**Universidade Estadual Paulista - UNESP**

**Departamento de Estatística, Matemática Aplicada e Computação**

**Bacharelado em Ciência da Computação - Integral**

Sistemas Operacionais I

**Atividade 2**

**Escalonadores de Processos**

Gabriel Correia Balbino

Lucas Bastelli Spagnol

Pedro Gabriel Gonçalves

Vinícius do Amaral Brunheroto

Prof. Alessandro Viola Pizzoleto

Rio Claro/SP

2021

**FIFO**

*Conceitos:*

FIFO (first in, first out, "primeiro a entrar, primeiro a sair") ou FCFS (first come, first served, "primeiro a chegar, primeiro a ser servido") é um algoritmo de escalonamento do tipo não-preemptivo para estruturas de dados do tipo fila. Apresenta o seguinte critério: o primeiro elemento a ser retirado é o primeiro que tiver sido inserido, é um algoritmo que entrega a CPU os processos pela ordem de chegada. Ele executa o processo como um todo do início ao fim não interrompendo o processo executado até ser finalizado, então quando um novo processo chega e existe um ainda em execução ele vai para uma fila de espera. Esta fila de espera é do uma fila que organiza os processos que chegam até eles serem atendidos pela CPU.

No FIFO todos os processos tendem a serem atendidos ao menos que um processo possua um erro ou loop infinito. O loop infinito irá parar a máquina, pois com o FIFO não terá como dar continuidade a execução dos processos que estão aguardando na fila de espera.

*Vantagens:*

A implementação é simples e fácil. Basta manter uma fila de todas as páginas na memória. Elas são adicionadas ao final da fila quando lidas para a memória. A página que sofre swap (tipo de cópia entre a memória principal e o disco) para o disco é sempre escolhida da frente da fila.

O sistema operacional faz o reescalonamento apenas quando é absolutamente necessário. A sobrecarga estão é pequena.

O processo utiliza a CPU sem ser interrompido. É justo, pois atende pela ordem de chegada. Também, impede adiamento indefinido.

Processos CPU-bound levam vantagem no uso do processador sobre processos I/O-bound.

*Desvantagens:*

Tempo de resposta alto, porque o algoritmo não garante um tempo de resposta rápido pois é extremamente sensível a ordem de chegada de cada processo e dos antecessores e se processos que tendem a demorar mais tempo chegarem primeiro o tempo médio de espera e o turnaround acabam sendo aumentados.

Possibilidade de processos CPU-bound de menor importância prejudicarem processos de I/O-bound mais proprietários.

Impossibilidade de prever-se quando um processo terá a sua execução iniciada, já que isso varia em função do tempo de execução dos demais processos na fila de pronto.

Processos longos fazem os curtos esperarem muito, pois podemos ter processos grandes que serão atendidos primeiro conforme a fila. Logo os processos menores terão tempo de espera maior.

Não se mostra eficiente para processos interativos

Não considera a importância de uma tarefa

*Utilização:*

Inicialmente foi implementado em sistemas monoprogramáveis com processamento batch e atualmente, sistemas de tempo compartilhado utilizam FIFO com variações.

Além disso, são usados em circuitos eletrônicos de buffer e controle de fluxo, que vai desde o hardware até o software. Na forma de hardware o FIFO consiste de um conjunto de ler e escrever ponteiros, armazenamento (SRAM, flip-flops, fechos ou qualquer outra forma adequada de armazenamento) e lógica de controle. Para o FIFO, de tamanho não trivial, a SRAM de porta dupla é usada quando uma porta é usada para a escrita e a outra para leitura.

O FIFO síncrono tem o mesmo clock usado para leitura e escrita e o assíncrono utiliza clocks diferentes. Uma aplicação comum de um assíncrono utiliza um código de Gray (código binário refletido), ou qualquer unidade de código a distância, para a ler e escrever os ponteiros

FIFO é aconselhável quando você utiliza itens que possuem alto giro e um tempo de validade que possibilite alguma ação caso algo dê errado. Bons exemplos são itens de mercearia básica como arroz e feijão. Outra boa alternativa é utilizar para produtos que não possuam validade, como eletrônicos, eletrodomésticos e móveis.

