Projeto Prático SO1 – Ciclo de Vida de um Processo

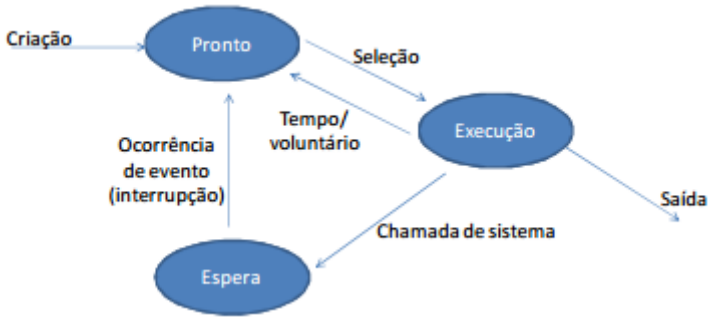
Descrição:

Existem partes de sistemas operacionais que cuidam da ordem em que os programas devem ser executados, denominados escalonadores. Por exemplo, em um sistema de computação de tempo-compartilhado ("*time-shared*") existe a necessidade de manter um conjunto de processos em uma fila, esperando para serem executados. Aqui iremos simular o processo de escalonamento de processos. Sendo assim voce deve implementar um escalonador capaz de resolver as decisões abaixo.

Cada processo é representado por um registro composto por um número identificador do processo e o tempo que ele despende para ser executado (tabela de processo ou PCB). Vale lembrar que cada processo poderá utilizar a CPU por no máximo 10 unidades de tempo por vez.

Pensando nisso, imagine **várias FILAS encadeadas** de controle de execução, onde cada uma tem a sua finalidade específica:

- **Pronto:** programas prontos para serem executados, apenas aguardando a sua vez. Deve ser definida a política de escolha do processo: FIFO ou Prioridade. Aqui a fila é do tipo **FIFO circular** ou round Robin e contém os PCB dos processos prontos.
- **Espera:** programas que iniciaram a execução, mas não terminaram ficando bloqueados por um tempo (sugestão: sortear a possibilidade de bloqueio (se ele estiver executando vai para espera) e desbloqueio de processo (ao ser desbloqueado o programa deverá ir para a fila de Pronto**)**. Aqui a fila é do tipo FIFO e contém os PCB dos processos bloqueados; Vale ressaltar que podem haver vários tipos de filas pois cada dispositivo/recurso é uma fila (fila para HD, fila para teclado, fila para mouse, etc.). Veja abaixo a situação de criação de processos filhos.
- **Execução**: possui apenas o programa que está sendo executado, completada a execução do programa este é finalizado, caso contrário, ele deverá ir para a fila de Espera, pois necessita de recursos que não estão disponíveis (sugestão: sortear a necessidade de recurso, caso precise e se o recurso não estiver disponível, o processo ficará bloqueado indo para a fila de Espera). Se o tempo necessário para finalizar o processo for maior que o tempo de CPU este deverá voltar para a fila de Pronto e aguardar uma nova execução. Independente da situação deverá ser selecionado um novo processo que esteja pronto para ser executado. Quando o processo sair para Bloqueado ou voltar para Pronto, o mesmo deve ser executado apenas durante o tempo restante (tempo total do processo – o tempo de CPU já executado).



Portanto, escreva um programa que seja capaz de executar a simulação citada deixando explícitas as implementações dos módulos abaixo:

- Incluir novos processos na Fila de **Prontos**;
- Retirar um processo da Fila de **Prontos** e colocar para **Execução**;
- Retirar um processo da **Execução** e colocar em **Espera;**
- Retirar um processo da **Espera** e colocar em **Pronto;**
- Quando finalizar o processo de simulação, os elementos que se encontram nas filas devem ser concluídos**.**

Além disso, os processos poderão criar processos filhos durante sua execução. Essa criação deve seguir a ideia da primitiva fork e wait visto em sala de aula. Vale lembrar que os processos filhos só podem finalizar se o processo pai executar um wait. Caso o pai execute um wait este deve ficar bloqueado esperando o filho finalizar sua execução. Após o desbloqueio o pai volta a executar. Para cada filho criado deve existir um wait.

Se um processo vai ou não criar processos filhos, isto deve ser feito de forma aleatória durante a execução do processo. Em algum momento este deve verificar se vai ou não criar filho, se sim criar o filho. Depois deverá ter um wait para esperar o filho acabar.

Os tempos que cada processo executar deve ser feito de forma aleatória.

Ao final da simulação informe:

- A quantidade de processos que foram executados por completo, ou seja, finalizados;
- A quantidade de processos que foram bloqueados e qual o tempo médio de bloqueio envolvendo todos os bloqueados;
- A quantidade de processos que ficaram entre os estados de Execução e Pronto**,** ou seja, quando o tempo de execução dele foi maior que o tempo de CPU e não esteve em estado de bloqueado.
- O tempo total de execução de cada processo. Vale lembrar que temos que contar o tempo de espera também.
- Quantos processos filho cada processo criou e seus tempo de bloqueio/esperar devido a isso.