Sistemas Operacionais

Sistemas Operacionais

Escalonamento de Processos

Prof. Arliones Hoeller

arliones.hoeller@ifsc.edu.br

11 de fevereiro de 2014

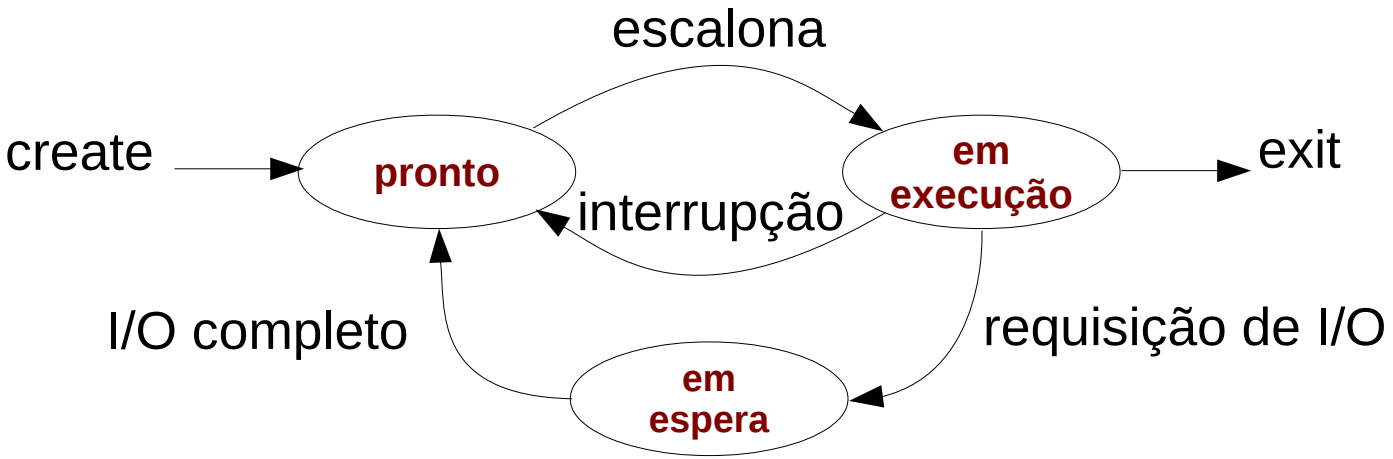
baseado no material do Prof. Fröhlich em
http://www.lisha.ufsc.br/~guto

Gerenciamento de processos

- Processo
  - É uma instância de um programa em execução
  - É uma entidade ativa
  - Possui *contexto* e *estado*
  - É executado sequencialmente
    - Cada instrução executada pertence a um processo
  - Outros nomes
    - *Job* em sistemas de lote (*batch systems*)
    - Tarefa (*task*) em sistemas com *time-sharing*

Estado de um Processo

- Estado de um processo
  - Em execução (*running*)
    - Processo cujas instruções estão sendo executadas
  - Em espera (*waiting*)
    - Processo aguardando por algum evento (ex.: operação de I/O)
  - Pronto (*ready*)
    - Processo está pronto para entrar em execução, mas precisa aguardar por um processador disponível



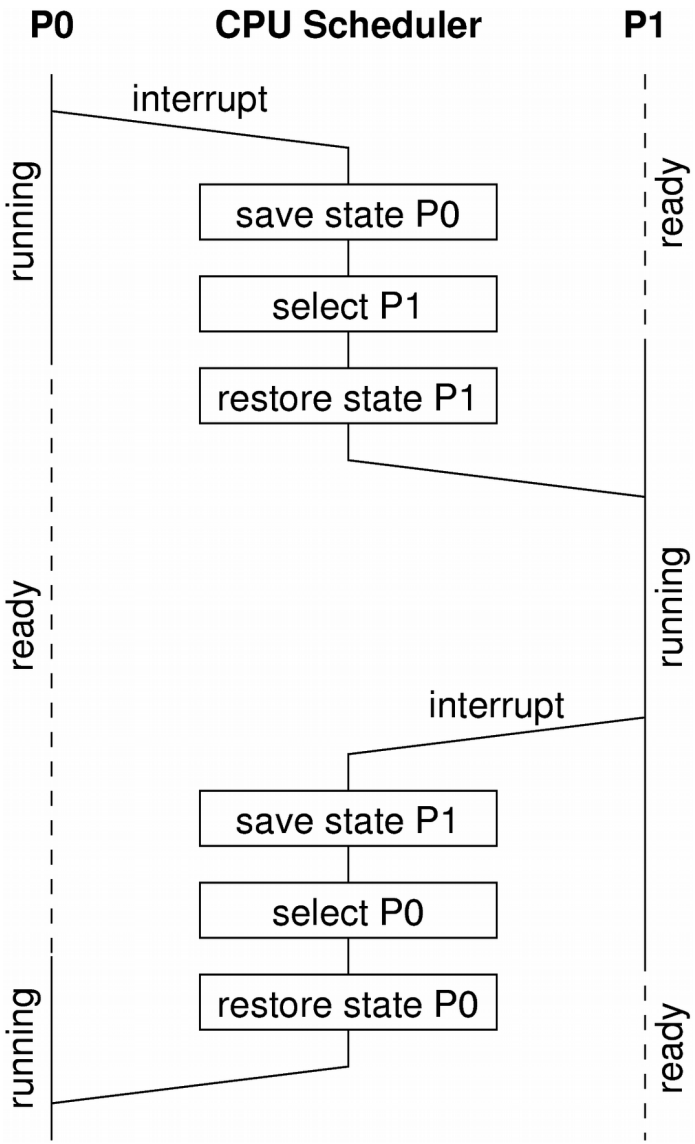
Contexto de Processo

- Contexto de Processo
  - Informação que permite ao SO retomar a execução de um processo
  - *Process Control Block* (PCB)
    - Estado
    - Registradores da CPU
    - Informação de escalonamento
    - Informação de memória
    - Inforamção de I/O
    - Informação de contabilização
  - Pilha do processo