Ele também é utilizado em empresa de logística e nas mais diversas áreas, desde produção até vendas e marketing, bem como por repositores e promotores.

**SJF**

*Conceitos:*

Shortest Job First (trabalho mais curto primeiro), ou Shortest Job Next (trabalho mais curto em seguida), ou Shortest Process Next (SPN, processo mais curto em seguida) é um algoritmo de escalonamento que seleciona para ser executado o processo com o menor tempo de execução.

Na sua concepção inicial (primeiros sistemas operacionais com processamento exclusivamente batch), o escalonamento SJF é não-preemptivo pois o processo que possui a CPU somente a libera quando termina sua execução ou quando se bloqueia. Vale citar, que o Shortest Remaining Time é uma variação preemptiva de SJF, já que se um outro processo chegar pico de CPU menor do que o restante do processo atual, há preempção.

No SJF, o processador é alocado ao processo com etapa de CPU mais breve. Em caso de empate se aplica outro algoritmo (normalmente o FIFO). Ele ordena a fila de processos prontos em função do tempo das seguintes etapas de CPU dos processos. O critério é o tempo de execução restante (burst), sendo o menor primeiro.

Uma maneira de implementar o escalonamento SJF em sistemas interativos foi considerar o comportamento do processo neste ambiente.

*Vantagens:*

É vantajoso por sua simplicidade e também porque minimiza o tempo médio que cada processo leva desde quando ele é criado até o fim de sua execução, incluindo aqui o tempo de espera entre o momento em que ele é criado e em que é selecionado para executar.

Deste modo, minimiza o efeito de priorizar processos do tipo cpu-bound, favorece os processos mais curtos, aumenta o rendimento (throughput), há redução do tempo médio de turnaround dos processos em relação sobre o FIFO.

*Desvantagens:*

Pode levar a inanição de processos com longos tempos de execução caso processos curtos sejam continuamente adicionados ao escalonador.

A necessidade de saber previamente os tempos para execução dos processos. Embora seja impossível prever os tempos de maneira exata, ou seja, não é possível ao SO saber quanto tempo um processo irá permanecer utilizando a CPU na próxima vez em que for escalonado. Assim, é difícil determinar a priori qual será a duração da seguinte etapa de CPU dos processos.

Deste modo, temos como desvantagens o fato que ele é baseado em estimativas de tempo, que há maior variância no tempo de espera (mais imprevisibilidade), baixo aproveitamento quando se tem poucos processos prontos para serem executados e não impede o adiamento indefinido.

*Utilização:*

Pode ser usado na estimativa de uso do próximo quantum assim para priorizar as tarefas orientadas a entrada/saída, que usam menos o processador.

Usa-se os tempos (deste algoritmo) para escalonar/selecionar o processo com o CPU burst mais pequeno. Quando dois processo têm o mesmo CPU burst, o desempate faz-se por FCFS.

Assim é usado na CPU que é atribuída a um processo, este não pode ser que não pode ser preempcionado até completar o seu CPU burst.

SJF é ótimo, uma vez que minimiza o tempo médio de espera de um dado conjunto de processos. Deste modo, ele como o FIFO pode ser utilizado em empresa de logística, produção, vendas e marketing.

**SRT**

*Conceitos:*

SRT (Shortest Remaining Time) ou SRTF (Shortest Remaining Time First) é a versão preemptiva do método de escalonamento SJF, neste algoritmo o processo com o menor tempo de execução restante até ser completado é o processo escolhido para ser executado.

O algoritmo avalia e posiciona os processos em uma fila de execução com seu tempo de chegada e seu tempo de execução estimado, quando um processo está em execução e chega um novo processo, se este possui tempo de execução previsto menor que o atual em execução ele é colocado para executar em seu lugar, o processo que estava em execução deve então ser reinserido na fila de execução levando em conta agora seu novo tempo restante.

*Vantagens:*

As vantagens desse método se dão pelo fato de que os processos de menor tamanho são gerenciados rapidamente.

Outra vantagem é que o sistema sofre pouca sobrecarga devido ao fato que o somente é feita quando um processo termina ou um processo novo é adicionado, e quando é adicionado apenas se compara o processo em execução no momento com os novos processos inseridos, ignorando os processos em espera.

