Processos

António Pinto
apinto@estg.ipp.pt

Escola Superior de Tecnologia e Gestão

Setembro, 2016

Sumário

Introdução

- Inicialmente só se executava um programa de cada vez, com acesso a todos os recursos e controlo total do *hardware*
- Atualmente podem-se executar vários programas em *simultâneo*
- Execução concorrente requer mecanismos de controlo
- Surge o conceito de processo [1]
- Processo assume-se como unidade de trabalho nos sistemas interativos atuais

[1] Processo como programa em execução

SO como conjunto de processos

- Sistema operativo pode ser visto com um conjunto de processos cooperativos, em execução concorrente

[2] Sem acesso directo ao *hardware*

[3] Com controlo total sobre ao *hardware*

SO como conjunto de processos

- Sistema operativo pode ser visto com um conjunto de processos cooperativos, em execução concorrente
  - Processos gerados pelos utilizadores correm em modo não privilegiado[2] $\rightarrow$ **Modo utilizador**

[2] Sem acesso directo ao *hardware*

[3] Com controlo total sobre ao *hardware*

SO como conjunto de processos

- Sistema operativo pode ser visto com um conjunto de processos cooperativos, em execução concorrente
  - Processos gerados pelos utilizadores correm em modo não privilegiado[2] $\rightarrow$ **Modo utilizador**
  - Processos que fazem parte do sistema operativo correm em modo privilegiado[3] $\rightarrow$ **Modo *root***

[2] Sem acesso directo ao *hardware*

[3] Com controlo total sobre ao *hardware*

Conteúdos

Representação de um processo

- Informação utilizada por um sistema operativo para representar um processo

Representação de um processo

- Informação utilizada por um sistema operativo para representar um processo e inclui, **além do código** do programa:
  - Dados das variáveis globais, locais, argumentos de funções, . . . (*stack*)

Representação de um processo

- Informação utilizada por um sistema operativo para representar um processo e inclui, **além do código** do programa:
  - Dados das variáveis globais, locais, argumentos de funções, . . . (*stack*)
  - *Program counter* (ou *instruction pointer*) que indica qual é a próxima instrução a ser executada

Representação de um processo

- Informação utilizada por um sistema operativo para representar um processo e inclui, **além do código** do programa:
  - Dados das variáveis globais, locais, argumentos de funções, . . . (*stack*)
  - *Program counter* (ou *instruction pointer*) que indica qual é a próxima instrução a ser executada
  - Recursos associados (tabelas de ficheiros, sinais, . . .)

Estados de um processo

Processos, desde que surgem e ao longo da sua existência vão transitando entre estados, até que terminam:

- **Novo (*new*):** Processo está a ser criado.

Estados de um processo

Processos, desde que surgem e ao longo da sua existência vão transitando entre estados, até que terminam:

- **Novo (*new*):** Processo está a ser criado.
- **Execução (*running*):** Código do processo está a ser executado pelo processador.

Estados de um processo

Processos, desde que surgem e ao longo da sua existência vão transitando entre estados, até que terminam:

- **Novo (*new*):** Processo está a ser criado.
- **Execução (*running*):** Código do processo está a ser executado pelo processador.
- **Espera (*waiting*):** Processo pediu um operação de I/O e aguarda que esta esteja concluída.

Estados de um processo

Processos, desde que surgem e ao longo da sua existência vão transitando entre estados, até que terminam:

- **Novo (*new*):** Processo está a ser criado.
- **Execução (*running*):** Código do processo está a ser executado pelo processador.
- **Espera (*waiting*):** Processo pediu um operação de I/O e aguarda que esta esteja concluída.
- **Pronto (*ready*):** Processo está suspenso mas pronto para ser continuado.

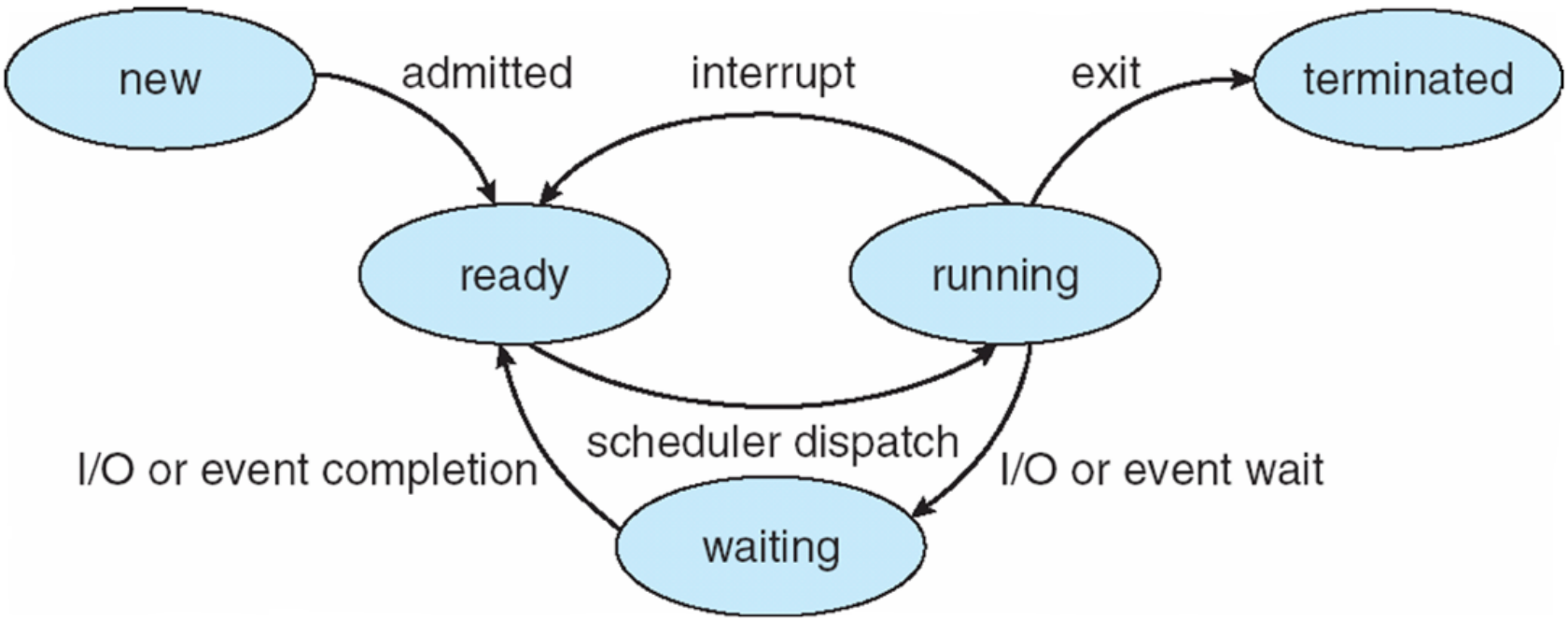
Estados de um processo

Processos, desde que surgem e ao longo da sua existência vão transitando entre estados, até que terminam:

- **Novo (*new*):** Processo está a ser criado.
- **Execução (*running*):** Código do processo está a ser executado pelo processador.
- **Espera (*waiting*):** Processo pediu um operação de I/O e aguarda que esta esteja concluída.
- **Pronto (*ready*):** Processo está suspenso mas pronto para ser continuado.
- **Terminado (*terminated*):** Processo acabou.

Estados de um processo

Diagrama de estados



Process control block (PCB)

- PCB é uma estrutura de dados usada pelo SO para guardar a informação de representação de um processo

Process control block (PCB)

- PCB é uma estrutura de dados usada pelo SO para guardar a informação de representação de um processo
- PCB estão armazenados em memória
- SO tem sempre visão atual de cada processo

Process control block (PCB)

Informação contida no PCB

- **Estado:** Estado do processo (*new, ready, running*, . . .

Process control block (PCB)

Informação contida no PCB

- **Estado:** Estado do processo (*new, ready, running*, . . .
- **Program counter:** Endereço da próxima instrução.

