1. Considere o seguinte conjunto de processos, com duração de surto de CPU expressa em milissegundos, conforme a tabela abaixo. Os processos são considerados como tendo chegado na ordem P1, P2, ..., P5, todos no instante zero.*

Processo	Ciclos CPU	Prioridade
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

   a) Desenhe quatro diagramas de Gantt que ilustrem a execução desses processos usando os escalonamentos FIFO, SJF, Prioridade não-preemptivo (um número de prioridade mais baixo implica prioridade mais alta) e RR (quantum=3).
   b) Qual é o tempo de retorno de cada processo para cada um dos algoritmos de escalonamento do item a) ?
   c) Qual é o tempo de espera de cada processo para cada um dos algoritmos de escalonamento do item a) ?
   d) Qual é o tempo médio de espera de cada um dos algoritmos?

2. 'Vamos supor que os seguintes processos cheguem para execução nos instantes indicados. Cada processo executará durante o período de tempo listado.*

Processo	Ciclos CPU	Chegada
P0	8	0
P1	4	0,4
P2	1	1,0

   a) Qual o tempo de espera médio para esses processos com o algoritmo de escalonamento FIFO?
   b) Qual o tempo de espera médio para o algoritmo SJF não-preemptivo?
   c) E para o SJF preemptivo?
   d) O algoritmo SJF não-preemptivo deve melhorar o desempenho, mas observe que escolhemos executar o processo P1 no tempo 0 porque não sabíamos que dois processos mais curtos chegariam logo a seguir. Calcule o tempo de espera médio se a CPU ficar inativa pela primeira unidade de tempo (1) e o escalonamento SJF não-preemptivo for utilizado. Lembre-se de que os processos P1 e P2 estão esperando durante esse tempo inativo, de modo que seu tempo de espera poderá aumentar. Esse algoritmo pode ser chamado de escalonamento de conhecimento futuro.

3. Em relação ao favorecimento de processos curtos, compare os seguintes algoritmos de escalonamento:
   a) FIFO
   b) RR
   c) Múltiplas Filas com Realimentação

4. Como pode ser determinado o valor do *quantum* no escalonamento *Round-Robin*?

5. Defina uma política de escalonamento baseada em múltiplas filas com realimentação. Devem existir duas filas. O algoritmo entre as filas deve trabalhar de forma que, com o passar do tempo, processos "i/o *bound*" vão para a fila 1 e processos "cpu *bound*" vão para a fila 2. Não deve ser possível a ocorrência de postergação indefinida de nenhum processo.

6. Considere o seguinte conjunto de processos, com duração de surto de CPU expressa em milissegundos, o instante de chegada e a prioridade (menor número indica prioridade mais alta), conforme a tabela a seguir. Considere o escalonamento destes processos com as políticas FIFO, SJF, Prioridade e Fatia de tempo com quantum 4 e para as que possuem, considere também suas versões preemptivas. Construa o Diagrama de Tempo ou de Gantt para cada uma das políticas e calcule o tempo médio de retorno e de espera dos processos. Qual dos escalonamentos foi mais eficiente para este caso?

Processo	Instante de Chegada	Surto de CPU	Prioridade
P0	0	10	3
P1	1	5	1
P2	2	2	3

7. Três programas devem ser executados em um computador. Todos os programas são compostos por 2 ciclos de processador e dois ciclos de E/S. A entrada e saída de todos os programas é feita sobre a mesma unidade de disco. Os tempos de cada programas são mostrados abaixo: *

Processo	Processador	Disco	Processador	Disco
P1	3	10	3	12
P2	4	12	6	8
P3	7	8	8	10

a) Construa diagramas de tempo ou de Gantt mostrando a ocupação do processador e do disco a cada momento, até todos os programas terminarem. Suponha que o algoritmo utilizado seja Fatia de Tempo, com fatia de 4 unidades. Qual a taxa de ocupação do processador e do disco?

b) O que acontece com as taxas de ocupação se for utilizado um disco com o dobro da velocidade de acesso (duração dos ciclos de E/S é dividida por 2)?

8. Quais os principais objetivos do escalonamento de processos?

9. Diferencie escalonamento preemptivo e escalonamento não-preemptivo.

10. Na política de escalonamento FCFS, considere a situação a seguir:

- Um processo CPU *bound* (P1) obtém e detém a CPU. Muitos processos I/O *bound* terminam sua operação de I/O e passam para o estado de prontos. O P1 passa para a operação de I/O.

Descreva como acontecerá a utilização da CPU e dos dispositivos. Apresente uma soluções para melhorar o uso dos recursos do sistema de computação.