*Desvantagens:*

Uma desvantagem dele alinhasse com a desvantagem do método SJF, sendo esta a possibilidade de um processo grande ficar por tempo indefinido sem ser completado, se processos curtos forem adicionados continuamente.

Outra desvantagem seria o fato que é necessário constantemente calcular de forma preciso o tempo estimado de execução para cada processo, o que faz o algoritmo ser usado frequentemente apenas em ambientes especializados.

*Utilização:*

Ele normalmente é usado em ambientes especializados.

**Múltiplas Filas de realimentação**

*Conceitos:*

Primeiro vamos apresentar o conceito de um escalonador com múltiplas filas, temos então um escalonador onde é implementado o conceito de filas, cada uma possuindo um certo nível de prioridade, contudo os processos não ficam em uma mesma fila até o termino do processamento. O sistema identifica dinamicamente o comportamento de cada processo, assim ajustando suas prioridades de execução e modo de escalonamento.

Esse algoritmo dá prioridade a processos curtos e processos I/O bound, também separa os processos em categorias baseados nas suas necessidades de processamento.

*Vantagens:*

Bem flexível, sendo possível de ser implementado em qualquer sistema operacional.

Permite processos diferentes se moverem entra filas diferentes.

Previne a inanição (processo nunca finalizado) ao mover um processo que está a muito tempo na espera de uma fila de baixa prioridade para um de maior.

*Desvantagens:*

Pode gerar um processamento excessivo no sistema.

É o algoritmo mais complexo de implementar, devido aos diferentes métodos utilizados e como escolher seus valores.

*Utilização:*

Processo criado entra no final da fila de mais alta prioridade.

Cada fila implementa um mecanismo desejado para escalonamento.

Quando o processo em execução deixa a CPU por preempção por prioridade ou por solicitação a um recurso do sistema ele é reescalonado para dentro da mesma fila.

Caso o processo esgote seu quantum de tempo, ele é redirecionado para um a fila de menor prioridade (preempção por tempo), e assim por diante.

A fila de mais baixa prioridade implementa o mecanismo de escalonamento circular.

O quantum em cada fila varia em função da sua prioridade. Quanto maior a prioridade, menor seu quantum de tempo.

**Escalonamento para vários processadores**

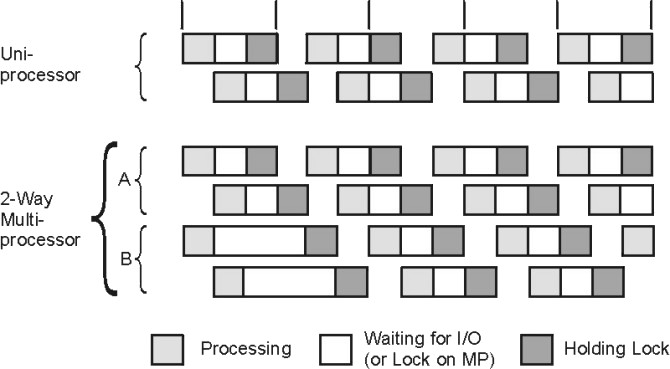
*Conceitos:*

Escalonamento para vários processadores ou computação paralela é o trabalho de muitos processadores simultaneamente para produzir um poder computacional excepcional e reduzir significativamente o tempo computacional total. Em tais cenários, escalabilidade ou escalonamento é amplamente usado para indicar a capacidade do hardware e software de fornecer maior poder computacional quando a quantidade de recursos é aumentada.

Escalabilidade é o grau em que o rendimento da carga de trabalho se beneficia da disponibilidade de processadores adicionais, é importante para a computação paralela ser eficiente. Uma escalabilidade garante a aceleração para um tamanho de problema fixo em relação ao número de processadores.

Geralmente é expresso como o quociente da taxa de transferência da carga de trabalho em um multiprocessador dividido pela taxa de transferência em um uniprocessador comparável. Por exemplo, se um uniprocessador atingiu 20 solicitações por segundo em uma determinada carga de trabalho e um sistema de quatro processadores atingiu 58 solicitações por segundo, o fator de escala seria 2,9.

