Sistemas Operacionais

Prof. Robson de Souza

Aulas 17 e 18

Conteúdo: Escalonamento de processos

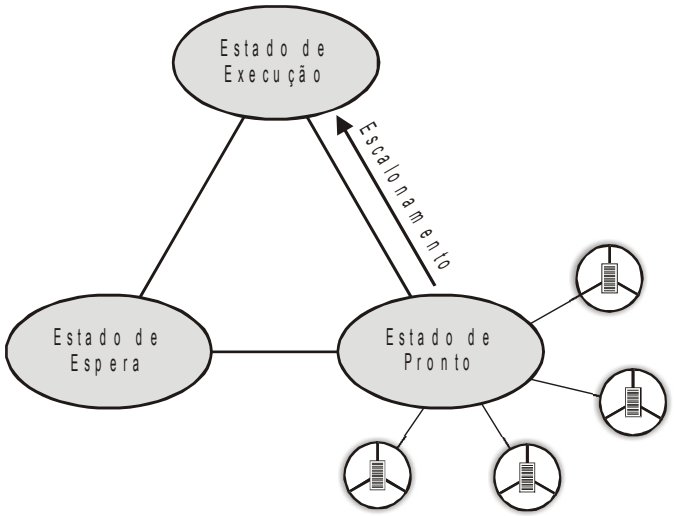
A partir do momento em que diversos processos podem estar no estado de pronto, critérios devem ser estabelecidos para determinar qual processo será escolhido para fazer uso do processador. O conjunto de critérios utilizados para esta seleção são conhecidos como ***política de escalonamento.***

A rotina responsável do s.o. por implementar a política de escalonamento chama-se ***escalonador (scheduler)*.**

Existem alguns critérios que devem ser observados quando o assunto é escalonamento, são eles:

- Utilização do processador → O processador deve manter-se ocupado na maior parte do tempo.
- *Throughput* → Representa o número de processos executados em um determinado intervalo de tempo, deve-se maximizar o *throughput.*
- Tempo de CPU → É o tempo que um processo leva no estado de execução, as políticas de escalonamento não influenciam nesse tempo.
- Tempo de espera → Tempo total que um processo permanece na fila de pronto
- Tempo de *turnaround* → Tempo total que um processo leva desde a sua criação até o seu término. As políticas de escalonamento minimizam este tempo.
- Tempo de resposta → Tempo decorrido entre uma requisição ao sistema ou à aplicação e o instante em que a resposta é exibida

O escalonamento passa a ser importante pois é comum haver múltiplos processos competindo pela CPU. Além de se preocupar em escolher o processo certo para executar, o escalonador também deve se preocupar em fazer um uso eficiente da CPU, pois chavear processos é muito custoso.



Os algorimtos de escalonamento podem ser **preemptivos** ou **não-preemptivos**, onde a preempção é a possibilidade de um s.o. interromper um processo em execução e substituí-lo por um outro.

- Escalonamento *não-preemptivo* → O processo só sai do estado de execução caso termine seu processamento. Nesse caso, o escalonador escolhe um processo para executar e deixa executar até que seja bloqueado ou voluntariamente libere a CPU. Nenhuma decisão de escalonamento é tomada durante as interrupções de relógio.

- Escalonamento *preemptivo* → O s.o. pode executar uma interrupção e passar o processo para o estado de pronto. Nesse caso, o escalonador escolhe um processo e o deixa em execução por um tempo máximo fixado. Ao final do intervalo o processo será suspenso mesmo se ainda estiver executando e o escalonador escolherá outro processo para executar.

Objetivos do algoritmo de escalonamento

Em um algoritmo de escalonamento, a **justiça** é algo importante, o que significa dar a cada processo uma porção justa da CPU. Obviamente que categorias diferentes de processos podem ser tratadas de formas diversas.

Outro fator importante é a **aplicação da política**, ou seja, verificar e assegurar que a política estabelecida é cumprida. O **equilíbrio** também é importante, o que significa manter ou tentar manter todas as partes do sistema ocupadas.

Dependendo do tipo de sistema ainda são importantes a **vazão** que é o número de tarefas realizadas por hora (o objetivo é maximizar) e o **tempo de retorno** que é o tempo médio do momento em que uma tarefa é submetida até o momento em que ela é terminada (o objetivo é minimizar).

Além disso, deve-se levar em consideração a **proporcionalidade**, ou seja, satisfazer às expectativas dos usuários, existem tarefas que os usuários esperam que sejam demoradas, outras não.

