departamento de informática

Sistemas Operativos

Trabalho prático 3

Sistema P2P para partilha de ficheiros

Versão 3.1.0

1 Introdução

O presente trabalho prático visa uma maior familiarização com a programação de mecanismos de sincronização e comunicação entre processos utilizados em sistemas operativos no ambiente Unix/Linux. É realizado em grupo. Esses grupos já estão definidos na unidade curricular. O trabalho está sujeito a apresentação e defesa, realizada individualmente por cada aluno. As defesas serão marcadas em data oportuna, comunicadas aos alunos e publicadas na plataforma de e-learning.

É, obviamente, interdita a cópia parcial ou integral de trabalhos e, a ser detetada, conduzirá à adequada penalização dos envolvidos. O trabalho deverá apresentar-se na forma de código fonte e de um relatório claro e conciso, que também será objeto de avaliação.

2 Contextualização

As especificações apresentadas neste enunciado poderão não corresponder à solução mais simples ou eficiente, devendo ser encaradas como um pretexto para conjugar, num único trabalho prático, um conjunto alargado de mecanismos de comunicação estudados, nomeadamente, ficheiros, sinais, pipes unidirecionais, pipes nomeados, filas de mensagens, memória partilhada, semáforos.

De modo a simplificar a avaliação deste trabalho, as indicações devem ser integralmente respeitadas. É, contudo, admissível, que sejam tomadas opções diferentes, desde que, claramente, não representem uma forma de "contornar" algum dos aspetos em estudo. Estas opções devem encontrar-se devidamente fundamentadas no respetivo relatório. As opções tomadas no sentido de resolver circunstâncias que aqui não sejam explicitadas deverão, também, ser devidamente documentadas.

3 Descrição do problema

Pretende-se implementar um sistema P2P (*peer-to-peer*) de partilha de ficheiros através de mecanismos aprendidos nas aulas, tais como pipes, pipes nomeados, filas de mensagens, memória partilhada, sinais, semáforos.

A aplicação deverá ser desenvolvida em linguagem C, no ambiente de desenvolvimento utilizado nas aulas práticas.

4 Implementação

A aplicação deve obedecer ao indicado na Figura 1 e aos requisitos explicados abaixo.

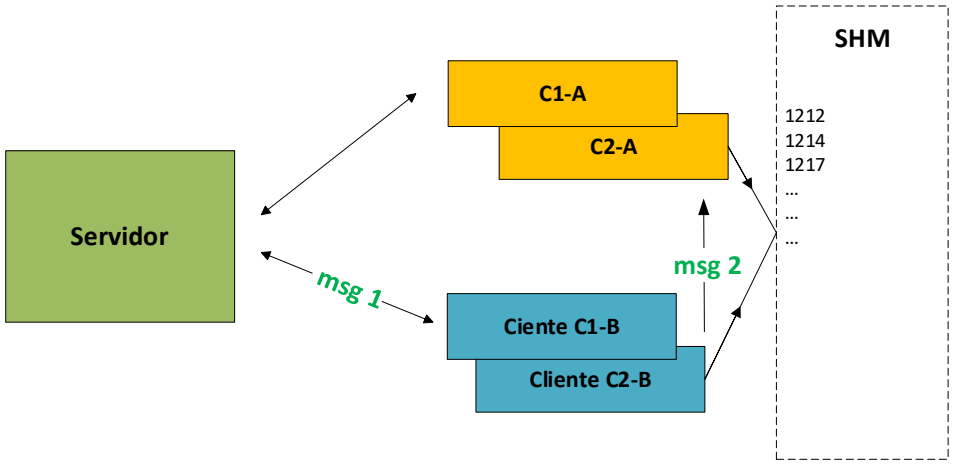

Figura 1 – Representação do sistema a implementar

msg 1 (Fila de mensagens 1) – serve para a troca de mensagens entre clientes e entre cliente – servidor.

msg 2 (Fila de mensagens 2) – serve para o envio de um ficheiro de um C2 para um C1.

SHM – Lista os PIDs dos C2 disponíveis.

Devem utilizar os sinais SIGINT, SIGCHLD, SIGUSR1 ou outros para controle dos processos (como terminação, etc.)

4.1 Funcionalidades do Cliente

Quando o processo cliente é executado, deve, através da primitiva fork(), dividir-se em dois: o C1, o pai, que disponibiliza um *prompt* (linha de comandos) para o utilizador, e o C2, o filho, que serve pedidos de ficheiros.

