

**Instituto Politécnico de Viseu  
Escola Superior de Tecnologia e Gestão de Viseu  
Departamento de Informática**

Unidade Curricular: Sistemas Operativos

Trabalho prático 3

**Realizado por:**

António Ramos- 23906

Gonçalo Abreu- 22996

Leandro Dias- 23028

Viseu, 2023

Índice

[**Introdução** 3](#_Toc123893870)

[**Plano Geral** 3](#_Toc123893871)

[**Plano do código** 3](#_Toc123893872)

[Librarias usadas: 3](#_Toc123893873)

[Plano do file “cliente.c”: 4](#_Toc123893874)

[Plano do file “msg.h”: 4](#_Toc123893875)

[**Desenvolvimento do código** 5](#_Toc123893876)

[Função tratarSinal 5](#_Toc123893877)

[**Funcionalidades do Cliente** 5](#_Toc123893878)

[Connect 5](#_Toc123893879)

[Procura e Quero 6](#_Toc123893880)

[Shutdown 8](#_Toc123893881)

[**Funcionalidades do Servidor** 8](#_Toc123893882)

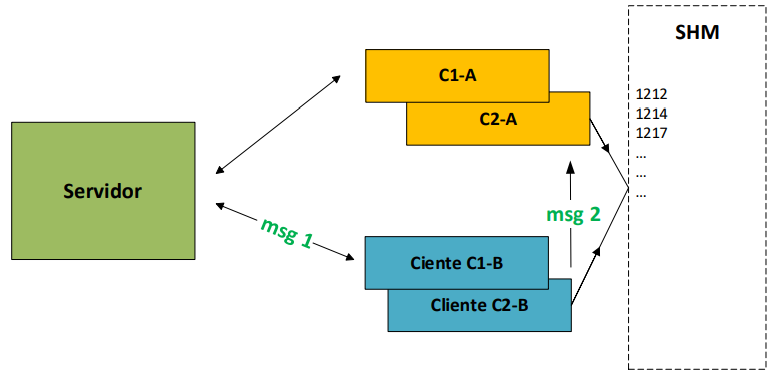
[**Demonstração do programa em execução** 10](#_Toc123893883)

[**Conclusão** 12](#_Toc123893884)

## **Introdução**

O presente relatório é relativo ao trabalho prático 3 da Unidade Curricular de Sistemas Operativos.   
Neste será desenvolvido um código em que se vai implementar um sistema P2P de partilha de ficheiros através de pipes, pipes nomeados, filas de mensagens, memória partilhada, sinais e semáforos.

## **Plano Geral**



***msg 1*** – Serve para a troca de mensagens entre clientes e cliente – servidor.

***msg 2*** – Serve para o envio de um ficheiro de C2 para C1.

***SHM*** – Lista os PID’s dos C2 disponíveis.

## **Plano do código**

### Librarias usadas:

1#include <errno.h>

2#include <sys/types.h>

3#include <sys/ipc.h>

4#include <sys/msg.h>

5#include <sys/shm.h>

6#include <signal.h>

7#include <string.h>

8#include <stdlib.h>

9#include <unistd.h>

10#include <stdio.h>

11#include <dirent.h>

### Plano do file “cliente.c”:

Neste file está inserido o tratamento de sinal, as funções relativas ao envio de mensagens tais como “void enviarMSG()“, leitura de ficheiros com a função “ void ficheiro()“. No main, ocorre a certificação que o SHM foi criado, depois ligam-se as filas de mensagens msg1 e mg2 e verificam-se o número de clientes atuais, isto é, se há <=10 clientes atuais. Posteriormente, conectam-se os processos pai e filho por meio de pipes, (pipe(pipeFd);(criação)).

### Plano do file “msg.h”:

Aqui estão presentes as estruturas (structs) das mensagens:

**struct** s\_msg {

**long** tipo;

**char** nome[**256**];

**char** conteudo[**1024**];

};

Organização SHM (memória partilhada):

**struct** s\_shm{

**int** pid[TAMANHO\_SHM];

**int** tamanho;

};

## **Desenvolvimento do código**

### Função tratarSinal

Esta função trata um sinal verificando se houve connect.

Se sim, este copia o PID do C1 e o PID do C2 em conteúdo.

De seguida envia uma mensagem de “shutdown” que de seguida acrescenta o PID do processo a ele associado através de “conteúdo”.

Uma imagem com texto

Descrição gerada automaticamente

## **Funcionalidades do Cliente**

### Connect

Para que esta função se ligue ao servidor através do envio de uma mensagem do tipo “1” usou-se msg.tipo= **1** e forneceu-se um *prompt*

Uma imagem com texto

Descrição gerada automaticamente

### Procura e Quero

O objetivo da função procura é pesquisar nos restantes clientes um ficheiro com o nome que se deu.

