Simulando o escalonamento preemptivo de processos com chamdas de sistema

Primeiro Trabalho de Sistemas de Computação – INF 1316

- Prazo de entrega (envio por E-mail): 10 de outubro, 23:59
- Data de apresentação (na bancada): 15 de outubro, 11-13 hs

O primeiro trabalho consiste em programar (em linguagem C e usando as primitivas fork/exec, sinais e comunicação Inter processos do Unix/Linux) um simulador de um kernel/núcleo de sistema operacional (**KernelSim**) que gerencia de 3 a 5 processos de aplicação (A1, A2, A3, etc.) e intercala as suas execuções a depender se estão esperando pelo término de uma operação de leitura ou escrita em um dispositivo de E/S simulado (denominados, D1 e D2), ou o sinal de aviso do termino de sua fatia de tempo (a.k.a. *time slice*). Para isso você deverá implementar também um processo adicional que emula o controlador de interrupções (**InterContoler Sim**), que gera as interrupções referentes ao relógio (*clock*) e ao término da operação de E/S nos dispositivos D1 e D2, sendo respectivamente a interrupção IRQ0 (TimeSlice), IRQ1 (dispositivo D1) e IRQ2 (dispositivo D2). Figura 1 mostra os elementos que devem fazer parte de seu sistema.

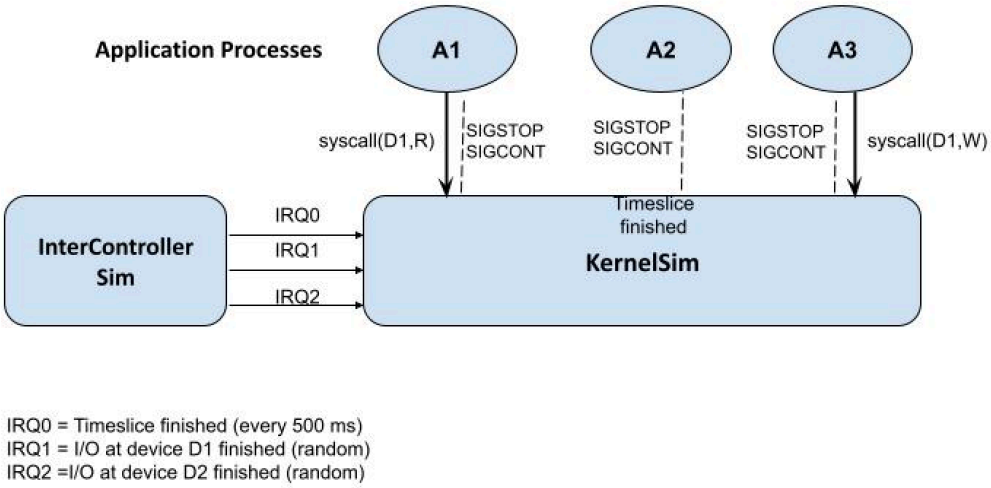

Figura 1: principais processos do seu sistema.

No seu programa, cada um dos elementos em azul-cinza da Fig. 1 deve ser um processo Unix. Tanto InterController Sim para KernelSim devem ser processos infinitos que executam em pararelo.

Uso de Sinais UNIX

Para emular as interrupções de InterController Sim para KernelSim, você pode usar qualquer mecanismo de comunicação, como pipes, Shmem ou FIFO, e deve usar os sinais SIGSTOP e SIGCONT para permitir que o KernelSim interrompa e retome, respectivamente, a execução dos processos A1, A2, A3, .., A6.

Time-sharing

Qualquer processo Ax pode executar ininterruptamente no máximo durante TimeSlice segundos. Portanto, quando chega um IRQ0, o KernelSim, envia um SIGSTOP para o processo que estava executando, escolhe outro processo de aplicação e novamente ativa o mesmo usando o sinal SIGCONT, a menos que este processo esteja esperando pelo retrono de uma syscall de E/S para um dispositivo D1 ou D2. Mais ainda, se dois processos A1 e A2 tiverem executado uma syscall para o mesmo dispositivo, digamos D2, então o primeiro IRQ2 indicará o término da primeira chamada (e desbloquear um dos A1/A2) e o segundo IRQ2 sinalizará o termino do segundo syscall (e irá desbloquear um dos A1/A2) independente a qual chamada de sistema ele corresponde.

