Escalonamento

- *Workload*
- Problema
- Métricas no escalonamento
- Primeiro a entrar, primeiro a executar (FIFO)
- Tarefa mais curta primeiro (SJF)
- Tarefa mais próxima de ficar completa primeiro (STCF)
- Nova métrica: tempo de resposta
- *Round Robin*
- Adicionando E/S
- Questões

Workload

- A carga de um sistema *(workload)* é definida como a soma dos processos que correm num sistema.
- Assumimos situações não realistas para começar no que designamos por “Operação de escalonamento totalmente operacional”:
  - **1ª assunção**:
    - Todas as tarefas (*jobs*) executadas têm a mesma duração;
  - **2ª assunção**:
    - Todos as tarefas estão disponíveis para executar ao mesmo tempo;
  - **3ª assunção**:
    - Uma vez iniciada, a tarefa é executada até ao fim;
  - **4ª assunção**:
    - Todas as tarefas usam apenas CPU (i.e., não há eventos de E/S);
  - **5ª assunção**:
    - O tempo despendido por cada tarefa em execução não é conhecido;

Métricas no escalonamento

- São necessárias métricas que permitam comparar diferentes políticas de escalonamento.
  - Usamos uma métrica designada **Tempo de retorno** (***turnaround time***):
    - Tempo de completação de um trabalho(*) subtraído ao tempo de chegada:

$$T_{ta} = T_{completion} - T_{arrival}$$

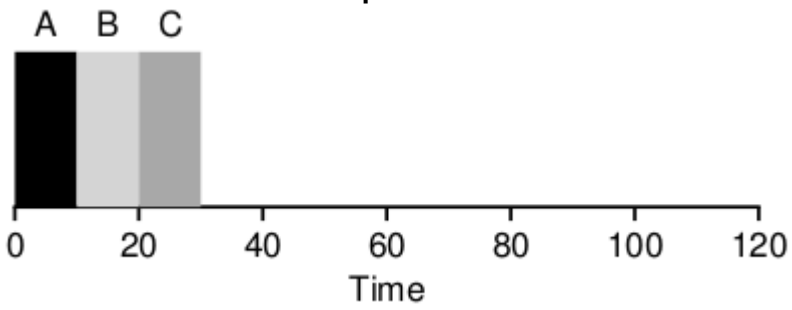
    - Assumindo que todos os trabalhos estão disponíveis ao mesmo tempo (***tempo de chegada***), $T_{arrival} = 0$.
  - O ***tempo de retorno*** (T_{ta})é a métrica de desempenho – performance, do sistema [será utilizada primeiro]
  - Outra métrica: a **equidade** – ***fairness***
    - *Certos trabalhos são impedidos de serem executados, apesar de otimizar o desempenho! A medida de equidade mede este aspeto.*

(*) Iremos utilizar tarefa ou trabalho. Em inglês, job.

FIFO

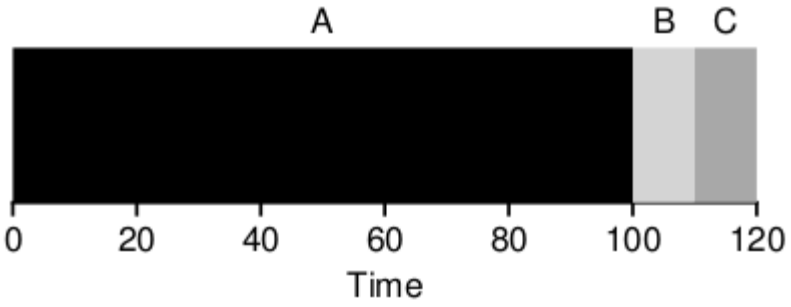
- Algoritmo o mais simples possível:
  - Simples e fácil de implementar;
  - Tem bons resultados, dadas as assunções;
- **1º CASO**
  - Imaginemos três trabalhos diferentes, A,B e C
    - Chegam sensivelmente ao mesmo tempo ($\mathbf{T_{arrival}}$=0).
    - Assume-se que cada trabalho é executado em 10 segundos.
      - **A** termina aos 10, **B** aos 20 e **C** aos 30 segundos;
      - O tempo de chegada é zero para cada um:
      - O valor médio do tempo de retorno é:

$$T_{ta} = \frac{10+20+30}{3} = 20$$



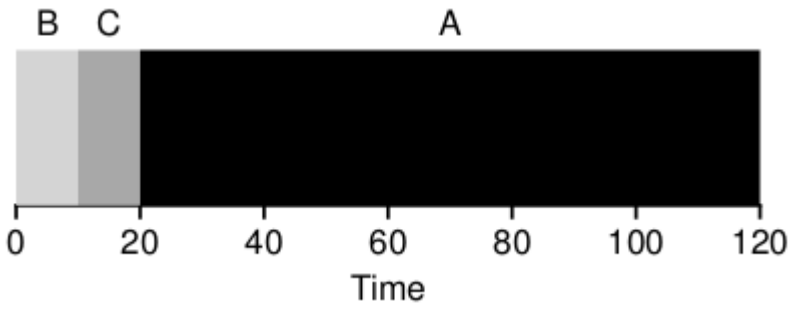
FIFO

- **2º CASO**
  - Relaxamos **1ª assunção**: Nem todos os processos têm o mesmo tempo de execução.
    - Assumimos três processos, mas o processo **A** demora agora 100 segundos.
    - O resultado agora é: $T_{ta} = \frac{100+110+120}{3} = 110$
    - **Efeito de Comboio ou engarrafamento** (***convoy effect***): o número de pequenos consumidores ficam em fila atrás de um "grande" consumidor.
  - Como é que se consegue resolver este problema em que existem diferentes processos com diferentes pesos no consumo do CPU?



Trabalho mais curto primeiro (SJF)

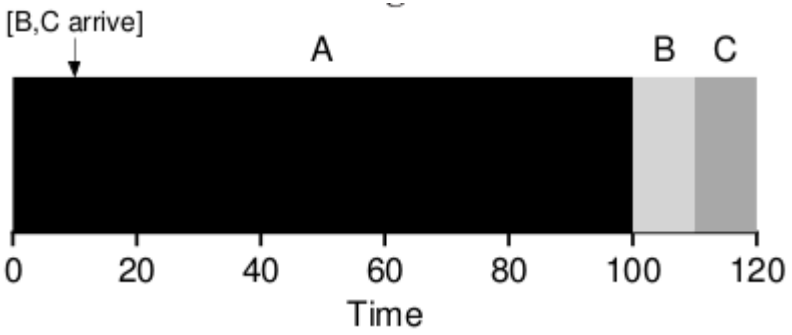
- Trabalho mais curto primeiro – ***Shortest Job First (SJF)***
  - Forma simples de resolver o problema;
  - A solução consiste em correr primeiro as tarefas mais curtas;
  - A *SJF* reduz a média de 110 para 50 segundos no caso anterior (**porquê? -** fazer contas)
  - Mantém-se o pressuposto de que todas as tarefas se iniciam ao mesmo tempo:
    - Este é um algoritmo de escalonamento ótimo.



Trabalho mais curto primeiro (SJF)

- Relaxamos a **2ª assunção**: Nem todos as tarefas são executadas ao mesmo tempo
  - Que problemas novos surgem?
    - **A** inicia a execução com *t=0;*
    - **B** e **C** iniciam a execução com *t=10*
  - O tempo de retorno da situação anterior é agora recalculado:
    - Existe o mesmo problema de "engarrafamento"