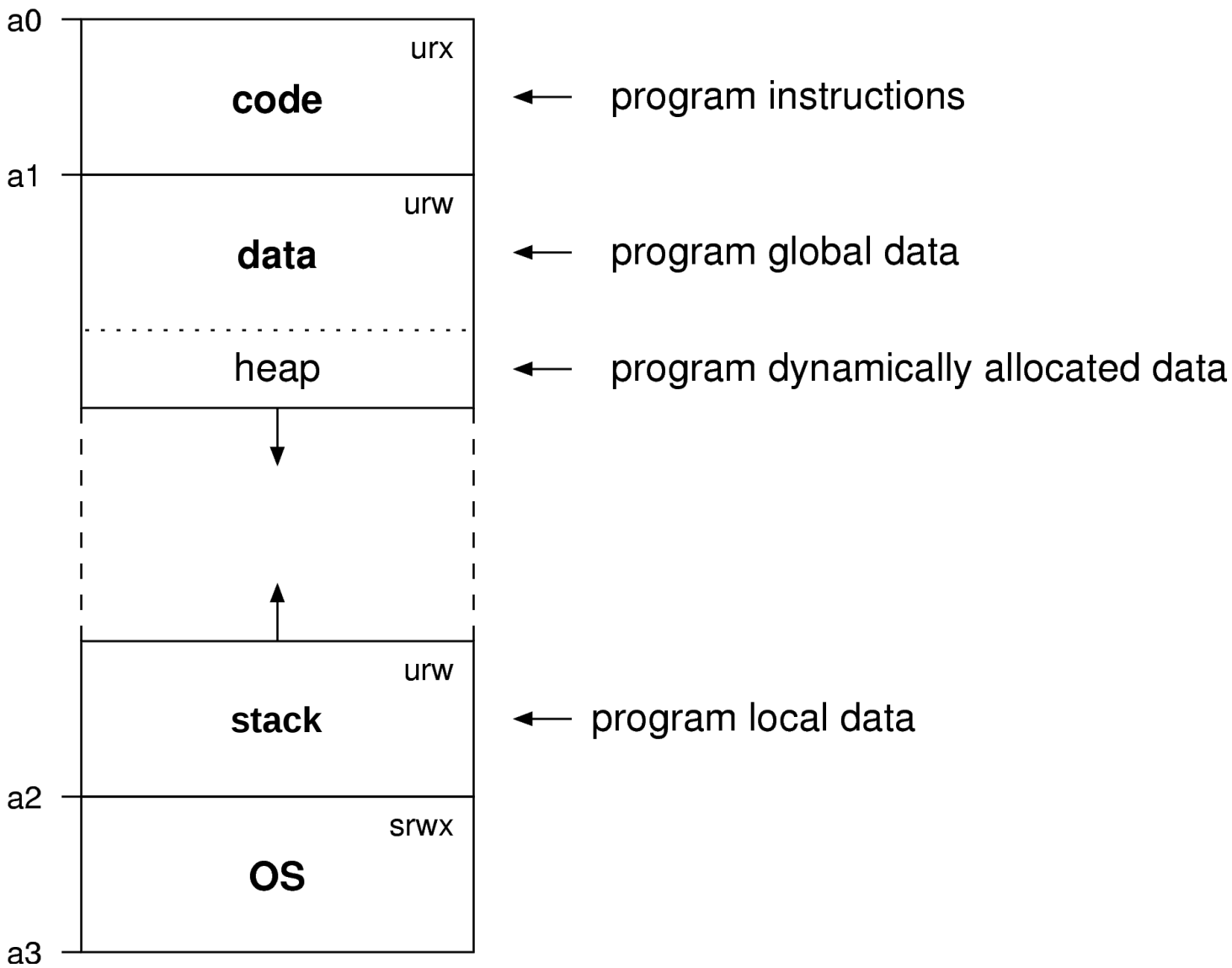
Process Control Block Típico

Process management	Memory management	File management
Registers	Pointer to text segment info	Root directory
Program counter	Pointer to data segment info	Working directory
Program status word	Pointer to stack segment info	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Troca de Contexto



