Threads e Escalonamento

Threads

- Núcleo do Linux trata processos como tarefas.
- Threads são segmentos de execução de processos.
- Um processo pode ter uma ou várias threads.
- Antigamente:
  - processo era recipiente de recursos e threads eram unidades de execução
- Atualmente (2000):
  - Chamada ao sistema *clone*

Threads

pid = clone(function, stack_ptr, sharing_flags, arg)

Flag	Significado quando marcado	Significado quando limpo
CLONE_VM	Cria um novo thread	Cria um novo processo
CLONE_FS	Compartilha umask e os diretórios-raiz e de trabalho	Não compartilha umask e os diretórios-raiz e de trabalho
CLONE_FILES	Compartilha os descritores de arquivos	Copia os descritores de arquivos
CLONE_SIGHAND	Compartilha a tabela do tratador de sinais	Copia a tabela do tratador de sinais
CLONE_PARENT	O novo thread tem o mesmo pai que o chamador	O chamador é o pai do novo thread

Escalonamento de Threads

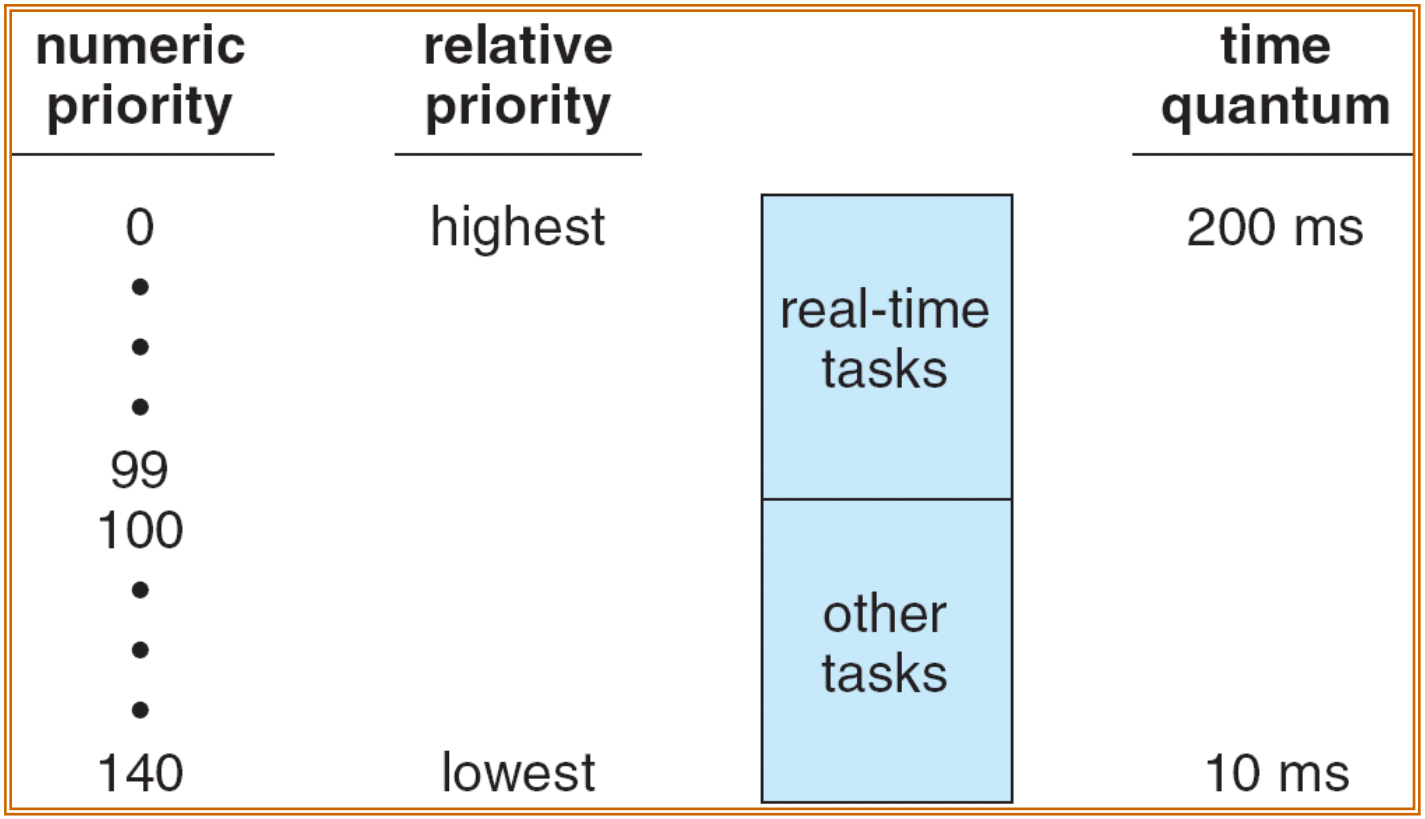
- Escalonamento no Linux é baseado em threads e não processos.
- Cada thread tem uma prioridade de escalonamento associada a si, com números maiores indicando prioridade mais baixa.
- A realimentação negativa no escalonamento de CPU torna difícil para uma única thread monopolizar o tempo de CPU.
- Envelhecimento da thread é empregado para prevenir que uma thread sofra inanição.
- Quando uma thread escolhe pela liberação da CPU, ela deve ser bloqueada em um evento.