$$T_{ta} = \frac{100 + (110 - 10) + (120 - 10)}{3} = 103.33$$



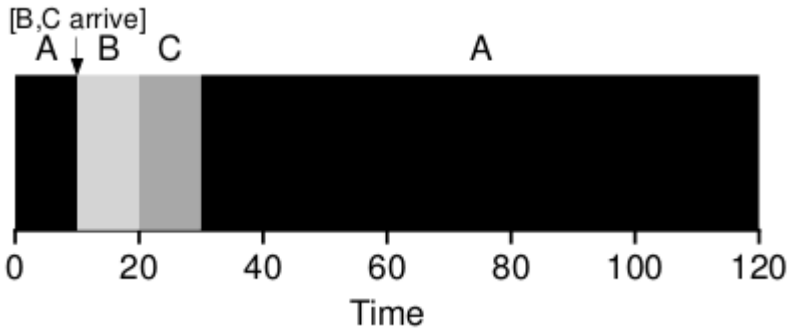
Tempo de completude mais curto primeiro (STCF)

- Tempo de completude mais curso primeiro – **Shortest Time-to-Completion First (STCF)**
  - Iremos relaxar a **3ª condição**: Um trabalho pode não ser executado até ao fim.
    - Diz-se que ocorre **preempção** quando o escalonador retira um trabalho em execução;
    - O escalonador pode retirar a tarefa **A** e decidir correr outra tarefa;

Tempo de completude mais curto primeiro (STCF)

- O STCF seleciona o trabalho que tem o tempo de execução mais pequeno.
  - A **preempção** ocorre, retirando a tarefa **A** e executando **B** e **C**, depois de eles “chegarem”
  - Só depois de **B** e **C** terminarem, a tarefa **A** é reiniciada.

$$T_{ta} = \frac{(120-0)+(20-10)+(30-10)}{3} = 50$$



Tempo de completude mais curto primeiro (STCF)

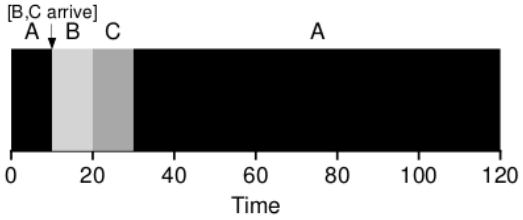
- Tal como nos exemplos anteriores, **STCF** é provavelmente a solução ótima na métrica utilizada – tempo de retorno.
- O que acontece se introduzirmos uma nova métrica?

Nova métrica: Tempo de resposta

- Com a introdução da partilha de computador por múltiplos utilizadores (*time-shared):*
  - Utilizadores necessitam de interagir com terminal;
  - O sistema tem de responder às suas necessidades;
- Nova métrica: **Tempo de resposta** (***response time***):
  - Tempo que demora uma tarefa a ser, pela primeira vez, escalonada ($T_{firstrun}$) desde o momento em que chega ao sistema – se torna executável ($T_{arrival}$):

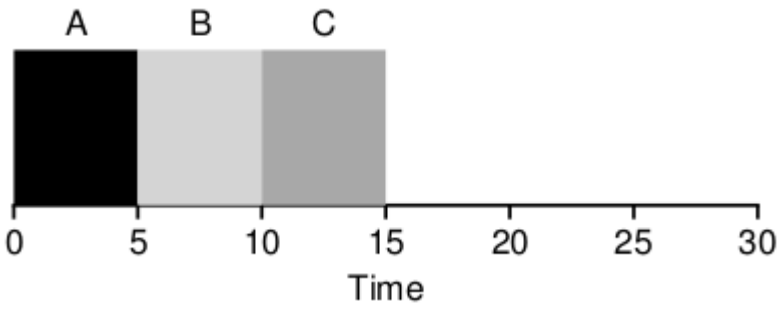
$$T_r = T_{firstrun} - T_{arrival}$$

  - Por exemplo, se **A** tem tempo de chegada 0 e **B** e **C** de 10, para o exemplo anterior, o tempo de resposta é 0 para **A**, 0 para **B** e 10 para **C** (média: 3,33).