Espaço de Endereçamento de um Processo



Criação de um Processo

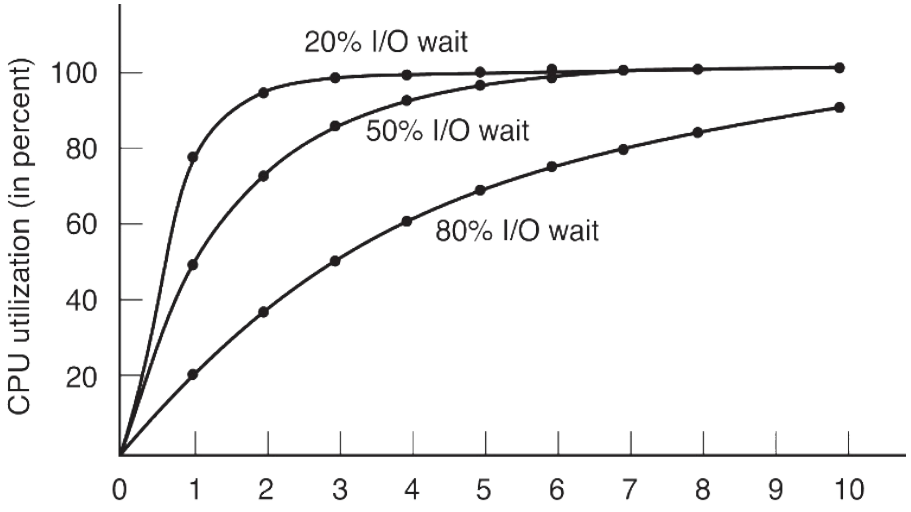
- Um processo é criado quando outro processo invoca a chamada de sistema correspondente (ex.: *fork*)
  - Criador = processo pai (*parent*)
  - Criado = processo filho (*child*)
  - Recursos do filho podem ser
    - Herdados do pai
    - Alocados no SO
  - Quem cria o primeiro processo?
    - Forjado pelo SO na inicialização
- Destruição de um Processo
  - Natural: quando um processo termina e chama *exit*
  - Forçado
    - Pelo SO quando um processo opera erroneamente (*abort*)
    - Por outro processo (pai) por qualquer razão (*kill*)

Processos Concorrentes

- Processos Concorrentes
  - Compartilhamento de recursos (concorrência)
  - Aceleração com múltiplos elementos de processamento
- Processos Independentes
  - Um prgrama sequencial em execução
  - Contexto privado
  - Saída depende exclusivamente da entrada
- Processos Cooperantes
  - Um programa paralelo em execução
  - Contexto compartilhado
  - Saída depende também da ordem relativa de execução

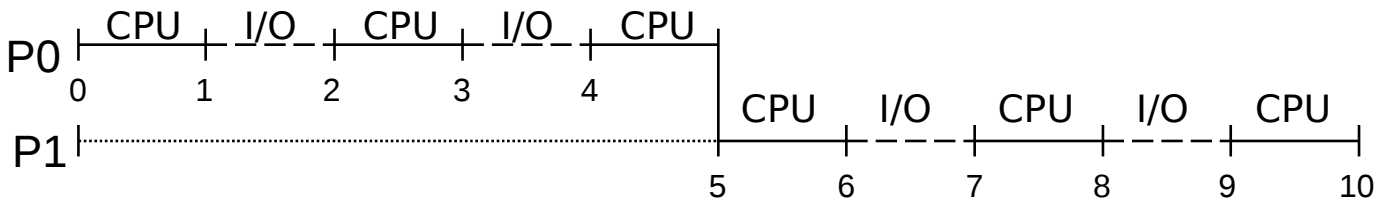
Multiprogramação

- Processamento paralelo em uma CPU
  - Processos bloqueados deixam a CPU
  - Processos prontos assumem a CPU
  - Não há processamento simultâneo (uma CPU)
- Métricas
  - Tempo de resposta (*turnaround time*)
  - Vazão (*throughput*)
  - Utilização da CPU