Process control block (PCB)

Informação contida no PCB

- **Estado:** Estado do processo (*new, ready, running*, . . .
- **Program counter:** Endereço da próxima instrução.
- **Registos CPU:** Guardados sempre que se suspende.

Process control block (PCB)

Informação contida no PCB

- **Estado:** Estado do processo (*new, ready, running*, . . .
- **Program counter:** Endereço da próxima instrução.
- **Registos CPU:** Guardados sempre que se suspende.
- **Escalonamento:** Prioridade, apontadores para filas de escalonamento, . . .

Process control block (PCB)

Informação contida no PCB

- **Estado:** Estado do processo (*new, ready, running*, . . .
- **Program counter:** Endereço da próxima instrução.
- **Registos CPU:** Guardados sempre que se suspende.
- **Escalonamento:** Prioridade, apontadores para filas de escalonamento, . . .
- **Memória:** Base e limite do segmento de memória, . . .

Process control block (PCB)

Informação contida no PCB

- **Estado:** Estado do processo (*new, ready, running*, ...
- **Program counter:** Endereço da próxima instrução.
- **Registos CPU:** Guardados sempre que se suspende.
- **Escalonamento:** Prioridade, apontadores para filas de escalonamento, ...
- **Memória:** Base e limite do segmento de memória, ...
- **Contabilidade:** Tempo de CPU utilizado, utilizador, número do processo, tempo real utilizado, ...

Process control block (PCB)

Informação contida no PCB

- **Estado:** Estado do processo (*new, ready, running*, ...
- **Program counter:** Endereço da próxima instrução.
- **Registos CPU:** Guardados sempre que se suspende.
- **Escalonamento:** Prioridade, apontadores para filas de escalonamento, ...
- **Memória:** Base e limite do segmento de memória, ...
- **Contabilidade:** Tempo de CPU utilizado, utilizador, número do processo, tempo real utilizado, ...
- **I/O:** Lista de ficheiros abertos, lista de sinais, ...

Process control block (PCB)

PCB do linux

Em Linux, cada processo é representado com a estrutura ***task_struct*** que contém campos como:

```
pid t_pid;                      /* ID processo                */
long state;                     /* Estado processo            */
unsigned int time_slice         /* Info escalonamento         */
struct task_struct *parent;     /* Pai                        */
struct list_head children;      /* Lista filhos               */
struct files_struct *files;     /* Lista ficheiros abertos    */
struct mm_struct *mm;           /* Segmento memoria           */
```

Conteúdos

Escalonamento de processos

- Sistemas multi-programados permitem a execução ***ao mesmo tempo*** de mais do que um processo

Escalonamento de processos

- Sistemas multi-programados permitem a execução ***ao mesmo tempo*** de mais do que um processo
  - Requer capacidade para alternar entre processos ativos (em execução)

Escalonamento de processos

- Sistemas multi-programados permitem a execução ***ao mesmo tempo*** de mais do que um processo
  - Requer capacidade para alternar entre processos ativos (em execução)
  - Só pode ser executado, em simultâneo, no máximo, um processo por núcleo (ou *core*)

Escalonamento de processos

- Sistemas multi-programados permitem a execução ***ao mesmo tempo*** de mais do que um processo
  - Requer capacidade para alternar entre processos ativos (em execução)
  - Só pode ser executado, em simultâneo, no máximo, um processo por núcleo (ou *core*)
- Pode-se escolher a **ordem de execução** dos processos!

Filas de escalonamento

- Quando um processo é criado este é **carregado para memória** principal e é colocado na **fila de processos prontos**

Filas de escalonamento

- Quando um processo é criado este é **carregado para memória** principal e é colocado na **fila de processos prontos**
- Fila processos prontos contem todos os processos existentes no sistema que podem ser executados

Filas de escalonamento

- Quando um processo é criado este é **carregado para memória** principal e é colocado na **fila de processos prontos**
- Fila processos prontos contem todos os processos existentes no sistema que podem ser executados
- Filas podem ser implementadas como listas ligadas, como vetores, ...

Filas de escalonamento

- Quando um processo é criado este é **carregado para memória** principal e é colocado na **fila de processos prontos**
- Fila processos prontos contem todos os processos existentes no sistema que podem ser executados
- Filas podem ser implementadas como listas ligadas, como vetores, ...
- Cada elemento destas listas tem a estrutura do PCB

Filas de escalonamento

- Quando um processo é criado este é **carregado para memória** principal e é colocado na **fila de processos prontos**
- Fila processos prontos contem todos os processos existentes no sistema que podem ser executados
- Filas podem ser implementadas como listas ligadas, como vetores, ...
- Cada elemento destas listas tem a estrutura do PCB
- Fila processo prontos pode ser construída com dois apontadores (cabeça e cauda) do mesmo tipo