*Vantagens:*



No multiprocessador do exemplo acima, dois processadores controlam a execução do programa, mas ainda há apenas um bloqueio. Para simplificar, toda a contenção de bloqueio é mostrada afetando o processador B.

*Desvantagens:*

Esse maior processamento vem com um custo maior.

*Utilização:*

O elemento fundamental de um núcleo (“Kernel”) de tempo-real é o escalonador, que tem como função permitir a execução de múltiplas tarefas num ambiente pseudo-paralelo, garantindo o respeito das suas metas temporais.

**Escalonamento em tempo real**

*Conceitos:*

O escalonamento de tempo-real pretende atribuir de uma forma óptima o processador às tarefas que os pretendem utilizar, garantindo o respeito das metas temporais (“deadlines”) associadas à execução de cada uma das tarefas.

*Vantagens:*

Implementação mais simples.

Baixo overhead.

Comportamento previsível.

*Desvantagens:*

Difícil lidar com eventos esporádicos.

Tudo deve operar em etapas sequenciais.

Código deve ser escalonado manualmente.

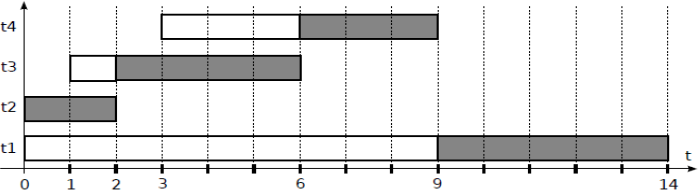
Comportamento temporalmente imprevisível.

*Utilização:*

No desenvolvimento de sistemas de tempo real de baixa complexidade. No entanto, as aplicações desenvolvidas poderão ter um comportamento temporalmente imprevisível.

Também, é usado em projetos com maior nível de exigência requisitos temporais ao longo de todo o processo de desenvolvimento.

**Comparação entre os Algoritmos**

O algoritmo de escalonamento que proporciona os menores tempos médios de execução e de espera é conhecido como menor tarefa primeiro, ou SJF (Shortest Job First). Consiste em atribuir o processador à menor (mais curta) tarefa da fila de tarefas prontas.

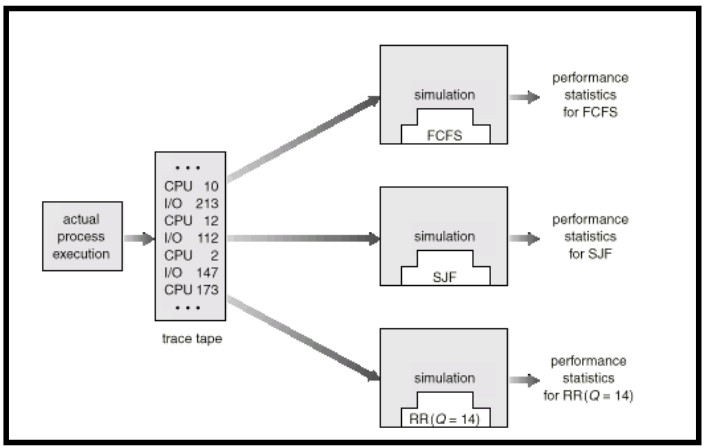
Nesse momento de decidir qual escalonador será utilizado no sistema operacional, cabe avaliar o cenário que o sistema será utilizado.

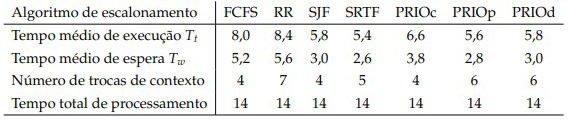
Deve-se ter cuidado com algumas variáveis como em casos que necessitam de mais processamento, ou seja, ação da CPU. Como com processos que necessitam de processamento, ocuparão a CPU por um tempo maior e não precisarão, ou de pouca, intervenção do usuário.

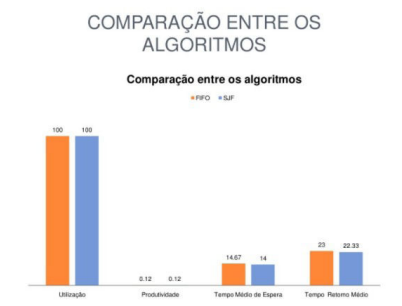
Enquanto isso, há processos que necessitam de mais entrada e saída de dados, ou seja, o processo necessita de intervenção do usuário. Ou seja, deve-se verificar o comportamento dos processos: se são orientados a Entrada e Saída (IN/OUT bound) ou orientados a orientados a CPU (CPU bound).

