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

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

```
filipe@filipe-VirtualBox:~/Desktop/SOPE/MP2/src$ valgrind ./c -t 10 /tmp/fifo_fi
lipe
==22375== Memcheck, a memory error detector
==22375== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==22375== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==22375== Command: ./c -t 10 /tmp/fifo_filipe
==10957==
==10957== HEAP SUMMARY:
==10957==     in use at exit: 0 bytes in 0 blocks
==10957==   total heap usage: 259 allocs, 259 frees, 71,550 bytes allocated
==10957==
==10957== All heap blocks were freed -- no leaks are possible
==10957==
==10957== For lists of detected and suppressed errors, rerun with: -s
==10957== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

3.2. CPPLINT

```
up201704805@gnomo:~/SOPE/MP2/src$ cpplint --filter=-whitespace,-legal/copyright,-readability/check --recursive client.c
Done processing client.c
up201704805@gnomo:~/SOPE/MP2/src$
```

3.3. INFER

```
up201907747@gnomo:~/SOPE$ infer run -- make
Capturing in make/cc mode...
gcc -Wall -DDELAY=100 -o s delay.c lib.o server.o -pthread
gcc -Wall -o c client.c -pthread
Found 2 source files to analyze in /usr/users2/2019/up201907747/SOPE/infer-out
2/2 [################################################################################] 100% 2.079s

  No issues found
```

3.4. SCRIPT

```
up201704805@gnomo:~/SOPE/MP2/src$ make
make: Nothing to be done for 'all'.
up201704805@gnomo:~/SOPE/MP2/src$ sh script.sh 1
:::::::: SOPE 2020/2021 MT2 ::::::::
:::::::: VALIDATION SCRIPT ::::::::
:::: Test case 1 - Server outlives client
Cleaning logs
Running ./c -t 10 /tmp/fifo_up201704805  ./s -t 20 -l 10 /tmp/fifo_up201704805 ...
Client PID: 512895, exit status: 0, cmd: ./c -t 10 /tmp/fifo_up201704805
Server PID: 512889, exit status: 0, cmd: ./s -t 20 -l 10 /tmp/fifo_up201704805
Assessing...
TOTALS CHECK
Client IWANT: 1510, Client GOTRS: 1510, Client CLOSD: 0, Client GAVUP: 0
ALL OK
Server RECVD: 1510, Server TSKEX: 1510, Server TSKDN: 1510, Server 2LATE: 0, Server FAILD: 0
ALL OK
ALL OK
ALL OK
ALL OK
ALL OK
TASKS CHECK
ALL OK (CLIENT-IDS)
ALL OK (CLIENT-RES)
ALL OK (SERVER-IDS)
ALL OK (SERVER-RES)
ALL OK
ALL OK
up201704805@gnomo:~/SOPE/MP2/src$
```

```
up201704805@gnomo:~/SOPE/MP2/src$ sh script.sh 2
:::::::: SOPE 2020/2021 MT2 ::::::::
:::::::: VALIDATION SCRIPT ::::::::
:::: Test case 2 - Server dies first
Cleaning logs
Running ./c -t 10 /tmp/fifo_up201704805  ./s -t 5 -l 10 /tmp/fifo_up201704805 ...
Client PID: 517189, exit status: 0, cmd: ./c -t 10 /tmp/fifo_up201704805
Server PID: 517187, exit status: 0, cmd: ./s -t 5 -l 10 /tmp/fifo_up201704805
Assessing...
TOTALS CHECK
Client IWANT: 600, Client GOTRS: 599, Client CLOSD: 1, Client GAVUP: 0
ALL OK
Server RECVD: 600, Server TSKEX: 599, Server TSKDN: 599, Server 2LATE: 1, Server FAILD: 0
ALL OK
ALL OK
ALL OK
ALL OK
ALL OK
TASKS CHECK
ALL OK (CLIENT-IDS)
ALL OK (CLIENT-RES)
ALL OK (SERVER-IDS)
ALL OK (SERVER-RES)
ALL OK
ALL OK
up201704805@gnomo:~/SOPE/MP2/src$
```

```
up201704805@gnomo:~/SOPE/MP2/src$ sh script.sh 3
:::::::: SOPE 2020/2021 MT2 ::::::::
:::::::: VALIDATION SCRIPT ::::::::
:::: Test case 3 - Server starts late
Cleaning logs
Running ./c -t 10 /tmp/fifo_up201704805  ./s -t 20 -l 10 /tmp/fifo_up201704805 ...
Client PID: 518993, exit status: 0, cmd: ./c -t 10 /tmp/fifo_up201704805
Server PID: 518995, exit status: 0, cmd: ./s -t 20 -l 10 /tmp/fifo_up201704805
Assessing...
TOTALS CHECK
Client IWANT: 1497, Client GOTRS: 1497, Client CLOSD: 0, Client GAVUP: 0
ALL OK
Server RECVD: 1497, Server TSKEX: 1497, Server TSKDN: 1497, Server 2LATE: 0, Server FAILD: 0
ALL OK
ALL OK
ALL OK
ALL OK
ALL OK
TASKS CHECK
ALL OK (CLIENT-IDS)
ALL OK (CLIENT-RES)
ALL OK (SERVER-IDS)
ALL OK (SERVER-RES)
ALL OK
ALL OK
up201704805@gnomo:~/SOPE/MP2/src$
```

4. Compilação e execução

4.1. Compilação

```
make
make clean
```

ou

```
gcc -Wall -DDELAY=100 -o s delay.c lib.o server.o -pthread
```

```
gcc -Wall -o c client.c -pthread
```

4.2. Execução

```
./c <-t nsecs> fifoname
```

```
./s <-t nsecs> [-l bufsz] fifoname
```

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%