Multiprogramação

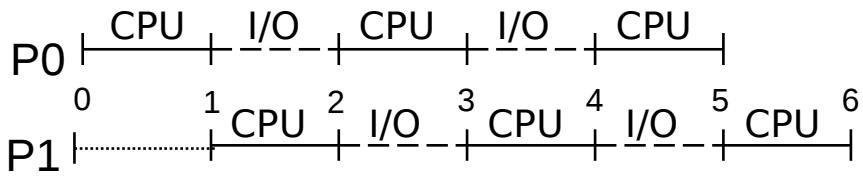
- SEM multiprogramação



tempo = 10 tu
vazão = 0.2 proc/tu

tempo médio de resposta = 7.5 tu
utilização da CPU = 60%

- COM multiprogramação

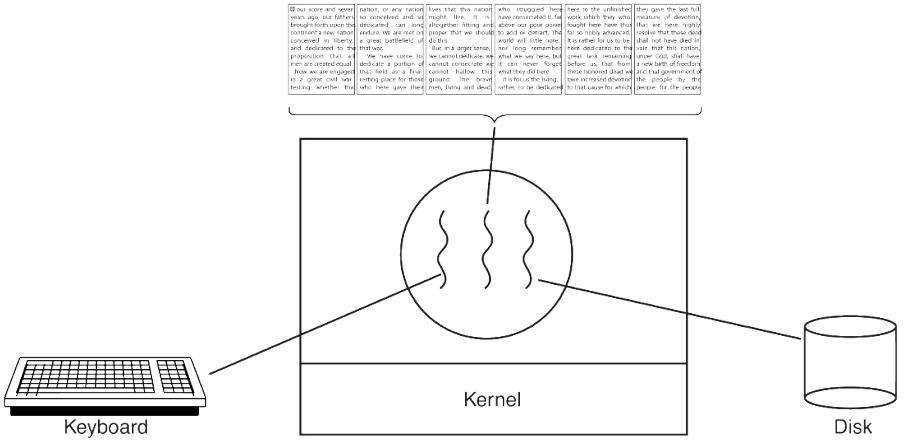


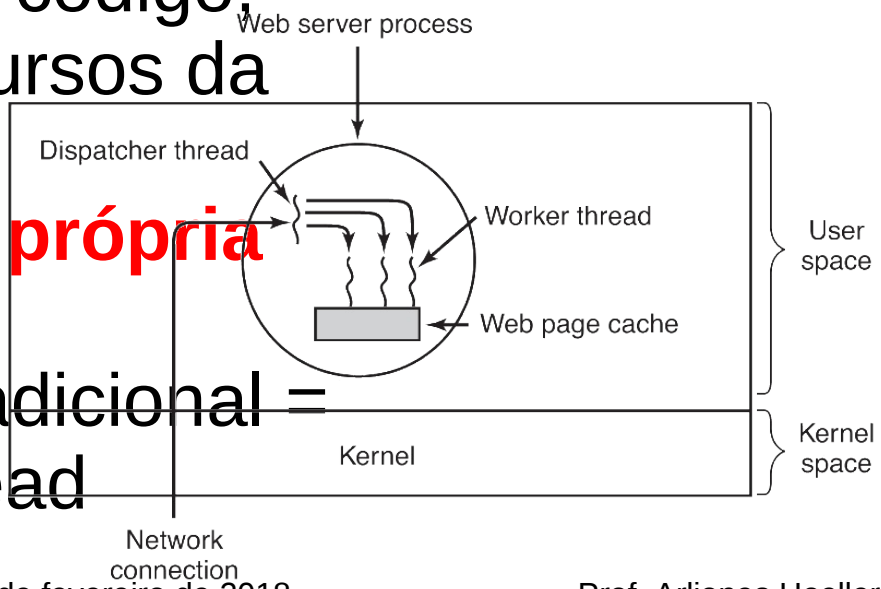
tempo = 6 tu
vazão = 0.33 proc/tu

tempo médio de resposta = 5.5 tu
utilização da CPU = 100%

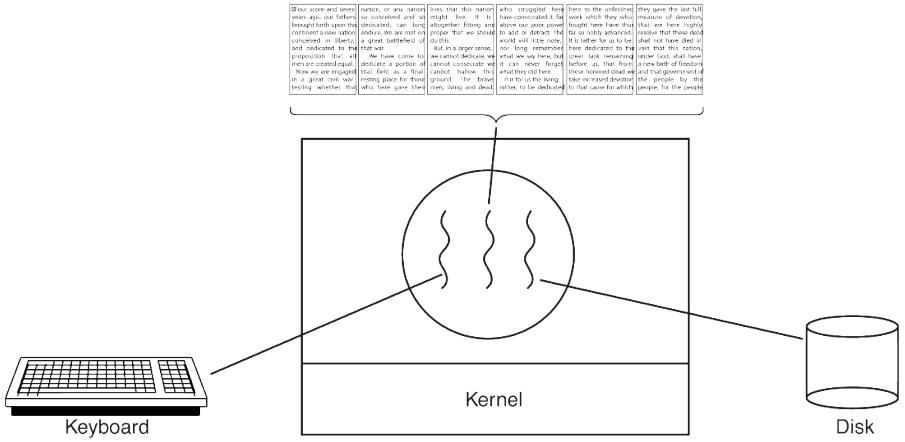
Threads

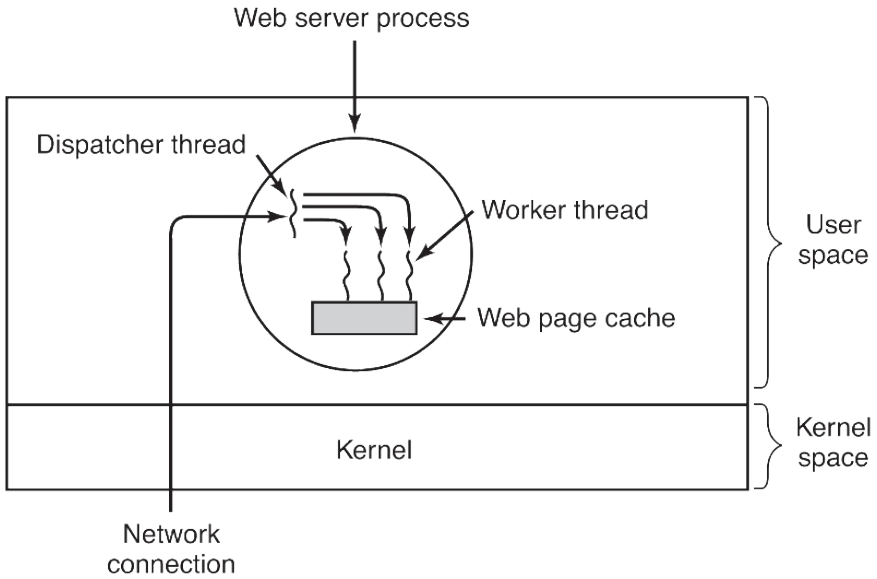
- Também chamados de processos leves (*lightweight*)
  - Baixo custo de criação
- A execução ocorre dentro de um processo (*task*)
- Compartilha código, dados e recursos da *task*
- **Possui sua própria pilha**
- Processo tradicional = task + 1 thread





