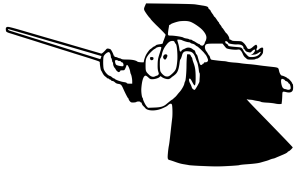
Gerência de processos - escalonamento de tarefas

EMB5632 - Sistemas Operacionais
Prof. Dr. Ricardo José Pfitscher
ricardo.pfitscher@ufsc.br

UFSC

Objetivos de aprendizagem



- Entender o conceito de escalonamento
- Conhecer os algoritmos e métricas de avaliação
- Construir diagramas de tempo de execução
- Avaliar criticamente os algoritmos

UFSC

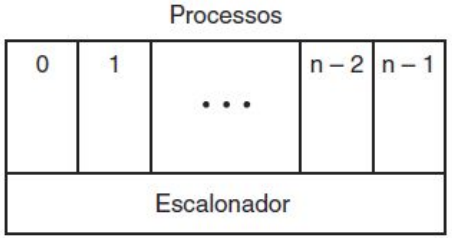
Cronograma

- Conceito de Escalonamento
- Comportamentos dos Processos
- Categoria dos algoritmos
- Objetivos do algoritmo de escalonamento
- Algoritmos de escalonamento

UFSC

Conceito de Escalonamento

- Sistemas multiprogramados:
  - Podem ter mais de um processo em estado de pronto
- Requisito básico: decidir qual o próximo processo a executar e por quanto tempo
  - O componente do SO que faz isso é o **escalonador** (*scheduler*)
    - Implementa um **algoritmo de escalonamento** (também chamada de política de escalonamento)
  - A diferença entre os algoritmos se dá nos objetivos O que eles devem priorizar?
    - Todos visam usar a CPU de forma eficiente
      - Chaveamentos de processos é muito custoso
        - Salvar, Estado atual do processo, Registradores, Mapa de memória, selecionar novo processo
  - Visão dos processos:



Tipos de tarefas

- Tarefas de tempo real
  - Exigem previsibilidade do tempo de execução
    - Controle de sistemas críticos (aeronaves, plantas industriais)
- Tarefas interativas
  - recebem eventos externos
    - editores de texto, navegadores internet, etc
- Tarefas em lote (batch)
  - sem requisitos temporais, sem interação com o usuário
    - Backup, antivírus, treinamento de IA, processamento de big data

Comportamento dos Processos [1/4]

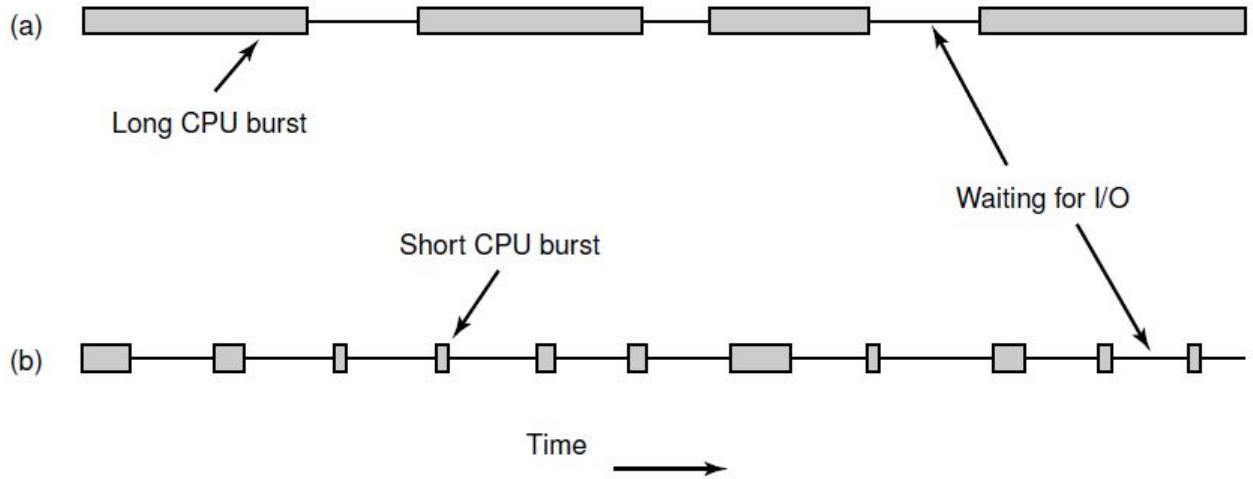
- Em geral, processos alternam ciclos de CPU com ciclos de E/S
  - Processo executa várias instruções de máquina e executa uma chamada de sistema solicitando um serviço do SO
    - Ex. ler um arquivo ou escrever nele, interagir com o usuário
- Existem duas grandes classes de processos:
  - Orientados a CPU (*CPU-Bound)*
  - Orientados a E/S (*IO-Bound)*
  - Também existem processos que equilibram (híbridos)

Comportamento dos Processos [2/4]

(a) Um processo orientado a CPU

(b) Um processo orientado a E/S

(a) Quando está em E/S o processo não executa em CPU, fica bloqueado



Comportamento dos Processos [3/4]

- Quando escalonar
  - Situações em que o escalonador do SO é invocado:
    - Na criação de um processo
      - Executa pai ou filho?
    - No encerramento de um processo
      - Outro processo deve ser escolhido
    - Quando um processo bloqueia
      - Qual dos prontos deve ser escolhido
    - Quando ocorre uma interrupção de E/S
      - O processo bloqueado fica pronto para execução
    - Quando ocorre interrupção de relógio
      - Escalonamento **preemptivo**
        - Execução por tempo máximo determinado

Comportamento dos Processos [4/4]

- Escalonamento **não preemptivo** e **preemptivo**
- Não preemptivo:
  - Processo só para de executar na CPU se quiser
    - Invocação de uma chamada de sistema
    - Liberação voluntária da CPU
- Preemptivo
  - Processo pode perder a CPU mesmo contra a sua vontade
    - Preempção por tempo (mais comum)
    - Preempção por prioridade
      - Chegada de um processo mais prioritário
    - As possibilidades do não preemptivo também se enquadram

Objetivos do algoritmo de escalonamento

- Que critérios podem ser usados para avaliar um algoritmo de escalonamento?
  - **Vazão (throughput):** Número de Jobs processados por tempo
  - **Tempo de retorno (execução)**: Tempo entre o momento em que uma *tarefa* foi submetida até o tempo que foi terminada ➔ tempo de retorno médio
  - **Tempo de resposta:** Tempo entre emissão de um comando e obtenção de um resultado
    - Entre digitar uma letra no teclado e ela aparecer na tela
  - **Tempo de espera**: tempo total perdido pela tarefa na fila de espera (pronta para executar, mas não executou)
  - **Justiça:** equidade da distribuição do processador entre tarefas de mesma prioridade
  - **Eficiência:** utilização do processador na execução de tarefas de usuário
    - muito tempo em troca de contexto não é interessante

