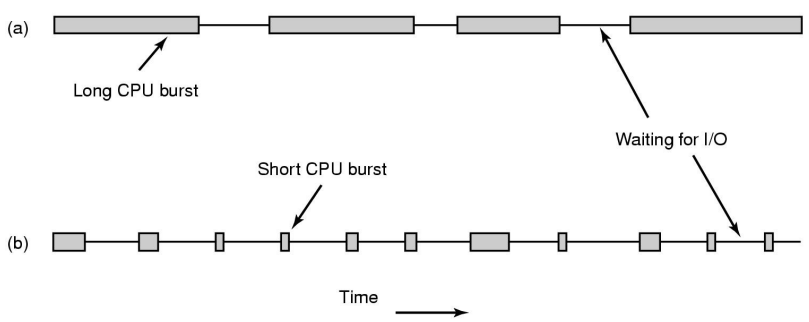
- Múltiplas filas
  - Cada fila tem uma prioridade
  - Cada fila pode ter sua própria política de prioridade
    - RR, FIFO, SJF, etc.



Implementação de Prioridade

- Prioridade Estática
  - Ao ser criado, é atribuída uma prioridade ao processo, que será a mesma ao longo de sua vida.
- Pode haver *starvation* em processos com baixa prioridade
- A prioridade estática pode prejudicar ou beneficiar determinados processos.
  - Processo pode seu perfil de uso de CPU e I/O modificado ao longo de sua execução.





Implementação de Prioridade - Unix

- Múltiplas filas com realimentação, como RR em cada fila.
- As prioridades são reavaliadas a cada segundo em função de:
  - Prioridade atual
  - Prioridade do usuário
  - Tempo recente de uso da CPU
  - Fator nice
- Prioridades são divididas em faixas de acordo com o tipo do usuário.
- A troca dinâmica das prioridades respeita os limites da faixa

Implementação de Prioridade - Unix

- Prioridade recebem valores entre 0 e 127 (menor o valor, maior a prioridade)
  - 0-49 → Processos do núcleo
  - 50-127 → Processos do usuário
- Ordem decrescente de prioridade
  - Swapper
  - Controle de dispositivos de entrada e saída orientados a blocos
  - Manipulação de arquivos
  - Controle de dispositivos de entrada e saída orientados a caractere
  - Processos de usuário

Implementação de Prioridade - Unix

- Cálculo de prioridade do processo de usuário
  - Fator nice: valor variando entre 0 (mais prioritário) a 39 (menos prioritário)
    - O default é 20.
  - Uso recente da CPU

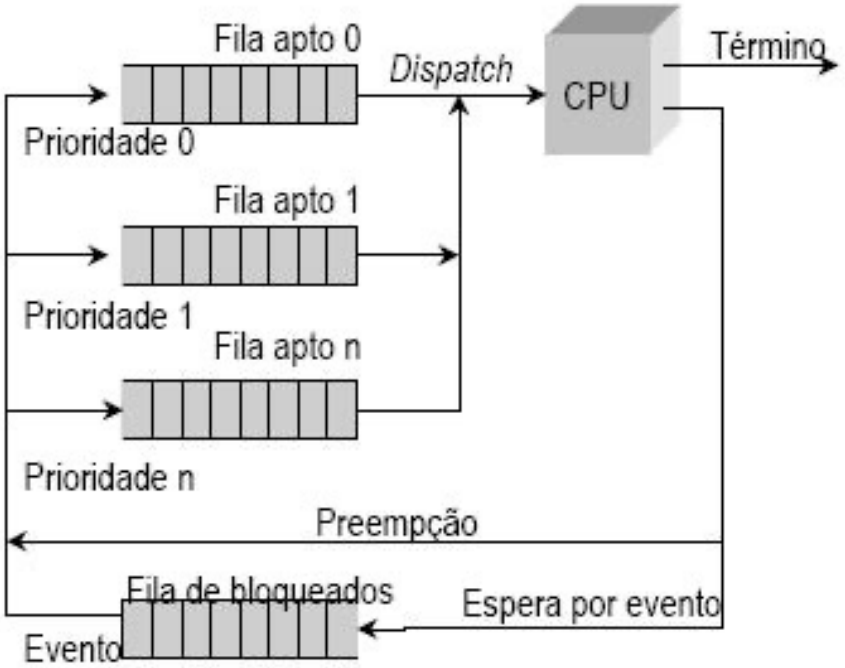
$$decay = \frac{2 \times load\_average}{(2 \times load\_average + 1)}$$