Nova métrica: Tempo de resposta

- Qual é o desempenho dos diferentes escalonadores utilizando a métrica tempo de resposta?
  - **STCF*** e políticas similares não apresentam bons resultados
    - Se o tempo de chegada de três tarefas é idêntico, o terceiro trabalho tem de esperar que os outros dois trabalhos executem até ao fim antes de ser escalonada uma primeira vez.
    - Imaginemos que uma das tarefas implica **atividade interativa**:
      - O tempo de resposta ao trabalho **C** obrigava o utilizador a ficar 10 segundos à frente do terminal.

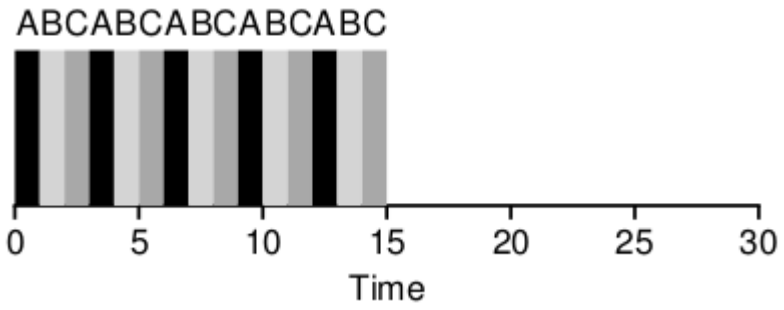


*Shortest Time-to-Completion First.

Round Robin

- **Round-Robin (RR)**
  - Em vez de executar as tarefas até ao fim, RR executa os trabalhos numa fatia de tempo – ***time slice***, muitas vezes designado quantum do escalonador (***scheduling-quantum***).
    - Utilizam-se mecanismos de interrupção para implementar o **RR**.
    - A duração de uma fatia do tempo tem de ser proporcional ao período de interrupção do relógio – ***timer-interrupt period*.**
      - Se o relógio de interrupção dispara cada 10 milisegundos, a fatia de tempo deve ser 10, 20 ou múltiplo desse valor.
      - No exemplo, em baixo, o **tempo de resposta** é baixo:

$$\frac{0+5+10}{3}=5$$



Round Robin

- A dimensão da fatia de tempo atribuído a cada trabalho é crítico:
  - Se diminuta, **RR** tem um bom desempenho para a métrica **tempo de resposta**.
    - Tornar esta fatia de tempo demasiado pequena?
      - O custo de mudança de contexto vai sobrepor-se ao bom desempenho.
    - Existe um compromisso (*trade-off*) entre a dimensão da fatia de tempo e o custo de mudança de contexto.
  - O que acontece se o desempenho for medido com a métrica **tempo de retorno**?
    - ***A***,***B*** e ***C*** executam durante 5 segundos e têm uma fatia de tempo de 1 segundo cada.
    - ***A*** termina ao fim de 13 segundos, ***B*** ao fim de 14 segundos e ***C*** ao fim de 15 segundos.
  - RR é uma das piores políticas de escalonamento caso se considere a métrica **tempo de retorno**.

Round robin

- **RR** tem o efeito de estender no tempo cada uma das tarefas ao executar apenas uma pequena parte de cada tarefa de cada vez.
- Qualquer política que é **justa,** no sentido em que que partilha o CPU entre processos ativos, tem um desempenho fraco em métricas como a métrica **tempo de retorno**.