Escalonamento de Threads

- Linux distingue três classes de threads para questões de escalonamento:
  - FIFO em tempo real
    - Tem garantia de ser selecionado para executar antes de qualquer processo kernel ou de tempo compartilhado.
    - Pode preemptar processos em kernel ou usuário.
  - Chaveamento circular em tempo real
    - Tem garantia de ser selecionado para executar antes de qualquer processo de tempo compartilhado.
  - Tempo compartilhado
    - Prioridade mais baixa

Escalonamento de Threads

- Threads em tempo real possuem prioridade de 0 a 99.
- Threads convencionais possuem prioridade de 100 a 139



Escalonamento de Threads

- A realimentação multinível usando FIFO
- Se uma thread em execução não bloquear ou completar dentro de 1s, é preemptado.
- As prioridades são recomputadas uma vez por segundo.
- O contador é incrementado a cada tique de relógio para a thread que está executando.

Exemplo do escalonamento de tempo-real do Linux

A	minimum
B	middle
C	middle
D	maximum

(a) Relative thread priorities

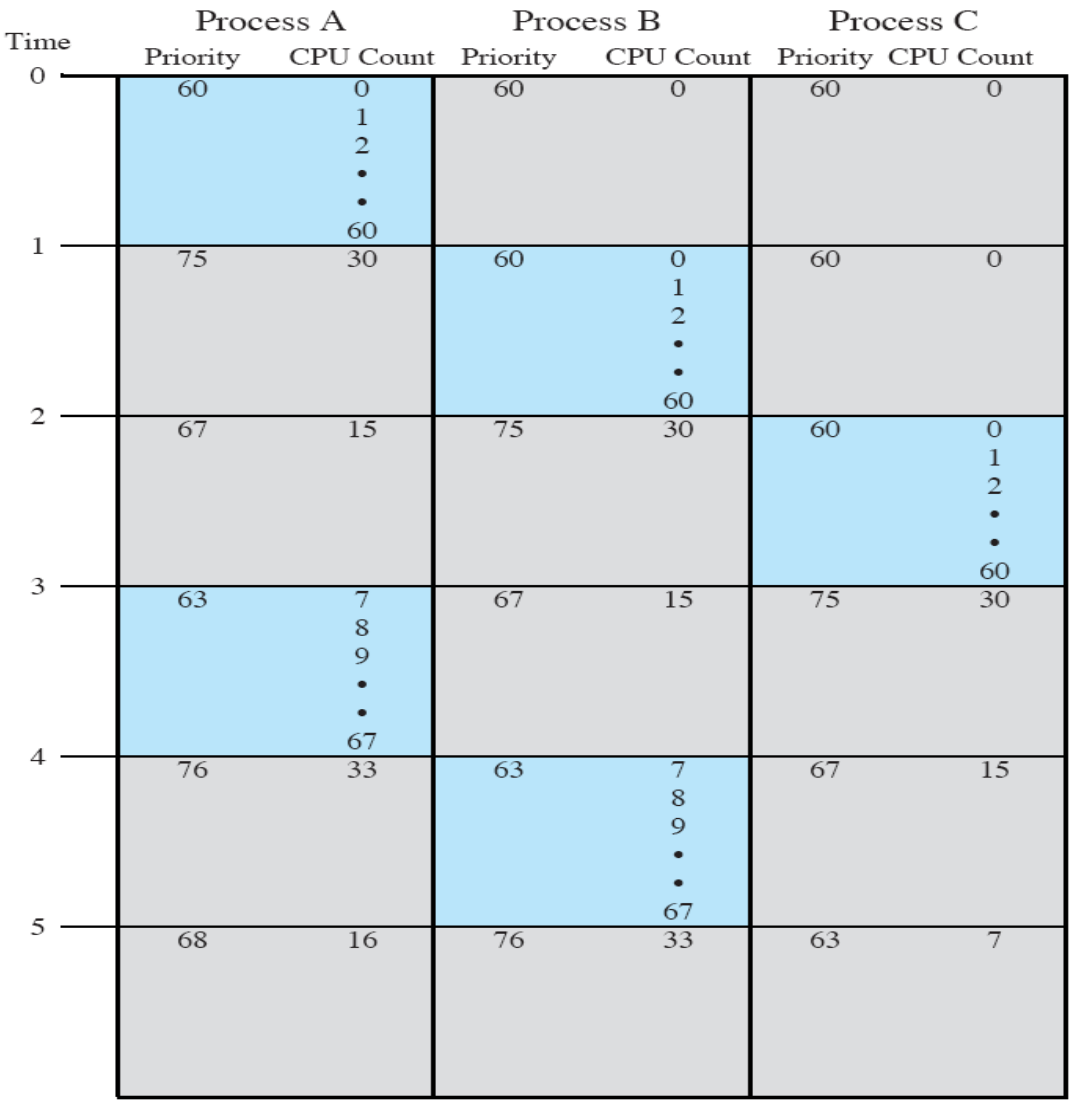
(b) Flow with FIFO scheduling

(c) Flow with RR scheduling

Escalonamento de Threads

- Recômputo:
  - contador = contador/2;
  - prioridade = prioridade_base + *nice* + contador/2;
- Um valor numericamente baixo implica em uma prioridade de escalonamento maior.
- O valor de nice é reajustado entre -20 e +20 (default 0).

Exemplo do escalonamento tradicional do Unix



Time	Process A		Process B		Process C	
	Priority	CPU Count	Priority	CPU Count	Priority	CPU Count
0	60	0 1 2 • • 60	60	0	60	0
1	75	30	60	0 1 2 • • 60	60	0
2	67	15	75	30	60	0 1 2 • • 60
3	63	7 8 9 • • 67	67	15	75	30
4	76	33	63	7 8 9 • • 67	67	15
5	68	16	76	33	63	7