Threads

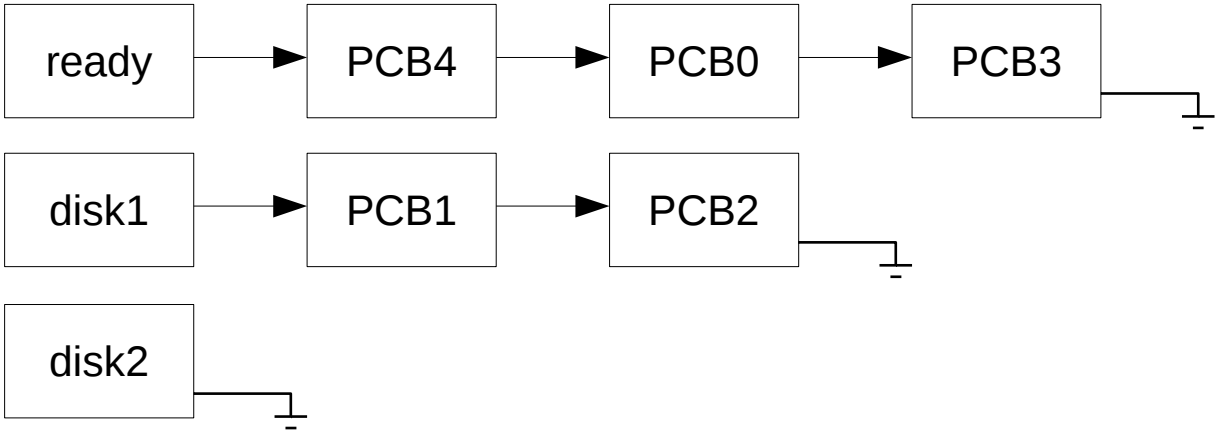




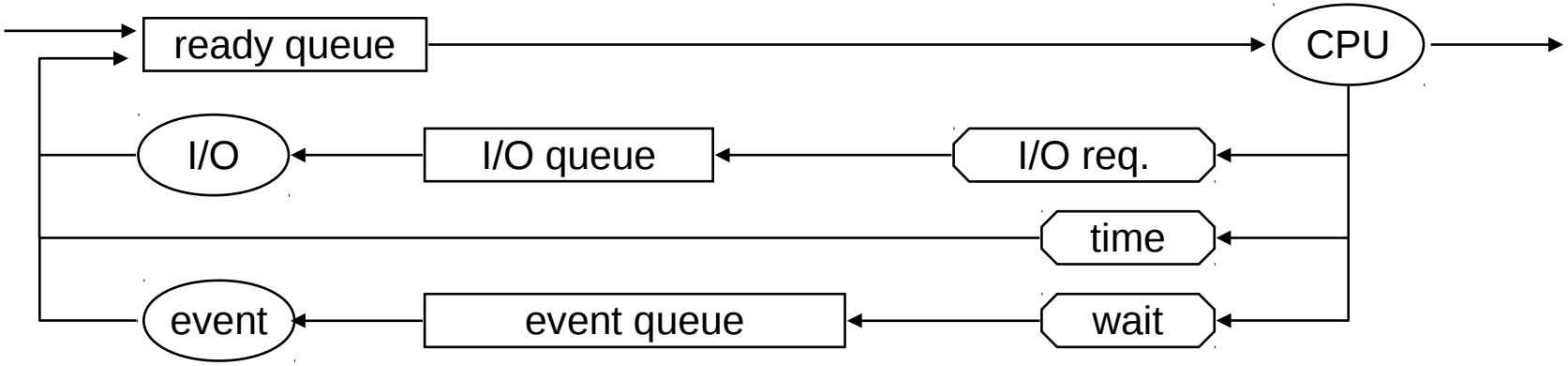
- Também chamados de processos leves (*lightweight*)
  - Baixo custo de criação
- A execução ocorre dentro de um processo (*task*)
- Compartilha código, dados e recursos da *task*
- **Possui sua própria pilha**
- Processo tradicional = task + 1 thread

