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

Prova Prática – Exame Normal – Pós-Laboral – Enunciado F

2025.01.13 / 20h00 Prova com consulta **Duração:** 2h30m

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 à participação 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_Normal**
  2) Seguidamente, copie a máquina virtual da UC para a pasta EI_PA_Normal 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_Normal do ambiente de trabalho e descompactá-lo.
- **Antes de iniciar a prova:**
  - Execute os seguintes comandos:

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

    ```
    cd ~/ENormal/R_NUMERO/;  tar cvf ENormal_YYYYMMDD_R_NUMERO.tar *
    ```

    (em que YYYYMMDD corresponde à data corrente (e.g., 20250113) e R_NUMERO obedece ao formato acima indicado);
  - Verifique que o arquivo ".tar" que criou não está vazio, através da execução de:

    ```
    tar tvf ENormal_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;

Pergunta 1 [10 valores]

(Escreva as suas respostas a esta pergunta no diretório **`"~/ENormal/R_NUMERO/Pergunta1"`**. 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 existentes no arquivo **EmptyProject-Template.zip (usar a versão disponível no Moodle)**.

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

Recorrendo à linguagem C, à norma Pthreads e ao utilitário gengetopt, implemente a aplicação ***guess_num*** cujo propósito é o de simular o jogo 'Adivinha o Número' com múltiplos jogadores. De um modo geral, este jogo realiza-se da seguinte forma:

1. É gerado um valor inteiro aleatório que, para esta aplicação, pode variar entre 2 e 100 (inclusive) e representa o número que se pretende adivinhar;
2. O jogo é disputado em rondas. A cada ronda, os jogadores têm uma tentativa para adivinhar o valor gerado no ponto anterior;
3. No final de cada ronda, após todos os jogadores efetuarem a sua tentativa, há um júri que vai verificar as jogadas realizadas, comparando a jogada de cada jogador com o valor gerado no ponto 1. Para cada jogador, o júri dá a indicação se o valor a adivinhar é inferior ou superior ao da jogada.
4. Se houver pelo menos um jogador que tenha adivinhado o valor gerado no ponto 1, o júri dá por terminado o jogo e a aplicação termina. Caso contrário, avança para a próxima ronda.

O jogo requer a utilização de um vetor de N inteiros, sendo N o número de jogadores que vão participar no jogo. Cada posição do vetor vai ser utilizada por um jogador para armazenar o valor da jogada de uma ronda. Após todos os jogadores efetuarem a sua tentativa, o júri acede a esse vetor de forma exclusiva e, para cada posição, atribui o valor -1 se o valor a adivinhar for inferior, ou 1 se o valor a adivinhar for superior ao da jogada. Se for igual, o júri deve indicar que o jogador daquela posição acertou no número. Depois de percorrer todas as posições do vetor e, caso tenham sido encontrados vencedores, o júri termina o jogo com a indicação do número de rondas jogadas.

Por exemplo, considere que o número a adivinhar é o 34 e que o jogo foi iniciado com 4 jogadores. Então, a aplicação deve criar um vetor de inteiros de 4 posições e, numa ronda, cada jogador deve preencher a sua posição respetiva com os valores aleatórios:

0	**1**	**2**	**3**
56	47	12	23

Após todos os jogadores efetuarem a sua tentativa, o júri acede a este vetor e faz a sua avaliação, atribuindo a cada posição o valor -1 ou 1, conforme o caso. Neste exemplo, o vetor deve ficar preenchido da seguinte forma, após avaliação do júri:

0	**1**	**2**	**3**
-1	-1	1	1

Com base nesta informação disponibilizada pelo júri, cada jogador pode ajustar o seu limite mínimo ou máximo para geração do valor aleatório da próxima ronda (no início, os limites mínimo e máximo têm valores de 2 e 100, respetivamente).

O júri e cada um dos jogadores são representados na aplicação através de *threads* que devem funcionar de forma sincronizada, ou seja, enquanto os jogadores estiverem a realizar as suas jogadas, o júri não pode aceder ao vetor e, por outro lado, enquanto o júri estiver a avaliar as jogadas, os jogadores têm de aguardar.

O número de jogadores e o número a adivinhar são controlados através dos seguintes parâmetros da linha de comandos:

-p/--num-players <int>: parâmetro obrigatório que indica o número de jogadores que vão participar no jogo. A aplicação deve validar que o valor é positivo.

-n/--number <int>: parâmetro opcional que indica o número a adivinhar. A aplicação deve validar que o valor está compreendido entre [2, 100]. Caso o parâmetro não seja indicado, a aplicação deve gerar um valor aleatório compreendido nesse intervalo.

