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

Prova Prática – Exame Normal – Diurno – Enunciado E

2025.01.13 / 14h30 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 (última a versão disponível no Moodle)**.

NOTA 3: Não é permitido o uso de Variable Length Arrays (VLA)

NOTA 4: código entregue que **não compile através da linha de comando "make"** (sem identificadores adicionais) 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, elabore o programa **work_processor** que visa simular o processamento de unidades de trabalho por *threads* especializadas.

O programa deve receber um parâmetro de entrada através da linha de comandos utilizando o parâmetro **-n/--work_size <número>,** que especifica o número total de unidades de trabalho a serem processadas. Cada unidade de trabalho poderá ser de tipo 1 ou 2. O programa deve então criar **quatro** threads: **duas** threads de **tipo 1** e **duas** threads de **tipo 2**.

O programa deve gerar um vetor de unidades de trabalho, cujo tamanho é dado pelo parâmetro de entrada work_size. Nesse vetor, cada unidade possui um tipo associado (1 ou 2), atribuído aleatoriamente. Seguidamente o programa deverá criar as quatro threads que irão fazer o processamento. O programa termina quando todas as unidades de trabalho tiverem sido processadas.

Cada thread deverá procurar por uma unidade de trabalho do seu tipo para processar. O processamento de cada unidade de trabalho leva **um** segundo. Depois de decorrida esta duração, a thread deverá marcar essa unidade como processada e escrever uma mensagem na saída padrão (stdout) como exemplificado abaixo. A thread deverá terminar quando não houver mais nenhuma unidade de trabalho do seu tipo para processar.

O programa deve garantir que as threads sincronizam corretamente o acesso ao vetor de unidades de trabalho para evitar *race conditions*. Deve também garantir que enquanto uma thread está a processar uma unidade de trabalho (duração 1s) as outras threads conseguem aceder às restantes unidades de trabalho, e que não há duas threads a processar a mesma unidade de trabalho. A aplicação deve ainda emitir informação apropriada no canal de saída padrão (stdout), seguindo o modelo apresentado no Exemplo 1.

Exemplo 1

```
$ ./task_processor -n 4
Work units:
1 - type 1
2 - type 2
3 - type 1
4 - type 2
Starting work!
worker thread #1 of type 1 is starting.
worker thread #0 of type 1 is starting.
worker thread #3 of type 2 is starting.
worker thread #2 of type 2 is starting.
Worker of type 2 has finished work item number 2.
worker thread #3 of type 2 is done.
Worker of type 1 has finished work item number 1.
worker thread #1 of type 1 is done.
Worker of type 1 has finished work item number 3.
worker thread #0 of type 1 is done.
Worker of type 2 has finished work item number 4.
worker thread #2 of type 2 is done.
All work done.
```

Exemplo 2

```
$ ./task_processor -n -10
ERROR: worksize must be > 0
```

Exemplo de código C para gerar valores aleatórios. Sugere-se a consulta de man **rand_r**

Função **rand_r:** gerar números aleatórios **Fora das threads**	Função **rand_r:** gerar números aleatórios **Nas threads**

`unsigned int r_state = time(NULL) ^ getpid();`
`int value1 = rand_r(&r_state);`
`int value2 = rand_r(&r_state);`
	`unsigned int r_state = time(NULL) ^ getpid() ^ pthread_self();`
	`int value1 = rand_r(&r_state);`
	`int value2 = rand_r(&r_state);`

Pergunta 2 [10 valores]

(Escreva as suas respostas a esta pergunta no diretório **"~/ENormal/R_NUMERO/Pergunta2"**. Deve indicar o seu nome completo e número de estudante IPLeiria no ficheiro **README.txt** a ser criado no diretório)
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

Recorrendo à linguagem C, pretende-se que implemente o serviço cliente/servidor UDP **stegano** cujo propósito é exemplificar técnicas de esteganografia. A esteganografia consiste em esconder uma mensagem dentro de outra mensagem, neste caso o objetivo será esconder texto dentro de números inteiros.

No serviço **stegano**, uma string com uma mensagem 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, e envia de volta para o cliente um vetor com números inteiros (de tipo uint32_t). Se a string enviada tiver **n** caracteres, o servidor devolve um vetor com **8n** números inteiros de 32 bits. Por uma questão de simplicidade, deve considerar que as strings a serem processadas são compatíveis com a língua inglesa, e consequentemente, cada caracter ocupa somente um *byte*.