Filas de escalonamento

Exemplo

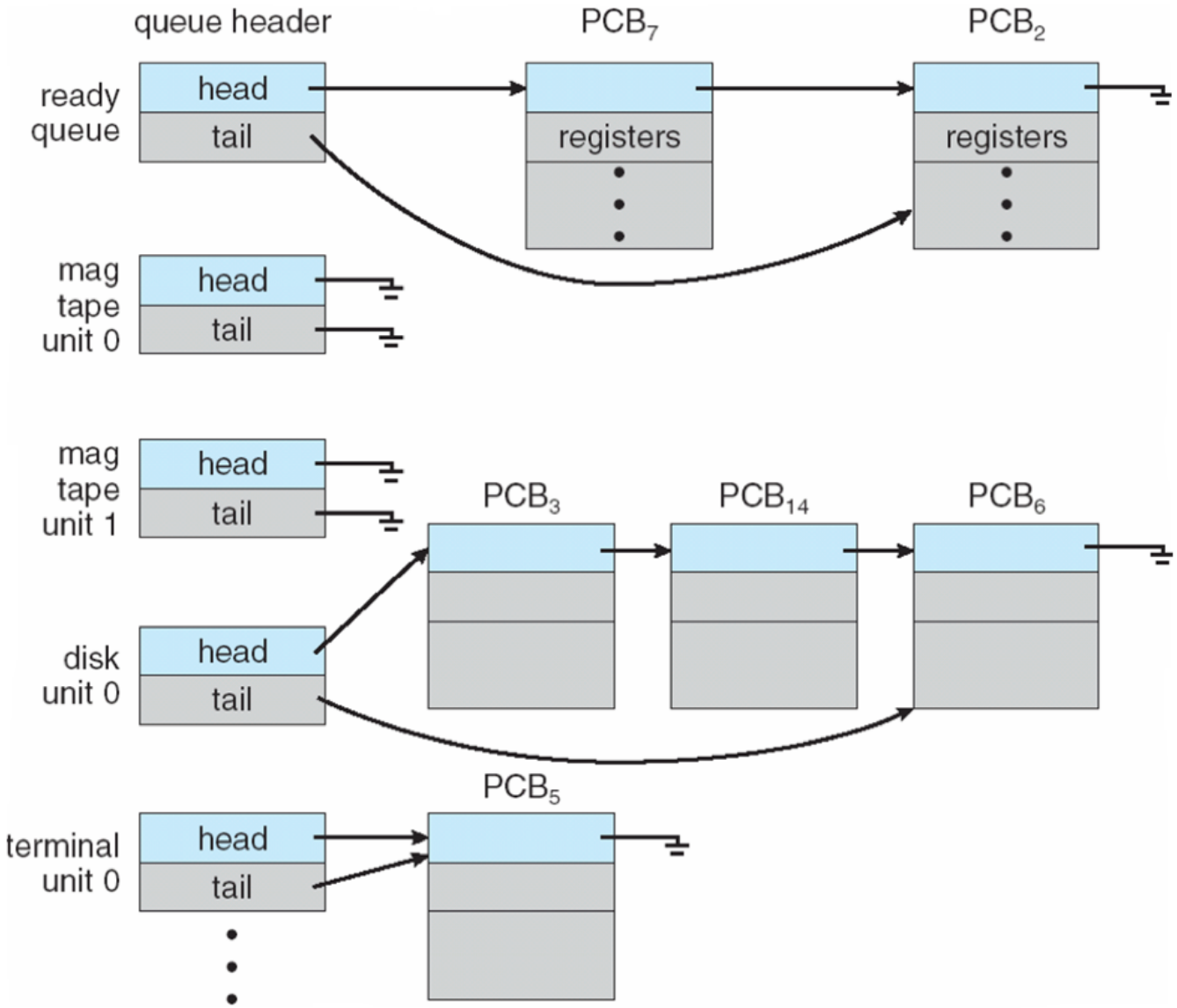


Diagrama de filas

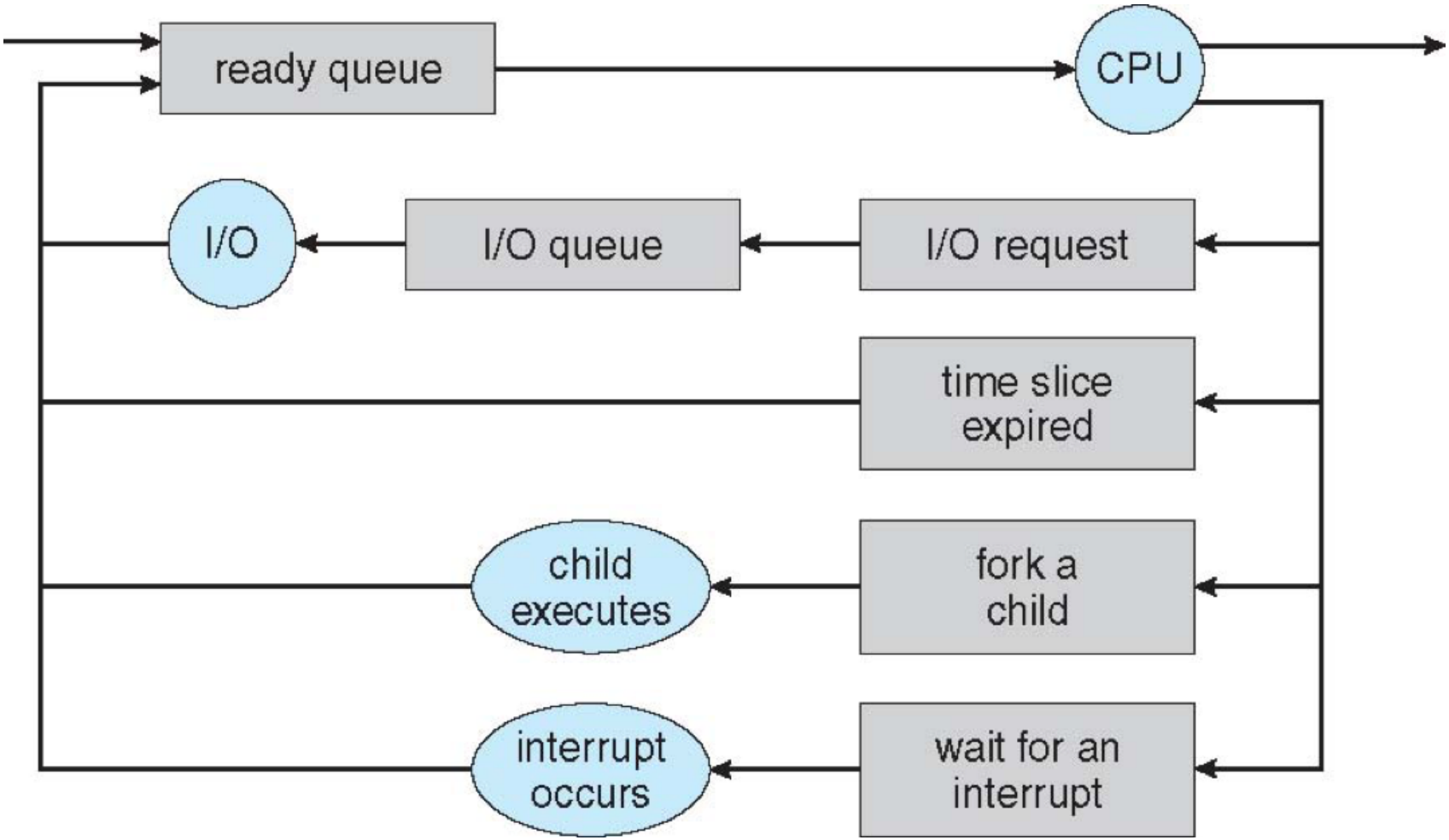


Diagrama de filas

- Durante a execução de um processo, pode:

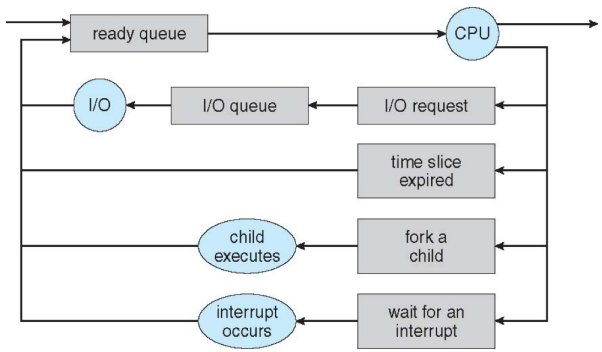


Diagrama de filas

- Durante a execução de um processo, pode:
  - Ser feito um pedido de I/O

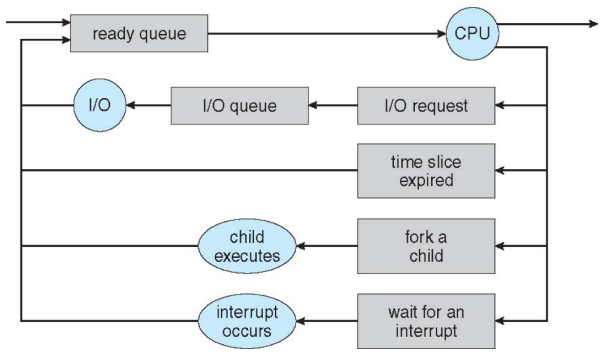


Diagrama de filas

- Durante a execução de um processo, pode:
  - Ser feito um pedido de I/O
  - Acabar o tempo de CPU atribuído

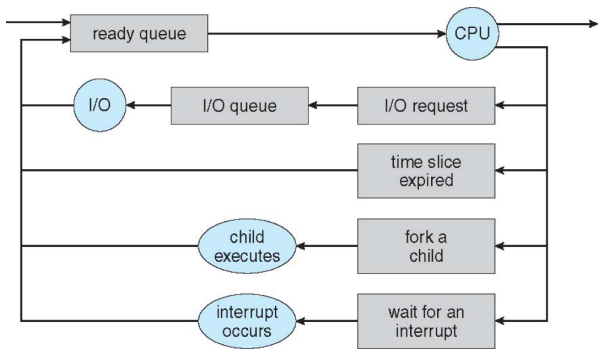


Diagrama de filas

- Durante a execução de um processo, pode:
  - Ser feito um pedido de I/O
  - Acabar o tempo de CPU atribuído
  - Criar um processo (filho)

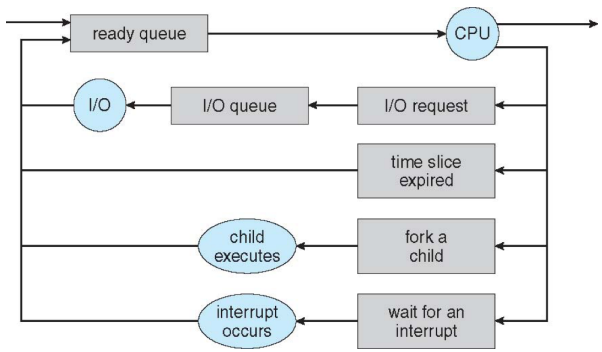


Diagrama de filas

- Durante a execução de um processo, pode:
  - Ser feito um pedido de I/O
  - Acabar o tempo de CPU atribuído
  - Criar um processo (filho)
  - Ser interrompido (sinal)