O C1 deverá disponibilizar um *prompt* que aceita os seguintes comandos:

4.1.1 connect

“connect” liga-se ao servidor através do envio de uma mensagem com o tipo “1” e acrescentando ao conteúdo o PID do C2. Por exemplo, enviando “connect 1111” na fila de mensagens 1, em que o “1111” representa o PID do processo C2 gerado pelo fork.

4.1.2 procura

“procura abcde” pesquisa nos restantes clientes um ficheiro com o nome “abcde”.

O C1 deve enviar a mensagem complementando com o seu próprio PID (ex: “procura abcde 1111”) para facilitar a resposta.

O envio é feito para todos os clientes C2 presentes na SHM. Ou seja, cada mensagem terá o PID no tipo e “procura YYYY XXXX” no conteúdo.

Periodicamente (de 30 em 30 segundos, por exemplo), esse C1 deve verificar se já alguém respondeu na fila de mensagens 1. Isto é, procura uma mensagem cujo tipo é igual ao seu PID. Um exemplo de uma mensagem de resposta seria “resposta abcde 1214”, onde “1214” corresponde a um exemplo de um PID do processo (do tipo C2) que tem o ficheiro e que foi o responsável pela resposta.

Nessa altura deve mostrar essa mesma informação ao utilizador: resposta abcde 1214

4.1.3 quero

“quero abcde 1214” significa que o utilizador pede o ficheiro “abcde”. O C1 antes de enviar a mensagem com a ordem para a fila de mensagens 1, deve identificar o PID do destinatário escolhido pelo utilizador (ex. 1214) e utilizar esse PID no tipo da mensagem.

Deve ainda acrescentar um 4º argumento ao conteúdo da mensagem que é o seu PID (ex: “quero abcde 1214 1111”, onde 1111 é o seu próprio PID).

Periodicamente deve escutar se o ficheiro foi enviado numa fila de mensagens 2. O ficheiro é enviado numa mensagem cujo tipo é o PID do processo C1 e o corpo é uma estrutura que contém o nome do ficheiro e o seu conteúdo. Ver na especificação do C2 a composição da estrutura.

4.1.4 shutdown

“shutdown” quando o utilizador se desliga, deve enviar uma mensagem ao servidor, ou seja mensagem do tipo “1”, acrescentando o PID do C2 ao conteúdo. Por exemplo, “shutdown XXX” onde XXX é o PID do processo C2 a ele associado. Envia também um sinal “SIGUSR1” ao C2.

4.1.5 O C2 deverá:

1. Assim que lançado, ler os ficheiros da diretoria atual (máximo 20) e manter os nomes em memória para poder responder a pedidos.

2. Após isso, deverá estar à escuta da fila de mensagens 1 por pesquisas ou pedidos e responder na fila de mensagens apropriada: a pesquisas na 1 e a pedidos na 2.

O C2 serve os primeiros 1024 bytes do ficheiro na fila de mensagens 2 numa estrutura do tipo:

```
typedef struct
{
        int type;
        char nome[256];
        char conteudo[1024];
} msgStruct;
```

O C2 deve armar o sinal SIGUSR1 quando é iniciado e quando o receber, terminar imediatamente.

4.2 Funcionalidades do Servidor

O Servidor deverá ler da fila de mensagens 1 o pedido de "connect" e "shutdown" por parte dos clientes e atualizar a SHM.

Só aceita no máximo 10 clientes em simultâneo. A partir desse valor descarta a mensagem do cliente e não atualiza a SHM.

Sempre que um cliente enviar a mensagem *shutdown* deverá remover o C2 da SHM e utilizar essa posição para futuros clientes.

4.3 Valorização do trabalho

As configurações, como as filas de mensagens, podem estar num FIFO, por exemplo, ou outro mecanismo IPC, o que valorizará o trabalho. Por exemplo, em vez de chamar o servidor ou o cliente com as filas de mensagem a utilizar, ou seja, em vez de

servidor 10 11 ou cliente 10 11

chamar, apenas, servidor ou cliente e as filas de mensagens serem lidas do mecanismo criado para o efeito.