UFSC

Algoritmos de escalonamento [1/12]

- Escalonamento para sistemas em lote
  - Primeiro a chegar, primeiro a ser servido (FCFS)
  - Job mais curto primeiro (SJF)
  - Próximo de menor tempo restante (SRTN)
- Escalonamento para sistemas interativos
  - Alternância circular (*round-robin*)
  - Por Prioridades
  - Filas Múltiplas
  - Fração Justa

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5

UFSC

Algoritmos de escalonamento [2/12]

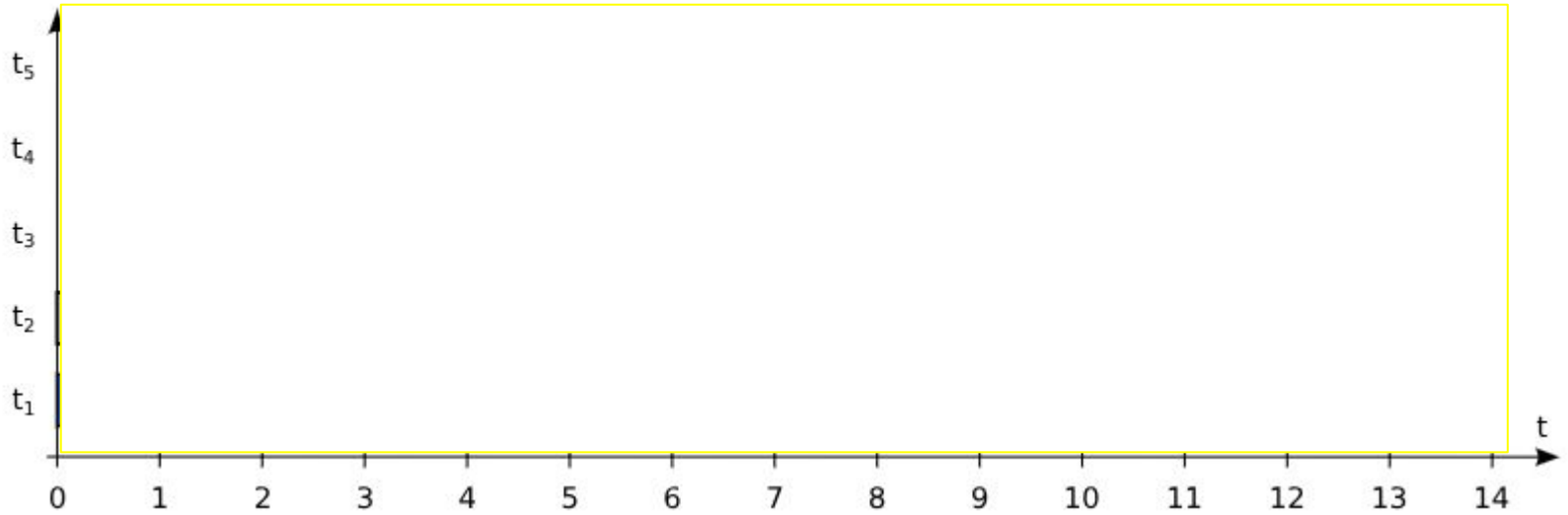
- FCFS [1/2]
  - Processos são atendidos por ordem de chegada
    - Primeiro a chegar, primeiro a ser servido
    - *First-come, first-served (FCFS)*
  - O processo escalonado usa a CPU por quanto tempo quiser ➜ **não preemptivo**
    - Até encerrar, bloquear, ou entregar o controle
  - Simples de implementar
  - Não diferencia processos orientados a CPU e orientados a E/S
    - Isso pode prejudicar os de E/S

Exemplo - FCFS

Tempo de espera e tempo de retorno médio?

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5



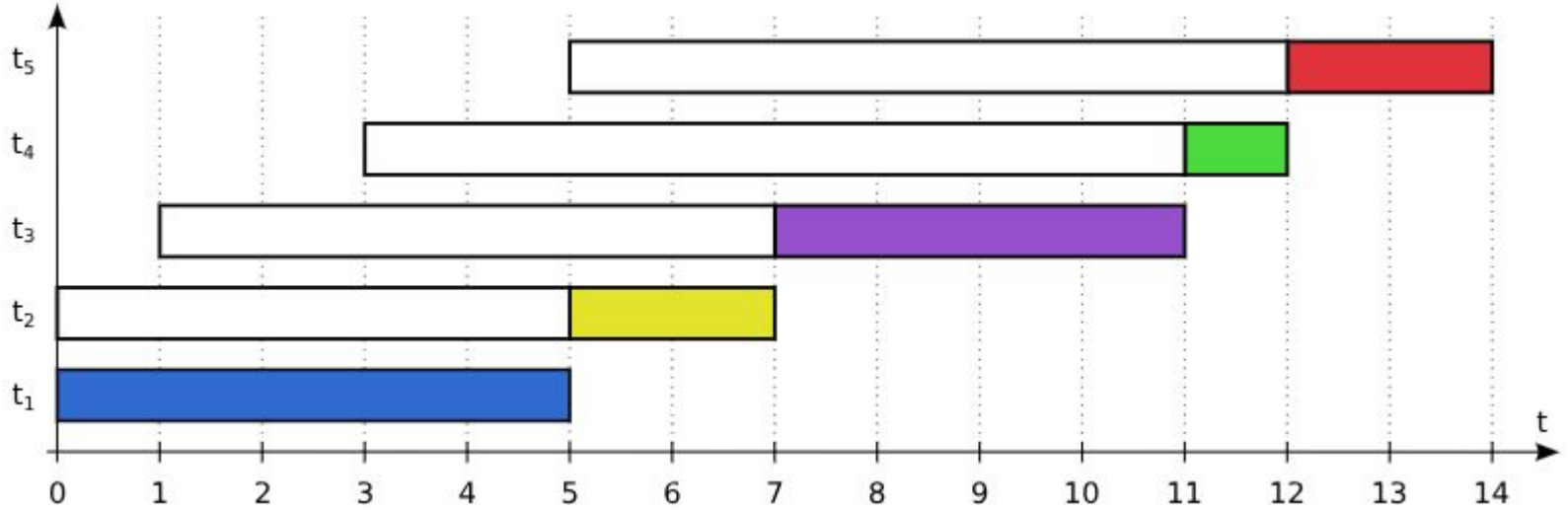
Fonte: (Maziero, 2019)

Exemplo - FCFS

Tempo de espera médio = 5,2s
Tempo de retorno médio = 8,0s

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5



Fonte: (Maziero, 2019)

Algoritmos de escalonamento [3/12]

- FCFS [2/2]
  - Imagine a situação:
    - Um processo CPU-bound que executa 1 segundo por vez (P1)
    - Muitos processos orientados a E/S que usem pouco tempo de CPU, mas que precisem realizar cada um mil leituras de disco (P2)
    - Depois que P1 executa por 1 segundo, bloqueia a espera de E/S
    - P2 podem executar e iniciam leituras de disco
    - P1 obtém E/S (desbloqueia), executa por mais 1 segundo
  - Resultado:
    - Cada P2 lê um bloco por segundo e, demorará mil segundos para terminar
    - Preempção por tempo resolveria, a cada dez milisegundos uma leitura □ 10 segundos para as mil leituras

Algoritmos de escalonamento [4/12]

- SJF (*Short Job First*)
  - Os processos mais curtos são atendidos primeiro
    - Mais curto menor tempo de CPU
  - **Não preemptivo**
  - Menor tempo médio de retorno
  - Premissas:
    - Todas as tarefas devem estar disponíveis simultaneamente
    - A duração dos ciclos de CPU deve ser conhecida

Como saber o tempo de execução?