Relaxar as últimas assunções

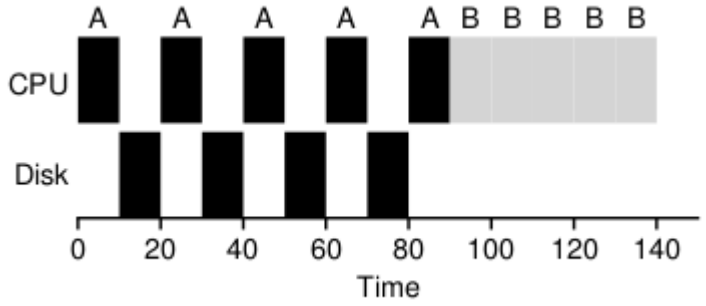
- Foram apresentados dois tipos de escalonadores:
  - ***SJF***, ***STCF*** : otimizam o tempo de retorno mas têm um tempo de resposta fraco;
  - **RR:** otimiza o tempo de resposta mas tem um tempo de retorno fraco.
- Iremos:
  - Incorporar E/S.
  - Considerar que o escalonador não conhece a duração de cada tarefa.

Incorporar E/S

- O escalonador tem de tomar uma decisão quando:
  - A tarefa inicia a execução de uma E/S.
    - A tarefa não irá usar CPU durante a execução da E/S.
    - O processo (associado à tarefa) fica no estado **bloqueado** (***blocked*****).**
    - O SO deverá escalonar outra tarefa para o CPU.
  - A E/S termina.
    - Surge uma interrupção.
    - OS altera o estado do processo de bloqueado para **executável**.

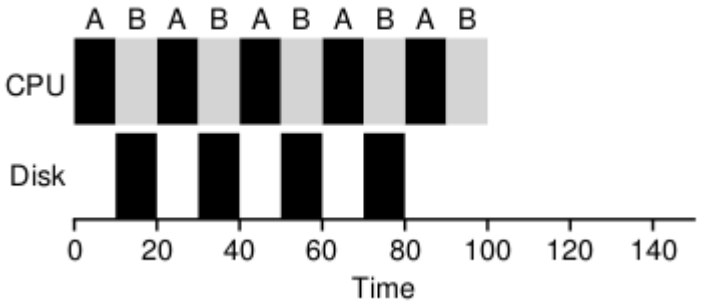
Incorportar E/S

- **Exemplo**
  - Assumimos que existem duas tarefas, ***A*** e ***B***, em que cada uma necessita de 50 milissegundos (ms) de *CPU*.
  - ***A*** corre 10 ms e depois faz um pedido de E/S (assume-se que cada E/S tem a duração de 10 ms).
  - ***B*** usa o CPU durante os 50 ms sem executar qualquer pedido de E/S.
- Assumindo que executamos primeiro **A** e depois **B**, o resultado é:



Incorporar E/S

- **Exemplo**
  - Utilizando **STCF**, se assumirmos que cada 10 ms de A é considerado um sub-trabalho de A (sub-job of A), medindo-se assim o tempo de 10 ms:
    - Sobrepõe-se a execução de um processo com o tratamento da E/S
  - Considerando cada momento de execução do CPU como trabalho, garantimos:
    - O escalonador garante que os processos que são “interativos” possam ser executados com a frequência necessária.
    - O escalonador manda executar os processos de uso mais intensivo de CPU nos intervalos em que há o processamento de E/S.



Sem Oráculo...

- **Problema:** Os **OS** conhecem, usualmente, muito pouco sobre a dimensão da tarefa que têm de executar.
  - Como podemos então propor uma solução que se comporta como **SJF** ou **STCF** sem ter esse conhecimento à *priori*?
  - Como poderemos incorporar algumas das ideias que foram propostas para o escalonador ***RR*** de forma que o desempenho seja bom tendo em conta a métrica **tempo de resposta**?
    - Usar o passado recente para predizer o futuro – **fila de retorno multi-nível** (***multi-level feedback queue - MLFQ***)

Questões

- Explique cada uma das estratégias de escalonamento:
  - SJF
  - STCF
  - Round-robin (RR)
- Existem duas métricas que permitem medir o desempenho de cada uma das estratégias, a saber, tempo de retorno e o tempo de resposta.
  - Explique a que corresponde o tempo de retorno.
  - Explique a que corresponde o tempo de resposta.
  - Explique a quais métricas respondem melhor as três estratégias de escalonamento estudadas.