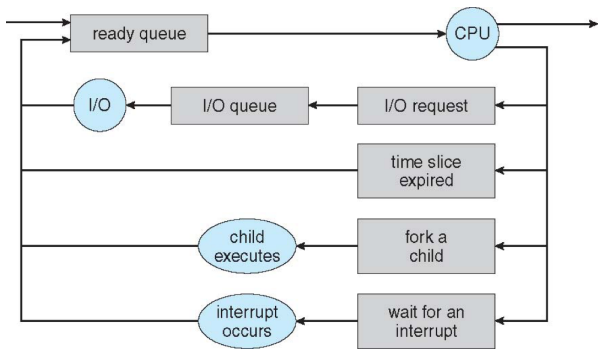
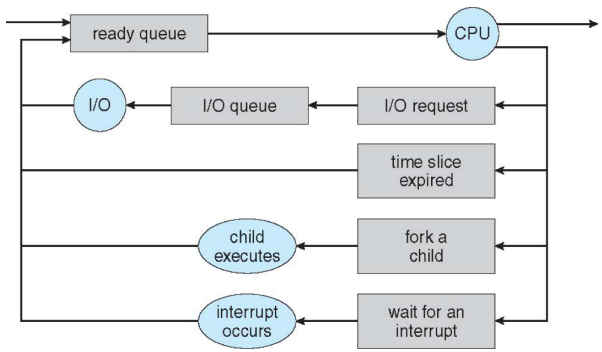


Diagrama de filas

- Durante a execução de um processo, pode:
  - Ser feito um pedido de I/O
  - Acabar o tempo de CPU atribuído
  - Criar um processo (filho)
  - Ser interrompido (sinal)
- Terminando o processo, o seu PCB é removido de todas as filas



Técnicas de escalonamento

- Processo, ao longo da vida, passa pelas várias filas de escalonamento

Técnicas de escalonamento

- Processo, ao longo da vida, passa pelas várias filas de escalonamento
- Seleção de processos deve ser com base em lógica

Técnicas de escalonamento

- Processo, ao longo da vida, passa pelas várias filas de escalonamento
- Seleção de processos deve ser com base em lógica
- Seleção é efetuada pelo *dispatcher*

Técnicas de escalonamento

- Processo, ao longo da vida, passa pelas várias filas de escalonamento
- Seleção de processos deve ser com base em lógica
- Seleção é efetuada pelo *dispatcher*
- Existem algumas abordagens
  - Escalonamento de curto prazo (*short-term*)
  - Escalonamento de médio prazo (*medium-term*)
  - Escalonamento de longo prazo (*long-term*)

Escalonamento de curto prazo

- Técnica mais comum e, por vezes, a única disponível

Escalonamento de curto prazo

- Técnica mais comum e, por vezes, a única disponível
- *Dispatcher* invocado com frequência muito elevada (ms)

Escalonamento de curto prazo

- Técnica mais comum e, por vezes, a única disponível
- *Dispatcher* invocado com frequência muito elevada (ms)
- Tem de executar muito rapidamente (100ms ou menos)

Escalonamento de curto prazo

- Técnica mais comum e, por vezes, a única disponível
- *Dispatcher* invocado com frequência muito elevada (ms)
- Tem de executar muito rapidamente (100ms ou menos)
- Limita a complexidade de escolha do próximo processo a executar

Escalonamento de curto prazo

- Técnica mais comum e, por vezes, a única disponível
- *Dispatcher* invocado com frequência muito elevada (ms)
- Tem de executar muito rapidamente (100ms ou menos)
- Limita a complexidade de escolha do próximo processo a executar
- Se *dispatcher* demorar 10ms e for executado a cada 100ms $\rightarrow$ 10% CPU utilizado para escalonamento!

Mudança de contexto

- Retrata o procedimento de substituição do processo ativo (em execução pelo CPU)

Mudança de contexto

- Retrata o procedimento de substituição do processo ativo (em execução pelo CPU)
- Requer que se guarde o estado do processo actual e que se carregue o estado do novo processo a executar

Mudança de contexto

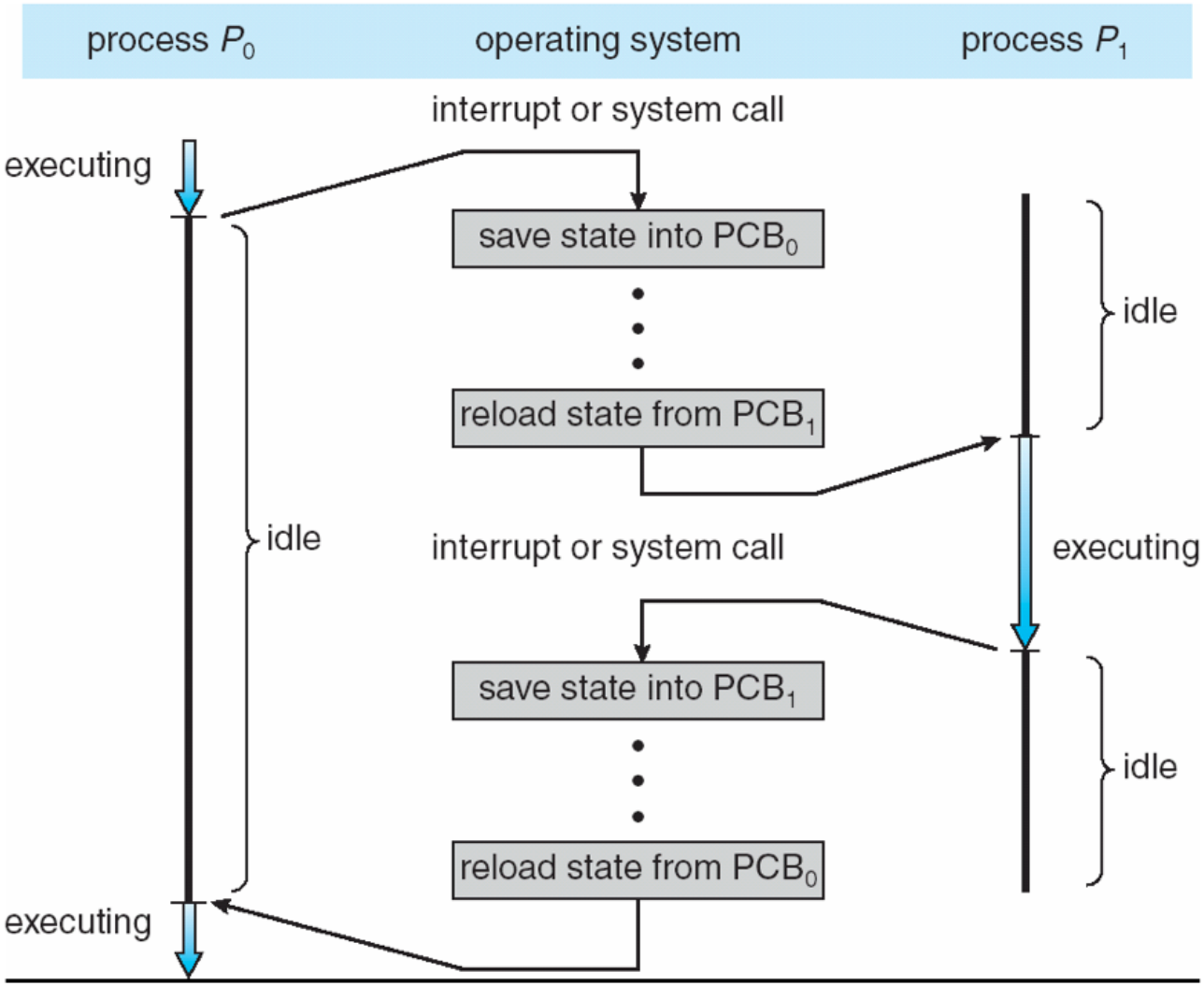
- Retrata o procedimento de substituição do processo ativo (em execução pelo CPU)
- Requer que se guarde o estado do processo actual e que se carregue o estado do novo processo a executar
- Demora entre 1 a 1000 μs (quando implica a reposição do estado dos registos do CPU)