Estruturas de Escalonamento de Processos

- Filas de prontos (*ready*) e E/S



- Diagrama de filas do sistema

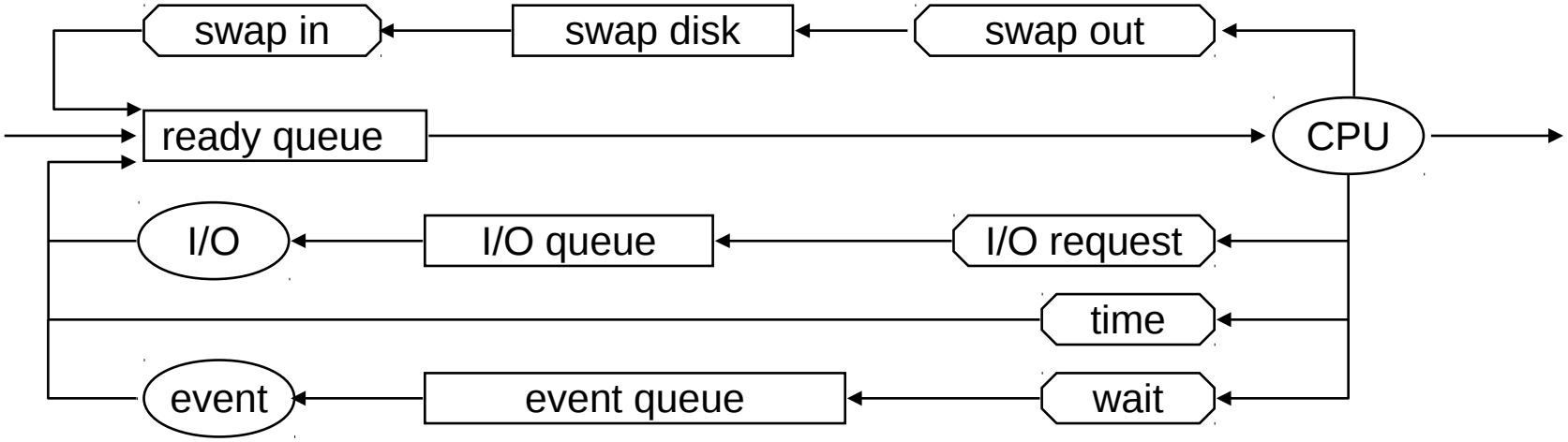


Escalonadores de Processos

- Escalonador de curto prazo (CPU)
  - Seleciona processos da fila de prontos
  - Executa frequentemente – precisa ser muito eficiente
- Escalonador de longo prazo (*jobs*)
  - Seleciona processos autorizados a executar
    - Controle de admissão de processos
  - Tenta balancear processos *I/O-bound* e *CPU-bound*

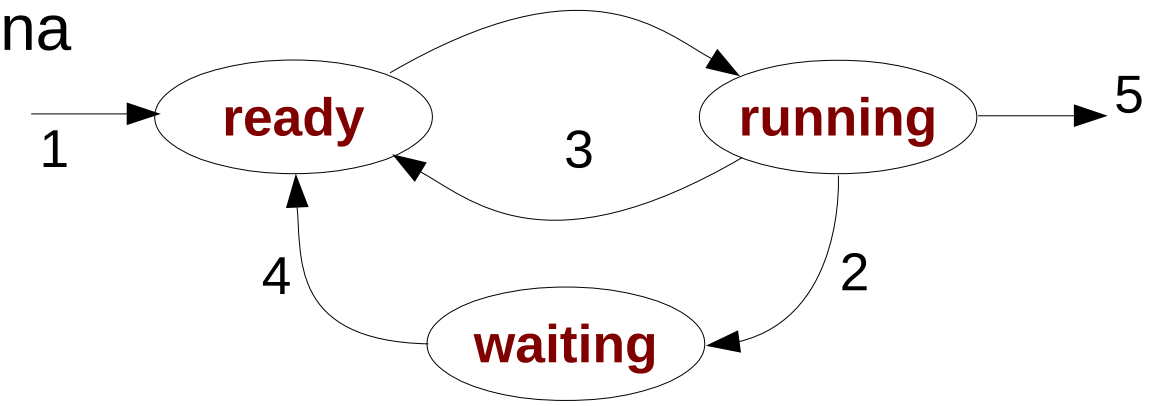
Escalonadores de Processos

- Escalonador de médio prazo (*swapper*)
  - Suspende processos temporariamente
    - Para manter balanço entre o uso de E/S e CPU
    - Devido ao esgotamento de memória



Escalonamento Preemptivo e Não-Preemptivo

- Os seguintes eventos modificam o estado de um processo e são observados pelo escalonador:
  1 - Novo processo no sistema
  2 - Muda seu estado de *executando* para *em espera*
  3 - Muda seu estado de *executando* para *pronto*
  4 - Muda seu estado de *em espera* para *pronto*
  5 - Termina



Critérios para escalonamento de processos

- Maximizar utilização da CPU
- Maximizar vazão do sistema (processos/tempo)
- Minimizar tempo de resposta (tempo total)
- Minimizar tempo de espera dos processos
- Minimizar e/ou estabilizar tempo de resposta ao usuário

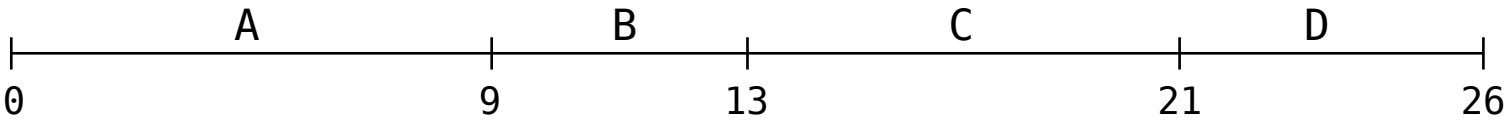
Políticas de Escalonamento de Processos

- *First-Come First-Served* (FCFS)
- *Shortest Job First*
- Prioridade Estática
- Prioridade Dinâmica
- *Round-Robin*
- Fila Multinível
- E milhares de outros...

