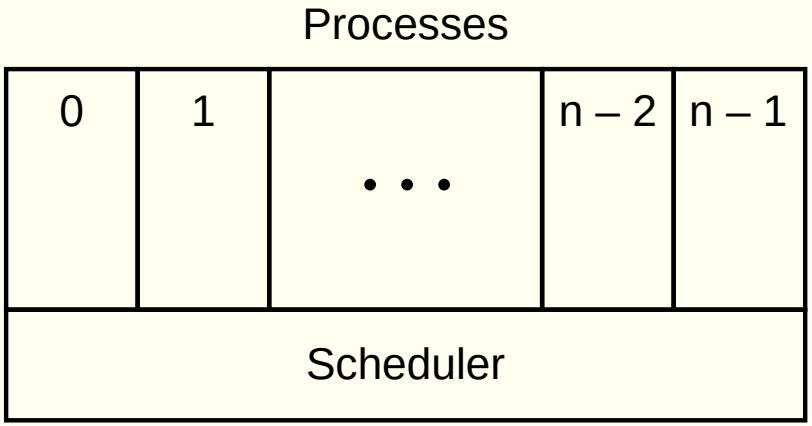
MC514–Sistemas Operacionais: Teoria e Prática

1s2008

Escalonamento de Processos e Threads

Escalonador



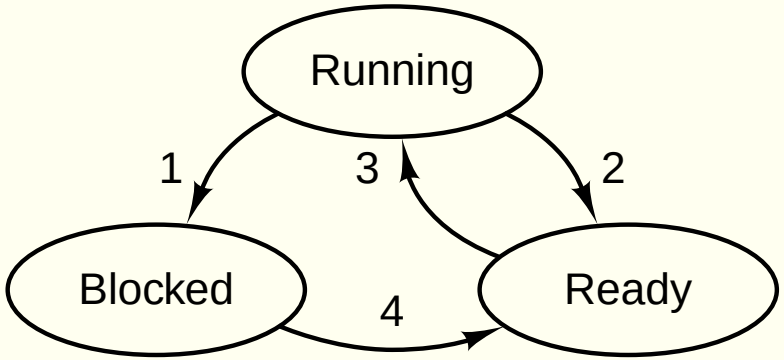
Tanenbaum: Figura 2.3

A função do escalonador é escolher qual deve ser o próximo processo a ser executado.

Quando escalonar

- Quando um processo é criado
- Quando um processo termina
- Quando um processo faz uma operação de I/O
- Interrupção de relógio (sistemas preemptivos)

Estado dos processos



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Tanenbaum: Figura 2.2

Por que algumas arestas estão faltando?

Campos gerenciados por processo

- Gerência de processos: registradores, contador de programa, *program status word*, estado, prioridades, identificador de processos, sinais
- Gerência de memória: apontadores para os segmentos de dados, texto e pilha.
- Gerência de arquivos: diretório raiz e corrente, descritores de arquivos, identificadores de usuário e grupo

$*$ Quais recursos são compartilhados pelas threads?

Mudança de contexto

1. Hardware stacks program counter, etc.
2. Hardware loads new program counter from interrupt vector.
3. Assembly language procedure saves registers.
4. Assembly language procedure sets up new stack.
5. C interrupt service runs (typically reads and buffers input).
6. Scheduler decides which process is to run next.
7. C procedure returns to the assembly code.
8. Assembly language procedure starts up new current process.