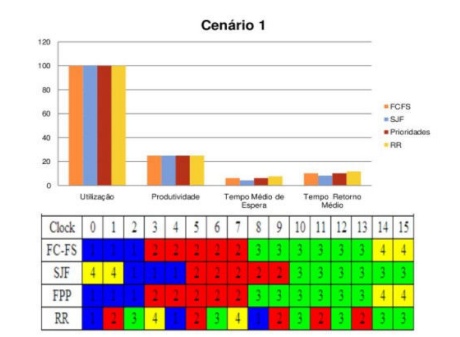
O escalonador ideal é aquele que consegue deixar a CPU 100% ocupada para maximizar a produtividade e minimizar o tempo de retorno, resposta e espera. Não existe nenhuma política de escalonamento ótima. A política de escalonamento conveniente depende do tipo de processo e do critério de otimização desejado

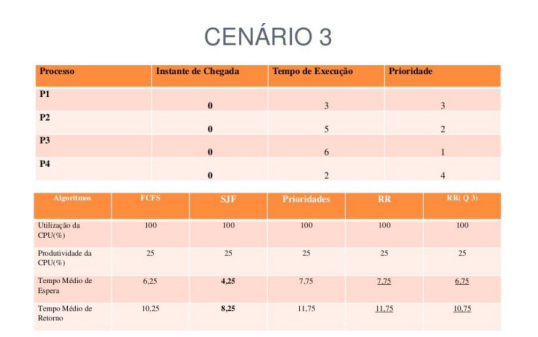
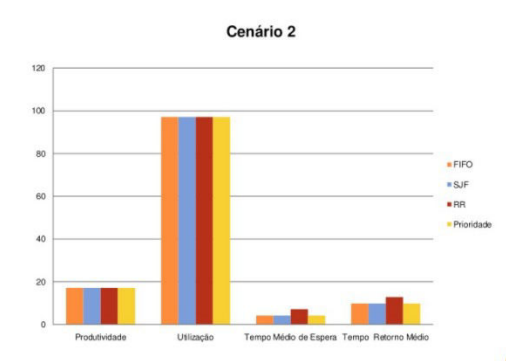
Maneiras de se fazer avaliação dos algoritmos de escalonamento: **Modelagem determinística:** considera uma carga de trabalho particular (pré-determinada) Define (calcula) o desempenho de cada algoritmo para a carga utilização de CPU, throughput, tempo de espera, tempo de turnarorund, etc; **Avaliação por simulação:** método mais preciso, utiliza um modelo de sistema de computação, as informações (processos, picos de CPU, chegadas, E/S, términos, etc.) podem ser geradas aleatoriamente. Resultados são usados para verificar o que ocorre na realidade e adota-se a distribuição adequada; **Avaliação por implementação:** mais realista porém tem alto custo: necessário implementar no kernel e testar sob as diversas situações reais.

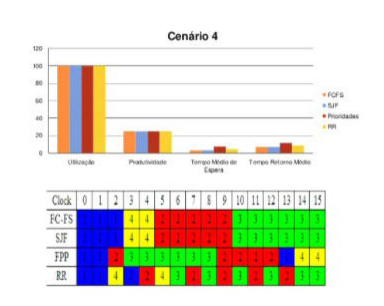
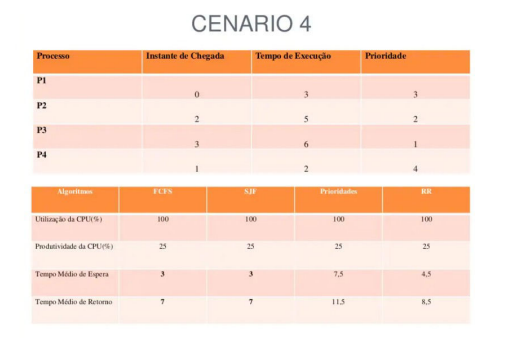
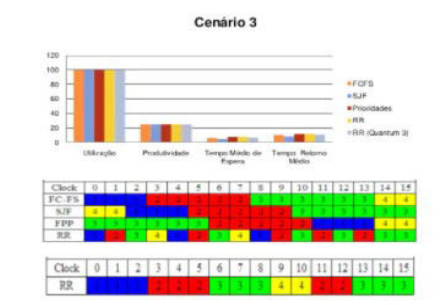
Exemplo de avaliação por simulação:

Pode-se observar que os algoritmos preemptivos (RR, SRTF e PRIOp) possuem um número de trocas de contexto maior que seus correspondentes cooperativos (ou não-preemptivos), o que era de se esperar. Também pode-se constatar que o algoritmo SRTF proporciona os melhores tempos médios de execução Tt e de espera Tw, enquanto os piores tempos são providos pelo algoritmo RR (que, no entanto, oferece um melhor tempo de resposta a aplicações interativas). Observa-se também que o tempo total de processamento é constante, pois ele só depende da carga de processamento de cada tarefa e não da ordem em que são executadas. Contudo, esse tempo pode ser influenciado pelo número de trocas de contexto, caso seja muito elevado.



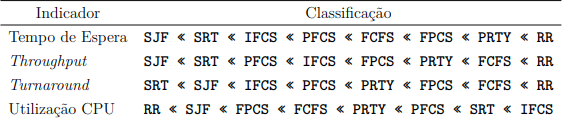


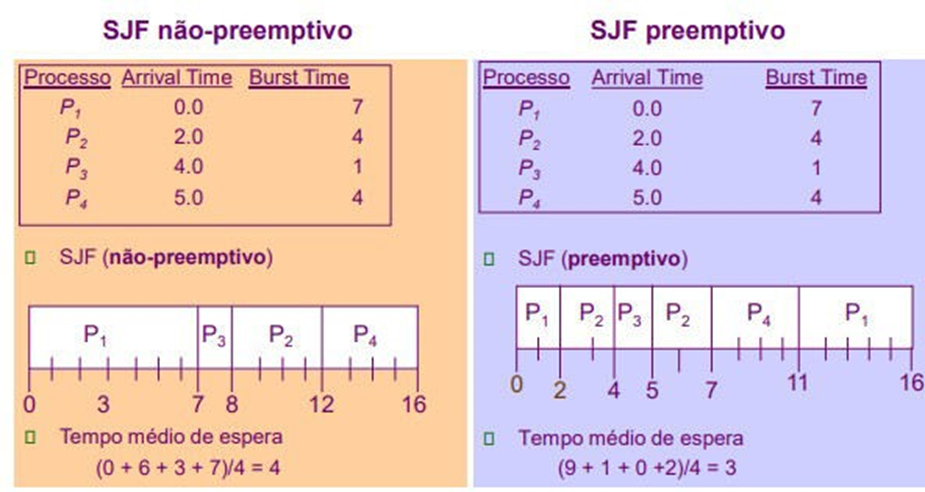


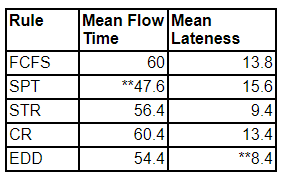


Essas simulações foram feitas por um grupo de alunos por meio de dois simuladores:

->SOSIM

→SimulaRSO





First Come First Serve (FCFS)

Shortest Processing Time (SPT)

Earliest Due Date (EDD)

Slack Time Remaining (STR) = time remaining before due date - remaining processing time

The shortest STR goes first

Critical Ratio (CR) = Time remaining before due date/Remaining processing time

The smallest CR goes first

**Melhor Algoritmo**

Após analisar todos os dados obtidos até então, fomos eliminando cada um dos algoritmos de escalonamento. Inicialmente, excluímos Escalonamento para vários processadores e Escalonamento em tempo real, isso porque eles apresentaram uma maior complexidade, além de uma dificuldade maior, em relação aos demais, na busca de dados.

Em seguida, conseguimos tirar os algoritmos FIFO, SJF. Apesar de terem simples implementações, eles eram um pouco mais lentos, perante o SJF. E sobre o Múltiplas Filas de realimentação, não conseguimos eliminar pois, ela além de poder implementar em qualquer sistema, é capaz de evitar alguns problemas.