First Come First Served (FCFS)

- Política
  - Fila de prontos (*ready*) utiliza política FIFO
  - Processos novos inseridos no final
  - Não-preemptivo
- Desempenho
  - Extremamente pobre quando um processo *CPU-bound* bloqueia um processo *I/O-bound*
- Exemplo

Process	A	B	C	D
CPU time	9	4	8	5
Arrival time	0	0	0	0

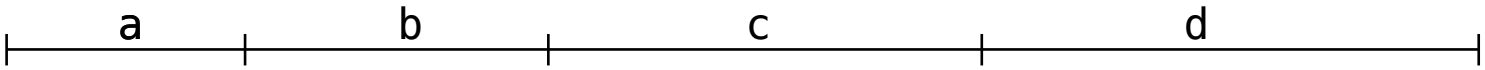


Tempo médio de resposta (TMR) = (9 + 13 + 21 + 26) / 4 = 17.25 tu

Tempo médio de espera (TME) = (0 + 9 + 13 + 21) / 4 = 10.75 tu

Shortest Job First (SJF)

- Política
  - Processo que utilizará menos CPU é executado antes
  - Preemptivo ou não-preemptivo
- Desempenho
  - Algoritmo ótimo em termos de tempo médio de resposta e tempo médio de espera



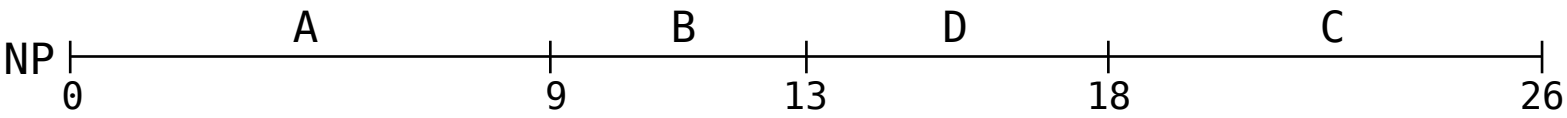

```
TMR = (a + (a+b) + (a+b+c) +(a+b+c+d)) / 4 = (4a + 3b + 2c + d) / 4 tu
TME = (0 + a + (a+b) + (a+b+c)) / 4 = (3a + 2b + c) /4 tu
```

- Útil para processos cujos tempos máximos de execução são conhecidos

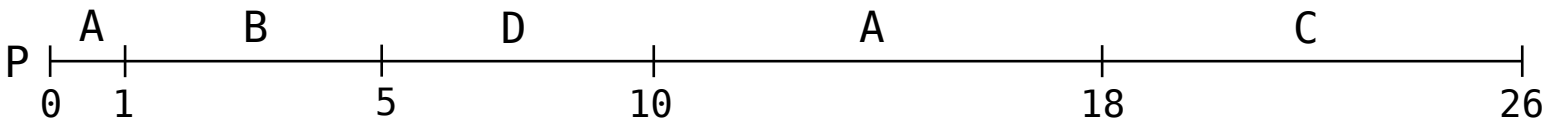
Shortest Job First (SJF)

- Exemplo

Process	A	B	C	D
CPU time	9	4	8	5
Arrival time	0	1	2	3



TMR = (9 + 12 + 24 + 15) / 4 = 15 tu
TME = (0 + 8 + 16 + 10) / 4 = 8.5 tu



TMR = (18 + 4 + 24 + 7) / 4 = 13,25 tu
TME = (9 + 0 + 16 + 2) / 4 = 6.75 tu

SJF Approximation

- Política
  - Estima tempo futuro baseado no passado recente
  - Processos que **têm tido** menor tempo de CPU é executado antes
- Fórmula

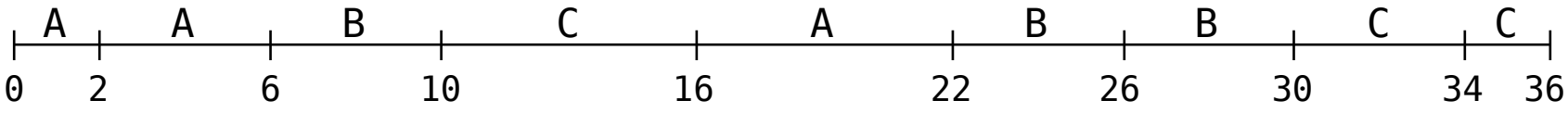
$$\bar{\pi}_{i+1} = a \cdot \pi_i + (1-a) \cdot \bar{\pi}_i$$

$\bar{\pi}_i$ = *estimativa no momento i*
a = *fator de relevância do passado*
π_i = *último tempo de execução do processo*

- Exemplo (α = 1/2)

```
TMR = (22 + 30 + 36) / 3 = 29.3 tu
TME= (10 + 18 + 24) / 3 = 17.3 tu
```

Process	π_0	t_0	π_1	t_1	π_2	t_2	π_3
A	1	2	1	4	2	6	4
B	1	4	2	4	3	4	3
C	1	6	3	4	3	2	2



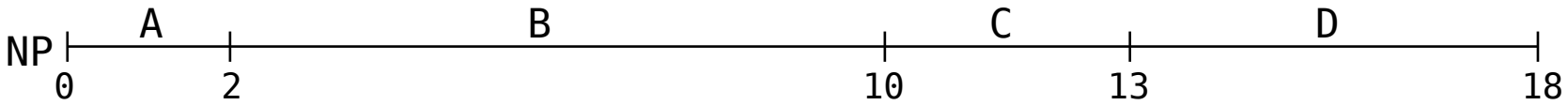
Prioridade

- Política
  - Processo com maior prioridade é executado antes
  - Proridades podem ser estáticas ou dinâmicas
  - Preemptivo ou não-preemptivo
- Processos podem nunca ser executados
  - Processos de baixa prioridade só executam quando processos com prioridade maior estão *em espera*
- Típico em sistemas de tempo real