Algoritmos de escalonamento [4/12]

- SJF (*Short Job First*)
  - Os processos mais curtos são atendidos primeiro
    - Mais curto menor tempo de CPU
  - **Não preemptivo**
  - Menor tempo médio de retorno
  - Premissas:
    - Todas as tarefas devem estar disponíveis simultaneamente
    - A duração dos ciclos de CPU deve ser conhecida

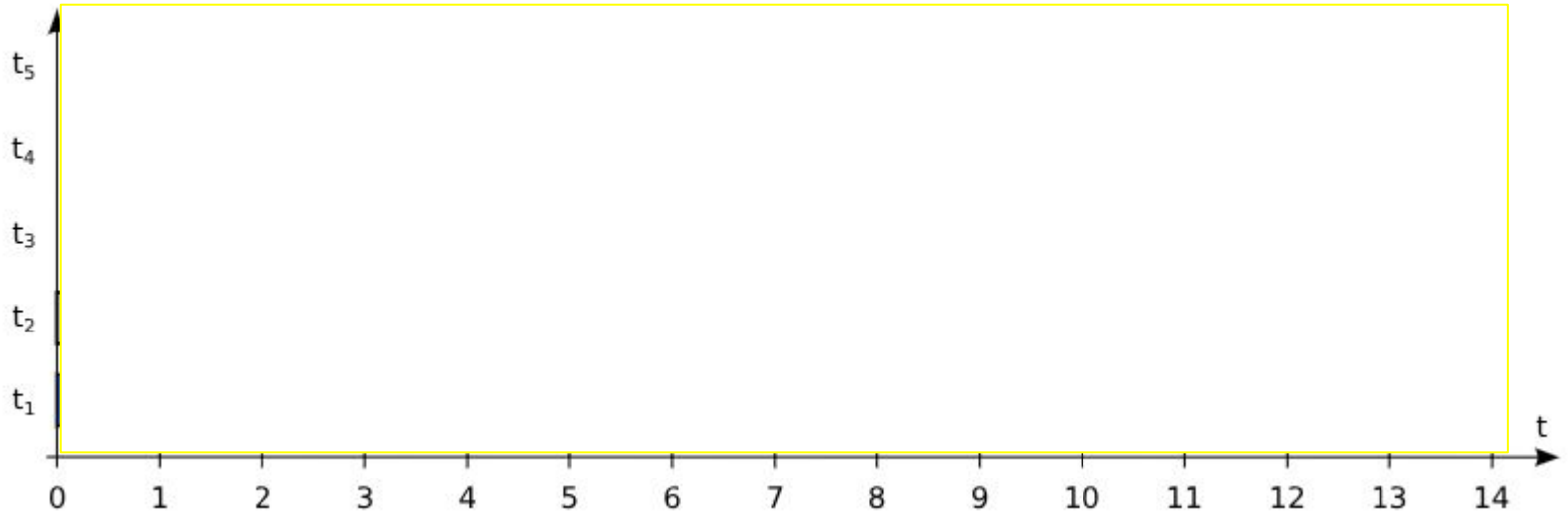
Como saber o tempo de execução?
R: histórico (impraticável)

Exemplo - SJF

Tempo de espera e tempo de retorno médio?

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5



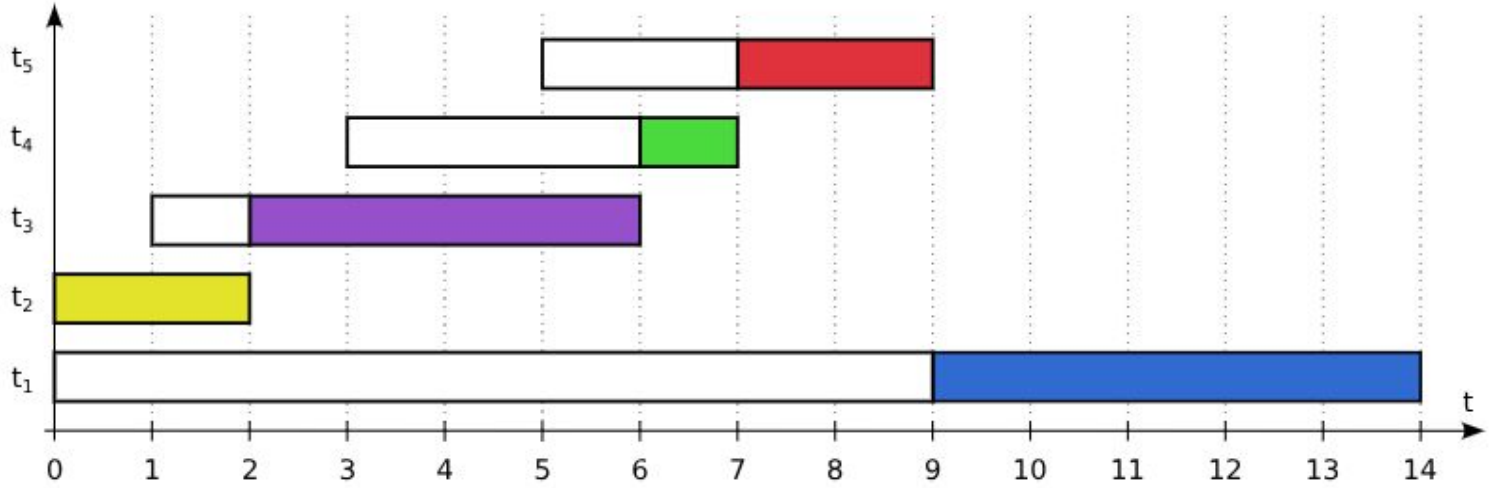
Fonte: (Maziero, 2019)

Exemplo - SJF

Tempo de espera médio = 5,8s
Tempo de retorno médio = 3,0s

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5



Fonte: (Maziero, 2019)

Algoritmos de escalonamento [5/12]