5 Tratamento de erros

Em acréscimo ao normal controlo de erros nas aplicações, a implementação deste trabalho deverá também contemplar o tratamento de determinadas situações de anomalia. A título indicativo, apresentam-se alguns exemplos:

- execução do programa cliente, antes de lançar o programa servidor;
- tentativa de utilizar um recurso IPC, entretanto removido;
- permanência de chaves IPC resultantes de execuções passadas mal sucedidas;
- erros nos pedidos feitos ao servidor (um pedido que não existe, etc).
- não eliminar os mecanismos de comunicação utilizados

6 Qualidade do código

O código deve ser construído de forma a tornar simples não só o seu desenvolvimento como a própria leitura/avaliação. Devem ser usados comentários (de forma coerente e consistente) de modo a que, por um lado, se torne fácil a interpretação de passagens mais complexas e que, por outro, se demonstre que quem escreveu as respetivas instruções está consciente da sua semântica e implicações. Este aspeto será relevante na avaliação do trabalho.

7 Relatório

O relatório, que se pretende breve, deverá justificar as opções tomadas, bem como eventuais desvios relativamente às especificações constantes deste enunciado. Devem ser identificadas as principais dificuldades encontradas e respetivas soluções (quando não mencionadas neste enunciado). No caso de o trabalho entregue não implementar todos as especificações referidas, as respetivas lacunas deverão necessariamente fazer parte desse relatório.

8 Entrega dos Trabalhos

O programa deve ser enviado para a plataforma de e-learning (https://moodle.estgv.ipv.pt), assim como um relatório com a descrição do programa implementado, respetivas limitações e problemas; no relatório deve constar uma explicação dos mecanismos IPC utilizados e para que circunstâncias.

A data limite de entrega está definida nesta atividade de submissão do trabalho. Todos os trabalhos entregues depois dessa data não são considerados.

9 Updates deste enunciado

De forma a viabilizar a eventual introdução de atualizações a este documento, chama-se a atenção de que é requisito deste trabalho a verificação da existência de versões atualizadas do enunciado. Nesse sentido, ao enunciado está associada uma versão.

10 Cenários de utilização – Simulação

Cliente inicia

```
quental@viriato:~/tp03$ cliente 10 11

 ========
Servidor não está a executar ...
=========
quental@viriato:~/tp03$
```

Servidor inicia

```
quental@viriato:~/tp03$ servidor 10 11
Servidor PID=77508
a receber pedidos de clientes...
```

Cliente liga-se ao servidor

```
quental@viriato:~/tp03/outro$ cliente 10 11
Cliente PID=77767
Escreva comandos ...
connect
? ... Escreva comandos ...
```

Servidor regista cliente

```
quental@viriato:~/tp03$ servidor 10 11
Servidor PID=77508
a receber pedidos de clientes...
C1 77767 C2 77768  connect
Ligar C2 1 com o pid 77768
```

Outro cliente liga-se

```
quental@viriato:~/tp03/um$ cliente 10 11
Cliente PID=78243
Escreva comandos ...
connect
? ... Escreva comandos ...
```

Servidor faz o registo do cliente

```
quental@viriato:~/tp03$ servidor 10 11
Servidor PID=77508
a receber pedidos de clientes...
C1 77767 C2 77768  connect
Ligar C2 1 com o pid 77768
a receber pedidos de clientes...
C1 78243 C2 78244  connect
Ligar C2 2 com o pid 78244
a receber pedidos de clientes...
```

Cliente pesquisa ficheiro

```
quental@viriato:~/tp03/outro$ cliente 10 11
Cliente PID=77767
Escreva comandos ...
connect
? ... Escreva comandos ...
procura fich
? ... Escreva comandos ...
```

Cliente recebe resposta

```
quental@viriato:~/tp03/outro$ cliente 10 11
Cliente PID=77767
Escreva comandos ...
connect
? ... Escreva comandos ...
procura fich
? ... Escreva comandos ...
? ...
 Recebi resposta! O ficheiro fich está no C2 7824
4
```

Cliente pede o ficheiro

```
quero fich 78244
Escreva comandos ...
Conteúdo do ficheiro:
 Olá
Sou o ficheiro fich
Estou na pasta um
Se me pediste, aí vou eu
E pronto! Terminei.
```

Cliente faz shutdown

```
shutdown
? ... Escreva comandos ...
Cliente 77768 vai terminar
? ... Cliente 77767 vai terminar
```

E servidor retira cliente

```
a receber pedidos de clientes...
C1 77767 C2 77768  shutdown
Desligar C2 2 PID=77768
a receber pedidos de clientes...
```

Servidor termina e terminam todos os clientes

Bom trabalho!!