Mudança de contexto

- Retrata o procedimento de substituição do processo ativo (em execução pelo CPU)
- Requer que se guarde o estado do processo actual e que se carregue o estado do novo processo a executar
- Demora entre 1 a 1000 μs (quando implica a reposição do estado dos registos do CPU)
- Constantes mudanças de contexto e maior necessidade de processamento requerem novos mecanismos e técnicas $\rightarrow$ Tarefas (*threads*)

Mudança de contexto

Troca de processo ativo num núcleo do CPU



Dispatcher

- Componente responsável pelo escalonamento do CPU

Dispatcher

- Componente responsável pelo escalonamento do CPU
- Escolhe o próximo processo a executar

Dispatcher

- Componente responsável pelo escalonamento do CPU
- Escolhe o próximo processo a executar
- Efetua a mudança de contexto

Dispatcher

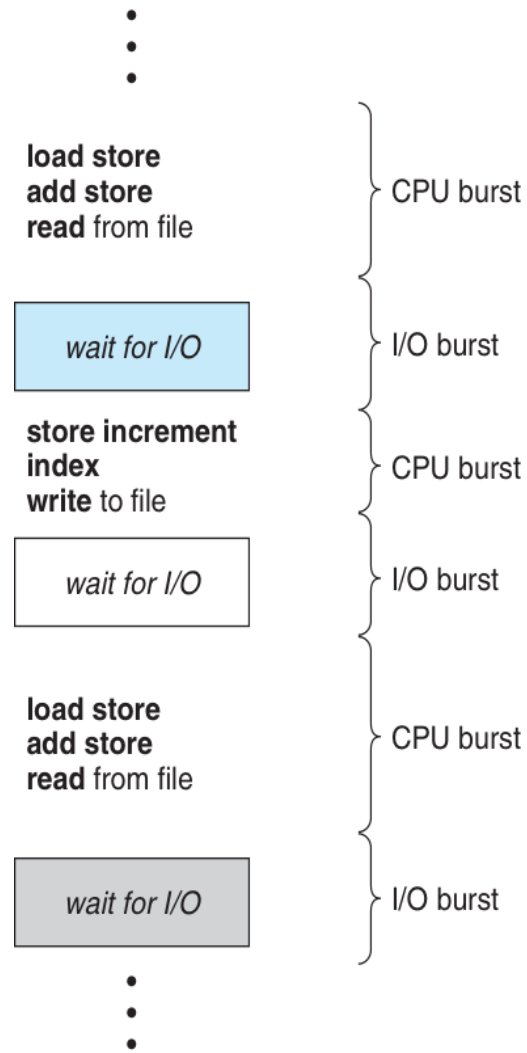
- Componente responsável pelo escalonamento do CPU
- Escolhe o próximo processo a executar
- Efetua a mudança de contexto
- Muda para o modo utilizador e posiciona-se na instrução seguinte

Dispatcher

- Componente responsável pelo escalonamento do CPU
- Escolhe o próximo processo a executar
- Efetua a mudança de contexto
- Muda para o modo utilizador e posiciona-se na instrução seguinte
- Tempo que demora chama-se *dispatcher latency*

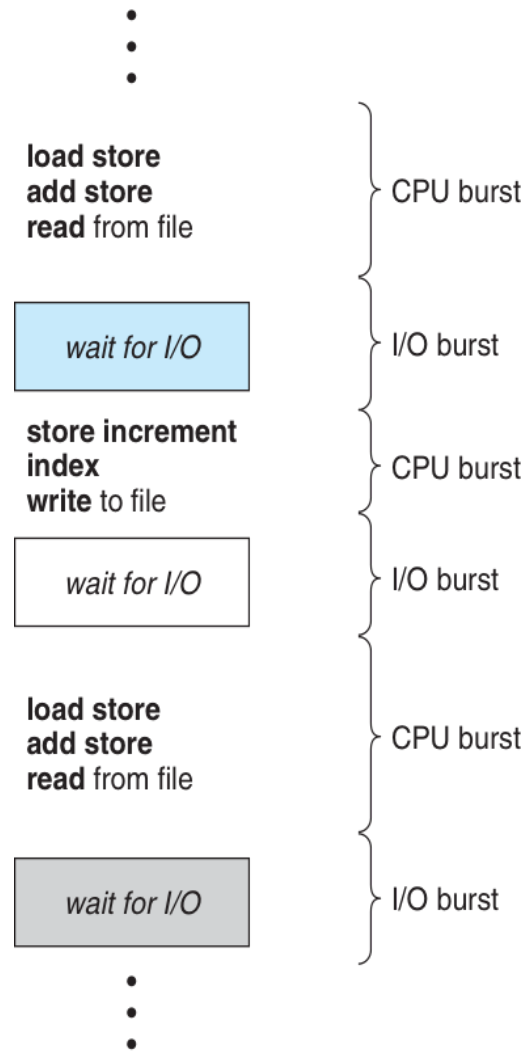
Ciclo CPU-IO

- Processo mudam entre estes dois estado



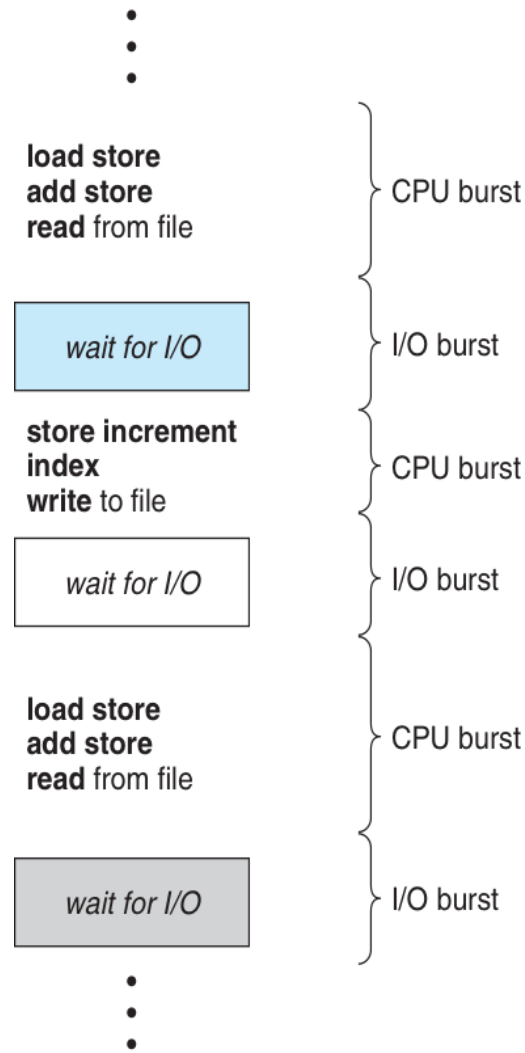
Ciclo CPU-IO

- Processo mudam entre estes dois estado
- Processo inicia com um ciclo de CPU (CPU *burst* inicial)



Ciclo CPU-IO

- Processo mudam entre estes dois estado
- Processo inicia com um ciclo de CPU (CPU *burst* inicial)
- Operação em alternância de ciclos de CPU com ciclos de E/S possibilita o escalonamento de processos



Tipos de escalonamento

O *dispatcher* executa quando:

Tipos de escalonamento

O *dispatcher* executa quando:

1. Processo passa para *Espera* (solicita E/S)

Tipos de escalonamento

O *dispatcher* executa quando:

1. Processo passa para *Espera* (solicita E/S)
2. Processo passa de *Execução* para *Pronto* (interrupção)

Tipos de escalonamento

O *dispatcher* executa quando:

1. Processo passa para *Espera* (solicita E/S)
2. Processo passa de *Execução* para *Pronto* (interrupção)
3. Processo passa de *Espera* para *Pronto* (E/S concluída)