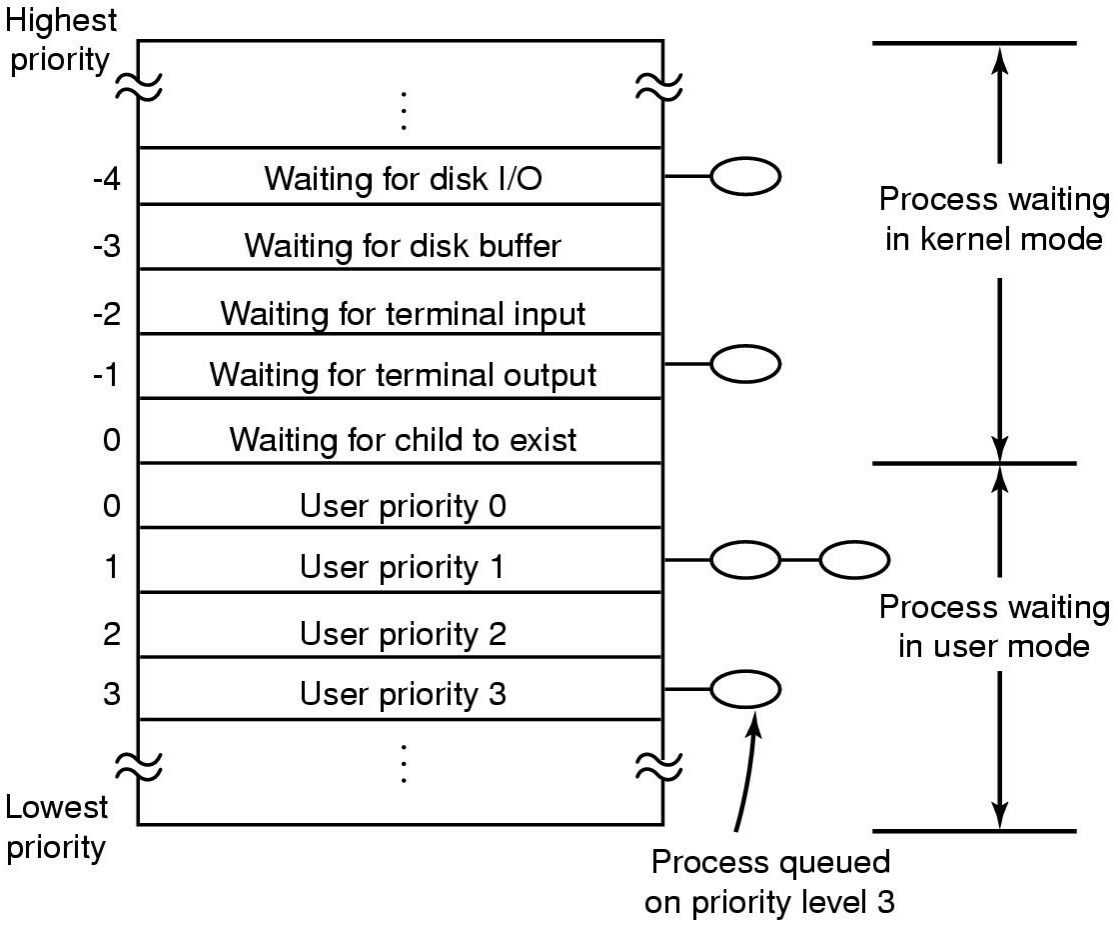
Colored rectangle represents executing process

Bandas

- A prioridade de base divide todas as threads em bandas fixas de níveis de prioridade.

- Ordem decrescente de prioridade
  - Swapper ou permutador
  - Controle de dispositivo de E/S de bloco
  - Tratamento de arquivos
  - Controle de dispositivo de E/S de caractere
  - Processos de usuário