$$p\_usrpri = PUSER + \frac{p\_cpu \times decay}{4} + 2 \times p\_nice$$

  - Sendo,
    - *load_average* é o número médio de processos aptos no último segundo.
    - PUSER é valor de base de prioridade para usuários (50)
  - Processos do usuário → Prioridade de 50 a 127

Implementação de Prioridade

- Prioridade dinâmica
  - A prioridade é ajustada de acordo com o estado de execução do processo ou sistema.
  - Múltiplas Filas com Realimentação.
    - Um processo pode ser movido entre as diversas filas, de acordo com alguma política pré-estabelecida.



Implementação de Prioridade - Linux

- Há duas classes em função do tipo de processos/threads
  - Processos interativos e batch
  - Processos de tempo-real
- Políticas de escalonamento – Padrão Posix
  - SCHED_FIFO: FIFO com prioridade estática
    - Válido apenas para processos de tempo real
  - SCHED_RR: Round-robin com prioridade estática
    - Válido apenas para processos de tempo-real
  - SCHED_OTHER: Filas multinível com prioridade dinâmicas (time-sharing)
    - Processos interativos e batch

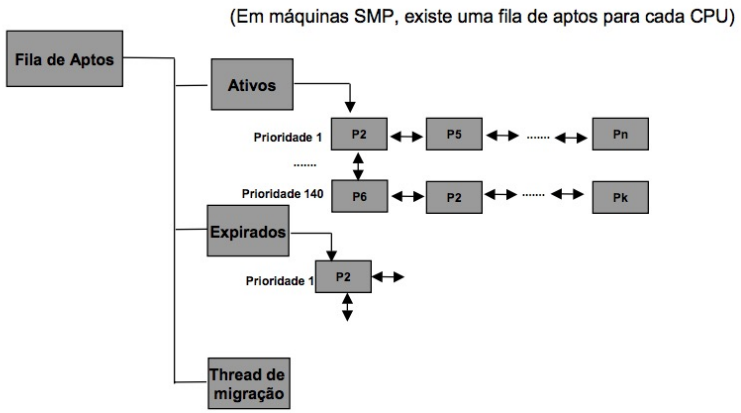
Implementação de Prioridade – Linux

- Processos de Tempo-real
  - Tempo real-soft
  - Prioridade fixa e definida pelo usuário ou por outros com privilégios especiais.
  - Têm maior prioridade do que os interativos e os batchs.
  - Processos com maior prioridade sempre executam primeiro
    - SCHED_FIFO ou SCHED_RR