Tipos de escalonamento

O *dispatcher* executa quando:

1. Processo passa para *Espera* (solicita E/S)
2. Processo passa de *Execução* para *Pronto* (interrupção)
3. Processo passa de *Espera* para *Pronto* (E/S concluída)
4. Processo termina

Tipos de escalonamento

O *dispatcher* executa quando:

1. Processo passa para *Espera* (solicita E/S)
2. Processo passa de *Execução* para *Pronto* (interrupção)
3. Processo passa de *Espera* para *Pronto* (E/S concluída)
4. Processo termina

Existem dois tipos de escalonamento de curto prazo

- Preemptivo (situações 2 e 3)
- Não preemptivo (situações 1 e 4)

Tipos de escalonamento

Preemptivo

- Permite a suspensão temporária da execução de um processo

Tipos de escalonamento

Preemptivo

- Permite a suspensão temporária da execução de um processo
- Sistema operativo mantém o controlo

Tipos de escalonamento

Preemptivo

- Permite a suspensão temporária da execução de um processo
- Sistema operativo mantém o controlo
- Requer mecanismos de controlo adicional
  - Suspender processo a meio da escrita do ficheiro implica bloquear acesso ao ficheiro até que suspensão termine

Tipos de escalonamento

Não preemptivo (ou cooperativo)

- Controlo é passado para o processo

Tipos de escalonamento

Não preemptivo (ou cooperativo)

- Controlo é passado para o processo
- Processo executa sistema operativo quando termina

Tipos de escalonamento

Não preemptivo (ou cooperativo)

- Controlo é passado para o processo
- Processo executa sistema operativo quando termina
- Escalonamento muito simples de implementar

Tipos de escalonamento

Não preemptivo (ou cooperativo)

- Controlo é passado para o processo
- Processo executa sistema operativo quando termina
- Escalonamento muito simples de implementar
- Não é adequado para os sistemas operativos atuais

Critérios de escalonamento

- Algoritmos de escalonamento do CPU podem ser avaliados sob vários critérios

Critérios de escalonamento

- Algoritmos de escalonamento do CPU podem ser avaliados sob vários critérios
- Dependendo do critério escolhido, um algoritmo pode ser considerado o melhor ou o pior

Critérios de escalonamento

- Algoritmos de escalonamento do CPU podem ser avaliados sob vários critérios
- Dependendo do critério escolhido, um algoritmo pode ser considerado o melhor ou o pior
- Não existe nenhum algoritmo ótimo

Critérios de escalonamento

Exemplos

- **Utilização do CPU** Quanto mais for utilizado o CPU, melhor. Valor em percentagem de tempo de utilização do CPU.

Critérios de escalonamento

Exemplos

- **Utilização do CPU** Quanto mais for utilizado o CPU, melhor. Valor em percentagem de tempo de utilização do CPU.
- **Rendimento** Número de processos concluídos por período de tempo.

Critérios de escalonamento

Exemplos

- **Utilização do CPU** Quanto mais for utilizado o CPU, melhor. Valor em percentagem de tempo de utilização do CPU.
- **Rendimento** Número de processos concluídos por período de tempo.
- **Tempo de vida** Tempo desde que um processo inicia a sua execução até que termina.

Critérios de escalonamento

Exemplos

- **Utilização do CPU** Quanto mais for utilizado o CPU, melhor. Valor em percentagem de tempo de utilização do CPU.
- **Rendimento** Número de processos concluídos por período de tempo.
- **Tempo de vida** Tempo desde que um processo inicia a sua execução até que termina.
- **Tempo de espera** Tempo gasto na fila de processos prontos e em filas de espera.

Critérios de escalonamento

Exemplos

- **Utilização do CPU** Quanto mais for utilizado o CPU, melhor. Valor em percentagem de tempo de utilização do CPU.
- **Rendimento** Número de processos concluídos por período de tempo.
- **Tempo de vida** Tempo desde que um processo inicia a sua execução até que termina.
- **Tempo de espera** Tempo gasto na fila de processos prontos e em filas de espera.
- **Tempo de espera inicial** Tempo gasto na fila de processos prontos pelo CPU *burst* inicial.

Conteúdos

Algoritmos de escalonamento de CPU

- *First-Come, First-Served* → Não preemptivo
- *Shortest-Job-First* → Não preemptivo
- *Shortest-Remaining-Time-First* → Preemptivo
- Prioridade → (Não) Preemptivo
- *Round-Robin* → Preemptivo

First-Come, First-Served (FCFS)

- Mais simples de todos
- Processos são executados por ordem de chegada
- Implementado com lista simplesmente ligada do tipo FIFO

First-Come, First-Served (FCFS)

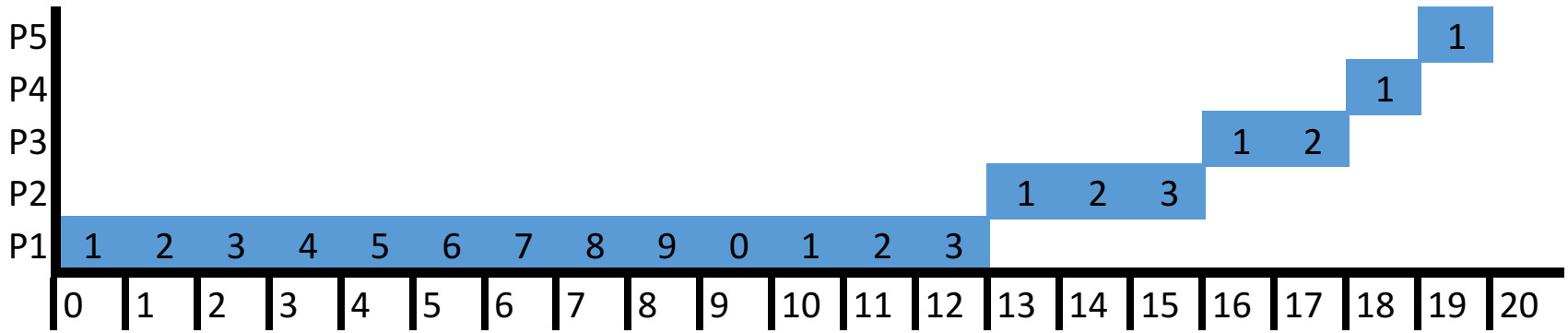
- Mais simples de todos
- Processos são executados por ordem de chegada
- Implementado com lista simplesmente ligada do tipo FIFO

#	Duração
1	13ms
2	3ms
3	2ms
4	1ms
5	1ms

First-Come, First-Served (FCFS)

- Mais simples de todos
- Processos são executados por ordem de chegada
- Implementado com lista simplesmente ligada do tipo FIFO

#	Duração
1	13ms
2	3ms
3	2ms
4	1ms
5	1ms



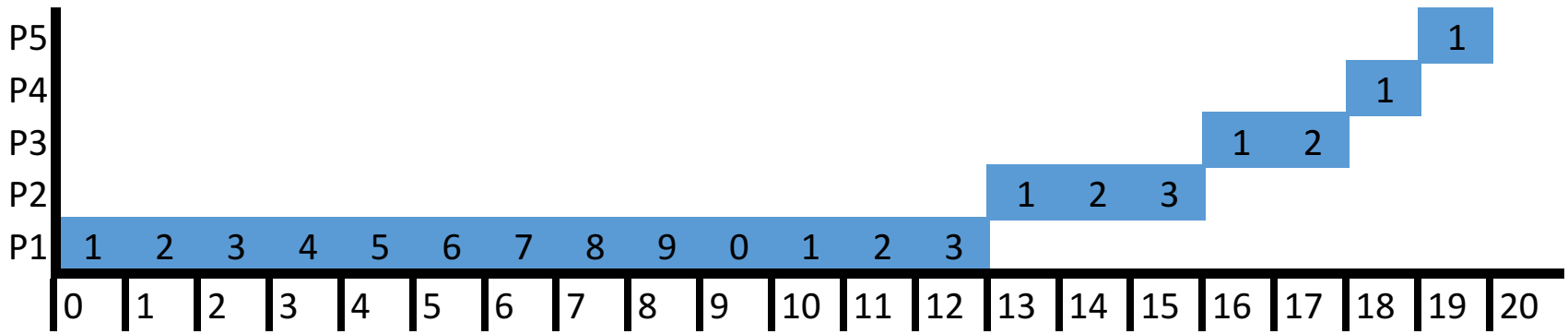
First-Come, First-Served (FCFS)