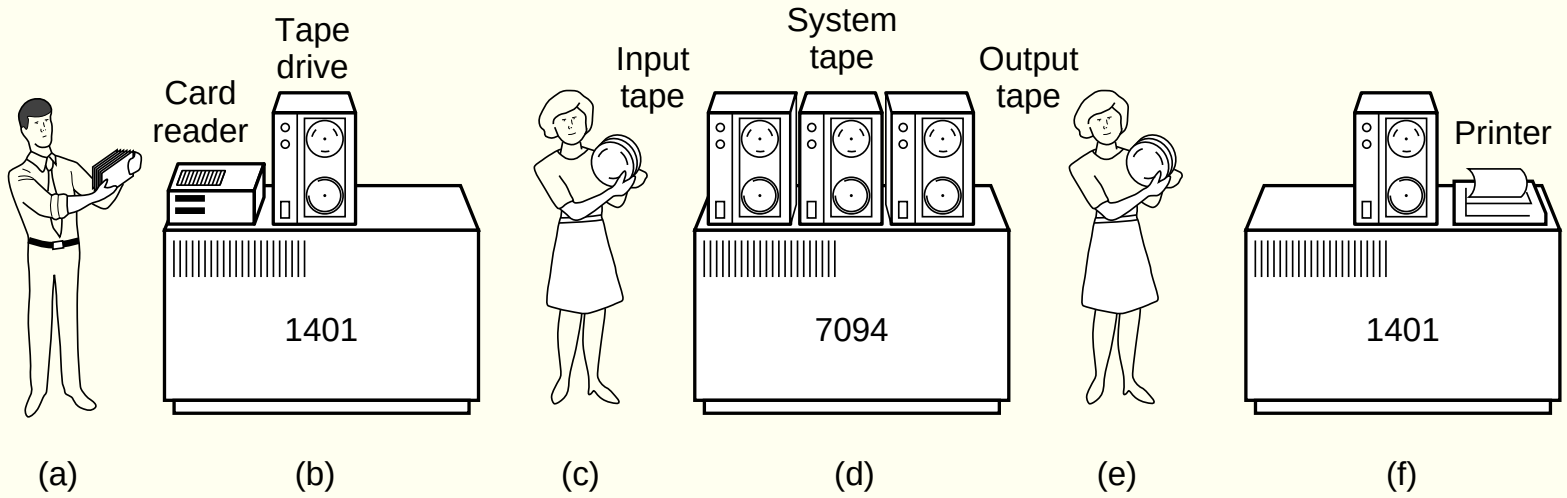
Tanenbaum: Figura 2.5

Vamos verificar em .../linux-2.6.24.2/kernel/sched.c? ;-)

Objetivos dos algoritmos de escalonamento

- Justiça
  - Cada processo deve receber a sua parte da CPU
- *Policy enforcement*
  - Respeito às políticas estabelecidas
- Equilíbrio
  - Todas as partes do sistema devem estar operando

Escalonamento em sistemas batch



Tanenbaum: Figura 1.2

Escalonamento em sistemas batch

- *Throughput*
  - o número de jobs por hora deve ser maximizado
- *Turnaround time*
  - o tempo entre a submissão e o término de um job deve ser minimizado
- Utilização da CPU
  - A CPU deve ficar ocupada o tempo todo

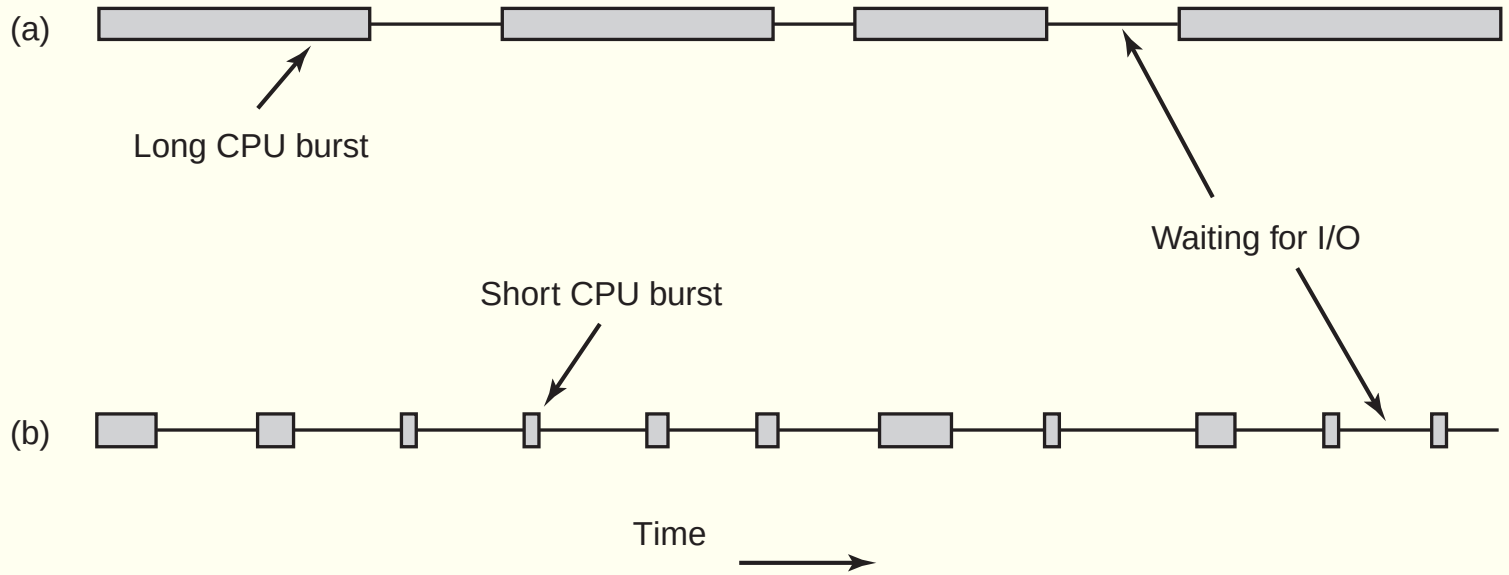
Escalonamento em sistemas batch

First-Come First-Served

- Processos obtêm a CPU na ordem de requisição
- Não preemptivo
- Aproveitamento ruim da CPU