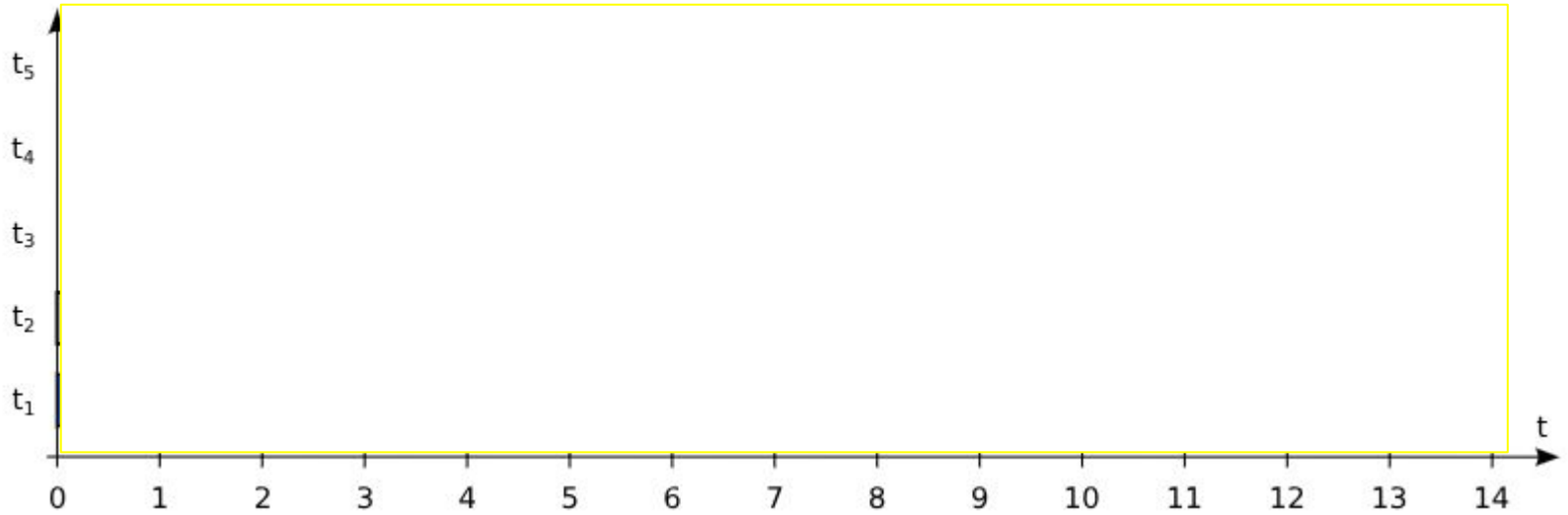
- Próximo de menor tempo restante
  - *Shortest remaining time next (SRTN) - shortest remaining time first (SRTF)*
  - Variante **preemptiva** do SJF
  - Quando chega um novo processo, seu tempo é comparado com o tempo restante do processo que está executando
    - Se for menor, o novo processo sofre preempção e o novo processo é escalonado em seu lugar
  - Garante bom desempenho para jobs curtos
  - Também requer tempos conhecidos de CPU

Exemplo - SRTF

Tempo de espera e tempo de retorno médio?

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5



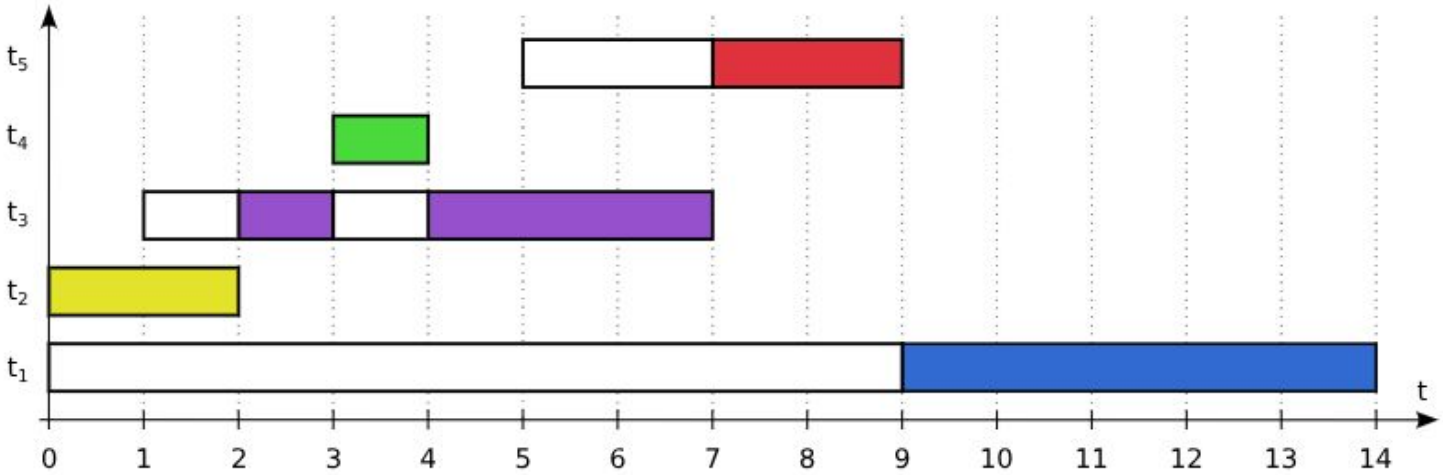
Fonte: (Maziero, 2019)

Exemplo - SRTF

Tempo de espera médio = 2,6s
Tempo de retorno médio = 5,4s

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5



Fonte: (Maziero, 2019)

Algoritmos de escalonamento [6/12]

- Alternância circular (*round-robin)* [1/2]
  - Cada processo que ganha a CPU executa por um determinado tempo (***quantum***)
  - Se ele não liberar a CPU ao final do *quantum* ele perde o processador e volta para a fila de prontos
    - **Algoritmo preemptivo**

Algoritmos de escalonamento [7/12]

