Aplicação de servidor

Servidor

Sistemas Operativos

Grupo 1 – Turma 3

Mestrado Integrado em Engenharia Informática e Computação

Angela Cruz	up201806781
Eduardo	
Filipe Pinto	up201907747
Luísa Mesquita	up201704805
Pedro Moreira	up201904642

Índice

1. Introdução

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

- Processamento dos argumentos introduzidos na linha de comandos.
- Tratamento do sinal SIGALRM.
- Abertura e fecho do FIFO público para leitura (O_RDONLY) e em modo *nonblocking* (O_NONBLOCK).
- Uso da função *task()* fornecida para obter a carga da tarefa pedida.
- Construção e uso de várias estruturas de dados (ver próxima seção)
- Escrita no *stdout* as mensagens: TSKEX, RECVD, FAILD, TSKDN
- Criação de um armazém (*cloud*) onde serão guardados os pedidos ao servidor.
- Criação de uma fila (*queue*) onde se guarda o primeiro e último pedido do armazém.
- Criação de várias funções auxiliares à *queue* que permitem colocar lá elementos, retirar elementos, verificar se está cheia ou se está vazia e verificar qual o elemento no topo da fila.
- Criação e tratamento das threads do tipo Consumidor e Produtor (é criado uma *thread* do tipo Consumidor e várias do tipo Produtor).
- 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.
- Aplicação de *mutexes* como mecanismo de sincronização de forma a evitar conflitos entre *threads* concorrentes.
- Criação dos FIFOS privados nas *threads* consumidoras.

2. Estruturas de dados e variáveis

As estruturas Message e Log são usadas tanto na parte do cliente como na parte do servidor, logo são guardadas num ficheiro em comum (common.h).

Foi criada uma estrutura de dados - Queue - como auxiliar ao trabalho, logo esta está colocada também num ficheiro à parte (queue.h).

As restantes estruturas e variáveis foram colocadas no ficheiro relacionado com o servidor (server.h).

2.1. 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.

```
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
};
```

2.2. Message

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

```
struct Message
{
   int rid;       // request id
   pid_t pid;     // process id
   pthread_t tid;  // thread id
   int tskload;    // task load
   int tskres;     // task result
};
```

2.3. Queue

Esta estrutura guarda o primeiro elemento e último elemento a guardar na fila.

```
struct Queue
{
   int first; //index of the first element of the queue (-1 if queue is empty)
   int last;  //index of the last element of the queue (-1 if queue is empty)
};
```

2.4. 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()*.

```
struct Arguments
{
   size_t nsecs;     //number of seconds
   int buffer_size;    //buffer size
   char public_fifo[100]; //public FIFO
};
```

2.5. ArgsThreadsProducer

Esta estrutura guarda variáveis a ser usadas na função *ThreadHandlerProd()*, ou seja, na *thread* produtora.

```
struct ArgsThreadsProducer
{   int rid;       // request id
   pid_t pid;     // process id
   pthread_t tid;  // thread id
   int tskload;    // task load
   int tskres;     // task result
   struct Message *cloud ;
   int nmax; //maximum number of elements in cloud
};
```

2.6. ArgsThreadsConsumer

Esta estrutura guarda variáveis a ser usadas na função *ThreadHandlerCons()*, ou seja, nas *threads* consumidoras.

```
struct ArgsThreadsConsumer
{
   struct Message *cloud;
   int nmax;
}
```

2.7. Variáveis globais

- **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 (*main*) parará de criar novas threads produtoras.
- **finishCons:** variável sempre a falso até a *thread* principal (*main*) parar de criar *threads* produtoras.

3. Testes

3.1. VALGRIND

```
luisa@luisa-VirtualBox:/media/sf_SOPE/MP2-2$ valgrind ./s -t 10 -l 100 /tmp/fifo
==3591== Memcheck, a memory error detector
==3591== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3591== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3591== Command: ./s -t 10 -l 100 /tmp/fifo
==3591==
==3591==
==3591== HEAP SUMMARY:
==3591==     in use at exit: 0 bytes in 0 blocks
==3591==   total heap usage: 7 allocs, 7 frees, 4,382 bytes allocated
==3591==
==3591== All heap blocks were freed -- no leaks are possible
==3591==
==3591== For lists of detected and suppressed errors, rerun with: -s
==3591== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
luisa@luisa-VirtualBox:/media/sf_SOPE/MP2-2$
```

3.2. CPPLINT

```
up201907747@gnomo:~/MP2-2$ cpplint --filter=-whitespace,-legal/copyright,-readability/check --recursive queue.c
Done processing queue.c
up201907747@gnomo:~/MP2-2$ cpplint --filter=-whitespace,-legal/copyright,-readability/check --recursive server.c
Done processing server.c
up201907747@gnomo:~/MP2-2$ cpplint --filter=-whitespace,-legal/copyright,-readability/check --recursive common.c
Done processing common.c
up201907747@gnomo:~/MP2-2$ cpplint --filter=-whitespace,-legal/copyright,-readability/check --recursive lib.c
Done processing lib.c
up201907747@gnomo:~/MP2-2$ cpplint --filter=-whitespace,-legal/copyright,-readability/check --recursive delay.c
Done processing delay.c
up201907747@gnomo:~/MP2-2$
```

3.3. INFER

```
up201704805@gnomo:~/SOPE/MP2-2$ infer run -- make
Capturing in make/cc mode...
make: Nothing to be done for 'all'.
Nothing to compile. Try running `make clean` first.
There was nothing to analyze.

  No issues found
up201704805@gnomo:~/SOPE/MP2-2$
```

3.4. SCRIPT

```
up201704805@gnomo:~/SOPE/MP2-2$ 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: 12986, exit status: 0, cmd: ./c -t 10 /tmp/fifo_up201704805
Server PID: 12950, exit status: 0, cmd: ./s -t 20 -l 10 /tmp/fifo_up201704805
Assessing...
TOTALS CHECK
Client IWANT: 667, Client GOTRS: 664, Client CLOSD: 0, Client GAVUP: 3
ALL OK
Server RECVD: 667, Server TSKEX: 667, Server TSKDN: 664, Server 2LATE: 0, Server FAILD: 3
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/MP2-2$
```

```
up201704805@gnomo:~/SOPE/MP2-2$ 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: 16127, exit status: 0, cmd: ./c -t 10 /tmp/fifo_up201704805
Server PID: 16124, exit status: 0, cmd: ./s -t 5 -l 10 /tmp/fifo_up201704805
Assessing...
TOTALS CHECK
Client IWANT: 264, Client GOTRS: 264, Client CLOSD: 0, Client GAVUP: 0
ALL OK
Server RECVD: 264, Server TSKEX: 264, Server TSKDN: 264, 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/MP2-2$
```

```
up201704805@gnomo:~/SOPE/MP2-2$ 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: 17448, exit status: 0, cmd: ./c -t 10 /tmp/fifo_up201704805
Server PID: 17450, exit status: 0, cmd: ./s -t 20 -l 10 /tmp/fifo_up201704805
Assessing...
TOTALS CHECK
Client IWANT: 595, Client GOTRS: 593, Client CLOSD: 0, Client GAVUP: 2
ALL OK
Server RECVD: 595, Server TSKEX: 595, Server TSKDN: 593, Server 2LATE: 0, Server FAILD: 2
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/MP2-2$
```

4. Compilação e execução

4.1. Compilação

```
make
make clean
```

ou

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

```
gcc -Wall -o s server.c queue.c lib.c delay.c common.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	50%
Eduardo	0%
Filipe Pinto	25%
Luísa Maria Mesquita	25%
Pedro Moreira	0%