

Fundamentos de Sistemas de Operação

FCT/UNL - 2009/2010

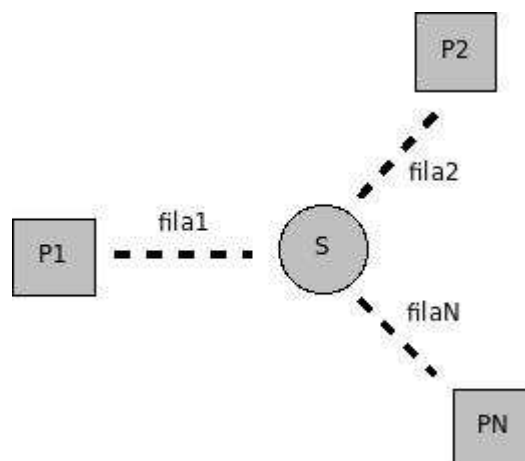
Trabalho prático 4

Comunicação entre processos – Filas de Mensagens

(2 aulas)

Introdução

Considere um sistema de conversação (*chat*) que permite a troca de mensagens entre vários processos. As mensagens (que na realidade são *strings*) são dirigidas a um servidor centralizado que tem como tarefa fazer a sua disseminação. Esta disseminação depende do tipo da mensagem: mensagens públicas devem ser entregues a todos os processos, excepto o que enviou; mensagens privadas devem ser entregues apenas ao destinatário.



Os processos têm à sua disposição uma interface de programação (API) que abstrai os detalhes do mecanismo de comunicação e sincronização, disponibilizando ao programador as seguintes funções:

- `void enviaMensagem(char *mensagem)` – envia uma mensagem pública.
- `void enviaMensagemPrivada(char *mensagem, int proc)` – envia uma mensagem privada para `proc`, onde `proc` distingue univocamente um processo.
- `char* recebeMensagem()` - recebe uma mensagem.

As funções de envio não são bloqueantes, i.e., não esperam que a mensagem seja entregue. No entanto, devem garantir que a mensagem é enviada. A função de recepção deve bloquear até que exista uma mensagem para ler.

Descrição genérica da utilização do sistema de *chat*

Para usar o sistema de *chat*, um utilizador procede da seguinte forma:

1. Numa janela, executa a aplicação servidor: `$ servidor <número>`, onde `<número>` especifica quantos clientes se vão ligar ao servidor.
2. Abre outras tantas janelas quanto o `<número>` acima especificado; em cada uma lança um cliente, escrevendo `$ cliente <id>`, onde `<id>` é um número que vai de 1 a `<número>`; por exemplo, se lançou o servidor com a linha `$ servidor 2`, deve abrir 2 janelas para clientes; na primeira escreve `$ cliente 1`, e na segunda, `$ cliente 2`.
3. Num cliente, a interacção com o utilizador desenrola-se da seguinte forma: o cliente afixa um *prompt*, e espera pela entrada de uma *string* com o seguinte formato: “<cabeçalho> texto” (aspas não incluídas); no caso do cabeçalho:
 - 3.1. ser formado por um `<id>` ou então pela palavra “*todos*”, a mensagem “texto” será enviada para o cliente `<id>` ou para todos os clientes, respectivamente (via servidor).
 - 3.2. ser formado pela palavra **fim**, o cliente (envia-a para o servidor e) termina.

Fase 1

- a) Implemente as funções da interface recorrendo às filas de mensagens Unix (*message queues*). Deve usar apenas uma fila para a ligação entre cada processo e o servidor. Para cada função assuma que essa fila já está criada e que o seu identificador (`msqid`) se encontra na variável global `fila`.
- b) Recorrendo ainda às filas de mensagens Unix implemente a função servidor que efectue a recepção e disseminação de mensagens públicas e privadas.

Fase 2

- a) Implemente o programa cliente, que é lançado como descrito em 3.
- b) Implemente o programa servidor, que é lançado como descrito em 1.
- c) Teste a aplicação...

Sugestões:

- Quando o servidor arranca, cria as `<número>` filas de mensagens usando um valor base fixo, conhecido, para chave da fila adicionado com o `<id>` do cliente que a vai usar; por ex., para dois clientes, seria `BASE_KEY+1` para chave de uma fila e `BASE_KEY+2` para a outra. Guarde os identificadores das filas (`msqid`) que vai criando num vector de dimensão suficiente.

- Defina o valor de `BASE_KEY` num *include*, `chat.h`, que usará tanto no servidor como no cliente (escolha um valor fácil de identificar se tiver de usar o comando **ipcs**).
- Use a seguinte codificação para as etiquetas das mensagens cliente/servidor: **<número>** para identificar um único destinatário; **-1** para representar “todos” (difusão); e **0** para representar “fim”.

Relatório

Entregue um pequeno texto em que propõe uma forma para resolver a questão da recepção bloqueante. [Nota: não se pretende código, mas uma descrição da solução que se propõe implementar, com algum detalhe que permita ajuizar se ela é exequível...]

Bibliografia

- Slides e apontamentos das aulas teóricas
- Capítulo 8 do livro *Unix System Programming* (secção 8.3.2)
- Manuais on-line (*man pages*)