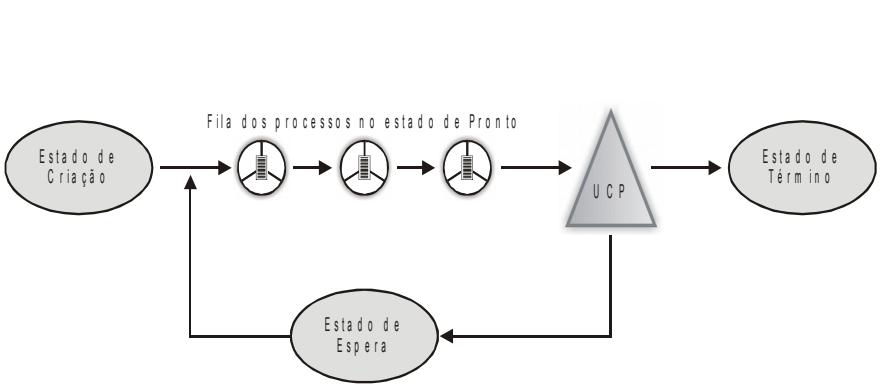
Algoritmos de escalonamento

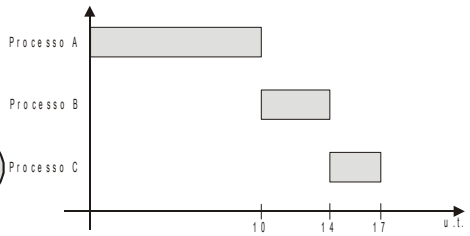
First Come, First Served (FCFS – Primeiro a Chegar, Primeiro a ser Servido):

Esse algoritmo é não-preemptivo e é provavelmente o mais simples algoritmo de escalonamento. A CPU é atribuída aos processos na ordem em que eles a requisitam, existe uma fila de processos prontos, o primeiro que chega é iniciado imediatamente, a partir daí a fila segue e os processos que chegam vão para o final dela.

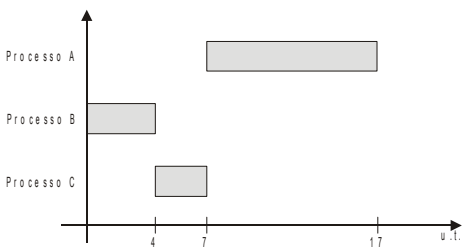
O processo executando não é interrompido, se ele for bloqueado, o primeiro da fila é o próximo a executar e o processo bloqueado que fica pronto vai para o final da fila.

Esse algoritmo é fácil de entender, fácil de programar e é justo. O problema dele é que se existe um processo que gaste muito tempo para executar e outros que executem muito mais rápido, estes tem que esperar muito todas as vezes enquanto o processo que gasta muito tempo precisar executar (e geralmente precisam executar várias vezes).





Processo	Tempo de processador (u.t.)
A	10
B	4
C	3



Shortest Job First (SJF – Tarefa Mais Curta Primeiro)

Seleciona o processo que tiver o menor tempo de processador ainda por executar. Um valor calculado estatisticamente é inserido no contexto de software. Na concepção inicial o escalonamento SJF é um escalonamento não-preemptivo, reduzindo o tempo médio de *turnaround* dos processos.

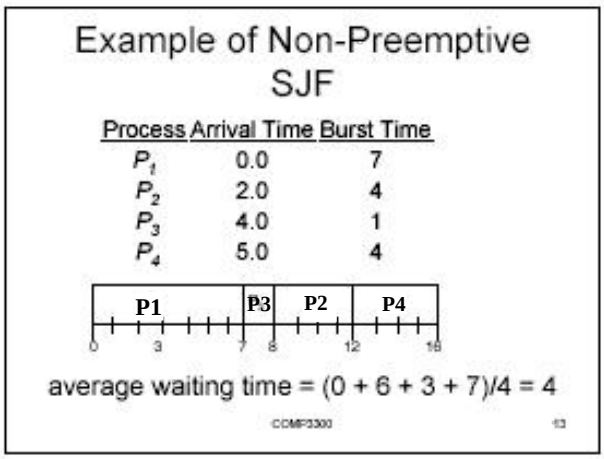
Considerando 4 tarefas com tempo de execução A, B, C e D respectivamente. A primeira termina no tempo A, a segunda no tempo A + B e assim por diante. O tempo médio de retorno nesse caso é (4A + 3B + 2C + D)/4. O mesmo argumento se aplica a qualquer número de tarefas.