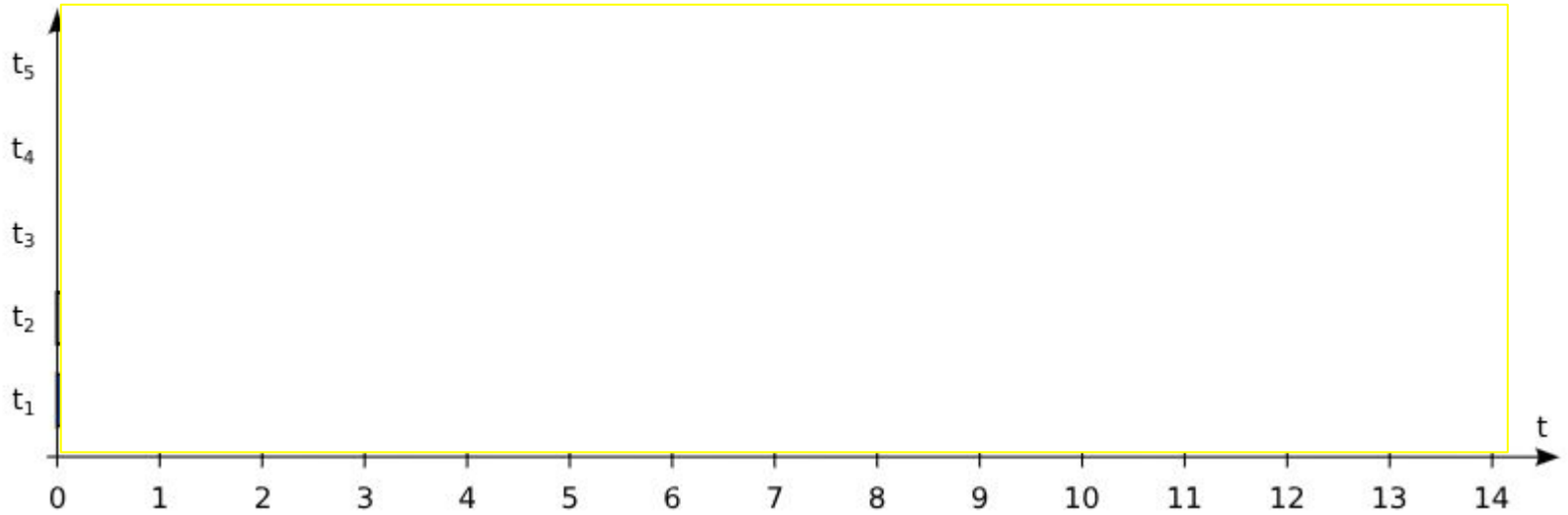
- Alternância circular (*round-robin)* [2/2]
  - Determinando o *quantum*
    - A decisão sobre o tamanho é a mais importante neste algoritmo
    - Quanto **menor** o *quantum*, **maior** o overhead
      - Tempo para chaveamento de contexto, se aproxima do tempo de execução
    - Quanto **maior** o *quantum*, **pior** o tempo de resposta
      - Ocorrem menos preempções
      - Processo demora mais a ser escalonado
      - Prejudica processos orientados a E/S
    - Na prática, utiliza-se entre 20 e 100 ms

Exemplo - RR - quantum=2

Tempo de espera e tempo de retorno médio?

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5



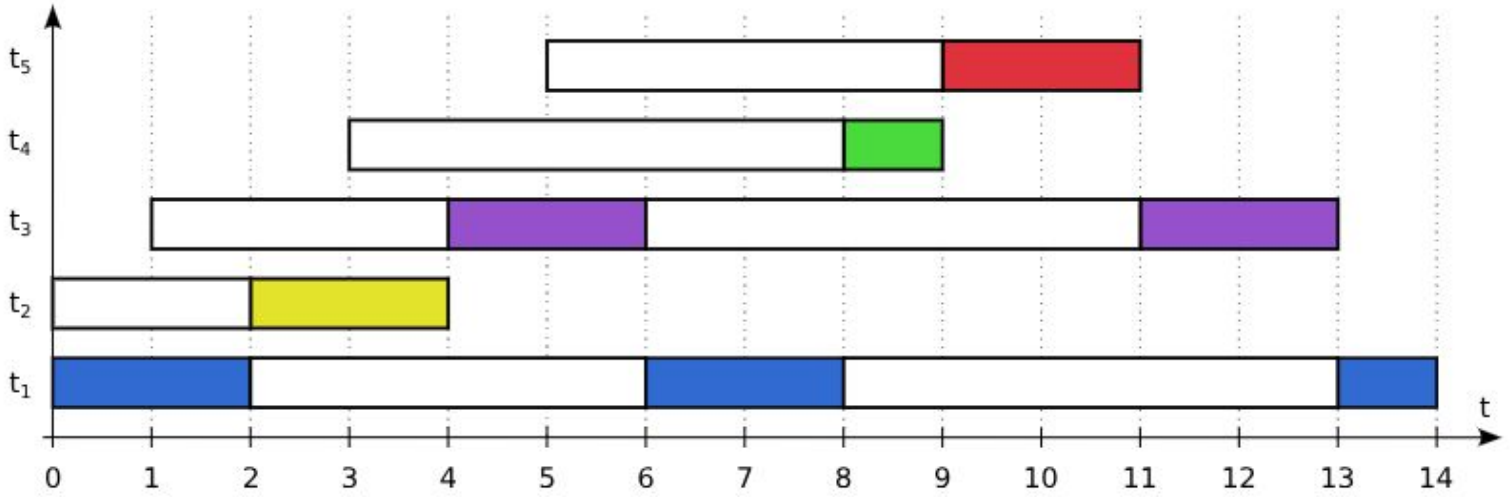
Fonte: (Maziero, 2019)

Exemplo - RR - quantum=2

Tempo de espera médio = 5,6s
Tempo de retorno médio = 8,4s

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5



Fonte: (Maziero, 2019)

Algoritmos de escalonamento [8/12]

- Escalonamento por prioridades [1/3]
  - Nem todos os processos têm a mesma prioridade
    - Um antivírus não deve prejudicar a exibição de um vídeo
  - Características
    - Cada processo recebe uma prioridade
    - O processo de maior prioridade executa
    - Para evitar que processos mais prioritários executem indefinidamente, a prioridade pode ser periodicamente reduzida
    - Prioridade preemptiva vs não-preemptiva
      - Preemptiva: Se alguém de maior prioridade chegar **Executa**
      - Não–Preemptiva (cooperativo): Se alguém de maior prioridade chegar **Espera**

