FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
DEPARTAMENTO DE INFORMÁTICA

Interação Cliente/Servidor
(Lab-08 – Dezembro 2019)

1. Objetivo

O objetivo deste trabalho é estudar a *interação cliente/servidor* em que o mecanismo de comunicação utilizado é a comunicação por *sockets datagram* (*UDP*). Em particular, um cliente envia um pedido a um servidor para lançar um processo e receber o output desse processo, e em que o cliente usa sockets datagram para o envio do pedido e receção da resposta.

2. Comunicação por sockets

Para realização deste trabalho recomenda-se a consulta dos slides da aula teórica bem como o capítulo 48 do livro da cadeira [1]. Estude o código da biblioteca em *udp_comm.c* que permite o uso de sockets UDP e que é constituída pelas funções seguintes:

- UDP_Open
- UDP_FillSockAddr
- UDP_Write
- UDP_Read

Considere o código disponibilizado online com os esqueletos de um servidor em *server.c* e de um cliente em *client.c*. Repare que estes dois programas utilizam as funções definidas em *udp_comm.c*. Para esse efeito incluem o ficheiro de definições *udp_comm.h*. Para obter o código do servidor e do cliente os comandos a invocar são os seguintes:

```
gcc –Wall –c udp_comm.c
gcc –Wall –o server server.c udp_comm.o
gcc –Wall –o client client.c udp_comm.o
```

Considere os esqueletos de um servidor em *server.c* e de um cliente em *client.c*. Repare que estes dois programas utilizam as funções definidas em cliente *udp_comm.c*. Para esse efeito incluem o ficheiro de definições *udp_comm.h*.

Escreva o código do cliente e do servidor cujas ações podem ser descritas como apresentado no texto que se segue.

Cliente

Um cliente é invocado com a linha de comando

```
./client host_name port_number nome_programa argumento
```

e executa as ações seguintes

1. Cria um socket UDP associado à porta *port_number + 1*
2. Forma uma mensagem com as 3a e 4as palavras da linha de comando que tem o formato

   ```
   nome_programa argumento
   ```

3. Envia a mensagem construída em 2 para a máquina *host_name*, porta *port_number*
4. Aguarda a resposta do servidor e escreve os bytes recebidos no standard output

Servidor

Um servidor é invocado com a linha de comando

```
./ server port_number
```

e começa por criar um socket UDP associado à porta *port_number*; após isto executa eternamente as acções que se seguem

1. Aguarda a chegada de uma mensagem com o formato *nome_programa argumento*
2. Cria um pipe
3. Faz um *fork( )*
4. Redirige o stdout do filho para o pipe
5. O filho faz *execvp* tentando usar *nome_programa* como nome do ficheiro executável e como primeiro argumento da linha de comando. A segunda e última palavra da linha de comando será *argumento*. Se o *execvp* falhar, o filho deve escrever no *stdout* "Programa não encontrado" e terminar
6. O pai lê os bytes que o filho escreve no pipe e envia-os de volta para o cliente
7. O pai espera que o filho termine e fecha os canais associados ao pipe

Qualquer programa invocado pelo cliente e lançado pelo servidor, deve usar *argv[1]* para calcular um valor e escrever esse valor no *stdout*. Ver o exemplo seguinte *strlen.c*.

```
#include <stdio.h>
int main( int argc, char *argv[ ]){
        // calcular um inteiro v a partir de argv[1]
        printf("resultado = %d\n", strlen(argv[1]));
        return 0;
}
```

Use os esqueletos dados para escrever os códigos do cliente e do servidor. Teste o seu programa com o programa *strlen.c* e outros exemplos mais úteis.

Para executar o servidor faça numa janela

```
./ server port_number
```

em que port_number é o valor da porta combinada entre o servidor e o cliente. Numa outra janela faça

```
./client host_name port_number nome_programa argumento
```

em que *hostname* é o nome da máquina em que reside o servidor (se for a mesma máquina onde reside o cliente será *localhost*).

3. Bibliografia

[1] Capítulo 48 do livro recomendado, *Operating Systems: Three Easy Pieces*, R. Arpaci-Dusseau, R. Arpaci-Dusseau, 2018 http://pages.cs.wisc.edu/~remzi/OSTEP/

[2] Páginas do manual sobre sockets UDP