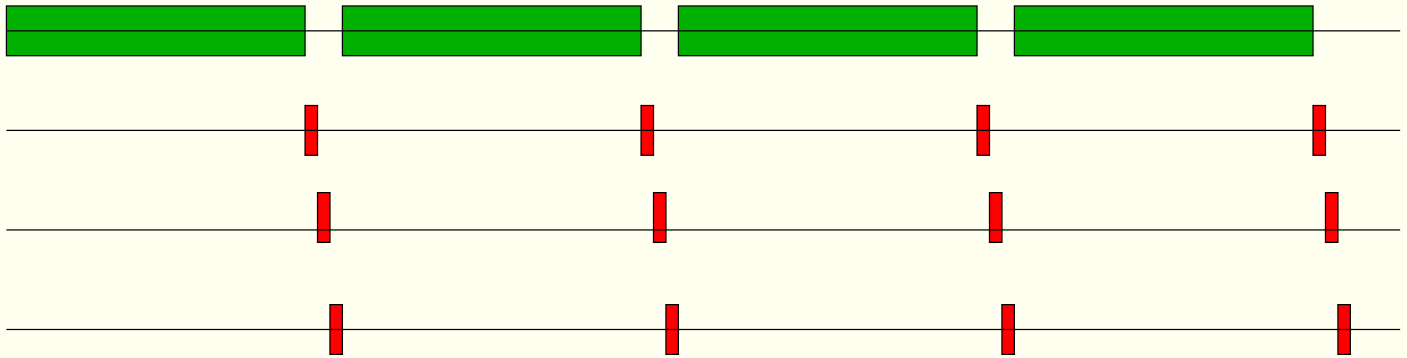
Aproveitamento da CPU

CPU-bound



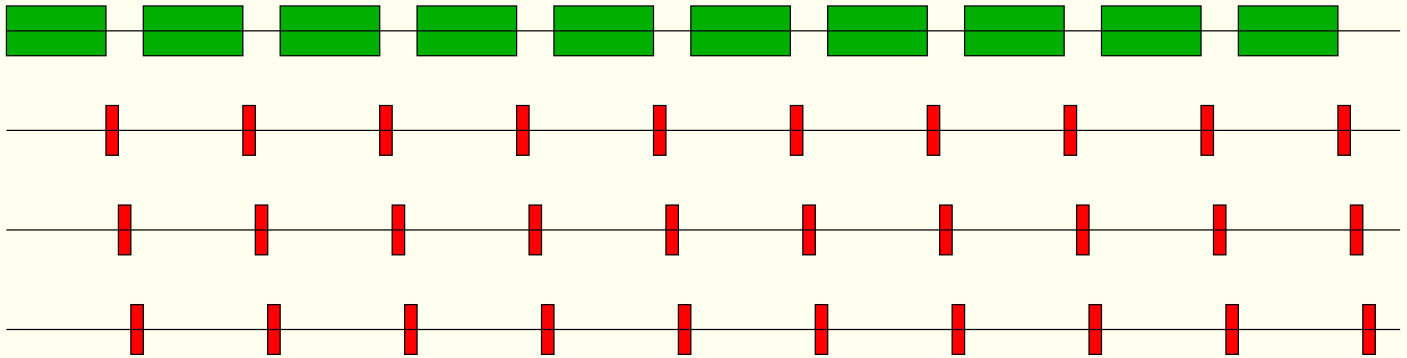
IO-bound

Aproveitamento da CPU



Processos I/O-bound conseguem poucos ciclos

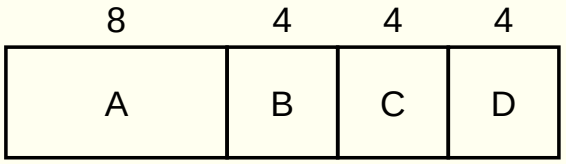
Aproveitamento da CPU



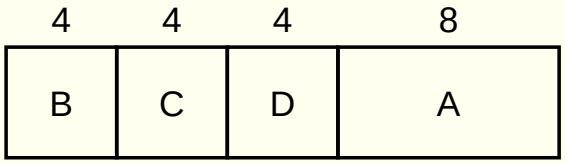
Processos I/O-bound conseguem mais ciclos

Escalonamento em sistemas batch

Shortest Job First



(a)



(b)