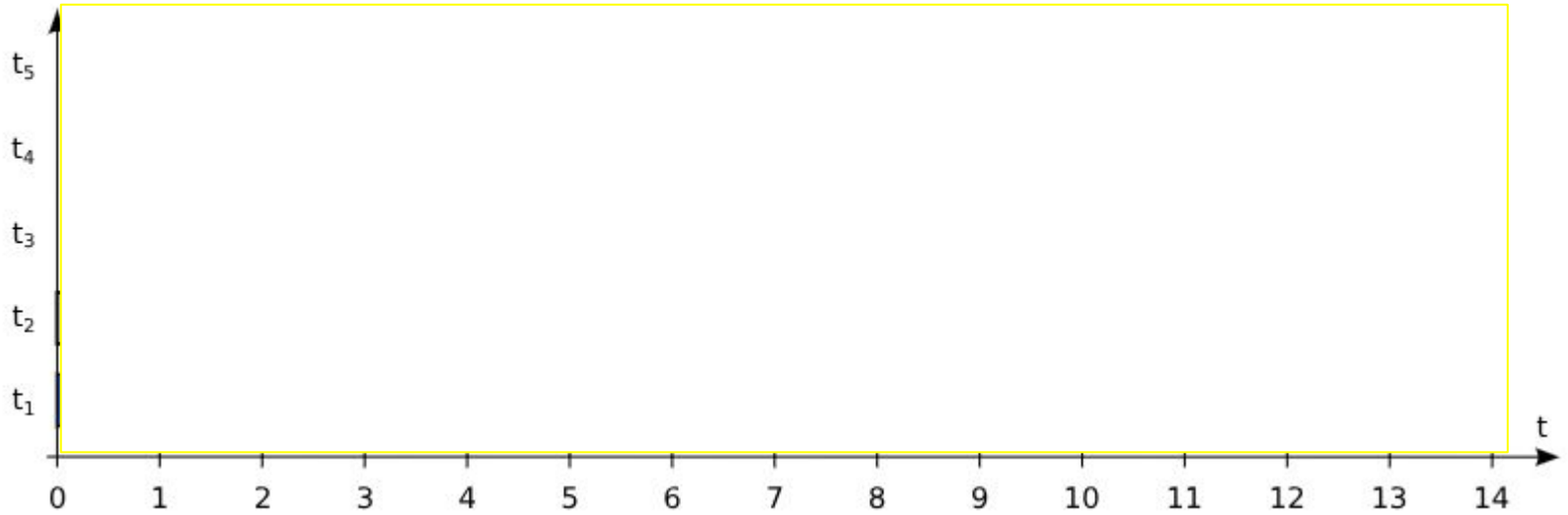
Exemplo - Prioridade_c

Tempo de espera e tempo de retorno médio?

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5

> valor ➜ > prioridade



Fonte: (Maziero, 2019)

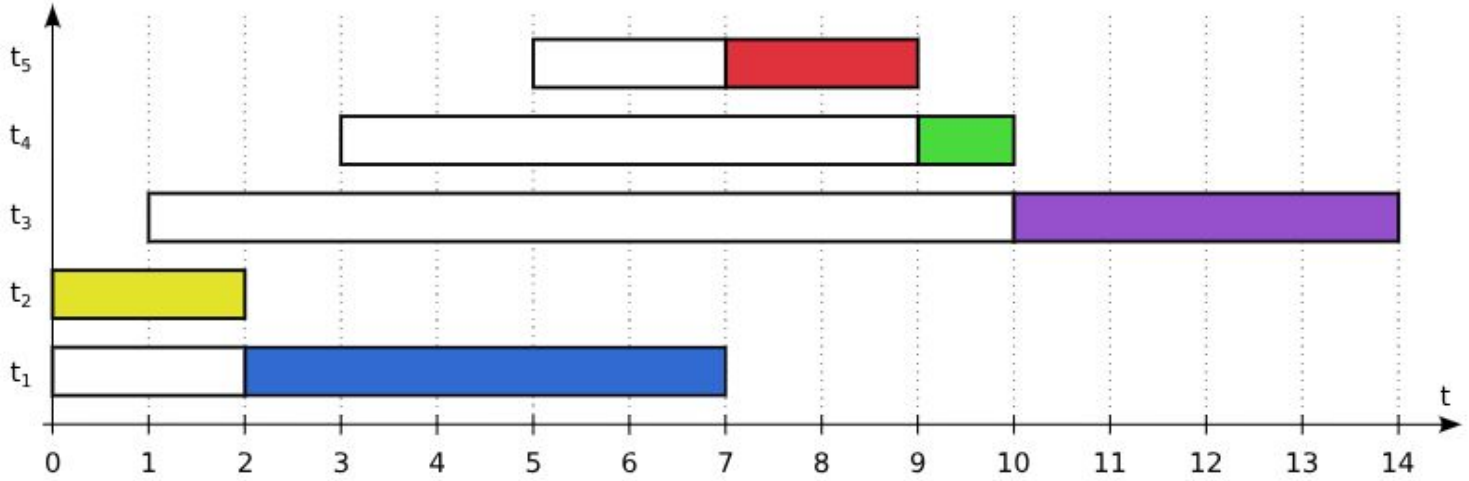
Exemplo - Prioridade_c

Tempo de espera médio = 3,8s
Tempo de retorno médio = 6,6s

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5

> valor ➜ > prioridade



Fonte: (Maziero, 2019)

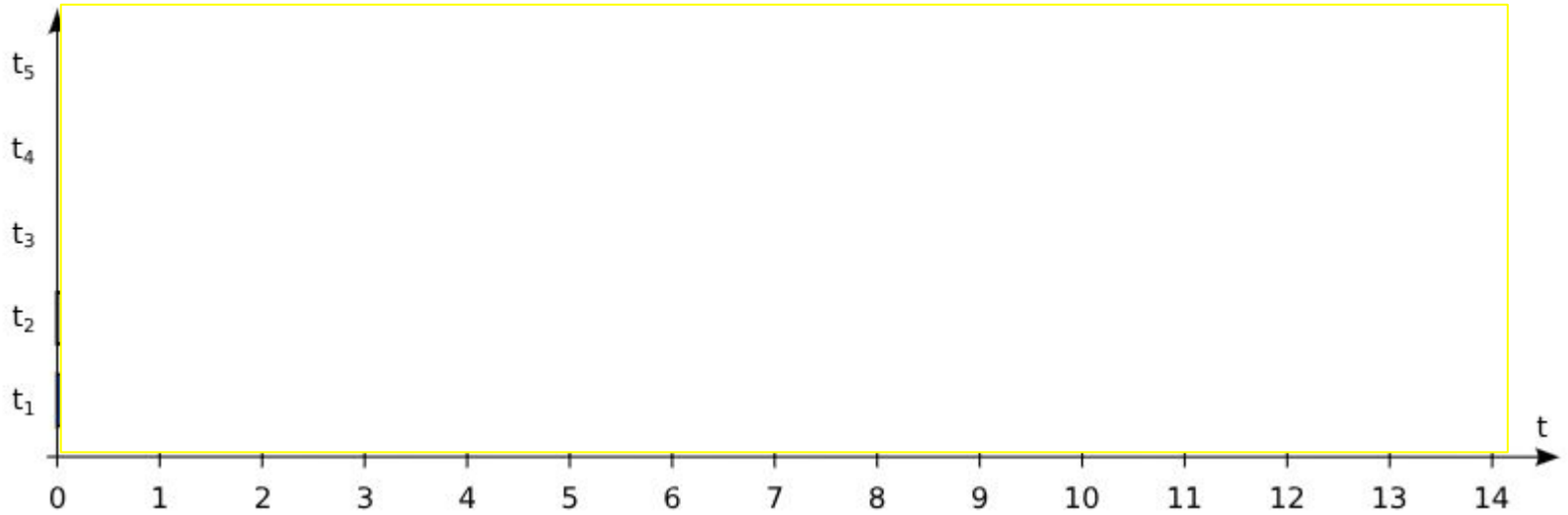
Exemplo - Prioridade_p

Tempo de espera e tempo de retorno médio?

