



Sistemas Operativos

Trabalho prático 3

Mecanismos de sincronização e comunicação entre processos

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, se necessário, 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, um conjunto alargado de mecanismos de comunicação estudados, nomeadamente, *ficheiros*, *sinais*, *pipes unidireccionais*, *pipes nomeados*, *filas de mensagens*, *memória partilhada* e, eventualmente, *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 e implementação

Pretende-se implementar um servidor que atende a pedidos de vários clientes utilizando mecanismos IPC tais como FIFO, FILAS DE MENSAGENS ou MEMÓRIAS PARTILHADAS e, se necessário, SEMÁFOROS para sincronização de processos e resolução de problemas relacionados com secções críticas, *race conditions*, etc

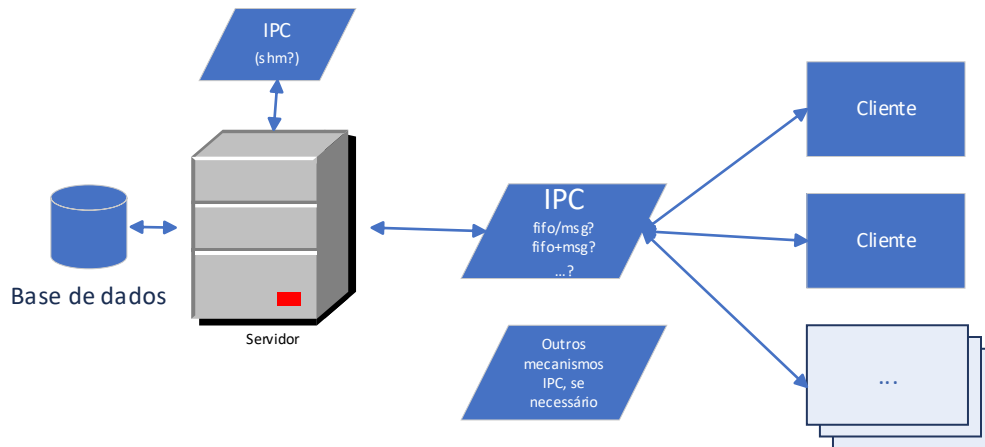


Figura 1 – Esquema simplificado

3.1 Servidor

O servidor contém uma base de dados de uma livraria para gestão das vendas dos livros. Cada registo é constituído por uma estrutura que contém diversos campos tais como o *código do livro*, o *título*, o *autor* e o respetivo *preço*. Estes campos são obrigatórios, mas poderá criar outros, caso os considere convenientes.

Quando recebe o pedido de um cliente (o código do livro ou o título ou a melhor forma que o estudante considerar), o servidor responde ao cliente, enviando os dados do livro que correspondem a esse código (Figura 9 e Figura 10).

Os seguintes itens devem ser levados em consideração:

1. Ao iniciar o servidor, este deve carregar a base de dados para a memória para poder responder a pedidos mais rapidamente (os alunos podem escolher outra solução).
2. Os clientes utilizam um ou vários dos mecanismos IPC (à escolha do aluno) para a comunicação entre servidor e cliente (preferencialmente filas de mensagens).
3. A resposta aos pedidos é feita de uma forma (pseudo) paralela, isto é, enquanto um processo está a pesquisar a base de dados para responder a um determinado pedido de um cliente, o processo servidor já deve estar disponível para receber um novo pedido de outro(s) cliente(s).
4. Deve garantir que os processos filho, quando terminam, não fiquem no estado zombie.

5. O servidor deve eliminar os mecanismos IPC utilizados, onde são colocados os pedidos, aquando da receção do sinal SIGINT.
6. Antes de finalizar, o servidor deve terminar os clientes ativos ou fazer com que os clientes terminem. Esta solução deve, preferencialmente, utilizar sinais.
7. O cliente e o servidor devem proceder à eliminação dos mecanismos IPC, utilizados por cada um deles, antes de terminarem.

3.2 Cliente

O programa cliente envia um pedido ao servidor e aguarda pela resposta. Quando obtém a resposta, escreve-a no monitor.

O cliente deve ter 2 processos:

- um lê os pedidos do *stdin*
- outro está à escuta de respostas do servidor.

O cliente deve permitir executar a Shell para executar comandos, tais como *ps -afH* para verificar o estado dos processos em execução (a forma interativa será valorizada).