- Tempo médio de espera inicial pode ser muito elevado

$$TEI_i = InicioExec_i - Chegada_i$$

$$\overline{TEI} = \frac{\sum_{i=1}^{n} TEI_i}{n} = \frac{66ms}{5} = 13,2ms$$

#	Duração	TEI
1	13ms	0ms
2	3ms	13ms
3	2ms	16ms
4	1ms	18ms
5	1ms	19ms
		66ms



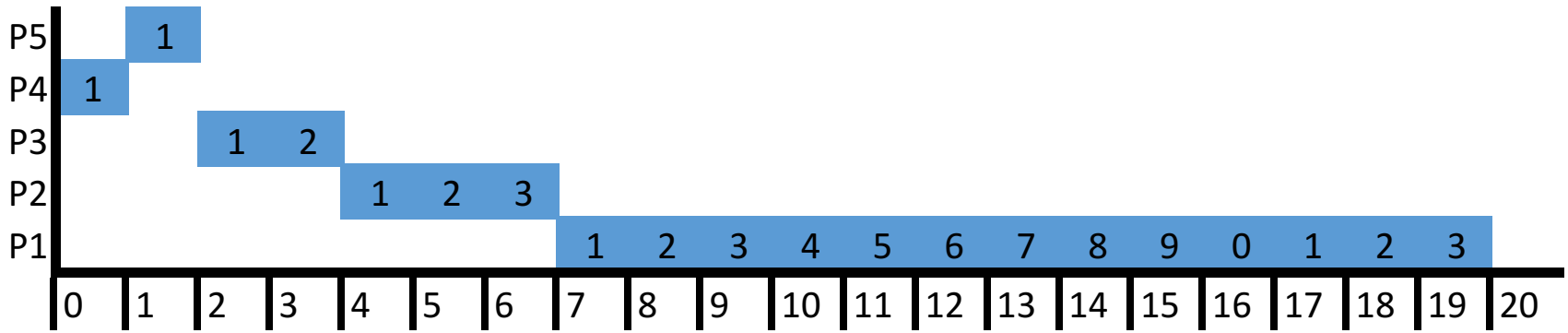
Shortest-Job-First (SJF)

- Primeiro executam os mais curtos
- Impacto dos curtos nos longos é menor
- Situações de igualdade usa-se FCFS

Shortest-Job-First (SJF)

- Primeiro executam os mais curtos
- Impacto dos curtos nos longos é menor
- Situações de igualdade usa-se FCFS

#	Duração
1	13ms
2	3ms
3	2ms
4	1ms
5	1ms

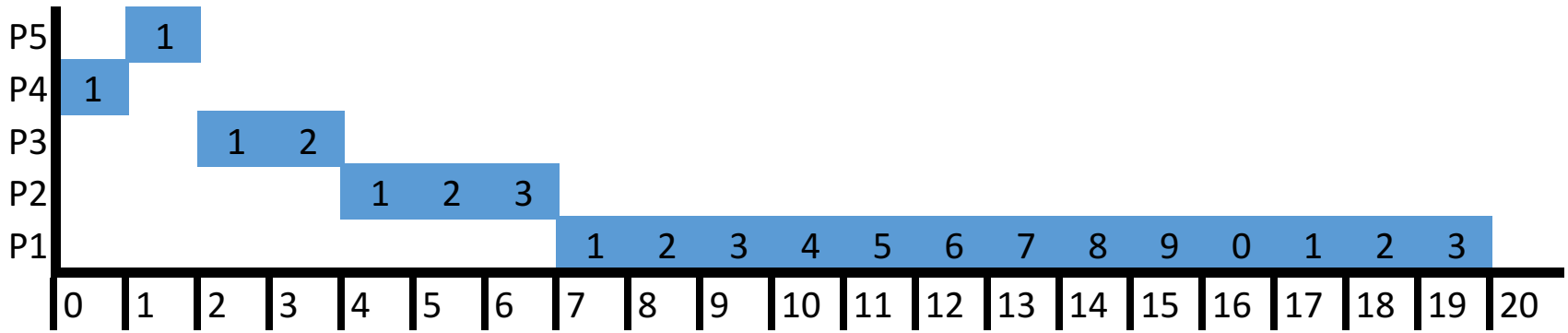


Shortest-Job-First (SJF)

- Obtém o mínimo tempo médio de espera (inicial)

$$\overline{TEI} = \frac{14}{5} = 2,8ms$$

#	Duração
1	13ms
2	3ms
3	2ms
4	1ms
5	1ms



Shortest-Job-First (SJF)

Estimação de tempos futuros

- Requer conhecimento antecipado da duração $\rightarrow$ Só por **estimativa**

Shortest-Job-First (SJF)

Estimação de tempos futuros

- Requer conhecimento antecipado da duração $\rightarrow$ Só por **estimativa**
- Duração do próximo ciclo de CPU é obtido com média exponencialmente ponderada durações dos ciclos anteriores

Shortest-Job-First (SJF)

Estimação de tempos futuros

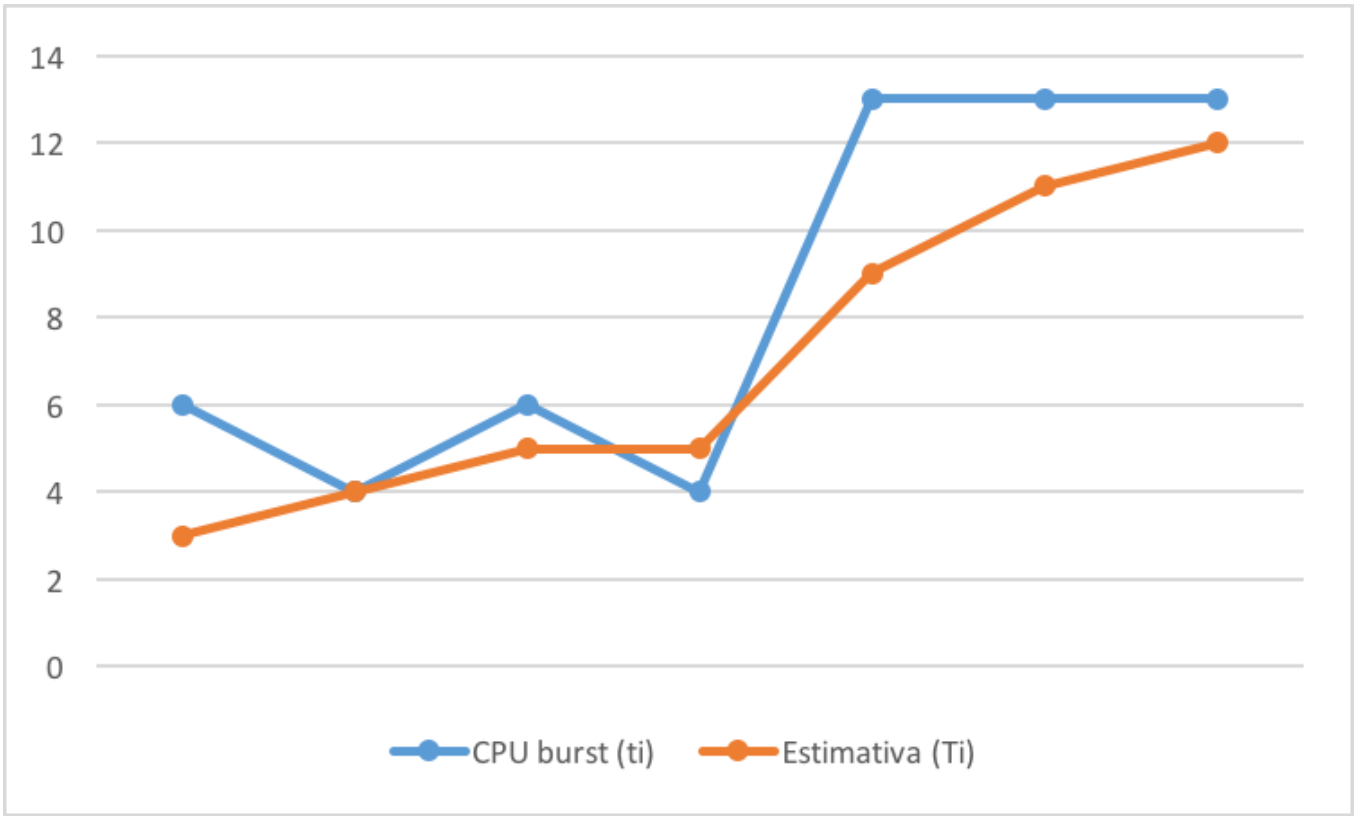
- Requer conhecimento antecipado da duração $\rightarrow$ Só por **estimativa**
- Duração do próximo ciclo de CPU é obtido com média exponencialmente ponderada durações dos ciclos anteriores
- Se t_n for duração do ciclo n, T_{n+1} a duração do ciclo seguinte e T_n o histórico, para um $\alpha, 0 \leq \alpha \leq 1$ temos

