Licenciatura em
Engenharia Informática
UC de Programação Avançada
2º ano – Engenharia Informática
Regime diurno/pós-laboral
Ano letivo 2024/2025 - 1ºSemestre

Teste Prático #2 – Enunciado C

2024.12.21 / 09h30 Prova com consulta **Duração:** 90 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.

- **Preparação da máquina virtual:**
  1) Crie, no ambiente de trabalho, a pasta **EI_PA_Prova2**
  2) Seguidamente, copie a máquina virtual da UC para a pasta EI_PA_Prova2 do ambiente de trabalho.
    a) Caso tenha a máquina virtual da UC numa PEN pode copiá-la para o ambiente de trabalho.
    b) Caso contrário, o ficheiro 7z da máquina virtual encontra-se na pasta C:\VM, pelo que pode copiar o ficheiro 7Z (não mover!) para a pasta EI_PA_Prova2 do ambiente de trabalho e descompactá-lo.
- **Antes de iniciar a prova (dentro da máquina virtual):**
  - Execute os seguintes comandos:

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

    ```
    cd ~/ProvaP/R_NUMERO/;  tar cvf ProvaP_YYYYMMDD_R_NUMERO.tar *
    ```

    (em que YYYYMMDD corresponde à data corrente, e.g., 20241221, e R_NUMERO obedece ao formato acima indicado);
  - Verifique que o arquivo ".tar" que criou não está vazio, através da execução de:

    ```
    tar tvf ProvaP_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 **"~/ProvaP/R_NUMERO/"**)
NOTA 1: Apenas é permitido a chamada a comandos externos através da função *system* ou de outra com funcionalidade similar para a implementação da funcionalidade de criação do ficheiro JPG
NOTA 2: A solução deve ser implementada com recurso aos ficheiros do diretório **EmptyProject-client-server-template_v2.5.zip**
NOTA 3: Não é permitido o uso de *Variable Length Arrays* (VLA)
NOTA 4: A gestão dos parâmetros da linha de comandos deve ser feita através do *gengetopt*
NOTA 5: Código entregue que **não compile** através do utilitário *make* e do respetivo *makefile* na máquina virtual da UC leva à atribuição da classificação de **0 (zero) valores** à resposta

(continua na página seguinte)

(não se esqueça de ler convenientemente as instruções da página anterior)

Recorrendo à linguagem C, pretende-se que implemente o serviço cliente/servidor UDP **rgb2hex** cujo propósito é a conversão de uma *string* com três octetos RGB -- um octeto para **Red**, um octeto para **Green** e um octeto para **Blue** para a representação de *hexadecimal color code,* também expressa em formato de string. Por exemplo, a string RGB "**11,22,33**" que corresponde a especificar uma cor com as componentes `red=11`, `green=22` e `blue=33` é representada pelo seguinte *hexadecimal color code* "**#0b1621**". Esse código pode ser empregue para a criação de um ficheiro JPEG com um bloco de cor através da seguinte linha de comandos:

```
convert -size 100x100 xc:#0b1621 file.jpg
```

No serviço **rgb2hex**, a *string* RGB é fornecida ao cliente através de um parâmetro da linha de comando, sendo a mesma enviada ao servidor através do protocolo de transporte UDP. O servidor processa a *string*, criando a representação *hexadecimal color code* em *string*, enviando essa nova *string* como resposta ao cliente. Finalmente, o cliente faz uso do utilitário convert para criar um ficheiro JPEG, cujo nome pode ser indicado pela linha de comando, e que representa um quadrado de 256x256 com a cor indicada.

O servidor UDP – **s_rgb2hex** – deve i) aguardar por um pedido de um cliente, e quando recebe um pedido, deve ii) proceder à criação da *string hexadecimal color code* e iii) enviá-la para o cliente. Deve ainda mostrar na saída padrão a *string* original (recebida do cliente) e a *string* criada.

O servidor UDP deve validar a string RGB recebida, verificando que tem o formato NUM1,NUM2,NUM3, e que cada NUM se encontra no intervalo inteiro [0,255]. Caso a string RGB seja inválida, o servidor deve enviar ao cliente a *string* "ERROR in RGB string".