A aplicação desenvolvida deve suportar multiprocessamento com vários clientes ativos e a solicitar pedidos simultâneos.

4 Algumas opções de implementação

Os alunos poderão utilizar estruturas como as que se indicam a seguir (exemplo.h):

// definição da estrutura da BD (*isto é, apenas, um exemplo*)

```
struct base_de_dados_produtos {  
    int  codigoLivro;  
    char tituloLivro[100];  
    char autorLivro[100];  
    float preco;  
    ...  
};
```

// exemplo de estrutura para a comunicação entre clientes e servidor

```
struct mensagem {  
    long para;  
    long de;  
    char texto[255];  
};
```

```
// exemplo de estrutura que guarda os pid dos clientes
struct clientes {
    long pid_do_cliente;
    ...
};
```

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);
- outros ...

6 Qualidade do código

O código deve ser construído de forma a tornar simples não só o seu desenvolvimento como a própria leitura/avaliação, devidamente indentado. 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 todas as especificações referidas, as respetivas lacunas deverão necessariamente fazer parte desse relatório.

8 Entrega dos Trabalhos

O programa fonte (.c e .h se existirem) e o relatório (.docx ou .pdf) devem ser enviados para a plataforma de *e-learning* (<https://moodle.estgv.ipv.pt>). Deverá enviar, apenas, um ficheiro compactado (*em formato ZIP*) identificado com o nome tipo *TP3-pv1000-pv1001-pv1002.zip*, em que os números indicados representam a identificação dos elementos do grupo. O programa também deve ter identificado os elementos do grupo no próprio código.

A data limite de entrega está definida nesta atividade de submissão do trabalho. 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.

Esta é a versão 3.1.0.

10 Reutilização de código de terceiros

Por favor, leia esta secção com muita atenção.

Todos os alunos são incentivados a resolver a tarefa de forma independente com o seu próprio código-fonte. Entretanto, ocasionalmente, pode haver razões justificáveis para usar código-fonte de terceiros. Observe que, se usou um ou mais trechos de código-fonte de terceiros no seu programa (isso inclui as situações em que fez pequenas modificações no código-fonte de terceiros), o seu programa será aceitável somente se tiver satisfeito todas as seguintes condições:

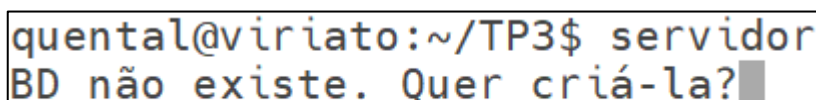
- O código-fonte de terceiros está bem identificado, incluindo os números de página e de linha, em seu código-fonte e também no relatório, e
- O código-fonte de terceiros está totalmente comentado na listagem do seu programa. Todas as variáveis, funções e principais estruturas de controle devem ser comentadas para mostrar claramente que entende a lógica do código, e
- O código-fonte de terceiros representa menos de 10% do seu programa (em termos de número de linhas), excluindo o código especificamente permitido para uso.
- O código-fonte não vem de um trabalho de estudantes, seja da Escola Superior de Tecnologia e Gestão ou não.

O não cumprimento de qualquer uma das condições acima resultará na atribuição da nota 0 (zero) para todo o seu trabalho.

11 Cenários de utilização – Simulação

Ver nas páginas seguintes um cenário de utilização.

1. Servidor inicia



```
quental@viriato:~/TP3$ servidor  
BD não existe. Quer criá-la?
```

Figura 2 – A primeira vez que o servidor executa, cria a base de dados

```
BD não existe. Quer criá-la?s

Codigo do livro: 1
Título: Sistemas Operativos

Nome do autor: Carlos Quental
Preço: 20

Codigo do livro: 2
Título: Linguagem C

Nome do autor: Dennis Ritchie
Preço: 10

Codigo do livro: 3
Título: C2X - Nova revisão da linguagem C

Nome do autor: Open Standards
Preço: 0

Codigo do livro: 4
Título:

Servidor iniciou...
```

Figura 3 – Inserção de dados na base de dados

```
quental@viriato:~/TP3$ servidor

Servidor iniciou...
```

Figura 4 – Servidor inicia

2. Clientes iniciam

```
quental@viriato:~/TP3$ cliente

0 servidor que tentou contactar nao esta disponivel

quental@viriato:~/TP3$ █
```