11. O escalonamento por Prioridade sempre resultará em um menor tempo médio de espera dos processos? Justifique.

12. O que é *starvation*? Quando pode acontecer? Qual a solução para este problema?

13. O desempenho do Round-Robin depende do tamanho do quantum. Quais as implicações no escalonamento se:

- for utilizado um quantum imediatamente superior ao tempo necessário para a troca de contexto?
- for utilizado um quantum cujo tamanho é imensamente superior a média dos surtos de CPU dos processos do sistema?

14. Em termos de aplicação, diferencie o escalonamento por Múltiplas Filas do escalonamento por Múltiplas Filas com Realimentação.

15. Em um sistema operacional, o escalonador de curto prazo utiliza duas filas. A fila “A” contém os processos do pessoal do CPD e a fila “B” contém os processos dos alunos. O algoritmo entre fila é fatia de tempo. De cada 11 unidades de tempo do processador, 7 são fornecidas para os processos da fila “A” e 4 para os processos da fila “B”. O tempo de cada fila é dividido entre os processos também por fatias de tempo, com fatias de 2 unidades para todos. A tabela a seguir mostra o conteúdo das duas filas no instante zero. Considere que está iniciando um ciclo de 11 unidades, e agora a fila “A” vai receber as suas 7 unidades de tempo. Considere que se terminar a fatia de tempo de uma determinada fila no meio da fatia de tempo de um dos processos, o processador passa para a outra fila, entretanto o processo que foi interrompido continua como sendo o primeiro da fila que foi interrompida.*

Fila	Processo	Duração Ciclo Processador
A	P1	6
A	P2	5
A	P3	7
B	P4	3
B	P5	8
B	P6	4

a. Mostre a sequência de execução dos processos, com os momentos em que acontecem as trocas
b. Este escalonamento faz uso de realimentação entre as filas?
c. Ao invés de ser utilizado fatia de tempo entre as filas, considere que a fila “A” tem prioridade sobre a fila “B”. Como ficaria a seqüência de execução destes processos, mantendo-se o escalonamento em cada fila por fatias de tempo, com quantum de 2 unidades?

* Exercícios adaptados de (OLIVEIRA, 2001)

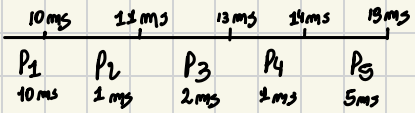
1. Algoritmos de Escalonamento

Temos 5 processos (P1, P2, P3, P4, P5), todos chegando ao mesmo tempo, mas com diferentes tempos de execução (ciclos de CPU) e prioridades.

a) Diagramas de Gantt: *Os diagramas de Gantt servem para visualizar a ordem e o tempo em que os processos são executados. Vamos criar um diagrama para cada algoritmo de escalonamento:*

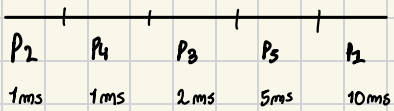
FIFO (First In, First Out): *Os processos são executados na ordem em que chegaram. P1 vai primeiro, seguido por P2, P3, P4 e P5.*

Exemplo de Gantt:



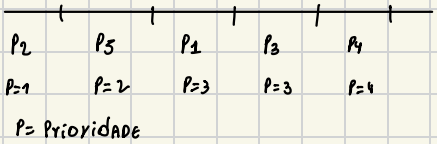
SJF (Shortest Job First): *Executa o processo mais curto primeiro. Aqui, a ordem é baseada na duração do ciclo de CPU.*

Exemplo de Gantt:



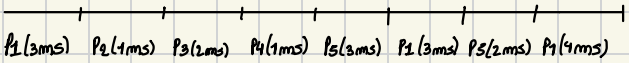
Prioridade (não preemptivo): *Processos com prioridade mais alta (número menor) são executados primeiro.*

Exemplo de Gantt:



RR (Round-Robin) com quantum de 3ms:
Cada processo recebe 3ms de tempo de CPU por vez até terminar.

Exemplo de Gantt:



b) Tempo de Retorno: *O tempo de retorno é o tempo total que o processo leva desde que chega até ser completamente finalizado*

FIFO:	SJF:	Prioridade:	RR:
P1: 10ms	P2: 1ms	P2: 1ms	P1: 13ms
P2: 11ms	P4: 2ms	P5: 6ms	P2: 4ms
P3: 13ms	P3: 4ms	P1: 16ms	P3: 5ms
P4: 14ms	P5: 9ms	P3: 18ms	P4: 6ms
P5: 19ms	P1: 19ms	P4: 19ms	P5: 11ms

c) Tempo de Espera: *O tempo de espera é o tempo que cada processo fica na fila esperando para ser executado.*