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5

> valor ➜ > prioridade



Fonte: (Maziero, 2019)

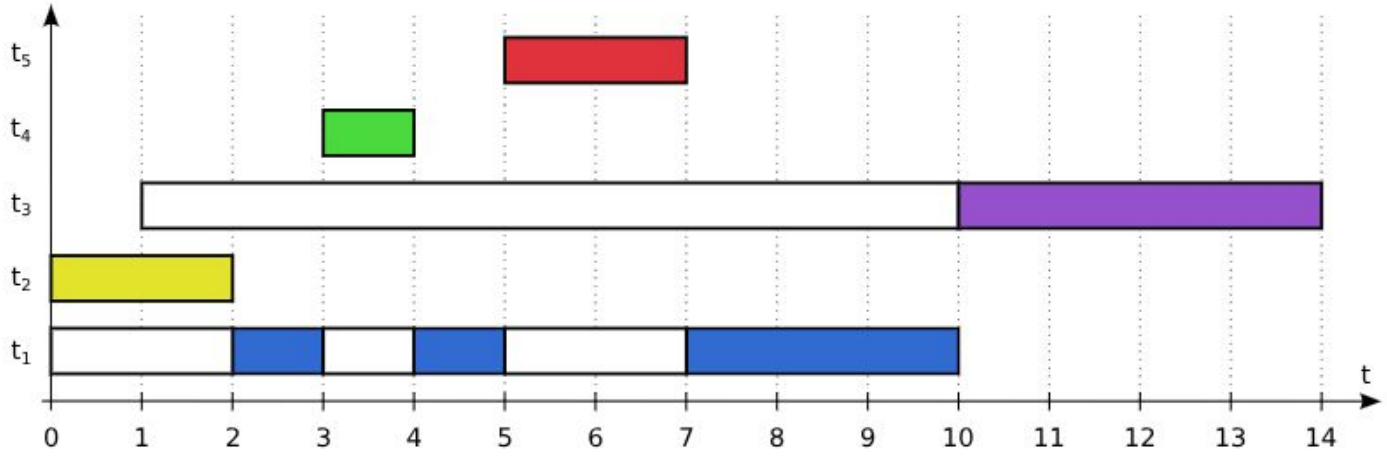
Exemplo - Prioridade_p

Tempo de espera médio = 2,8s
Tempo de retorno médio = 5,6s

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5

> valor ➜ > prioridade



Fonte: (Maziero, 2019)

Algoritmos de escalonamento [9/12]

- Escalonamento por prioridades [2/3]
  - Prioridades podem ser estáticas ou **dinâmicas**
    - Igual a fração do último *quantum* usado
      - Ex.: Definir uma prioridade 1/*f*, onde *f* é a fração de uso do *quantum*. Em uma última execução um processo utilizou 2ms de um *quantum* de 100ms, na próxima execução sua prioridade será 50
  - Envelhecimento: a prioridade das tarefas aumenta a cada rodada por um fator fixo
  - É comum agrupar os processos em classes de prioridades
    - Prioridade entre as classes
    - *Round-robin* dentro de cada classe

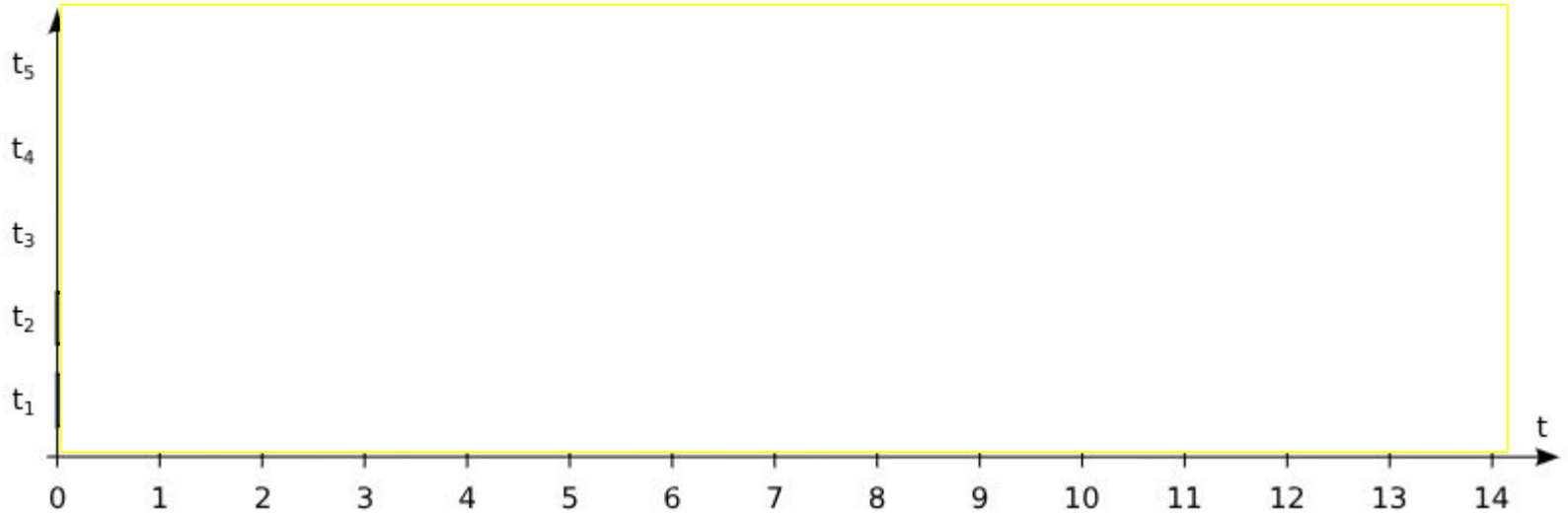
Exemplo - Prioridade_d

Tempo de espera e tempo de retorno médio?

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5

> valor ➜ > prioridade



Fonte: (Maziero, 2019)