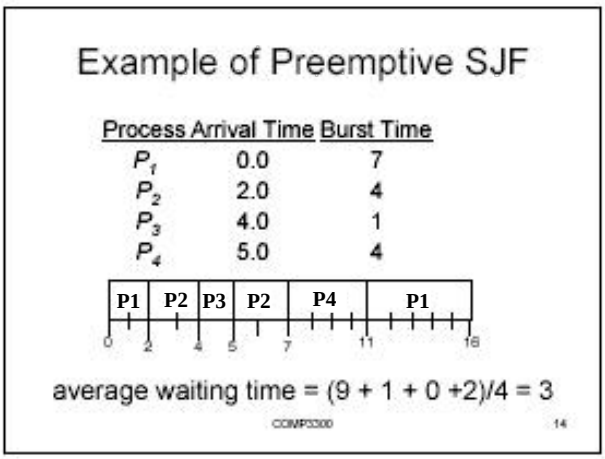
Esse algoritmo é adequado somente para situações em que todas as tarefas estejam disponíveis simultaneamente. O problema principal é que se uma tarefa muito importante precisar de muito tempo para executar, ela vai demorar a ser executada, pois terá de esperar todas as tarefas mais curtas executarem primeiro.

Próximo de menor tempo restante

Essa é uma versão preemptiva da tarefa mais curta primeiro. O escalonador sempre escolhe o processo cujo tempo de execução restante seja o menor, novamente o tempo de execução deve ser previamente conhecido.

Quando chega uma nova tarefa, seu tempo total é comparado com o tempo restante do processo em curso, se a nova tarefa precisar de menos tempo para terminar do que o processo em curso, este será suspenso e a nova tarefa será iniciada.





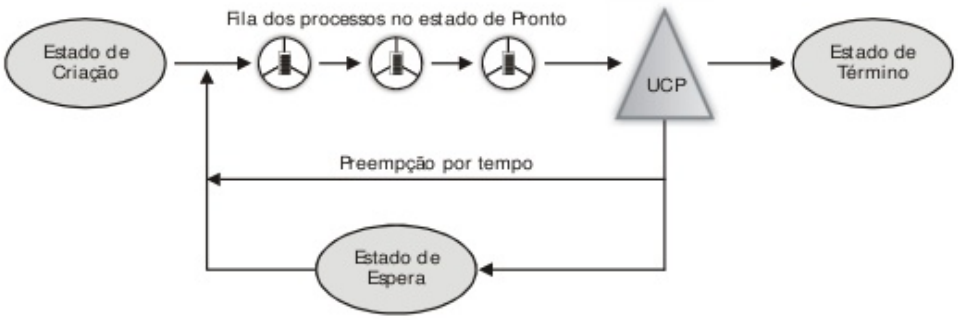
https://stackoverflow.com/questions/27679120/can-shortest-job-first-scheduling-be-subject-to-convoy-effect

Escalonamento circular (round-robin)

Esse algoritmo é um dos mais antigos, simples, justos e amplamente usados. A cada processo é atribuído um intervalo de tempo (quantum), no qual ele é permitido executar. Se ao final do quantum o processo ainda estiver executando, a CPU sofrerá preempção e será dada a outro processo.

Se o processo foi bloqueado ou terminou antes que o quantum tenha decorrido, a CPU é chaveada a outro processo. O escalonador só precisa manter uma lista de processos executáveis. Quando o processo usa todo o seu quantum, ele é colocado no final da fila.

O mais importante aqui é o tamanho do quantum, pois se ele for muito curto, perde-se muito tempo chaveando processos, por outro lado, se o quantum for muito alto e existirem muitos processos, pode-se perder desempenho.



https://www.slideshare.net/leinylson/sistemas-operacionais-aula-9-gerencia-do-processador

Escalonamento cooperativo

Um processo em execução pode voluntariamente liberar o processador, retornando à fila de pronto e possibilitando que um novo processo seja escalonado. A tarefa de liberar o processador é realizada exclusivamente pelo processo em execução. O processo em execução verifica periodicamente a fila de processos em estado de pronto.

Referências bibliográficas:

TANENBAUM, Andrew. 2ª ed. **Sistemas Operacionais Modernos,** Editora Pearson, 2003.

SILBERSCHATZ, Abraham. **Sistemas Operacionais com JAVA,** 6ª ed. Editora Campus

MACHADO, Francis B. **Arquitetura de Sistemas Operacionais,** 4ª ed, LTC, 2007.