FIFO:	SJF:	Prioridade:	RR:
P1: 0ms	P2: 0ms	P2: 0ms	P1: 3ms
P2: 10ms	P4: 1ms	P5: 1ms	P2: 0ms
P3: 11ms	P3: 2ms	P1: 6ms	P3: 3ms
P4: 13ms	P5: 4ms	P3: 16ms	P4: 5ms
P5: 14ms	P1: 9ms	P4: 17ms	P5: 6ms

d) Tempo Médio de Espera (O tempo médio de espera é a média do tempo que todos os processos passam esperando para serem executados)

FIFO: (0 + 10 + 11 + 13 + 14) / 5 = 9.6ms

SJF: (0 + 1 + 2 + 4 + 9) / 5 = 3.2ms

Prioridade: (0 + 1 + 6 + 16 + 17) / 5 =. 8ms

RR: (3 + 0 + 3 + 5 + 6) / 5 = 3.4ms

2. Chegada Diferenciada de Processos

Aqui temos processos que chegam em tempos diferentes. O desafio é calcular o tempo de espera médio para cada algoritmo.

a) FIFO
Com o algoritmo FIFO, os processos são executados na ordem de chegada, independentemente da duração dos seus ciclos de CPU.

P0 (chegada 0, 8ms), P1 (chegada 0.4, 4ms), P2 (chegada 1.0, 1ms) O tempo de espera médio para FIFO seria calculado baseado no tempo de cada um desses processos.

b) SJF Não-preemptivo
No **SJF não-preemptivo**, o processo mais curto é executado primeiro, mas uma vez que um processo começa, ele não é interrompido.

c) SJF Preemptivo
No **SJF preemptivo**, o processo pode ser interrompido se outro processo mais curto chegar. Isso gera mais eficiência para processos curtos.

d) Escalonamento de Conhecimento Futuro
Este algoritmo assume que temos conhecimento de todos os processos futuros, o que permite pausar a execução até o momento ideal para melhorar o tempo de espera médio.

3. Favorecimento de Processos Curtos

Aqui, vamos comparar três algoritmos diferentes e como eles favorecem ou não os processos com menor duração de CPU.

a) FIFO (First In, First Out)
No FIFO, o processo que chega primeiro é o que será executado primeiro, sem levar em conta a duração de seus ciclos de CPU. Isso significa que processos mais longos podem acabar bloqueando processos curtos, o que não é ideal para sistemas onde rapidez é importante.

b) RR (Round-Robin)
O **Round-Robin** favorece processos curtos, pois cada processo recebe uma pequena "fatia" de tempo para ser executado. Processos curtos conseguem terminar rapidamente, já que não têm que esperar por um processo longo para liberar a CPU. Contudo, o desempenho depende do **tamanho do quantum** (a fatia de tempo atribuída para cada processo).

c) Múltiplas Filas com Realimentação
Neste algoritmo, os processos são divididos em diferentes filas baseadas em sua natureza. **Processos curtos e com mais operações de entrada/saída** (I/O bound) vão para uma fila de maior prioridade. Já os **processos que usam mais CPU** (CPU bound) são movidos para outra fila com menos prioridade. Isso garante que processos curtos sejam priorizados, evitando que fiquem esperando por muito tempo.

4. Determinação do Quantum no Round-Robin

O **quantum** é o tempo que cada processo recebe para ser executado no algoritmo Round-Robin. A escolha do tamanho do quantum é crucial:
Se o quantum for **muito pequeno** (perto do tempo de troca de contexto), o sistema gastará mais tempo trocando entre processos do que executando-os.
Se o quantum for **muito grande**, o algoritmo se aproxima de um escalonamento FIFO, e processos curtos podem ficar esperando por muito tempo.
O valor ideal do quantum deve equilibrar entre a frequência de trocas de contexto e a eficiência da execução.

5. Política de Múltiplas Filas com Realimentação

Esta política de escalonamento envolve duas filas:

Fila 1 (alta prioridade): Para processos que fazem mais operações de entrada/saída (I/O bound). Esses processos costumam ser rápidos e precisam de resposta imediata.
Fila 2 (baixa prioridade): Para processos que exigem mais tempo de CPU (CPU bound). Esses processos tendem a ser mais longos e menos sensíveis ao tempo de resposta imediato.

O algoritmo de realimentação funciona da seguinte maneira:
Quando um processo termina rapidamente sua execução ou faz muitas operações de I/O, ele permanece na fila 1.
Se um processo usa muito tempo de CPU sem fazer I/O, ele é movido para a fila 2.
Prevenção de postergação indefinida: Deve haver um mecanismo para garantir que nenhum processo fique esperando para sempre na fila de baixa prioridade (fila 2). Por exemplo, processos que esperam muito tempo podem eventualmente ser movidos de volta para a fila de alta prioridade.