- Vazão (throughput) excelente
- Turnaround time

(a) $(8 + 12 + 16 + 20)/4 = 14$

(b) $(4 + 8 + 12 + 20)/4 = 11$

Escalonamento em sistemas batch

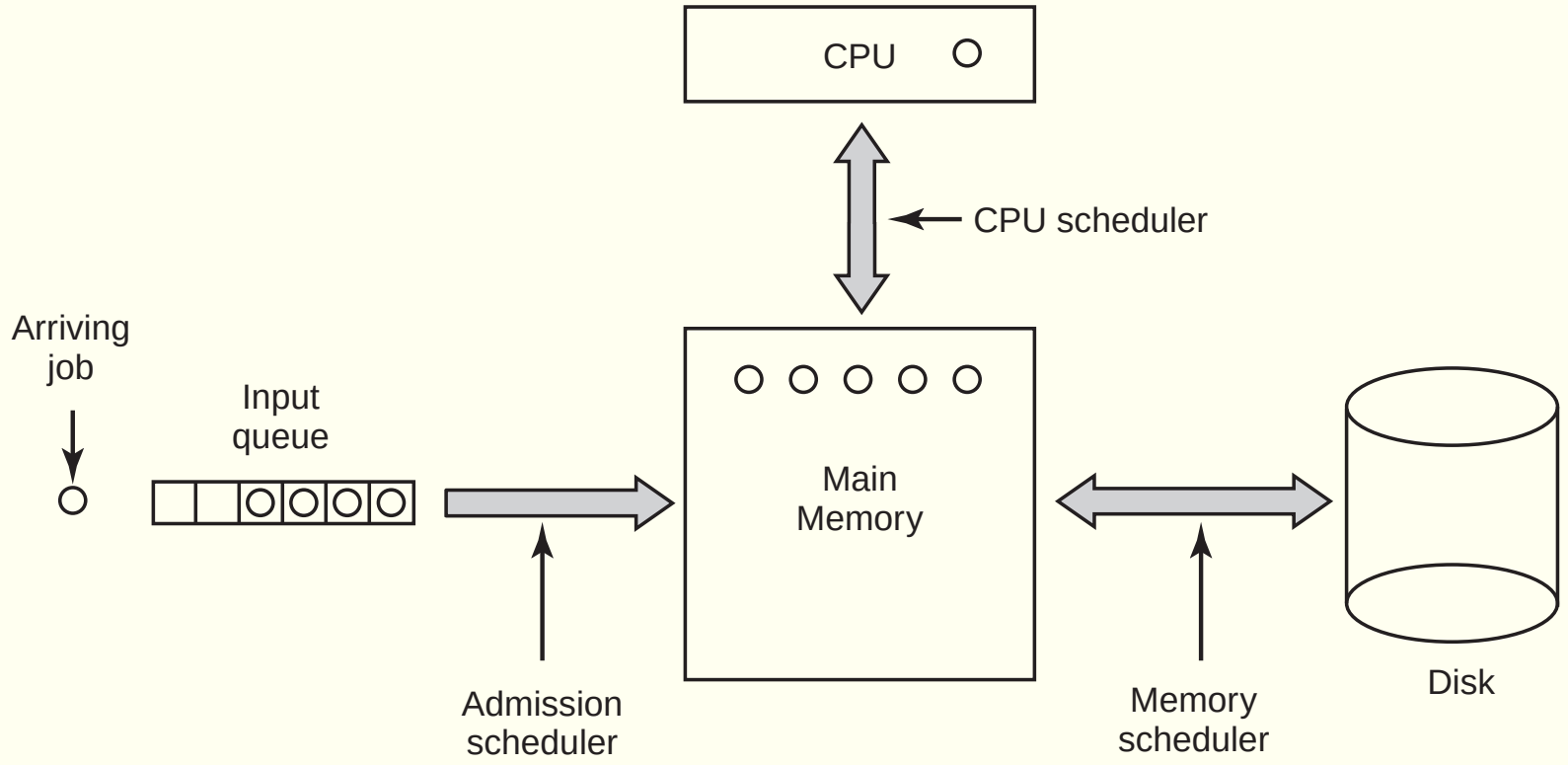
Shortest Job First

- Todos jobs precisam ser conhecidos previamente
  - Processos no tempo 0: 8 10
  - Processos no tempo 3: 4 4 8 10
- Se jobs curtos chegarem continuamente, os jobs longos nunca serão escalonados
  - Processos no tempo 100: 4 4 4 4 4 4 4 4 4 4 8 10

Shortest Remaining Time Next

- Versão preemptiva do algoritmo anterior
- Tenta ficar livre logo dos processos :-)
- Bom throughput
- Bom serviço para jobs curtos novos