$$T_{n+1} = \alpha \cdot t_n + (1 - \alpha) \cdot T_n$$

Shortest-Job-First (SJF)

Exemplo de estimativa

CPU *burst* (t_i)	6	4	6	4	13	13	13
Estimativa (T_i)	3	4	5	5	9	11	12



$T_0 = 0, \alpha = 0,5$

Shortest-Remaining-Time-First (SRTF)

- SJF Preemptivo
- Tempo considerado é o tempo que resta até acabar
- Sempre que surge um processo com menor duração, é de imediato processado

Prioridade

- Existe como preemtivo e não preemptivo

Prioridade

- Existe como preemtivo e não preemptivo
- Atribui-se uma prioridade cada processo na criação

Prioridade

- Existe como preemtivo e não preemptivo
- Atribui-se uma prioridade cada processo na criação
- Processos executados por ordem de prioridade

Prioridade

- Existe como preemtivo e não preemptivo
- Atribui-se uma prioridade cada processo na criação
- Processos executados por ordem de prioridade
- Situações de igualdade, recorre-se ao FCFS

Prioridade

- Existe como preemtivo e não preemptivo
- Atribui-se uma prioridade cada processo na criação
- Processos executados por ordem de prioridade
- Situações de igualdade, recorre-se ao FCFS
- Requer esquema de prioridades

  ex.: 0 a 5 (5 é menos prioritário)

Prioridade

Definição de prioridades

- Definição interna
  - Efetuada pelo sistema operativo
  - Definida no momento de criação do processo

Prioridade

Definição de prioridades

- Definição interna
  - Efetuada pelo sistema operativo
  - Definida no momento de criação do processo
- Definição externa
  - Definidas pelo utilizador

Prioridade

Definição de prioridades

- Definição interna
  - Efetuada pelo sistema operativo
  - Definida no momento de criação do processo
- Definição externa
  - Definidas pelo utilizador
- Existe a possibilidade de um processo de baixa prioridade nunca ser executado (*starvation)*
  - Solução passa pelo envelhecimento

Round-Robin (RR)

- Processa por ordem de chegada (similar ao FCFS)

Round-Robin (RR)

- Processa por ordem de chegada (similar ao FCFS)
- Define um período máximo de execução
  - *Quantum* (fatia de tempo)

Round-Robin (RR)

- Processa por ordem de chegada (similar ao FCFS)
- Define um período máximo de execução
  - *Quantum* (fatia de tempo)
- Lista de processo prontos passa a ser circular

Round-Robin (RR)

- Processa por ordem de chegada (similar ao FCFS)
- Define um período máximo de execução
  - *Quantum* (fatia de tempo)
- Lista de processo prontos passa a ser circular
- Percorre a lista de processos prontos, permitindo que cada processo execute durante uma fatia de tempo

Round-Robin (RR)

- Processa por ordem de chegada (similar ao FCFS)
- Define um período máximo de execução
  - *Quantum* (fatia de tempo)
- Lista de processo prontos passa a ser circular
- Percorre a lista de processos prontos, permitindo que cada processo execute durante uma fatia de tempo
- Se processo não conseguir terminar, é colocado no final da lista de processos prontos

Round-Robin (RR)

- Eficiência deste algoritmo está totalmente dependente do tamanho da fatia de tempo (*quantum*)

Round-Robin (RR)

- Eficiência deste algoritmo está totalmente dependente do tamanho da fatia de tempo (*quantum*)
- Adequado para sistemas interativos se *quantum* for de 10 a 100ms

Round-Robin (RR)

- Eficiência deste algoritmo está totalmente dependente do tamanho da fatia de tempo (*quantum*)
- Adequado para sistemas interativos se *quantum* for de 10 a 100ms
- *Quantum* muito pequeno implica muitas trocas de contexto

Round-Robin (RR)

- Eficiência deste algoritmo está totalmente dependente do tamanho da fatia de tempo (*quantum*)
- Adequado para sistemas interativos se *quantum* for de 10 a 100ms
- *Quantum* muito pequeno implica muitas trocas de contexto
- *Quantum* muito grande, deixa de fazer sentido

Round-Robin (RR)

- Avaliação de algoritmos como o RR com base no TEI não é adequado

Round-Robin (RR)

- Avaliação de algoritmos como o RR com base no TEI não é adequado
- Alternativa calcular o tempo médio de vida (TV)

$$TV_i = FimExec_i - InicioExec_i$$

$$\overline{TV} = \frac{\sum_{i=1}^{n} TV_i}{n}$$

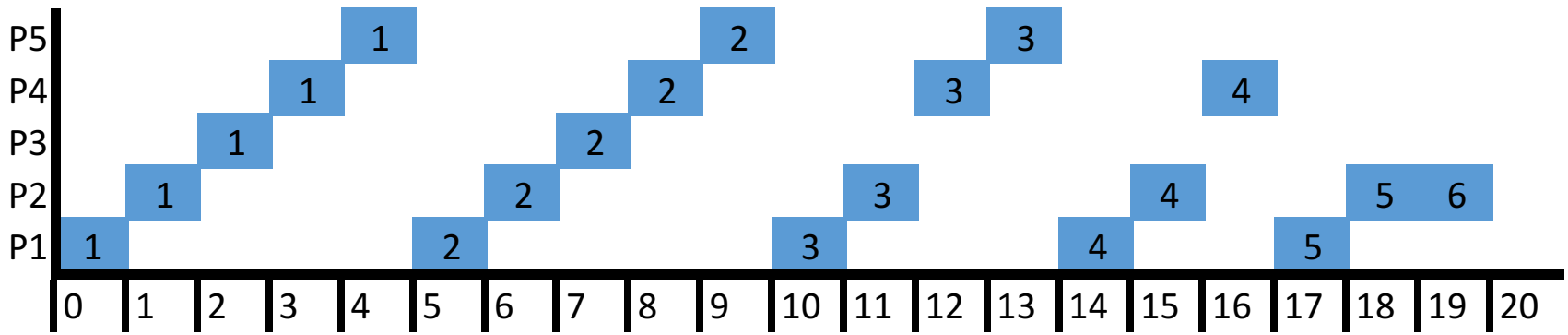
Round-Robin (RR)

Exemplo

- *Quantum=1*

$$\overline{TV} = \frac{77}{5} = 15,4$$

#	Duração
1	5ms
2	6ms
3	2ms
4	4ms
5	3ms



Bibliografia

- Baseado na bibliografia da unidade curricular.