Adaptação de guião desenvolvido por Artur Carneiro Pereira

universidade de aveiro deti departamento de electrónica, telecomunicações e informática

IPC— Filas de mensagens

Objectivo

Estudo da comunicação entre processos usando filas de mensagens em Unix.

Guião

1. *Introdução ao uso de filas de mensagens*

   O directório `base` contém uma implementação, agora usando filas de mensagens, do mesmo modelo cliente-servidor já descrito no módulo sobre memória partilhada e semáforos.

   a) Analise o código do módulo descrito nos ficheiros `message.h` e `message.c` que definem um conjunto de operações efectuadas sobre uma fila de mensagens e respectiva implementação em Unix.

   b) Analise o código do programa descrito pelo ficheiro `server.c`. Este programa representa aquilo que se costuma designar de *servidor*, isto é, um programa que presta serviços a outros que lhos solicitam, os chamados *clientes*. Recorde que ele 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 cabeçalho `comm.h`.

   c) Analise o código do programa descrito pelo ficheiro `client.c`, que 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 cabeçalho `comm.h`.

   d) O módulo `comm-msg.c` representa a implementação das funções de comunicação, quer do lado do servidor, quer do lado do cliente, recorrendo a filas de mensagens. Procure entender como se processa a interacção, nomeadamente, qual é o papel desempenhado pelas duas filas de mensagens.

   e) Filas de mensagens 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.

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) Analise o código do programa descrito pelo ficheiro `client2.c`. Este constitui uma variante do *cliente* que interactua com o *servidor* referido acima em que o cliente aguarda autorização do utilizador antes de recolher a mensagem de resposta.

m) Execute o servidor numa janela terminal, crie o executável do novo cliente (*make client2*) e execute-o em pelo menos duas janelas terminal. Envie mensagens pelos dois processos cliente, mas dê autorização para ler a resposta primeiro no processo que enviou a mensagem em último lugar. Constate que as respostas não chegam correctamente a cada cliente. Procure explicar porquê.

n) Explique quais as alterações que permitem garantir que todas as mensagens são entregues correctamente. Implemente essas alterações e teste a sua solução.

Tarefa 1 – Construa um programa que, usando filas de mensagens como canal de comunicação, permita a troca de mensagens entre dois terminais.

Tarefa 2 – Entre no directório `dinner` e analise o código dos programas descritos pelos ficheiros `probMsgFilos.c` e `msgFilos.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 passagem de mensagens.

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

Altere os ficheiros `probMsgFilos.c` e `msgFilos.c`, designando as novas versões de `probMsgFilosAlt.c` e `msgFilosAlt.c`, de modo a implementar a versão de negação da condição de *não libertação* do mesmo problema.