Escalonamento em três níveis

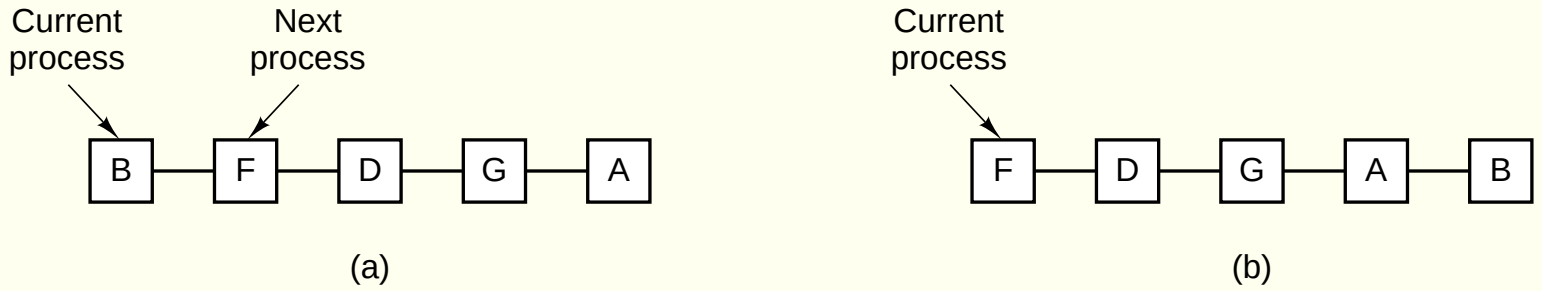


Tanenbaum: Figura 2.40

Escalonamento em sistemas interativos

- Tempo de resposta
  - O usuário quer respostas rápidas
- Proporcionalidade
  - É necessário respeitar as expectativas de tempo (tarefas fáceis versus tarefas difícies)

Round-Robin

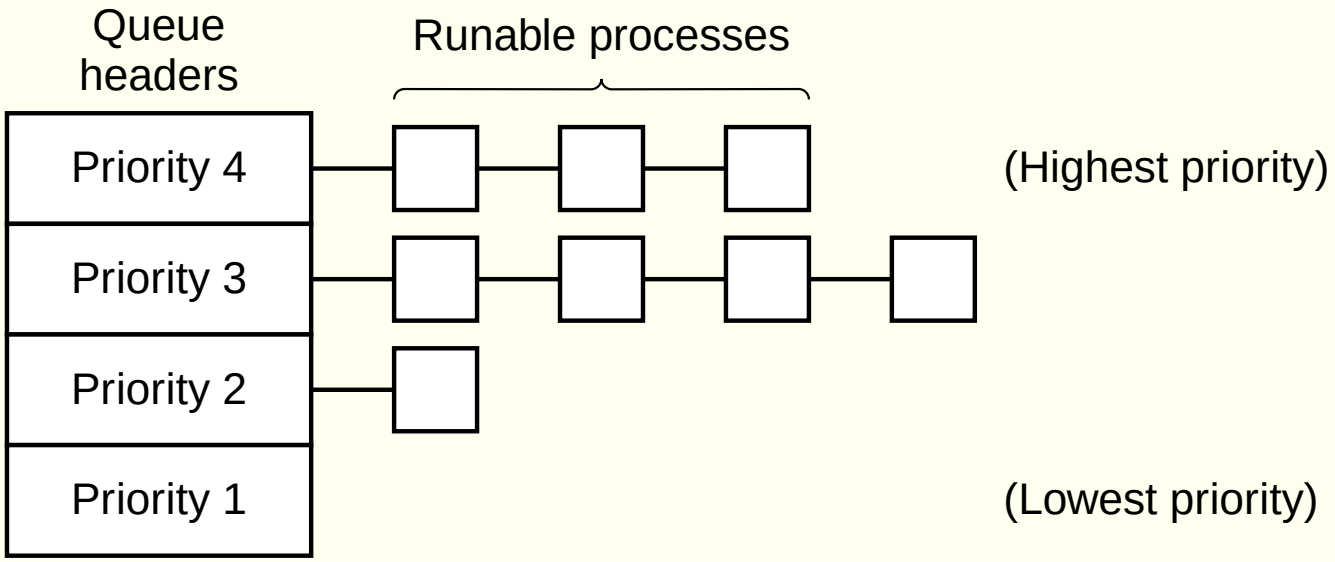


Tanenbaum: Figura 2.41

- Preemptivo
- Time quantum

  Como saber o valor ideal?

Prioridades para escalonamento



Tanenbaum: Figura 2.42

- Prioridades estáticas ou dinâmicas
- Comando `nice`

CTSS

Compatible Time Sharing System

- É mais eficiente rodar programas CPU-bound raramente por períodos longos do que frequentemente por períodos curtos
- Como determinar a classe de um processo?