Prioridade Estática

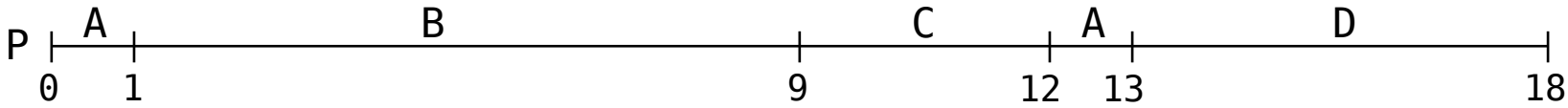
- Example

Process	A	B	C	D
CPU time	2	8	3	5
Priority	3	1	2	3
Arrival time	0	1	2	3



TMR = (2 + 9 +11 + 15) / 4 = 9.25 tu

TME = (0 + 1 + 8 + 10) / 4 = 4.75 tu



TMR = (13 + 8 + 10 + 15) / 4 = 11.5 tu

TME = (11 + 0 + 7 + 10) / 4 = 7 tu

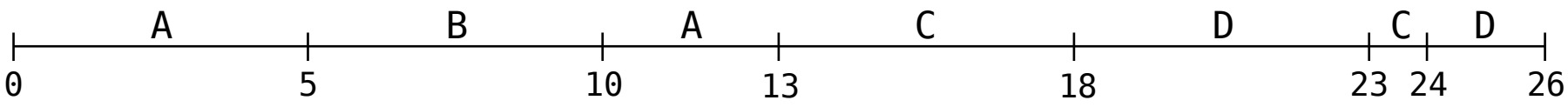
Round-Robin

- Política
  - Processos são re-escalonados periodicamente baseados num *quantum* de tempo
  - Fila circular FIFO
  - Preemptivo
- Fórmula
  - Para um dado conjunto de processos com **n** elementos e um *quantum* **q**:
    - Cada processo recebe 1/**n** de CPU em ciclos não maiores que **q** unidades de tempo
    - Tempo máximo de espera = (**n** – 1) x **q**
- Típico em sistemas com *time-sharing*

Round-Robin

- Exemplo (q = 5 tu)

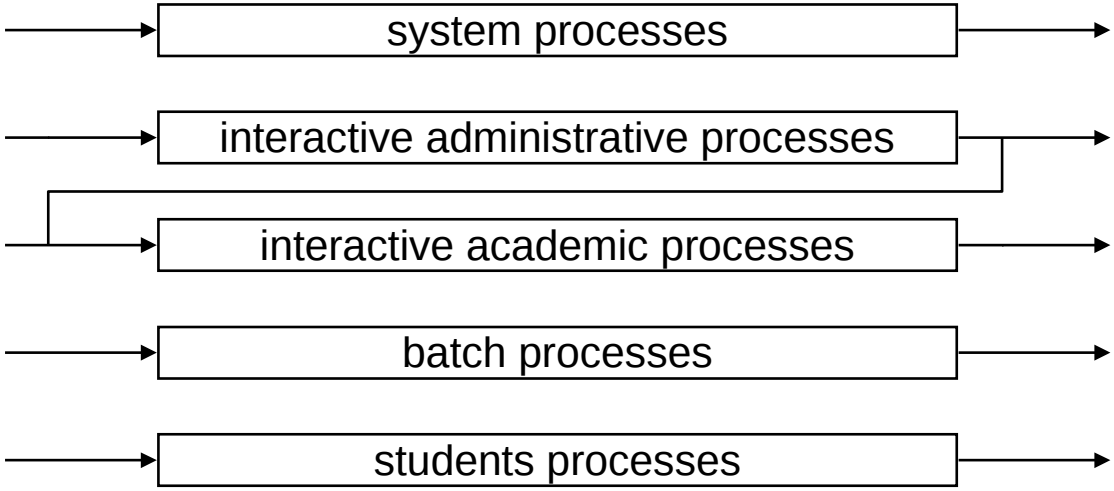
Process	A	B	C	D
CPU time	8	5	6	7
Arrival time	0	4	9	14



TMR = (13 + 6 + 15 + 12) / 4 = 11.5 tu
TME = (5 + 1 + 9 + 5) / 4 = 5 tu

Fila Multinível

- Política
  - Processos são agrupados
    - Ex.: sistema, interativos, *batch*
  - Cada grupo tem sua própria fila sob uma política específica
  - Processos podem trocar de grupos



Exercícios

- Calcule o tempo médio de resposta e o tempo médio de espera do sistema considerando o algoritmo SJF approximation.

Processo	π_0	t_0	π_1	t_1	π_2	t_2	π_3
A	1	6					
B	1	9					
C	1	4					

Processo	t_1	t_2
A	4	6
B	4	4
C	4	2

Exercícios

- Calcule o tempo médio de resposta e o tempo médio de espera do sistema considerando o Round-Robin com o quantum = 3;

Processo	A	B	C	D
Tempo de CPU	3	2	8	5
Chegada(tempo)	0	3	5	10