Entretanto, debatemos mais um pouco, e vimos que ela não tem uma implementação muito simples. Vale citar, que não conseguimos muitos dados práticos sobre o Múltiplas Filas de realimentação.

Deste modo, por ser preemptivo, ter uma simples implementação, ter dados práticos, ela também tinha um menor tempo de execução, deste modo pensando em uma escala maior, a diferença iria aumentar, e se escolhêssemos o SRT, ela iria poupar tempo e recurso.

Portanto, concluímos que o SRT é o melhor algoritmo, ao considerar todos os dados obtidos que estão presentes nesse documento.

**Referências**

<https://deinfo.uepg.br/~alunoso/2016/FIFO/index.html>

<https://deinfo.uepg.br/~alunoso/2016/FIFO/implementacao.html>

<https://deinfo.uepg.br/~alunoso/2016/FIFO/vantagens_desvantagens.html>

<https://pt.wikipedia.org/wiki/FIFO>

<https://www.gsigma.ufsc.br/~popov/aulas/so1/cap8so.html>

<http://www.inf.ufrgs.br/~johann/sisop1/aula10.scheduling.pdf>

<https://homepages.dcc.ufmg.br/~scampos/cursos/so/aulas/aula3.html>

<https://intranet.ifs.ifsuldeminas.edu.br/kleber.rezende/anteriores/2sem2015/2Redes/FSO_slides_07_Ger%C3%AAncia%20da%20CPU.pdf>

<https://docente.ifrn.edu.br/rodrigotertulino/disciplinas/2015.2/sistemas-operacionais-de-redes/slides/ifrn-introducao-aos-sistemas-operacionais-processos>

<https://www.longa.com.br/armazenagem/fifo-o-que-e-e-quando-utilizar/#:~:text=Ele%20%C3%A9%20usado%20nas%20mais,garantir%20a%20execu%C3%A7%C3%A3o%20dessa%20ferramenta>.

<https://pt.wikipedia.org/wiki/Shortest_job_first>

<http://www.univasf.edu.br/~andreza.leite/aulas/SO/ProcessosEscalonamento.pdf>

<http://wiki.icmc.usp.br/images/3/33/Aula05.pdf>

<http://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=socm:socm-06.pdf>

<https://joaoricardao.files.wordpress.com/2012/07/algoritmos_escalonamento.pdf>

<https://www.inf.pucrs.br/~emoreno/undergraduate/CC/sisop/class_files/Aula04.pdf>

<http://www.di.ubi.pt/~operativos/teoricos/capitulo5.pdf>

<http://nasemanadaprova.blogspot.com/2015/12/algoritmos-de-escalonamento-de-processos.html>

<http://www.univasf.edu.br/~andreza.leite/aulas/SO/ProcessosEscalonamento.pdf>

<https://www.oficinadanet.com.br/post/12781-sistemas-operacionais-o-que-e-escalonamento-de-processos>

<https://www.formiga.ifmg.edu.br/documents/2018/Biblioteca/TCCs_e_Artigos/Daniel_da_Silva_Alves.pdf>

<http://www.di.ubi.pt/~operativos/teoricos/capitulo5.pdf>

<http://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=socm:socm-06.pdf>

<https://pt.slideshare.net/TallesNascimentoRodrigues/sistemas-operacionais-escalonamento-de-processos>

<https://www.semanticscholar.org/paper/A-Priority-Based-MLFQ-Scheduler-for-CPU-Power-Kashif-Helmy/fc22e5f6ff039a71097d1d1f06b57bdbf77b9f84>

<https://www.uky.edu/~dsianita/300/schedule.html>

<https://www.dca.ufrn.br/~affonso/DCA_STR/aulas/escalonamento-STR-parte1.pdf>

<https://www.ibm.com/docs/en/aix/7.2?topic=workloads-multiprocessor-throughput-scalability>

<https://strufs.files.wordpress.com/2008/07/rtos.pdf>

<https://www.dca.ufrn.br/~affonso/DCA_STR/aulas/escalonamento-STR-parte1.pdf>