Classe 0 (1 quantum)	$\rightarrow$	P1 P2 P5 P7
Classe 1 (2 quanta)	$\rightarrow$	P0 P3
Classe 2 (4 quanta)	$\rightarrow$	P4
Classe 3 (8 quanta)	$\rightarrow$	P6

Shortest Process Next

- Baseado no algoritmo shortest job first
- Comandos $\equiv$ jobs
- Estimativas de tempo (*aging*)
  - T_0
  - $T_0/2 + T_1/2$
  - $T_0/4 + T_1/4 + T_2/2$
  - $T_0/8 + T_1/8 + T_2/4 + T_3/2$

Justiça em sistemas interativos

- Escalonamento garantido
  - O SO faz promessas e deve mantê-las (e.g. 1/n CPU)
- Loteria
  - Baseado na distribuição de tickets
  - Fácil dar pesos distintos aos processos
- *Fair-share*
  - Cada usuário receberá uma parte adequada do poder de processamento da CPU

Escalonamento em sistemas de tempo real

- Respeitar deadlines
- Previsibilidade
- Hard real time e soft real time
- Tratamento dos eventos periódicos $\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$

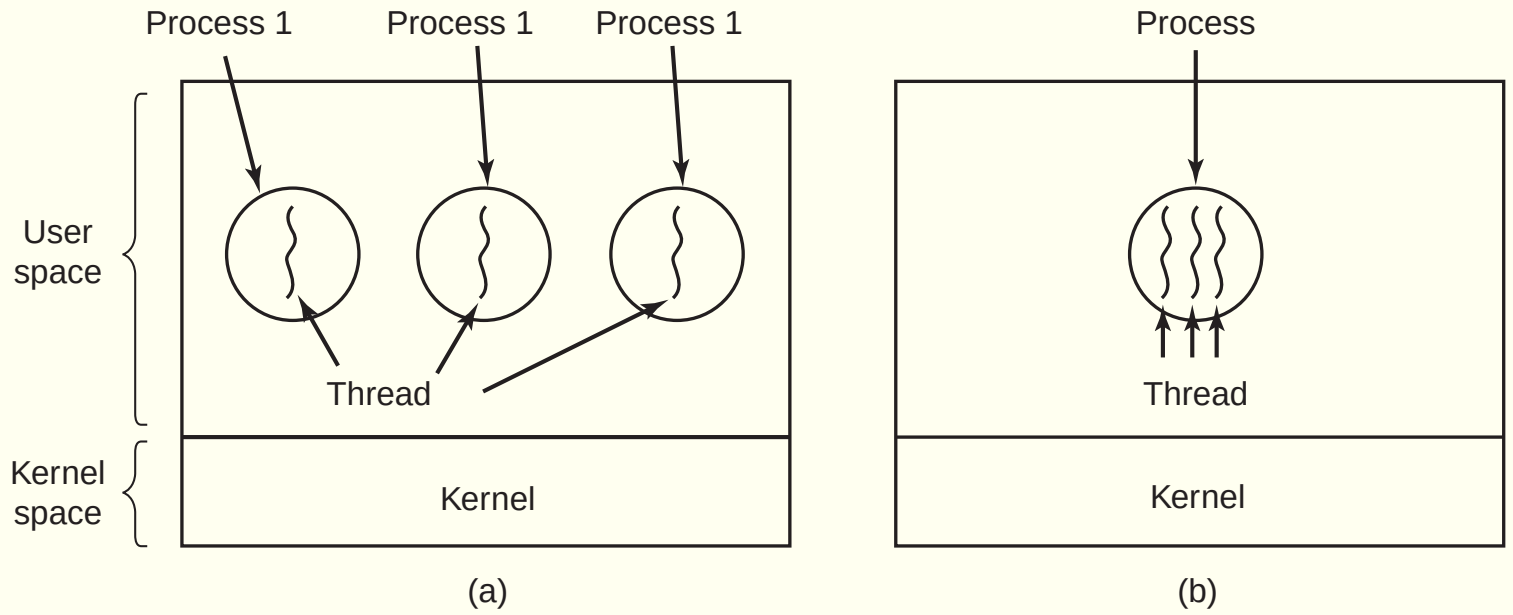
  P_i: periodicidade C_i: tempo de CPU

Política versus Mecanismo

- Considere o seguinte cenário:
  - um processo pai e vários processos filhos
  - os processos filhos atendem requisições
  - o processo pai sabe quais são as mais prioritárias
- O escalonador provê o mecanismo de escalonamento
- O processo pai sabe a política a ser adotada

