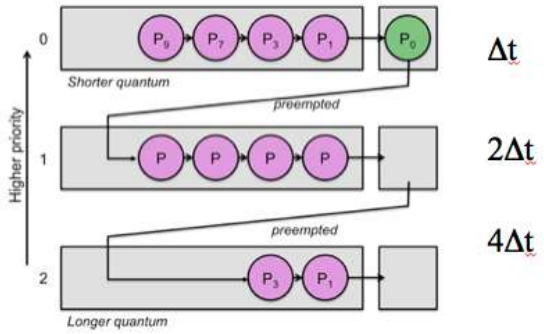
Primeiro Trabalho de Sistemas de Computação – INF 1019 – 2017_2

Escalonamento em Múltiplos Níveis com Feedback (MLF)

O primeiro trabalho consiste em programar na linguagem C um interpretador de comandos e um escalonador de programas. O interpretador deverá ler de `stdio` os nomes de programas-executáveis e parâmetros, e solicitar ao escalonador a execução concorrente dos processos correspondentes a esses programas. O escalonador por sua vez executa de forma intercalada os processos de acordo com uma política de escalonamento que é baseada em filas com níveis de prioridade que dependem do comportamento de cada processo (*cpu bound* ou *I/O bound*). Quando um processo é *I/O bound* ele deve ter sua prioridade aumentada, diferentemente dos processos *cpu bound,* que devem ter sua prioridade diminuída.

O Escalonador

O escalonador deve ter 3 filas de prioridade, F1, F2, F3. Cada vez que um processo (em execução) em uma fila F1 ou F2 apresentar um comportamento cpu bound (por exemplo, quando ele esgotar o quantum de tempo característico da fila atual) ele deve ser transferido para a fila imediatamente abaixo, de prioridade menor, ou seja F2 ou F3, respectivamente. E de modo análogo, sempre que um processo de uma fila, F2 ou F3, apresentar um comportamento mais I/O bound (por exemplo, ele não esgotou o quantum de tempo, foi interrompido por operação de entrada e saída) ele deve ser promovido a fila imediatamente superior, ou seja, a F1 ou F2, respectivamente. Ao serem criados todos os processos são inseridos na fila F1, de mais alta prioridade. Em uma mesma fila, os processos deverão ser escalonados em modo Round-Robin, ou seja, cada processo executando um quantum de tempo, e sendo interrompido preemptivamente, dando a vez ao próximo, também com o mesmo quantum, etc.



No entanto, os processos-usuário deverão apresentar um perfil (CPU e I/O bound) variável durante a sua execução, sendo ora mais CPU e ora mais I/O bound. Isso será determinado pela sequência de tempos (de rajadas de CPU) indicados pelos parâmetros passados ao interpretador, por exemplo:

exec <programa-exec-1> (2, 10, 4) // rajadas de 2, 10, 4 segundos, interrompidas por I/O
exec <programa-exec-1> (20, 20, 2) // rajadas de 20, 20, e 2 segundos, interrompidas por I/O

Detalhes de Implementação

A idéia é que o escalonador seja o processo pai de todo os processos usuário e que use sinais SIGSTOP e SIGCONT para intercalar a execução entre os processos. Ao contrário do que ocorre com um escalonador executando no núcleo do S.O, que "saberia" quando um processo bloqueou por causa de uma chamada I/O, no nosso caso teremos que simular/emular esse "bloqueio por I/O", também através de um envio de sinal ao escalonador por parte do processo que irá "simular um bloqueio por I/O". Para simplificar as coisas, vamos assumir que todos os bloqueios por I/O têm a mesma duração, de 3 segundos. Ou seja, apenas decorrido esse período de "quarentena" o processo bloqueado deverá ser recolocado na fila de prontos correta, de acordo com a politica de escalonamento mencionada acima. Note que uma vez que o processo-usuário terminar ele emite um sinal, o SIGCHD, para o pai, ou seja para o escalonador, que o elimina do sistema.

Todos os seus programas-usuário terão o mesmo comportamento: deverão conter vários comandos for, um para cada rajada de CPU, contendo em seu corpo apenas as linhas `printf("%d\n",getpid())` e `sleep(1)`, e fazendo o numero de interações de acordo com os parâmetros passados para o interpretador e recebidos internamente em `argc/argv`. E em cada final de um comando for cada programa-usuário deve notificar o escalonador, informando que o processo supostamente entrou em um estado "waiting-for-I/O" (w4IO). Para isso deve emitir um sinal Unix para o seu pai (o escalonador). Então, a partir desse sinal w4IO o escalonador poderá ver o que aconteceu com ao processo em execução: se recebeu esse sinal w4IO de bloqueio no I/O antes do quantum (da fila) se esgotar, ou se o quantum da fila chegou ao fim e tal sinal w4IO de bloqueio não chegou. Assim, o escalonador poderá mudar o processo de fila e saberá quantos segundos ainda restaram para a sua execução.

A comunicação/controle entre os processos, portanto, ocorre assim:

Interpretador >>(IPC)>> Escalonador <<(S)>> Processos-usuário

Onde (IPC) é qualquer método de comunicação interprocesso, de sua escolha e (S) são sinais.

Observações Gerais

O trabalho pode ser feito de forma individual ou em dupla, que devem ser enviado via e-mail para luizfernandobessaseibel@gmail.com e jessicatsalmeida@gmail.com até 25/10 às 23:59:59. O trabalho será avaliado na aula de laboratório seguinte, com a presença de todos os membros do grupo. A ausência no dia da apresentação acarretará em nota zero.

Deve ser entregue o código fonte e um relatório indicando que programas serão executados em seu teste, a ordem de entrada para o escalonador e a ordem de execução determinada pelo escalonador, juntamente com uma análise crítica dos alunos sobre o que, de fato, ocorreu (se a execução foi a prevista ou não). Essa explicação também será objeto de avaliação.