Implementação de Prioridade – Linux 2.6

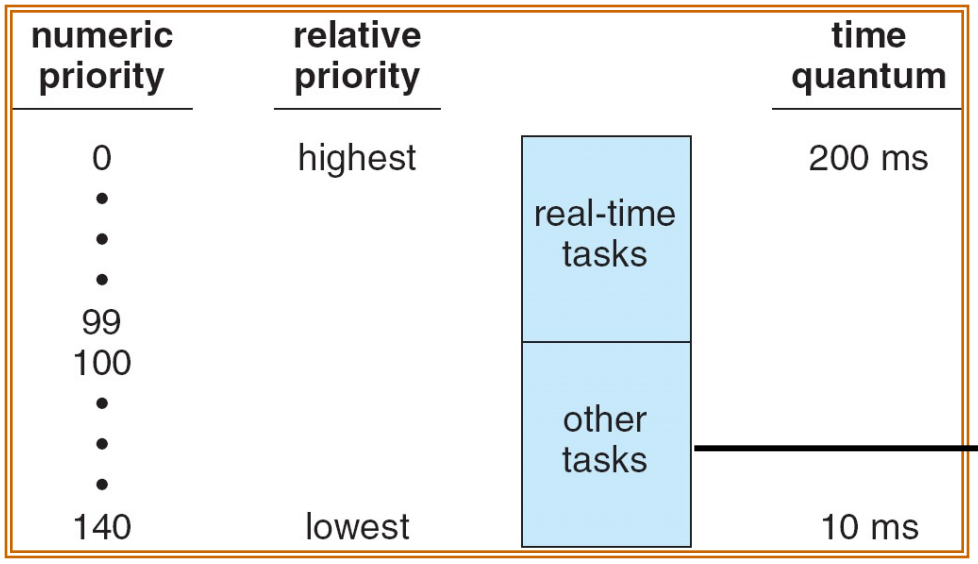
- Processos de Não Tempo-Real: Prioridade dinâmica baseada em créditos
  - O processo com maior crédito é o selecionado
  - A cada interrupção de tempo, o processo em execução perde um crédito.
  - Ao zerar o seu crédito, o processo é suspenso.
  - Se na fila de aptos não houver processos com créditos, é realizada uma distribuição de créditos para todos os processos (em todas as filas)
    - Créditos = 0.5*créditos + prioridade

Escalonamento no Kernel 2.6

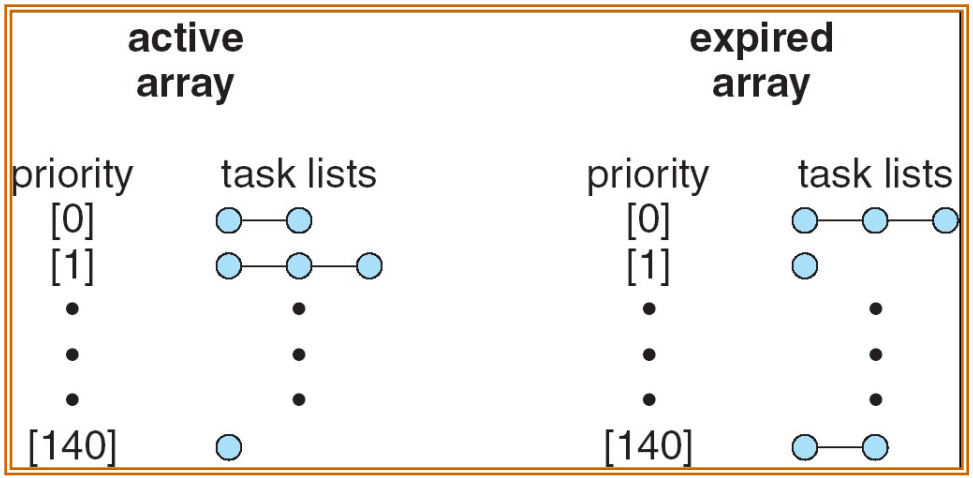


Quando o array Ativos se torna vazio, o array Expirados passa a ser o ativo, com uma simples troca de apontadores.

