Adaptação de guião desenvolvido por Artur Carneiro Pereira

universidade de aveiro deti departamento de electrónica, telecomunicações e informática

IPC— Memória partilhada e Semáforos

Objectivo

Estudo da comunicação entre processos usando memória partilhada e semáforos em Unix.

Guião

1. *Introdução ao uso de memória partilhada*

   a) Entre no directório `base` e analise o código do módulo descrito nos ficheiros `sharedMemory.h` e `sharedMemory.c` que definem um conjunto de operações efectuadas sobre um bloco de memória partilhada e respectiva implementação em Unix.

   b) O programa `reg-cri.c` ilustra os passos de criação / ligação, destruição, anexação / desanexação e acesso a uma região de memória partilhada. Tratando-se de uma aplicação concorrente, evidencia também a necessidade de acesso com exclusão mútua a uma região crítica. Procure identificar o código que constitui a região crítica.

   c) Crie o ficheiro executável (*make reg-cri*) e execute-o diversas vezes, usando as opções disponíveis, procurando determinar a sequência pela qual elas devem ser utilizadas. Em particular, faça-o incrementar 1000 vezes a variável, e procure interpretar o resultado obtido em sucessivas execuções. Compare os resultados com aqueles obtidos com o programa `incrementer` do guião sobre *threads*.

   d) Altere o código de modo a que as regiões críticas sejam executadas em exclusão mútua. Dica: analise o ficheiro `semaphore.h` e use um semáforo para obter a exclusão mútua.

2. *Introdução ao uso de semáforos*

   a) Agora no directório `servcli`, analise o código do módulo descrito nos ficheiros `semaphore.h` e `semaphore.c`, idênticos aos usados no exercício 1, que definem um conjunto de operações efectuadas sobre agregados de semáforos e respectiva implementação em Unix.

   b) O programa descrito no ficheiro `server.c` representa aquilo que se costuma designar de *servidor*, isto é, um programa que presta serviços a outros que lhos solicitam, os chamados *clientes*. Neste caso, recebe *strings* e converte os seus caracteres alfabéticos minúsculos em caracteres alfabéticos maiúsculos, antes de os devolver ao programa que solicitou a conversão. A comunicação é implementada através de quatro funções cujos protótipos estão declarados no ficheiro interface `comm.h`.

   c) Analise o código do programa descrito no ficheiro `client.c`, constitui o *cliente* que interactua com o *servidor* referido acima. A comunicação é também implementada

através de quatro funções cujos protótipos estão declarados no ficheiro interface `comm.h`.

d) O módulo `comm-shm.c` representa a implementação das funções de comunicação, quer do lado do *servidor*, quer do lado do *cliente*, recorrendo a memória partilhada e a semáforos. Procure entender como se processa a interacção e qual é o papel desempenhado por cada um dos semáforos presentes: R, S e A.

e) Regiões de memória partilhada e semáforos são recursos do sistema de operação. Em Unix, eles designam-se de recursos IPC. O comando *ipcs* lista os recursos IPC correntemente atribuídos. Consulte no manual *on-line* a descrição do comando *ipcs* (*man ipcs*). Execute-o e interprete a listagem apresentada.

f) Crie o executável servidor (*make server*) e execute-o numa janela terminal.

g) Execute de novo o comando *ipcs* (numa outra janela terminal) e constate as alterações entretanto ocorridas.

h) Crie o executável cliente (*make client*) e execute-o numa outra janela terminal. Lance pelo menos mais um processo cliente numa nova janela terminal. Constate como decorre a interacção comutando entre as diferentes janelas.

i) Os programas servidor e clientes não contemplam mecanismos de terminação. Termine-os, usando a combinação de teclas CRTL-C. Considere as várias situações.

j) Volte a lançar o servidor e procure entender o que ocorre.

k) Consulte no manual *on-line* a descrição do comando *ipcrm* (*man ipcrm*) que possibilita a remoção de recursos IPC entretanto atribuídos pelo sistema de operação e efectue a remoção dos recursos que tinham sido reservados pelo servidor.

l) Como poderia tornar a interacção independente da velocidade relativa de execução dos clientes? Note que, na versão actual, enquanto um cliente não ler o resultado do processamento da sua mensagem pelo servidor, mais nenhum cliente consegue processar o que quer que seja.

Tarefa – Entre no directório `dinner` e analise o código dos programas descritos pelos ficheiros `probSemFilos.c` e `semFilos.c`. Trata-se da implementação do *problema do jantar dos filósofos* de Dijkstra, que foi discutido nas aulas teóricas, na versão de negação da condição de *espera com retenção*, usando semáforos e memória partilhada.

Crie os ficheiros executáveis `probSemFilos` e `semFilos` (*make all*), execute o programa `probSemFilos` algumas vezes e interprete os resultados obtidos.

Altere os ficheiros `probSemFilos.c` e `semFilos.c`, designando as novas versões de `probSemFilosAlt.c` e `semFilosAlt.c`, de modo a implementar a versão de negação da condição de *não libertação* do mesmo problema.