O cliente UDP – **c_rgb2hex** – deve: i) receber na linha de comando a *string* RGB através da opção -r/--rgb; ii) enviar a referida *string* ao servidor; iii) aguardar a resposta do servidor; iv) mostrar a string *hexadecimal color code*; e v) criar um ficheiro JPEG composto por um quadrado de 256 por 256 píxeis com a cor a ser definida pela *string hexadecimal color code*.

Sugestões: 1) uso da função **system** para executar o comando convert; **2)** função **sscanf** ou **strtok/atoi** para análise de *strings*

Servidor – s_rgb2hex

O programa servidor deve suportar os seguintes parâmetros da linha de comandos, a serem implementadas com o gengetopt:
-p, --port <int>: porto de escuta do servidor. Deve estar compreendido no intervalo **[1024, 65535].** Parâmetro obrigatório.

Cliente – c_rgb2hex

O programa cliente deve suportar os seguintes parâmetros da linha de comandos, a serem implementadas com o gengetopt:
--ip/-i <IPv4>: endereço IPv4 do servidor. Caso o valor indicado não corresponda a um endereço IPv4 válido, a aplicação termina com uma mensagem apropriada no canal de erro padrão. Parâmetro obrigatório.
--port/-p <int>: porto do servidor. Caso o valor esteja fora da gama do intervalo fechado **[1024, 65535]**, a aplicação termina com uma apropriada mensagem no canal de erro padrão. Parâmetro obrigatório.
--rgb/-r <string>: *string* RGB a ser enviada ao servidor. Parâmetro obrigatório.
--jpg/-j <filename>: nome do ficheiro JPEG a ser criado. Caso não seja especificado, deve ser assumido o nome "pa.jpg".

Considere os seguintes exemplos de execução das aplicações cliente e servidor.

Exemplo 1

```
./s_rgb2hex -p 1234
[SERVER] waiting for client requests
[SERVER] (10 bytes received)
[SERVER] RGB:'127,10,128'
[SERVER] '#7f0a80'
[SERVER] Sending response back to client
[SERVER] (7 bytes sent)

[SERVER] waiting for client requests
[SERVER] (9 bytes received)
[SERVER] RGB:'255,0,128'
[SERVER] '#ff0080'
[SERVER] Sending response back to client
[SERVER] (7 bytes sent)
```

```
./c_rgb2hex -p 1234 -i 127.0.0.1 -r 127,10,128
[CLIENT] Data to server '127,10,128'
[CLIENT] (10 bytes sent)
[CLIENT] waiting for server response
[CLIENT] Received: '#7f0a80'
[CLIENT] Created JPG file 'pa.jpg'


./c_rgb2hex -p 1234 -i 127.0.0.1 -r 255,0,128 -j x.jpg
[CLIENT] Data to be sent to server '255,0,128'
[CLIENT] (9 bytes sent)
[CLIENT] waiting for server response
[CLIENT] Received: '#ff0080'
[CLIENT] Created JPG file 'x.jpg'
```

Exemplo 2

```
./s_rgb2hex -p 4444
[SERVER] waiting for client requests
[SERVER] (7 bytes received)
[SERVER] RGB:'912,0,1'
[SERVER] 'ERROR in RGB values '912,0,1''
[SERVER] Response 'ERROR in RGB values '912,0,1'' to client
[SERVER] (29 bytes sent)
[SERVER] waiting for client requests
```

```
./c_rgb2hex -p 4444 -i 127.0.0.1 -r 912,0,1
[CLIENT] Data to be sent to server '912,0,1'
[CLIENT] (7 bytes sent)
[CLIENT] waiting for server response
[CLIENT] Received: 'ERROR in RGB values '912,0,1''
[CLIENT] '912,0,1' invalid RGB - no JPG file created
```

Exemplos 3

```
./s_rgb2hex -p 99999999
[SERVER] invalid port
```

```
./c_rgb2hex -p 4444 -i 999.0.0.1 -r 1,1,1
ERROR - Invalid Network Address
```