Nome:_____

Nº ESTG: _____

Aval. Periódica - EI/Prog. Avançada

Teste Prático 2 – 2018.12.19 /C

Licenciatura em
Engenharia Informática
UC de Programação Avançada
2º ano – Engenharia Informática
Regime diurno
Ano letivo 2018/2019 - 1ºSemestre

2º Teste Prático – Enunciado C

2018.12.19 / 15h00 Prova com consulta **Duração:** 100 minutos

Nome Completo:_____

N.º de Estudante:_____ **Regime:** [ ] Diurno [ ] Pós-laboral

IMPORTANTE

É expressamente proibido o recurso à Internet durante a prova. Qualquer utilização não autorizada da Internet leva à anulação da prova e ao reportar da situação às autoridades competentes. O mesmo sucede com outros tipos de tentativa de fraude.

- **Antes de iniciar a prova:**
  - Execute os seguintes comandos:

    `cd; mkdir -p ~/Prova02/R_NUMERO/`

    (em que **R** deve ser substituído pela letra **D** se for do regime diurno e **N** se for aluno do regime pós-laboral e **NUMERO** deve ser substituído pelo seu número ESTG);
  - Para garantir que o seu diretório de trabalho seja o correto, faça:

    `cd ~/Prova02/R_NUMERO/`
- **Após ter terminado a prova:**
  - Deverá proceder à criação de um arquivo TAR, fazendo uso do seguinte comando:

    `cd ~/Prova02/R_NUMERO/; tar cvf Prova02_YYYYMMDD_R_NUMERO.tar *`

    (em que YYYYMMDD corresponde à data corrente, e.g., 20181219, e R_NUMERO obedece ao formato acima indicado);
  - Verifique que o arquivo ".tar" que criou não está vazio, através da execução de:

    `tar tvf Prova02_YYYYMMDD_R_NUMERO.tar`
  - Entregue o arquivo ".tar" através da plataforma moodle, no espaço reservado para o efeito. Em caso de dúvidas, pergunte ao professor;
  - Informe o professor para este validar a receção dos seus ficheiros.

Pergunta [20 valores]

(Escreva as suas respostas a esta pergunta no diretório **`"~/Prova02/R_NUMERO/Pergunta"`**. Deve indicar o seu nome completo e número de estudante IPLeiria no ficheiro **README.txt** a ser criado no diretório)

NOTA 1: não é permitida a chamada a comandos externos através da função *system* ou de outra com funcionalidade similar.

NOTA 2: a solução deve ser implementada com recurso aos ficheiros do diretório **EmptyProject-client-server-template.v2.02.zip**

NOTA 3: código entregue que **não compile** através do utilitário *make* e do respetivo *makefile* leva à atribuição da classificação de **0 (zero) valores** à resposta.

Pretende-se que elabore a aplicação cliente/servidor TCP concorrente `cipher` cujo propósito é o de cifrar/decifrar uma mensagem utilizando o operador XOR. Para o efeito, a aplicação cliente **cipher_clnt** recebe, através da opção de linha de comandos `--message <message>`, uma mensagem a cifrar/decifrar e opcionalmente o parâmetro `--key <key>`. O parâmetro `--key <key>` é um inteiro de 8 bits (uint8_t) que é empregue como chave na operação de cifragem.

O comportamento do serviço `cipher` é o seguinte: **i)** Caso o parâmetro `--key` **não** seja fornecido, o cliente deve solicitar ao servidor que cifre a mensagem (pedido CIFRAR). O servidor responderá enviando duas mensagens TCP: uma que contém uma chave criada aleatoriamente, seguido de outra que contém a mensagem cifrada; **ii)** Caso o parâmetro `--key` **seja fornecido**, o cliente deve solicitar ao servidor a decifragem da mensagem (pedido DECIFRAR). Para o efeito, o cliente envia ao servidor a chave indicada na linha de comando. O servidor irá responder com a mensagem decifrada.

A aplicação **`cipher_svr`** desempenha o papel de servidor. Esta aplicação aguarda por pedidos de serviço, sendo que quando recebe um pedido deve proceder consoante o tipo de pedido (**sugestão:** uint8_t).

Se o pedido enviado pelo cliente for do tipo **CIFRAR**, o servidor deve:

- Gerar um número aleatório entre 1 e 127 (**uint8_t**);
- Efetuar a receção da mensagem a cifrar (assuma um máximo de 1024 caracteres);
- Substituir cada caracter da mensagem pela operação: **`mensagem[i] XOR chave;`**
- Caso o resultado da operação de cifragem de um caracter seja inferior ao valor **32**, deverá ser mantido o caracter original[1];
- Após efetuar a cifragem, o servidor deve enviar ao cliente, em duas mensagens separadas, a chave, seguida da mensagem cifrada, fechando de seguida a ligação.

Se o pedido for de o tipo **DECIFRAR**, o servidor deve:

- Efetuar a receção da chave;
- Efetuar a receção da mensagem (assuma um máximo de 1024 caracteres);
- Reverter o algoritmo de cifragem, substituindo cada caracter da mensagem com valor >= **32** pela operação XOR com a chave;
- Depois de efetuar a conversão, o servidor deve enviar ao cliente a mensagem decifrada e terminar a ligação.

Tenha em atenção que **`cipher_svr`** é um servidor concorrente, pelo que deve ser capaz de atender vários clientes simultaneamente.

Os parâmetros da linha de comando da aplicação cliente **`cipher_clnt`** são:
`--ip/-i <IPv4_address>`: endereço IPv4 do servidor. Caso não seja especificado, deve ser assumido o endereço "127.0.0.1".
`--port/-p <int>`: porto remoto (TCP) do servidor. **Este parâmetro é obrigatório**.
`--message/-m <message>`: mensagem a cifrar/decifrar. **Este parâmetro é obrigatório**.
`--key/-k <key>`: chave a utilizar na operação de decifrar. **Parâmetro não obrigatório**, sendo que, caso não seja especificado, a operação a realizar é CIFRAR.

Por sua vez, o servidor **`cipher_svr`** recorre ao seguinte parâmetro da linha de comando:
`--port/-p <int>`: porto de escuta TCP da aplicação. **Este parâmetro é obrigatório**.

Ambas as aplicações devem validar os parâmetros, nomeadamente o porto (valor entre 1 e 65535), e o endereço IP (aplicação cliente). No caso de algum parâmetro estar incorreto, deve ser escrito no canal de erro padrão uma apropriada mensagem de erro e, seguidamente, terminar a aplicação.

Considere os seguintes exemplos de execução:

Exemplo 1 (CIFRAR)

```
./cipher_clnt --ip 127.0.0.1 --port 1234 -m 'Hello World'
>> KEY: 3
>> MESSAGE: Kfool#Tlqog
```

Exemplo 2 (DECIFRAR)

```
./cipher_clnt --ip 127.0.0.1 --port 1234 -m 'Kfool#Tlqog' -k 3
>> MESSAGE: Hello World
```

[1] Um valor inferior a 32 corresponde a um caracter dito não "*printável*".