Lauro Cruz e Souza - 156175

Pedro Emílio Machado de Brito - 137264

Atividade 2.2

server.c

O servidor foi implementado de forma a se comportar exatamente como o da atividade 2.1, isto é, esperar em loop por conexões, e também em um loop, ler uma linha de texto, imprimi-la, mandar eco, e tentar ler mais texto. A diferença, nesse caso, é puramente de implementação.

Uma limitação do servidor da atividade 2.2 é que ele suporta um número máximo de conexões simultâneas, devido ao limite imposto por `FD_SETSIZE`. Esse limite é impresso no início da execução.

Todas as funções que tratam da conexão com a rede tem seus códigos de erro verificados e tratados, se necessário, assim como no trabalho 1 e atividade 2.1

client.c

O cliente utilizado é exatamente o mesmo da atividade 2.1.

Funções utilizadas nos programas:

Além das funções utilizadas na atividade 2.1, que foram todas mantidas (com exceção de `fork`), utilizamos também:

`select`: Bloqueia até que algum socket de um determinado conjunto esteja pronto, podendo ser para leitura, escrita, ou alguma condição de exceção.

Também foram utilizadas as seguintes macros.

`FD_ZERO`: Limpa uma estrutura de dados de conjunto de sockets. `FD_SET`: Adiciona um socket a um conjunto. `FD_ISSET`: Checa se um socket está presente em um conjunto.

Um exemplo de sessão (idêntico ao da atividade 2.1):

Em um terminal na máquina local, rodamos:

```
$ ./server
```

Em outro na mesma máquina rodamos

`$ ./client localhost`

estabelecendo uma conexão. A porta usada é implicitamente 12345, pelas constantes definidas em cada arquivo fonte.

Temos então os primeiros outputs com as infos:

```
$ ./client localhost
Local IP: 127.0.0.1
    Port: 46704
```

```
$ ./server
1023 concurrent connections supported
CLIENT CONNECTED
  IP: 127.0.0.1
PORT: 46704
```

Várias linhas de texto são digitadas no console com o cliente aberto:

```
$ ./client localhost
oi
oi
teste
teste
batata
batata
the quick brown fox jumps over the lazy dog
the quick brown fox jumps over the lazy dog
```

```
$ ./server
1023 concurrent connections supported
CLIENT CONNECTED
  IP: 127.0.0.1
PORT: 34552

From 127.0.0.1:
oi

From 127.0.0.1:
teste

From 127.0.0.1:
batata

From 127.0.0.1:
the quick brown fox jumps over the lazy dog
```

Nesse ponto conectamos mais um cliente no servidor, rodando `./client <hostname>` na máquina `xaveco` do IC (por SSH). Obtemos assim:

```
$ client

Local IP: 143.106.16.163
    Port: 35344
oi2
oi2
teste2
teste2
batata2
batata2
the quick brown fox jumps over the lazy dog 2
the quick brown fox jumps over the lazy dog 2
```

```
$ server

1023 concurrent connections supported
CLIENT CONNECTED
  IP: 127.0.0.1
PORT: 34552

From 127.0.0.1:
oi

From 127.0.0.1:
teste

From 127.0.0.1:
batata

From 127.0.0.1:
the quick brown fox jumps over the lazy dog

CLIENT CONNECTED
  IP: 143.106.16.163
PORT: 35344

From 143.106.16.163:
oi2

From 143.106.16.163:
teste2

From 143.106.16.163:
batata2

From 143.106.16.163:
the quick brown fox jumps over the lazy dog 2
```

Mesmo um cliente tendo terminado, o servidor continua ativo, esperando, outras conexões.