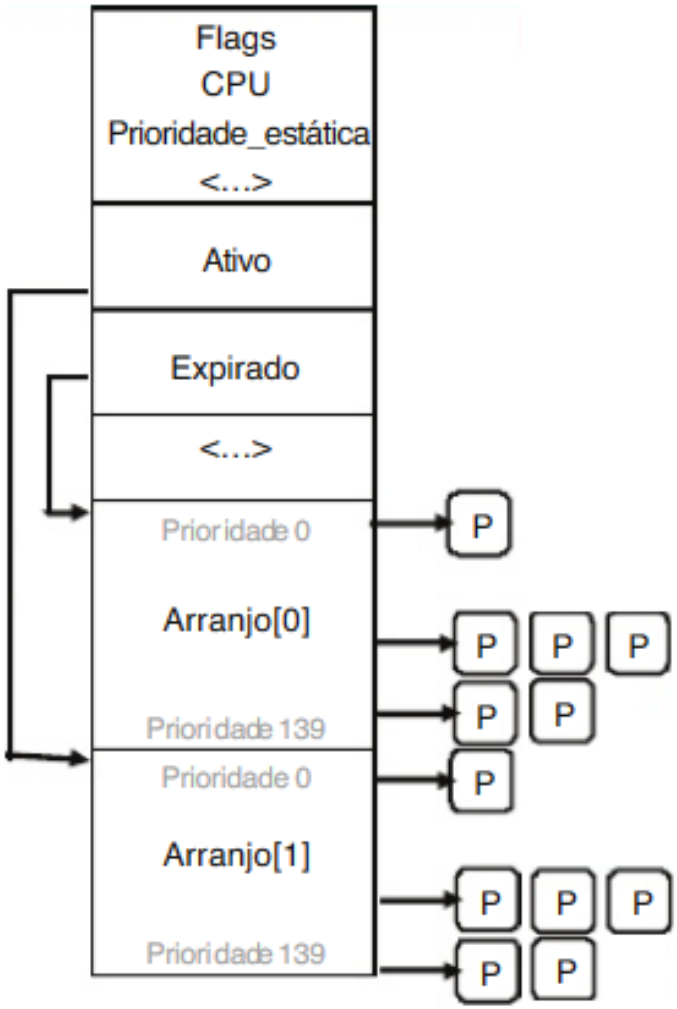
Escalonador do UNIX



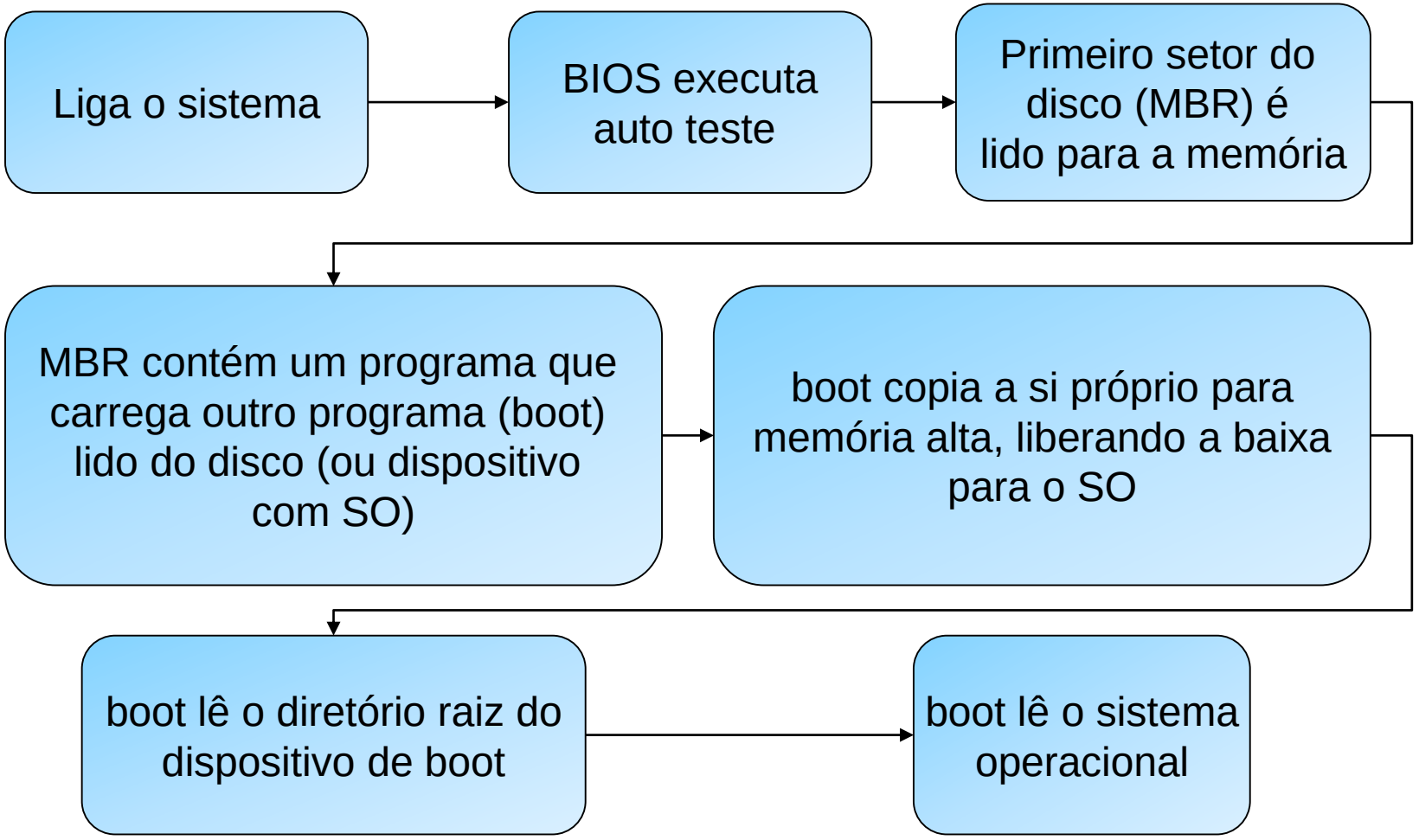
Baseado em estrutura de filas multinível

Escalonamento

- Uma estrutura de dados essencial utilizada pelo escalonador é a fila de execução.



Inicialização Linux



Inicialização Linux

- Código de disparo do núcleo é escrito em assembly = *hardware*.
- Código em C também realiza a inicialização = *software*