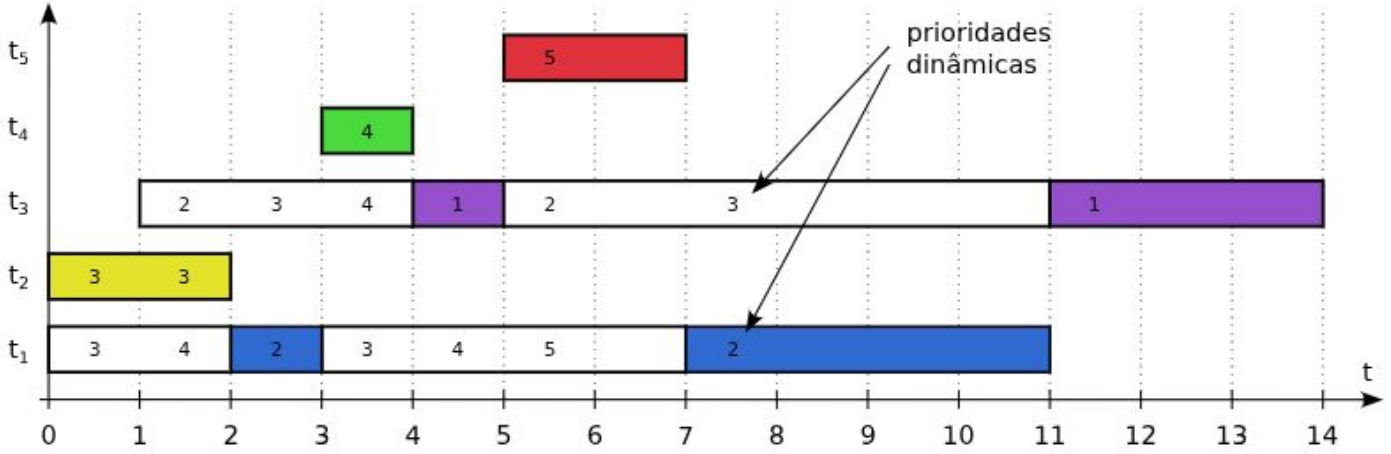
Exemplo - Prioridade_d

Tempo de espera médio = 3,0s
Tempo de retorno médio = 5,8s

Exemplo base

Tarefa	t_1	t_2	t_3	t_4	t_5
Ingresso	0	0	1	3	5
Duração	5	2	4	1	2
Prioridade	2	3	1	4	5

> valor ➜ > prioridade

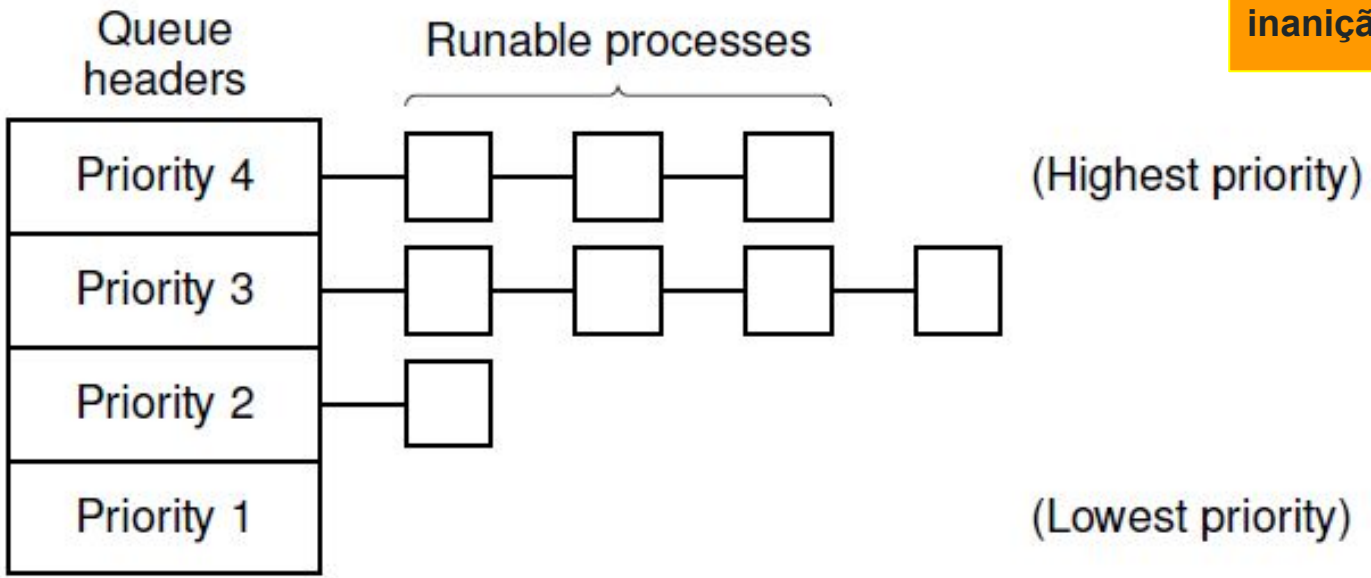


Fonte: (Maziero, 2019)

Algoritmos de escalonamento [10/12]

- Escalonamento por prioridades [3/3]

é necessário ajustar dinamicamente as prioridades para que processos nas classes mais baixas não morram por inanição (*starvation*)



Algoritmos de escalonamento [11/12]

- Filas Múltiplas
  - Variação do escalonamento por prioridades
  - Cada classe de prioridade tem um *quantum*
    - Classes mais prioritárias têm *quantum* menor
    - Se o *quantum* acaba antes que o processo conclua o ciclo de CPU, ele muda a prioridade (desce de fila)
  - Reduz a quantidade de chaveamento de contexto para processos orientados a CPU
  - Processos interativos têm alta prioridade
    - Usuários de processos em lote descobriram que poderiam acelerar seus processos usando o **terminal** (tem alta prioridade)

Algoritmos de escalonamento [12/12]

- Algoritmos de escalonamento “reais”
  - Tomando como referência o Linux
  - As tarefas são divididas em classes de escalonamento (Filas múltiplas)
    - SCHED_DEADLINE ➜ as tarefas tem tempo limite de execução (EDF)
    - SCHED_FIFO ➜ tarefas executam em FCFS
    - SCHED_RR ➜ preempção por tempo, quantum = 100ms
    - SCHED_NORMAL ➜ classe padrão, prioridades dinâmicas e RR (q in [0,75 e 6 ms]
    - SCHED_BATCH ➜ similar ao anterior, tarefas são CPU-Bound, q= 1,5 segundos
    - SCHED_IDLE ➜ menor prioridade, só executa se não tiver outra tarefa

SCHED_DEADLINE> SCHED_FIFO/RR > SCHED_NORMAL > SCHED_BATCH > SCHED_IDLE

Exercício

- Considere os seguintes processos e os respectivos tempos de CPU, todos chegando no instante 0, na ordem A-B-C-D-E:

Processo	Tempo de CPU	Tempo de Disco	Prioridade
A	10	1	3
B	1	2	1
C	2	1	3
D	1	2	4
E	5	1	2

A) Elaborar diagramas temporais usando FCFS, SJF, prioridade não preemptiva (maior valor maior prioridade) e *round-robin* com *quantum = 4.*

B) Qual o tempo de retorno de cada algoritmo?

C) Qual o tempo de espera de cada processo em cada algoritmo?

D) Quais algoritmos representam o maior e o menor tempo de espera médio, quais esses tempos?

Referências

- *MAZIERO, C. Sistemas Operacionais: Conceitos e Mecanismos. Editora da UFPR, 2019. 456 p. ISBN 978-85-7335-340-2*
- Andrew S. Tanenbaum. Sistemas Operacionais Modernos, 3a Edição. Capítulo 2.Pearson Prentice-Hall, 2009.