A aplicação deve ainda emitir informação apropriada no canal de saída padrão (*stdout)*, seguindo o modelo apresentado nos exemplos.

Exemplo 1

```
$ ./guess_num -p 4
[INFO] Number to guess: 7
[JURY] Waiting for round #1 to finish...
[JURY] Player 1 guessed 30.
[JURY] Player 2 guessed 27.
[JURY] Player 3 guessed 94.
[JURY] Player 4 guessed 59.
[JURY] Waiting for round #2 to finish...
[JURY] Player 1 guessed 3.
[JURY] Player 2 guessed 17.
[JURY] Player 3 guessed 43.
[JURY] Player 4 guessed 11.
[JURY] Waiting for round #3 to finish...
[JURY] Player 1 guessed 12.
[JURY] Player 2 guessed 7.
[JURY] Player 2 is a WINNER!
[JURY] Player 3 guessed 7.
[JURY] Player 3 is a WINNER!
[JURY] Player 4 guessed 9.
[JURY] Game finished in 3 round(s).
```

Exemplo 2

```
$ ./guess_num -p 4 -n 34
[INFO] Number to guess: 34
[JURY] Waiting for round #1 to finish...
[JURY] Player 1 guessed 38.
[JURY] Player 2 guessed 35.
[JURY] Player 3 guessed 20.
[JURY] Player 4 guessed 67.
[JURY] Waiting for round #2 to finish...
[JURY] Player 1 guessed 35.
[JURY] Player 2 guessed 24.
[JURY] Player 3 guessed 56.
[JURY] Player 4 guessed 34.
[JURY] Player 4 is a WINNER!
[JURY] Game finished in 2 round(s).
```

Exemplo 3

```
$ ./guess_num -p -10
[ERROR] Number of players must be positive.
```

Exemplo 4

```
$ ./guess_num -p 10 -n 213
[ERROR] Number to guess must be between 2 and 100.
```

Nota: Pode utilizar a seguinte função para gerar um valor aleatório compreendido no intervalo fechado [min,max].

```
int rand_between(int min, int max) {
    unsigned int seed = time(NULL) ^ getpid() ^ pthread_self();
    return min + rand_r(&seed) % (max - min + 1);
}
```

Pergunta 2 [10 valores]

(Escreva as suas respostas a esta pergunta no diretório **"~/ENormal/R_NUMERO/Pergunta2"**.

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_v2.5.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)

Recorrendo à linguagem C, pretende-se que implemente o serviço cliente/servidor UDP **xmaslights** cujo propósito é gerir um ou mais conjuntos de luzes de Natal, permitindo obter ou alterar o estado atual das luzes de cada conjunto. Um conjunto possui 32 luzes, e é representado no serviço através de um inteiro de 32 bits sem sinal onde cada bit representa o estado atual de uma luz: se o valor do bit for zero, a luz está desligada, mas se o valor for um, a luz está ligada. As luzes de um conjunto apenas podem ser ligadas ou desligadas de acordo com uma lista de padrões predefinida. A tabela seguinte apresenta a lista de padrões existentes e os seus IDs. Os IDs são usados pelos clientes do serviço para atribuir o padrão correspondente a um determinado conjunto de luzes:

ID do Padrão	MSB Padrão LSB
1 (ALL OFF)	00000000000000000000000000000000
2 (ALL ON)	11111111111111111111111111111111
3 (PAIR ODD)	00110011001100110011001100110011
4 (PAIR EVEN)	11001100110011001100110011001100
5 (ALTERNATE ODD)	01010101010101010101010101010101
6 (ALTERNATE EVEN)	10101010101010101010101010101010

No serviço **xmaslights**, a quantidade de conjuntos de luzes a gerir é fornecida através de um parâmetro da linha de comandos. Quando o servidor inicia, todos os conjuntos devem ser inicializados com as luzes desligadas, ou seja, com todos os bits a zero (o padrão identificado pelo ID 1). De seguida, o servidor regista-se num porto (também indicado através de um parâmetro da linha de comandos) à escuta de pedidos de clientes. O servidor aceita pedidos de 2 bytes, sendo que o primeiro byte deve conter o nº do conjunto de luzes e o segundo byte deve conter o ID do padrão que se pretende aplicar a esse conjunto. Por exemplo, se o servidor receber o pedido representado na tabela seguinte, significa que o cliente pretende aplicar o padrão com o ID 5 ao conjunto de luzes nº 6:

Primeiro byte	Segundo byte
6	5

Por outro lado, **se o segundo byte do pedido for zero**, isto significa que o cliente apenas pretende obter o padrão atual do conjunto de luzes nº 6. O nº do conjunto e o ID do padrão a aplicar são fornecidos ao cliente através de parâmetros da linha de comando, sendo os mesmos enviados ao servidor através do protocolo de transporte UDP. O servidor processa o pedido, validando o nº do conjunto e o ID do padrão. De seguida, o servidor obtém (se o ID do padrão for zero) ou atribui o padrão solicitado pelo cliente, respondendo sempre com um número de 32 bits sem sinal, em formato de rede, e que representa precisamente o padrão atualmente aplicado ao conjunto de luzes. Em caso de sucesso, o cliente deve mostrar na saída padrão os bits da resposta enviada pelo servidor (do bit mais significativo para o menos significativo). Em caso de erro (p.e., o nº do conjunto de luzes ou o ID do padrão é inválido), o servidor deve responder com o código de erro 255 (0x000000FF em hexadecimal) e o cliente deve apresentar uma mensagem de acordo com os exemplos mostrados abaixo.

Servidor – s_xmaslights

O programa servidor deve suportar os seguintes parâmetros da linha de comandos:

`-p, --port <int>:` porto de escuta do servidor. Deve estar compreendido no intervalo [**1024,65535]**. Parâmetro obrigatório.

`-s, --lightsets <int>:` quantidade de conjuntos de luzes a gerir pelo servidor. Deve ser um valor positivo. Parâmetro obrigatório.

Cliente – c_xmaslights

O programa cliente deve implementar os seguintes parâmetros da linha de comandos:

--ip/-i <IPv4>: endereço IPv4 do servidor. Caso o valor indicado não corresponda a um endereço IPv4 válido, a aplicação termina com uma apropriada mensagem 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.

--command/-c <string>: tipo de pedido ao enviar ao servidor. Assume apenas um dos seguintes valores: "get", "set". Se o comando for "get", então o cliente deve obter o padrão atualmente aplicado ao conjunto de luzes indicado no parâmetro '-s'. Parâmetro obrigatório.

--lightset/-s <short>: O nº do conjunto de luzes. Deve ser um valor positivo. Parâmetro obrigatório.

--pattern/-P <short>: O ID do padrão a aplicar ao conjunto de luzes. Parâmetro obrigatório, caso o comando seja "set", caso contrário este parâmetro deve ser ignorado.

Exemplo 1

```
./s_xmaslights -p 1234 -s 10
[SERVER] Managing 10 lightsets
[SERVER] waiting for client requests on port: 1234
[SERVER] Sending light set #7 pattern to 127.0.0.1:53309
[SERVER] Sending light set #7 pattern to 127.0.0.1:53311
[SERVER] Sending light set #1 pattern to 127.0.0.1:53313
```

```
./c_xmaslights -p 1234 -i 127.0.0.1 -c get -s 7
PATTERN(7) = 00000000000000000000000000000000
./c_xmaslights -p 1234 -i 127.0.0.1 -c set -s 7 -P 3
PATTERN(7) = 00110011001100110011001100110011
./c_xmaslights -p 1234 -i 127.0.0.1 -c set -s 1 -P 5
PATTERN(1) = 01010101010101010101010101010101
```

Exemplo 2

```
./s_xmaslights -p 1234 -s 15
[SERVER] Managing 15 lightsets
[SERVER] waiting for client requests on port: 1234
[SERVER] Sending error to 127.0.0.1:53313
[SERVER] Sending error to 127.0.0.1:53315
```

```
./c_xmaslights -p 1234 -i 127.0.0.1 -c get -s 20
[ERROR] An error occurred processing your request.
Check your light set number and pattern ID.
./c_xmaslights -p 1234 -i 127.0.0.1 -c set -s 1 -P 9
[ERROR] An error occurred processing your request.
Check your light set number and pattern ID.
```

Exemplo 3

```
./s_xmaslights -p 1234 -s 5
[SERVER] Managing 5 lightsets
[SERVER] waiting for client requests on port: 1234
```

```
./c_xmaslights -p 1234 -i 127.0.0.1 -c set -s 2
[ERROR] Pattern ID is mandatory if command is 'set'.
```

Exemplo 4

```
./s_xmaslights -p 99999999 -s 2
[SERVER] invalid port
```

```
./c_xmaslights -p 4444 -i 999.0.0.1 -c get -s 1
ERROR - Invalid Network Address
```