Escalonamento de threads

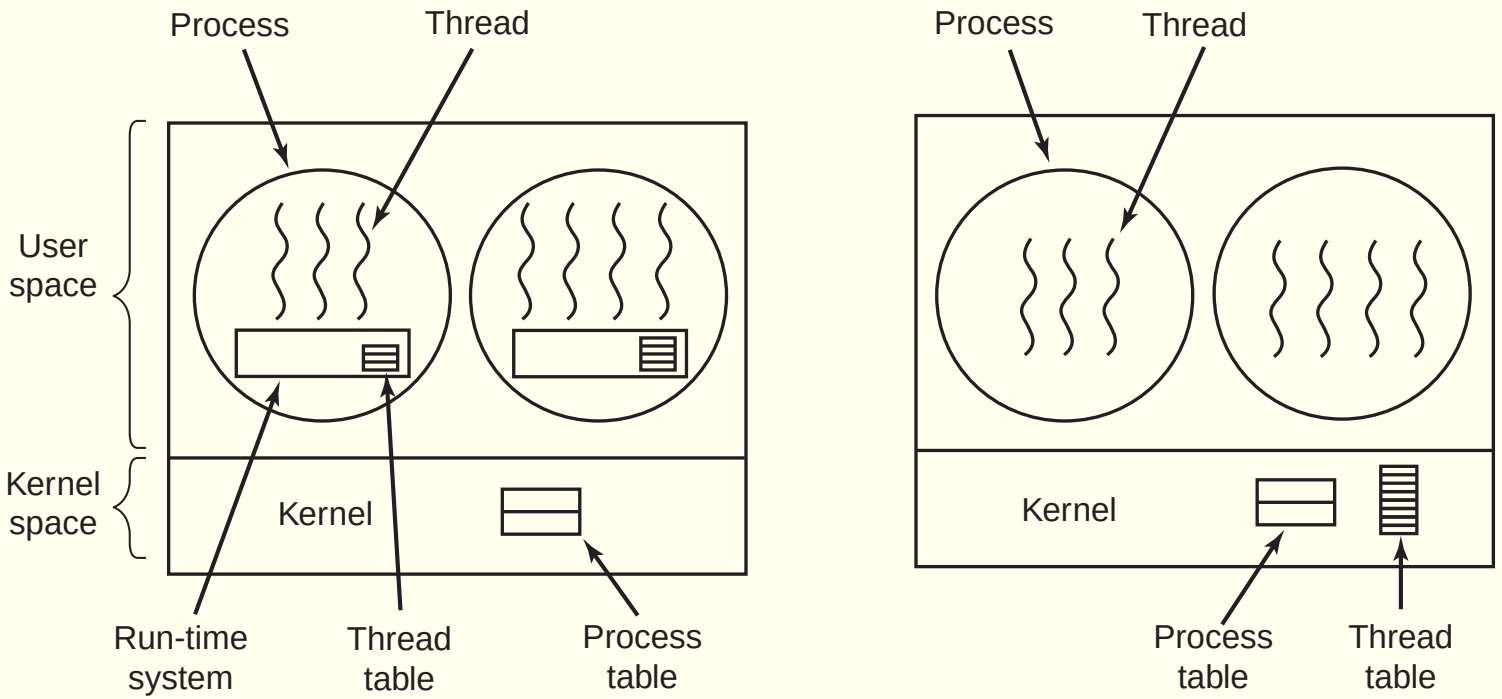
- Escalonamento em dois níveis: processos e threads