Implementação de Prioridade - Linux



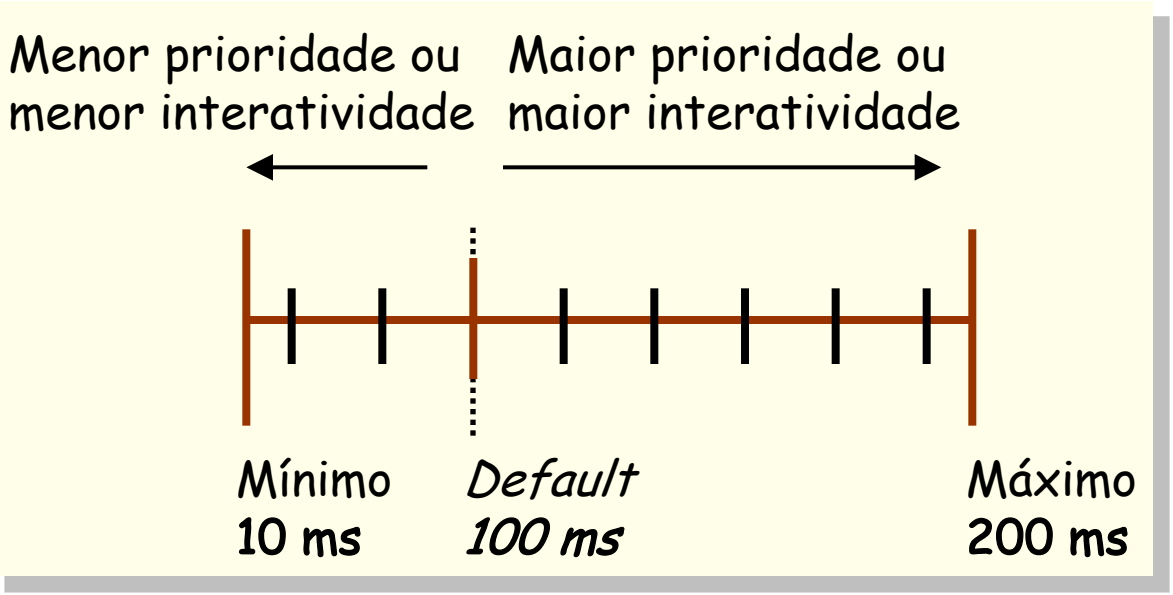
O padrão é 120



Implementação de Prioridade - Linux

Características principais do mecanismo de prioridades do *Linux*:

- O *timeslice* destinado a cada processo é calculados com base na sua **prioridade dinâmica**;



- processos **filhos** ganham a metade do tempo restante no *timeslice* do seu pai.

Implementação de Prioridade - Linux

Características principais do mecanismo de prioridades do *Linux*:

- Duas escalas de prioridade distintas: ***nice values*** e ***real-time priorities.***
  - *nice values*: -20 a 19
  - *real-time*: 1 a *MAX_RT_PRIO-1 = 99*

- Tanto o usuário quanto o sistema influenciam na determinação das prioridades dos processos.
- Procura beneficiar processos ***interativos.***
  - *task_struct.sleep_avg*
  - *0 ---- MAX_SLEEP_AVG* = 10 ms.
- **Real-time priorities** implementadas de acordo com o padrão ***POSIX.***

Prioridade de Processos

- Exercício:
  - Compile e execute o programa prioridade.cpp
    - g++ prioridade.cpp –o prioridade
    - ./prioridade
  - Em outro terminal, execute o programa Htop (ou top) ou o comando ps:
    - ps -o uid,pid,ppid,pri,ni,cmd
    - ps -o uid,pid,ppid,pri,ni,comm. <-no Mac
  - Analise o resultado
    - Com relação à prioridade do processo "prioridade"
  - Rode o programa com:
    - nice -n12 ./prioridade
    - Analise o resultado.

Prioridade de Processos

- Exercício:
  - Excecute o programa prioridade:
    - ./prioridade
  - Em outro terminal, execute o comando renice
    - sudo renice 0 -p <PID>
    - Analise o resultado
  - Execute o programa prioridade com:
    - nice -n12 ./prioridade
    - Analise o resultado.
  - Descomente as linhas associadas à função nice() no programa prioridade.cpp.
    - Compile novamente e execute o programa.
    - Analise os resultados.

