****

Aplicação de servidor

**Cliente**

Sistemas Operativos

Grupo 1 – Turma 3

Mestrado Integrado em Engenharia Informática e Computação

Angela Cruz up201806781

Eduardo

Filipe Pinto up201907747

Luísa Maria Mesquita up201704805

Pedro Moreira up201904642

**Índice**

[1. Introdução 2](#_Toc70687091)

[2. Estruturas de dados e variáveis 2](#_Toc70687092)

[2.1. Arguments 2](#_Toc70687093)

[2.2. Log 3](#_Toc70687094)

[2.3. ArgsThread 3](#_Toc70687095)

[2.4. Message 4](#_Toc70687096)

[2.5. Mutexes 4](#_Toc70687097)

[2.6. Outras variáveis globais 4](#_Toc70687098)

[3. Testes 5](#_Toc70687099)

[3.1. VALGRIND 5](#_Toc70687100)

[3.2. CPPLINT 5](#_Toc70687101)

[3.3. INFER 5](#_Toc70687102)

[4. Compilação e execução 7](#_Toc70687103)

[4.1. Compilação 7](#_Toc70687104)

[4.2. Execução 7](#_Toc70687105)

[5. Conclusão 7](#_Toc70687106)

# 

# 1. Introdução

O objetivo do trabalho consiste em desenvolver uma aplicação do tipo cliente-servidor, sendo que nesta primeira parte o foco foi na parte do cliente. Os seguintes pontos definem o desenvolvimento do trabalho:

* Processamento dos argumentos introduzidos na linha de comandos.
* Tratamento dos sinais SIGALRM e SIGPIPE
* Uso da função *alarm()* como medição do tempo de execução, sendo que um SIGALRM é gerado após decorrido o número de segundos introduzido na linha de comandos.
* Uso de várias funções essenciais para recolha de informações: *getpid()*, *pthread\_self()*, *time()*...
* Abertura e fecho do FIFO público para escrita (O\_WRONLY) e dos FIFO 's privados para leitura (O\_RDONLY). E posterior criação dos FIFO's privados, usando a função *mkfifo()* em modo 0666.
* Aplicação de *mutexes* como mecanismo de sincronização de forma a evitar conflitos entre threads concorrentes.
* Uso da função *rand\_r()* para gerar a carga da tarefa a pedir ao servidor.

# 2. Estruturas de dados e variáveis

## 2.1. Arguments

Esta estrutura guarda todas as informações introduzidas na linha de comandos e que são depois usadas ao longo do programa. A inicialização dos campos da estrutura é feita na função *ParseArguments()*.

/\*\*

\* @brief Struct with the arguments introduced in the command line.

\*/

struct Arguments

{

size\_t nsecs; //number of seconds

char \*public\_fifo; //public FIFO

};

## 2.2. Log

Esta estrutura guarda a informação a ser imprimida na consola. Sempre que ocorre uma determinada operação *oper*, atribui-se o respetivo valor a estes campos e chama-se a função *WriteLog()*, que imprime essa informação para o stdout.

/\*\*

\* @brief Struct with information to print to stdout

\*/

struct Log

{

time\_t inst; //return value of time() function

int i; //unique request number

int t; //task load

pid\_t pid; //process ID

pthread\_t tid; //thread ID

int res; //task result

char \*oper; //operation made

};

## 

## 2.3. ArgsThread

Esta estrutura guarda o PID do processo e o descritor de ficheiros do FIFO público para que estes possam ser usados na função *ThreadHandler()*, ou seja, em cada thread criado.

/\*\*

\* @brief Struct with arguments to use in function ThreadHandler().

\*/

struct ArgsThread

{

pid\_t pid; //process id of the program

int fd\_public\_fifo; //file descriptor of the public FIFO

};

## 

## 2.4. Message

Estrutura solicitada no enunciado que guarda as informações necessárias para mandar ao servidor.

/\*\*

\* @brief Struct for exchange of messages between client and server

\*/

struct Message

{

int rid; // request id

pid\_t pid; // process id

pthread\_t tid; // thread id

int tskload; // task load

int tskres; // task result

};

## 2.5. Mutexes

Duas variáveis que permitem guardar o estado do mutex (aberto ou fechado).

pthread\_mutex\_t lock1;

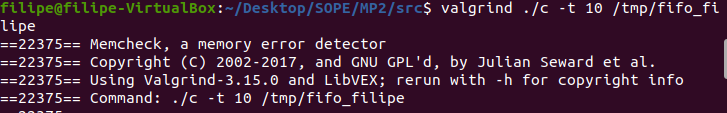
pthread\_mutex\_t lock2;

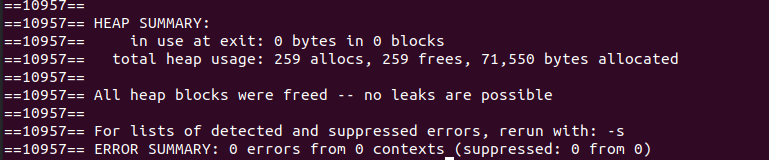
## 2.6. Outras variáveis globais

* **cont:** contador que permite guardar o número de threads criados pela thread principal. Sempre que um thread novo é criado, esta variável é incrementada.
* **errno:** guarda o número do último erro ocorrido.
* **finish:** variável sempre a falso até o *handler* do sinal SIGALRM colocá-la a verdadeiro, altura em que a thread principal parará de criar novas threads.