Tanenbaum: Figura 2.6

Escalonamento de threads

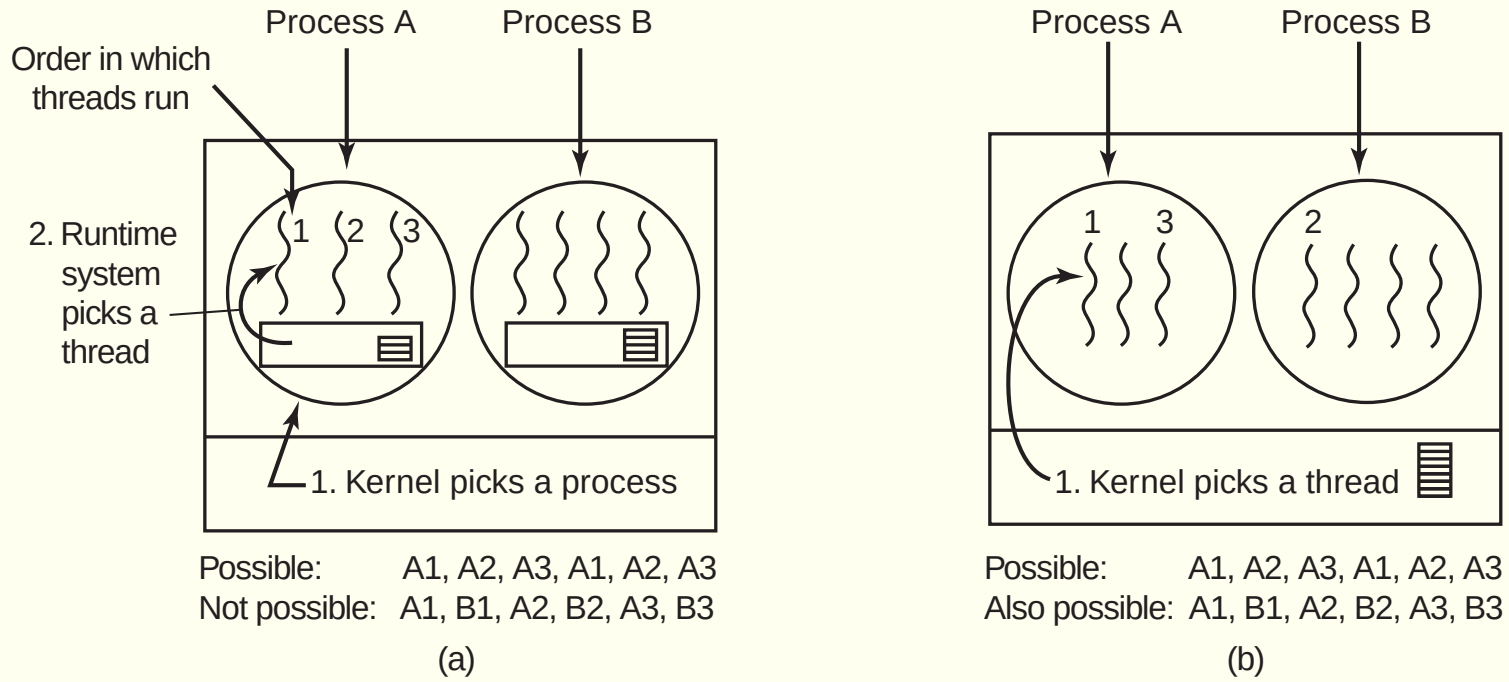
Threads de usuário e de kernel



Tanenbaum: Figura 2.13

Escalonamento de threads

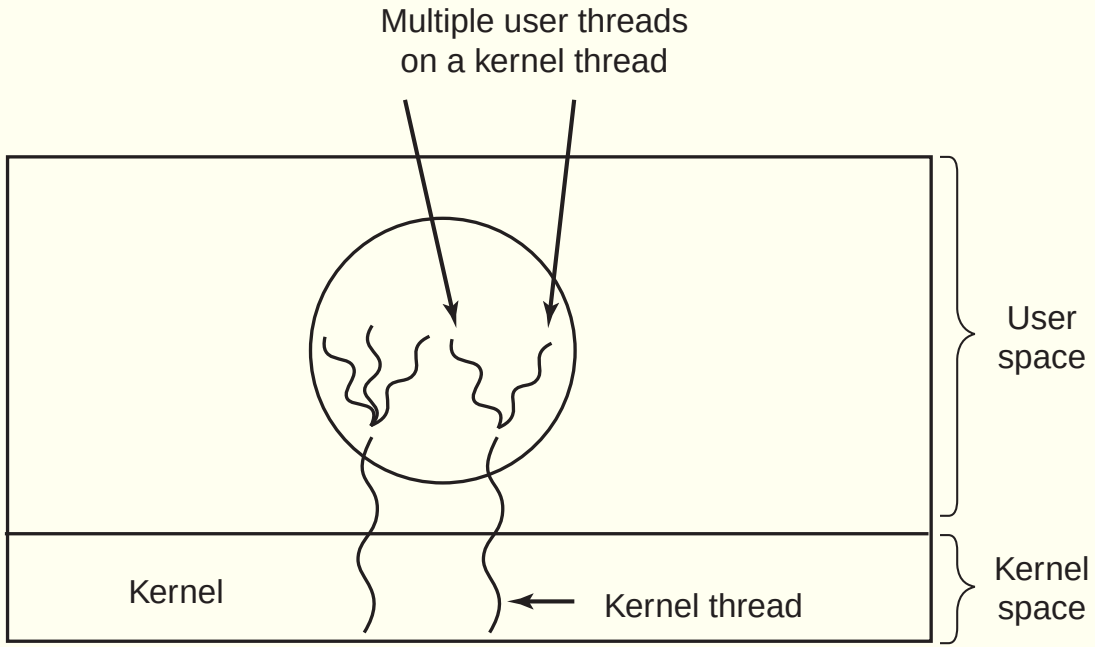
Threads de usuário e de kernel



Tanenbaum: Figura 2.43

Escalonamento de threads

Implementação híbrida



Tanenbaum: Figura 2.14