6. Escalonamento com Processos de Entrada e Saída

Agora, temos três processos que alternam entre uso de CPU e E/S (entrada/saída) em um disco. A ideia é construir diagramas de Gantt para entender como a CPU e o disco são ocupados durante a execução desses processos.

a) Diagrama de Gantt com Fatia de Tempo (Quantum = 4 unidades)
Neste caso, cada processo recebe até 4 unidades de tempo de CPU antes de ser interrompido. Após utilizar a CPU, o processo faz operações de E/S (que utilizam o disco), e a CPU pode ser atribuída a outro processo.
Aqui, teríamos dois diagramas de Gantt: um para a ocupação da CPU e outro para a ocupação do disco. A taxa de ocupação de cada recurso depende de quanto tempo os processos estão utilizando aquele recurso.

b) Disco com Velocidade Dobrada
Se o disco operar com o **dobro da velocidade**, o tempo de E/S é reduzido pela metade. Isso significa que os processos podem liberar o disco mais rapidamente e voltar a usar a CPU, o que aumentaria a **taxa de ocupação da CPU**, pois ela ficaria menos tempo ociosa esperando pelos processos que estão fazendo E/S.

7. Escalonamento Preemptivo vs. Não-preemptivo

Escalonamento Preemptivo: A CPU pode ser retirada de um processo em execução para ser atribuída a outro processo mais urgente. Isso é útil para sistemas onde a responsividade é importante, mas pode gerar mais trocas de contexto, aumentando a sobrecarga do sistema.
Escalonamento Não-preemptivo: Uma vez que um processo começa a executar, ele mantém o controle da CPU até terminar. Isso simplifica o gerenciamento da CPU, mas pode causar **starvation** (fome), onde processos de menor prioridade esperam indefinidamente.

8. Problemas de Uso de Recursos no FCFS

No escalonamento **FCFS (First Come, First Served)**, um processo CPU-bound (que usa muito tempo de CPU) pode monopolizar a CPU enquanto processos I/O-bound (que necessitam de pouca CPU e mais operações de E/S) ficam esperando. Isso resulta em um uso ineficiente da CPU e dos dispositivos de I/O.
Solução: Implementar uma política de escalonamento que dê mais prioridade a processos curtos ou que alterne entre processos CPU-bound e I/O-bound, como o algoritmo **Múltiplas Filas com Realimentação**.

9. Starvation

Starvation ocorre quando um processo de baixa prioridade espera indefinidamente para ser executado, porque processos de maior prioridade continuam chegando e ocupando a CPU.
Solução: Uma técnica comum para evitar starvation é a **prioridade envelhecida**. Com o tempo, a prioridade dos processos que estão esperando aumenta, até que eles finalmente sejam executados.

10. Impacto do Tamanho do Quantum no Round-Robin

*O tamanho do quantum no escalonamento **Round-Robin** afeta diretamente o desempenho:*

Se o quantum for **ligeiramente maior que o tempo de troca de contexto**, o sistema perde tempo trocando entre os processos, o que é ineficiente.
Se o quantum for **muito grande** (muito maior que o tempo de execução médio dos processos), o algoritmo se comporta como FIFO, e os processos curtos podem sofrer longos tempos de espera.

11. Escalonamento Múltiplas Filas vs. Múltiplas Filas com Realimentação

Múltiplas Filas: Os processos são divididos em várias filas, cada uma com sua própria prioridade. Um processo nunca muda de fila, e filas de maior prioridade são executadas primeiro.
Múltiplas Filas com Realimentação: Os processos podem mudar de fila com base no seu comportamento (uso de CPU ou E/S). Isso dá flexibilidade para adaptar o escalonamento às necessidades dinâmicas dos processos.

12. Exemplo de Sistema Operacional com Duas Filas

Neste exemplo, temos duas filas:

Fila A (para o pessoal do CPD), com 7 unidades de tempo.
Fila B (para alunos), com 4 unidades de tempo.
Cada fila é escalonada com o algoritmo Round-Robin, e a CPU alterna entre as filas.

a) Sequência de Execução
Aqui, desenharíamos a sequência de execução dos processos, alternando entre as filas de acordo com o tempo atribuído.

b) Realimentação
Este esquema de escalonamento pode ser modificado para incluir realimentação entre as filas, permitindo que processos da fila B eventualmente subam para a fila A com base em critérios definidos.

c) Prioridade para Fila A
Se a **Fila A** tiver prioridade sobre a Fila B, os processos da Fila B só serão executados quando não houver mais processos na Fila A.