# 3. Testes

## 3.1. VALGRIND

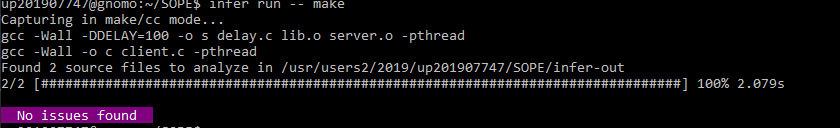




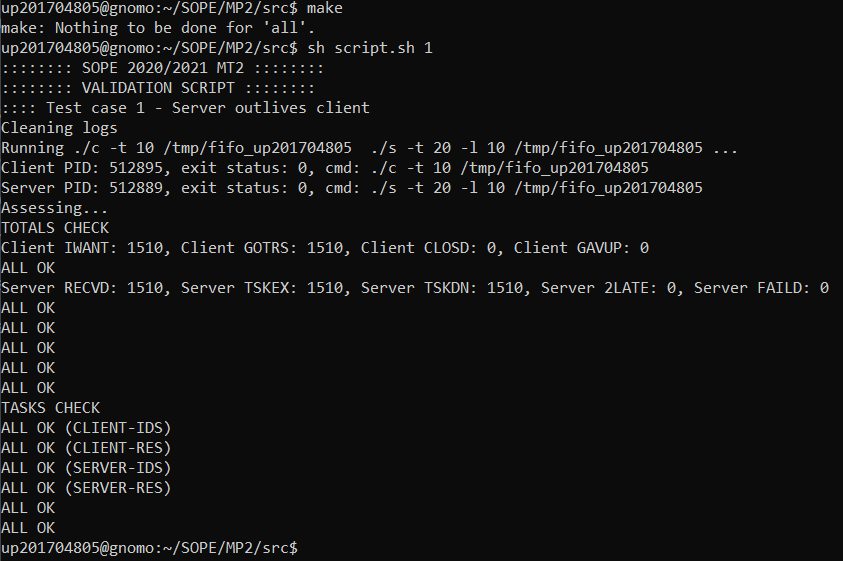
## 3.2. CPPLINT

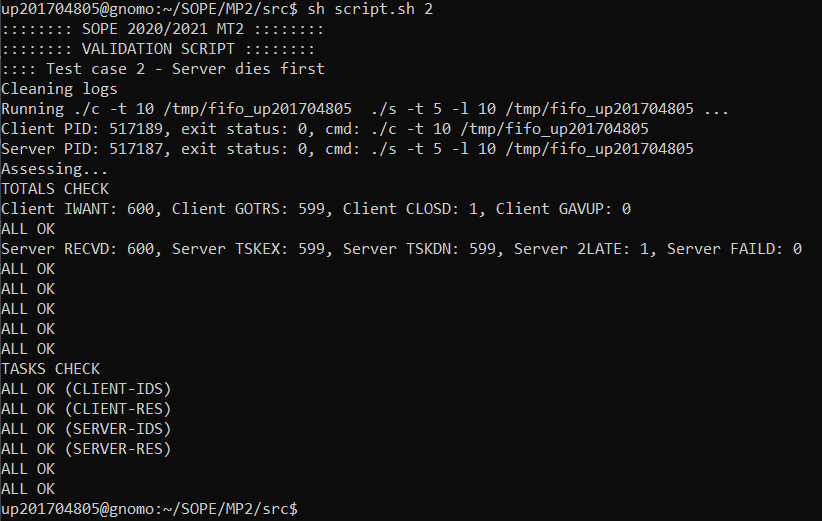


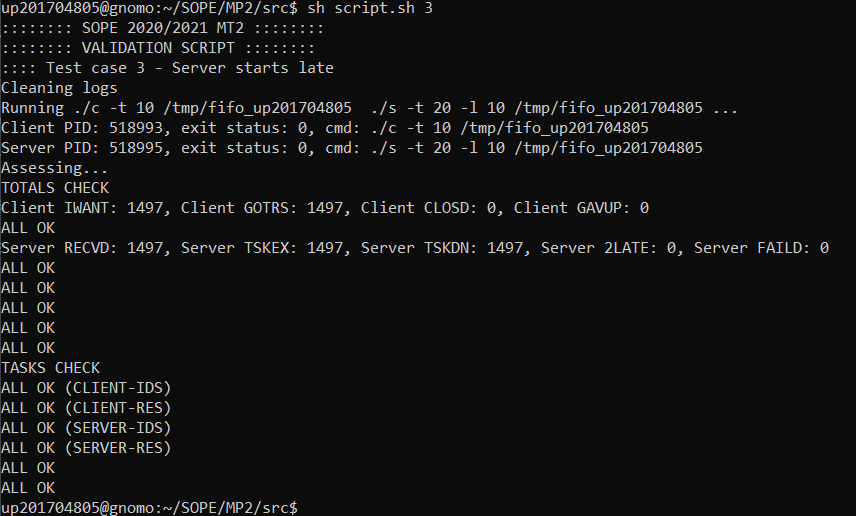
## 3.3. INFER



3.4. SCRIPT

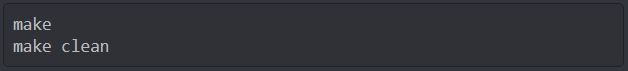






# 4. Compilação e execução

## 4.1. Compilação



ou





## 4.2. Execução





# 5. Conclusão

As seguintes percentagens correspondem à dedicação e consistência ao longo do tempo por parte de cada um a esta parte do trabalho.

|  |  |
| --- | --- |
| Angela Cruz | 100% |
| Eduardo | 0% |
| Filipe Pinto | 90% |
| Luísa Maria Mesquita | 80% |
| Pedro Moreira | 5% |