Chamada de Sistema (syscall)

Portanto, outra possibilidade é um processo de aplicação, digamos A1, executar um syscall(D2,W), o que faz com que seja imediatamente interrompido pelo Kernel Sim através de um sinal SIGSTOP. E esse mesmo KernelSim coloca então o processo A1 na fila processos bloqueados pelo dispositivo D2, em uma fila interna. Eventualmente, InterController Sim irá gerar um novo IRQ2, o que causará a liberação do primeiro processo que estava na fila dos bloqueados por D2, e o KernelSim irá poder desbloquear o A1 usando SIGCONT.

Em vez de simularmos um tempo específico que cada um dos dispositivos D1 e D2 precisam para executar as operações R/W solicitadas pela syscall(), faremos o InterConntollerSim gerar aleatoriamente interrupções para cada dispositivo (D1/D2) que indicam o termino de qualquer pedido de IO no dispositivo correspondente.

Para isso, implemente o InterConntoller Sim para gerar:

- Um IRQ0 a cada 500 ms (use sleep() dentro do corpo do loop)
- Um IRQ1 com probabilidade P_1 = 0.1 (a cada 500 ms)
- Um IRQ com probabilidade P_2 = 0.05 (a cada 500 ms)

Obs: Se você achar que essas frequências estão dificultando visualizar a execução, você pode alterar a duração do Time-slice e de P_1 e P_2.

Processos de Aplicação

Cada processo de aplicação Ax deve conter um laço (loop) de até MAX iterações e ter um contador de interações chamado de ProgramCounter (PC).

No corpo do loop deverá haver um sleep(1), e uma escolha aleatória, com baixa probabilidade, de executar um syscall(Dx, R-or-W-or-X)., onde D1 ou D2, R, W ou X são aleatórios.

Exemplo:

```
while (PC < MAX) {
       sleep(0.5);
       if (d = rand()%100 +1 < 15) { // generate a random syscall
              if (d % 2) Dx = D1
              else Dx= D2;
              if (d % 3 == 1) Op = R
              else if (d % 3 = 1) Op = W
              else Op = X;
              // generate syscall (Dx, Op)
       }
       sleep(0.5);
 }
```

Troca de Contexto

No seu sistema a troca de contexto acontecerá a cada vez que houver o chaveamento de um processo de aplicação para um outro. E no caso especifico do seu programa esse contexto deve ser o contador PC do processo. Então, ao reativar um processo antes suspenso o seu PC anteriormente guardado deve ser restaurado, para garantir que todos os processos de aplicação executem exatamente MAX interações. Além disso, se se a interrupção de um processo se deveu a um syscall, então os parâmetros desse syscall também devem fazer parte do contexto salvo e restaurado do processo.

Visualizando os estados dos processos

Além disso, seu simulador, em especial o InterController Sim, deve poder ser interrompido e reiniciar do ponto em que parou através do uso de sinais da linha de comando (tipo *kill X PID_simulador*) e quando estiver parado, deve mostrar as seguintes informações sobre todos os processos APx:

- valor do PC
- em que estado está
- se estiver bloqueado, em qual dispositivo e com qual operação
- se está executando (porque o Kernel Sim enviou um SIGCONT e ainda não um SIGSTOP para ele)
- desde o inicio da execução e até esse valor de PC, quantas vezes o processo AP acessou cada um dos dispositivos.
- e se estiver terminado

Observações Finais

Como acontece a avaliação?

O trabalho pode ser feito de forma individual ou em dupla. Deverá ser enviado na Data de Entrega (vide acima) para os e-mails do professor e do monitor, e deverá ser apresentado e explicado na Data de Apresentação na aula de laboratório (necessariamente com a presença da dupla). Cada dia de atraso acarreta um desconto de 1 ponto na nota máxima.

O que deve ser enviado/entregue?

Deve ser entregue o código fonte e um relatório indicando que programas fazem parte do seu trabalho incluindo eventuais programas de teste. Essa explicação também será objeto de avaliação.