Prioridade de Processos

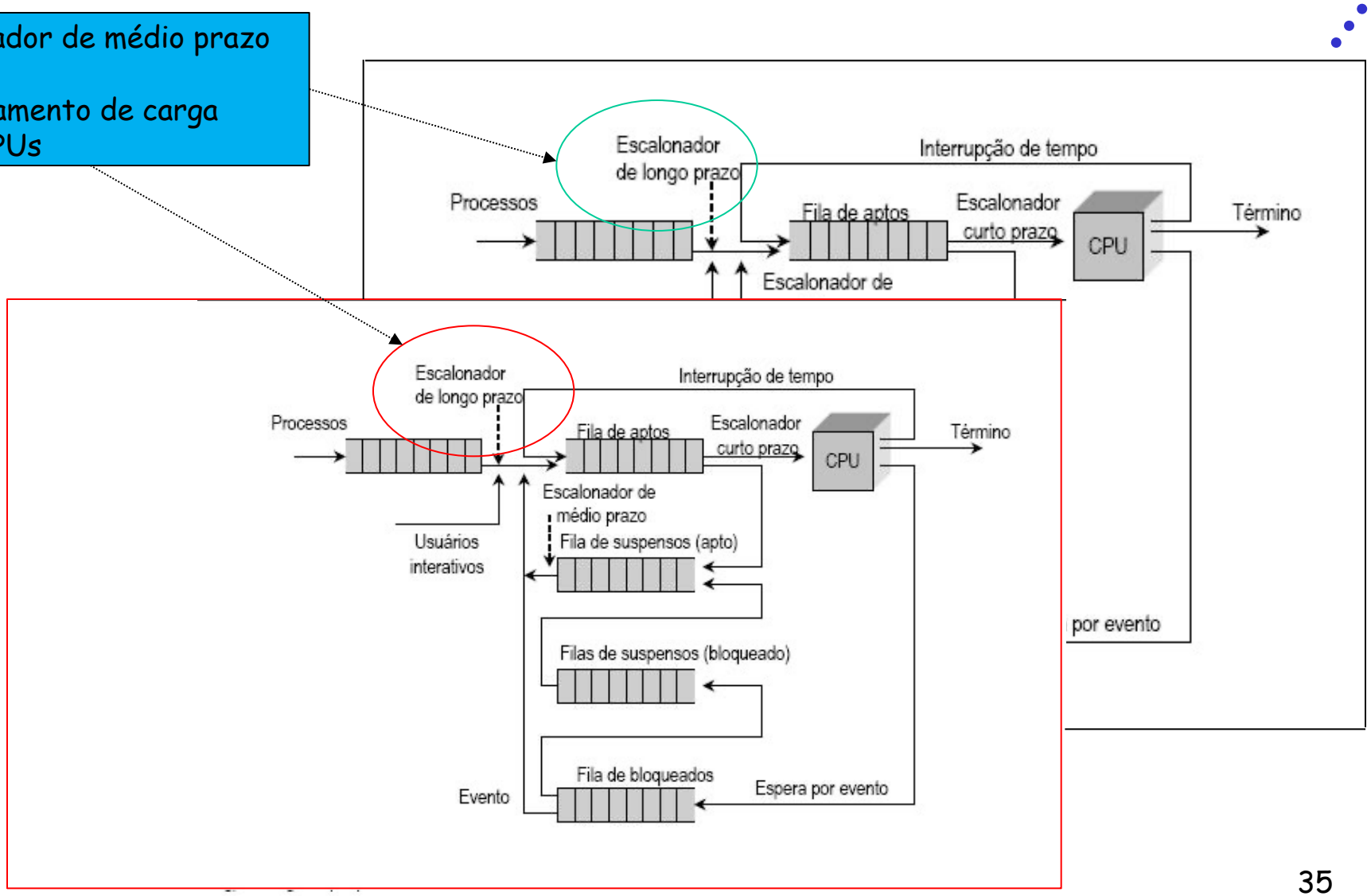
- Exercício:
  - Excecute o programa prioridade:
    - ./prioridade
  - Em outro terminal, execute o comando renice
    - sudo renice 0 -p <PID>
    - Analise o resultado
  - Execute o programa prioridade com:
    - nice -n12 ./prioridade
    - Analise o resultado.
  - Descomente as linhas associadas à função nice() no programa prioridade.cpp.
    - Compile novamente e execute o programa.
    - Analise os resultados.