Concretamente, para cada caracter da *string*, o servidor gera **oito números inteiros** e para cada um destes números inteiros **o bit menos significativo** deverá corresponder a um dos 8 bits do caracter. Por exemplo se um dos caracteres enviados for o caracter 'a' cuja representação em binário corresponde a `01100001`, então serão gerados 8 números inteiros de 32bits para esse caracter em que o primeiro número terá bit menos significativo 1, o segundo número terá bit menos significativo 0, o terceiro número terá bit significativo 0, e assim por diante. Os restantes 31 bits mais significativos de cada número inteiro serão aleatórios ou zero dependendo da opção da linha de comandos `-r/--use_rand`.

Depois de receber o vetor do servidor, o cliente deverá escrever cada elemento do vetor na saída padrão em formato decimal, conforme ilustrado nos Exemplos 1 e 2.

O servidor UDP – **s_stegano** – deve i) aguardar por um pedido de um cliente, e quando recebe um pedido, deve ii) proceder à criação do vetor com n * 8 números inteiros (uint32_t) e iii) enviar o vetor para o cliente. Deve ainda mostrar na saída padrão a string recebida do cliente.

O cliente UDP – **c_stegano** – deve i) Receber na linha de comando uma string da opção -m/--message ii) a string deve ser enviada ao servidor; iii) ficando o cliente a aguardar a resposta do servidor; iv) o cliente deve escrever na saída padrão cada um dos números recebidos.

Sugestão: uso da expressão `srand(time(NULL)^getpid())` para inicializar o gerador de números aleatórios, e `rand()` para obter um número aleatório.

Servidor – s_stegano

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.

`-r, --use_rand <int>:` Deve ter o valor 0 ou 1. Se o valor for 1, preencher o vetor com números aleatórios antes de colocar os bits menos significativos, caso contrário, usar o número 0. Parâmetro obrigatório.

Cliente – c_stegano

O programa cliente deve implementar os seguintes parâmetros da linha de comandos:
-i/--ip <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.

-p/--port <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.

-m/--messsage <string>: string a ser escondida. Tamanho máximo: 48 caracteres. Parâmetro obrigatório.

(Exemplos na página seguinte)

Exemplo 1

NOTA: o valor em hexadecimal do caracter '**P**' é **0x50** (0101.0000) e **0x41** (0100.0001) para a letra '**A**'. As linhas "[CLIENT] 0" até "[CLIENT] 7" correspondem à codificação dos bits da letra 'P', enquanto as restantes linhas representam a codificação da letra 'A'.

./stegano_s -p 1234 -r 0	**./stegano_c -i 127.0.0.1 -p 1234 -m PA**
[SERVER] Listening on port: 1234 [SERVER] Request from addr: 127.0.0.1:41370 [SERVER] string received: PA	[CLIENT] 0: 0 [CLIENT] 1: 0 [CLIENT] 2: 0 [CLIENT] 3: 0 [CLIENT] 4: 1 [CLIENT] 5: 0 [CLIENT] 6: 1 [CLIENT] 7: 0 [CLIENT] 8: 1 [CLIENT] 9: 0 [CLIENT] 10: 0 [CLIENT] 11: 0 [CLIENT] 12: 0 [CLIENT] 13: 0 [CLIENT] 14: 1 [CLIENT] 15: 0

Exemplo 2

./stegano_s -p 1234 -r 1	**./stegano_c -i 127.0.0.1 -p 1234 -m PA**
[SERVER] Listening on port: 1234 [SERVER] Request from addr: 127.0.0.1:54146 [SERVER] string received: PA	[CLIENT] 0: 1236188486 [CLIENT] 1: 612149398 [CLIENT] 2: 1850040426 [CLIENT] 3: 1212016672 [CLIENT] 4: 413190309 [CLIENT] 5: 2049304040 [CLIENT] 6: 234378215 [CLIENT] 7: 1081965658 [CLIENT] 8: 1112483915 [CLIENT] 9: 230284504 [CLIENT] 10: 2042884918 [CLIENT] 11: 775114598 [CLIENT] 12: 783560770 [CLIENT] 13: 1987760426 [CLIENT] 14: 165454573 [CLIENT] 15: 1450623062

Exemplos 3

./stegano_c -p 99999999 -r 1	**./stegano_c -i 999.0.0.1 -p 1234 -m Ola**
[SERVER] invalid port	[CLIENT] ERROR - Invalid Network Address

Exemplos 4

./stegano_c -i 127.0.0.1 -p 1234 -m string_muito_muito_muito_muito_muito_muito_grande

[CLIENT] ERROR: string to encode is 55 chars. Max accepted size is 48 chars.