Através da função criada “enviarMSG”, enviou-se a mensagem complementando com o seu próprio PID.

Nesta função terá de se verificar se o utilizador fez *connect* através de **if**( strcmp(texto,"connect") == **0** ) que indica que sim. Nesse caso envia-se a mensagem para a fila e altera-se o valor de i para 1 já que ele fez *connect*.

Caso no tratamento do sinal tenha “shutdown” na fila, envia-se a mensagem “texto” para a fila e altera-se o valor de i para 0 para voltar a verificar o *connect*.

Senão, se o utilizador estiver conectado (i=1) e o argumento1 seja “procura” ou “quero”, ocorre o envio da mensagem para a fila.

Uma imagem com texto

Descrição gerada automaticamente

É necessário que se crie uma comunicação pipe do filho para o pai para o pai receber o PID do filho C2.

Tal é possível graças à seguinte função.

Uma imagem com texto

Descrição gerada automaticamente

Relacionado ao filho, é necessária uma ligação pipe para que este envie o PID para o pai. Tal é possível graças à seguinte função.

Uma imagem com texto

Descrição gerada automaticamente

A função abaixo foi criada para o tratamento do ficheiro. Esta servirá para ler o nome do ficheiro e o seu conteúdo.

Uma imagem com texto

Descrição gerada automaticamente

No processo filho C2 inicia-se com o envio do PID de si próprio para o processo pai.

De seguida chama-se a função ficheiro (apresentada acima) para ler o nome do ficheiro e o conteúdo.

Define-se o tipo da mensagem destino (neste caso é a 2).

De seguida usa-se a função msgsnd que insere uma mensagem na fila (a mensagem alocada em msg)

Uma imagem com texto

Descrição gerada automaticamente

### Shutdown

Esta funcionalidade deverá enviar uma mensagem ao servidor quando o utilizador se desliga

Uma imagem com texto, ecrã, captura de ecrã, fechar

Descrição gerada automaticamente

## **Funcionalidades do Servidor**

Inicialmente procedemos à realização de um código que permite ordenar o vetor se algum valor de PID fosse removido no meio

Uma imagem com texto

Descrição gerada automaticamente

O servidor deverá ler a fila de mensagens 1, o pedido de *connect* e *shutdown* e para isso, recorremos à função msgrcv.

Uma imagem com texto, captura de ecrã, ecrã

Descrição gerada automaticamente

A seguinte função avisa que o servidor pode ter no máximo 10 clientes.

Uma imagem com texto

Descrição gerada automaticamente

A seguinte função permite atualizar a SHM caso o utilizador tenha feito *connect*. Nesse caso, a função escreve uma mensagem (printf("Ligar C2 %d com o pid %d**\n**",ptr->tamanho+**1**,pidC2);).

Uma imagem com texto

Descrição gerada automaticamente

Caso o utilizador tenha enviado um *shutdown*, a função deverá remover o C2 da SHM.

De seguida, visto que o array SHM, após se remover o valor, poder-se-á encontrar com um espaço vazio no meio, procedemos à chamada da função ordenarVetor que permitirá colocar os valores todos para a esquerda para que se possa adicionar um novo cliente (como explicado acima).

Uma imagem com texto

Descrição gerada automaticamente

## **Demonstração do programa em execução**

Ligar o servidor:

Uma imagem com texto

Descrição gerada automaticamente

Conectar o primeiro cliente:

Uma imagem com texto

Descrição gerada automaticamente

No servidor atualiza o shm com o PID do C2:

Uma imagem com texto

Descrição gerada automaticamente

Conectar o segundo cliente na outra pasta:

Uma imagem com texto

Descrição gerada automaticamente

No servidor, atualiza o SHM com o PID do C2 do cliente 2:

Uma imagem com texto

Descrição gerada automaticamente

Fazer o shutdown no cliente:

Uma imagem com texto

Descrição gerada automaticamente

Tirar o cliente 2 do SHM:

Uma imagem com texto

Descrição gerada automaticamente

Tratamento de sinal CTRL+C. Não dá para fazer shutdown sem o cliente ter feito o connect primeiro

Uma imagem com texto

Descrição gerada automaticamente

No servidor:

Uma imagem com texto

Descrição gerada automaticamente

## **Conclusão**

Ao longo deste trabalho pudemos desenvolver melhores noções quanto à arquitetura P2P (peer to peer) e poder trabalhar num caso prático que é a melhor maneira de nos fazer sentir mais próximos à linguagem.