Figura 5 - Cliente tenta ligar ao servidor

```

quental@viriato:~/TP3$ cliente

=====
Para terminar prima as teclas Ctrl+C
Como exemplo, se quiser executar o comando ps -afH prima as teclas Ctrl+Z

=====

Insira os códigos dos produtos e prima ENTER:

```

Figura 6. Cliente inicia

```

quental@viriato:~/TP3$ servidor

Servidor iniciou...
Cliente 500378 iniciou
Cliente 500431 iniciou

```

Figura 7 – Servidor mostra os clientes que iniciam

```

^ZEscreva o comando a executar: ps -afH

```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
quental	500378	471018	0	12:46	pts/3	00:00:00	cliente
quental	500379	500378	0	12:46	pts/3	00:00:00	cliente
quental	500800	500378	0	12:47	pts/3	00:00:00	ps -afH
quental	500431	470852	0	12:46	pts/2	00:00:00	cliente
quental	500432	500431	0	12:46	pts/2	00:00:00	cliente
quental	500335	435426	0	12:46	pts/0	00:00:00	servidor

```

Ok. Voltámos

```

Figura 8 – Processo permite ir à SHELL executar um comando – podem ver-se os vários clientes e respetivos filhos

3. Clientes efetuam pedidos

```

quental@viriato:~/TP3$ cliente

=====
Para terminar prima as teclas Ctrl+C
Como exemplo, se quiser executar o comando ps -afH prima as teclas Ctrl+Z

=====

Insira os códigos dos produtos e prima ENTER:
4
Cliente 544197. Código inexistente
1
Cliente 544197.
Título: Sistemas Operativos
Autor: Carlos Quental
Preço: 20.00

```

Figura 9 – Cliente pede produto com código inexistente e, de seguida, com código existente

```
Insira os códigos dos produtos e prima ENTER:  
2  
Cliente 544943.  
Título: Linguagem C  
Autor: Dennis Ritchie  
Preço: 10.00
```

Figura 10 – Outro cliente faz solicitação de livro

```
Servidor iniciou...  
Cliente 544197 iniciou  
Cliente 544197 solicitou o código 4  
Cliente 544197 solicitou o código 1  
Cliente 544943 iniciou  
Cliente 544943 solicitou o código 2
```

Figura 11 – Servidor mostra pedidos de clientes

4. Cliente termina

```
Insira os códigos dos produtos e prima ENTER:  
4  
Cliente 544197. Código inexistente  
1  
Cliente 544197.  
Título: Sistemas Operativos  
Autor: Carlos Quental  
Preço: 20.00  
^CCliente 544197 vai terminar e informou o servidor.  
quental@viriato:~/TP3$
```

```
Servidor iniciou...  
Cliente 544197 iniciou  
Cliente 544197 solicitou o código 4  
Cliente 544197 solicitou o código 1  
Cliente 544943 iniciou  
Cliente 544943 solicitou o código 2  
Cliente 544197 terminou
```

Figura 12 – Cliente termina e informa o servidor. O servidor retira cliente da lista de clientes

5. Servidor termina

```
Servidor iniciou...  
Cliente 549602 iniciou  
Cliente 549602 solicitou o código 1  
Cliente 549684 iniciou  
Cliente 549684 solicitou o código 2  
^CA informar cliente 549602  
A informar cliente 549684  
A encerrar a fila de mensagens...  
quental@viriato:~/TP3$ █
```

Figura 13 – Servidor termina e informa clientes

```
Insira os códigos dos produtos e prima ENTER:  
1  
Cliente 549602.  
Título: Sistemas Operativos  
Autor: Carlos Quental  
Preço: 20.00  
O servidor terminou. Cliente 549602 vai terminar.  
quental@viriato:~/TP3$  
A fila foi eliminada ...  
  
quental@viriato:~/TP3$ █
```

```
Insira os códigos dos produtos e prima ENTER:  
2  
Cliente 549684.  
Título: Linguagem C  
Autor: Dennis Ritchie  
Preço: 10.00  
O servidor terminou. Cliente 549684 vai terminar.  
quental@viriato:~/TP3$  
A fila foi eliminada ...  
  
quental@viriato:~/TP3$ █
```

Figura 14 – Clientes recebem informação do servidor e terminam

Bom trabalho!!