Balanceamento de Carga de CPU

- Contexto de multi-core

Escalonador de médio prazo

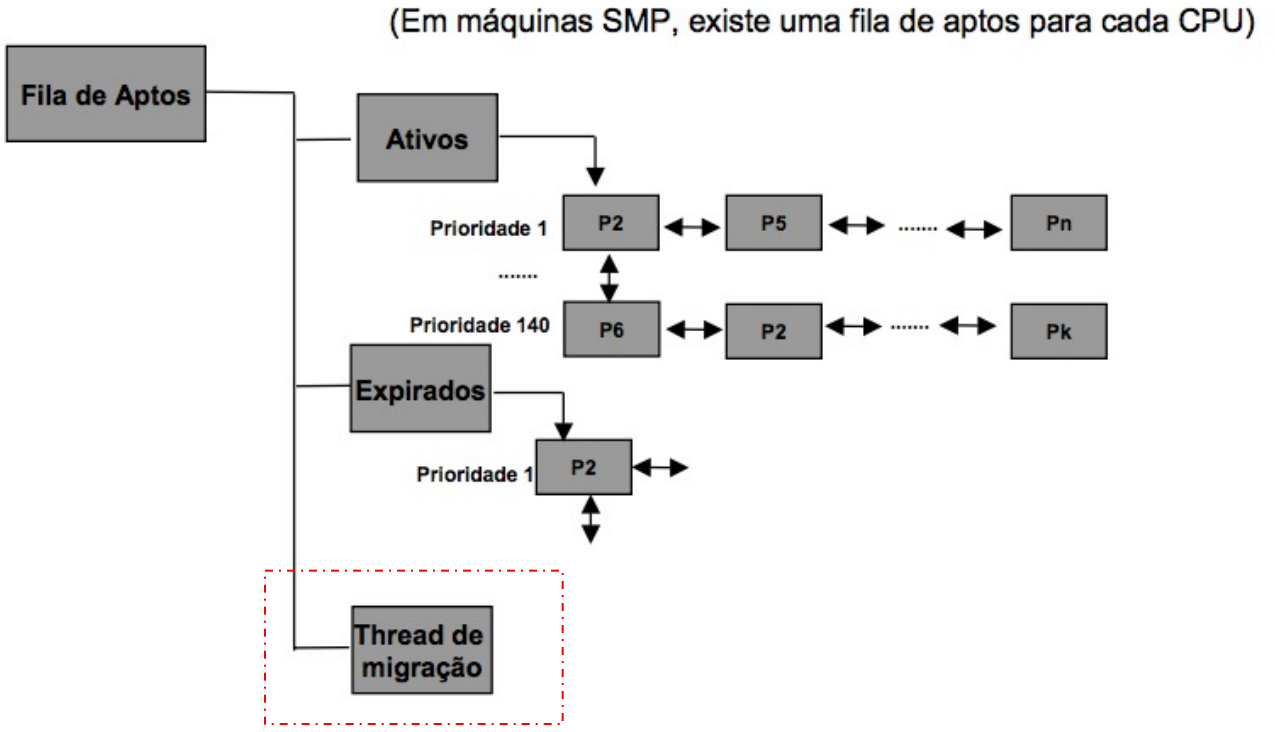
Balanceamento de carga entre CPUs



Balanceamento de Carga de CPU

- Contexto de multi-core

Escalonamento no Kernel 2.6



Quando o array Ativos se torna vazio, o array Expirados passa a ser o ativo, com uma simples troca de apontadores.

Conceitos de Prioridade e Alocação de CPU de Processos