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

Prova Prática – Exame Recurso – Pós-Laboral – Enunciado H

2025.02.03 / 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.
- O uso de dispositivos de armazenamento de dados (PEN, discos externos, etc.) só é permitido nos primeiros quinze minutos de prova.
- Os dispositivos *smartphones*, *smartwatches* e *fones* devem ser guardados em mochila/saco.

- **Preparação da máquina virtual:**
  1) Crie, no ambiente de trabalho, a pasta **EI_PA_Recurso**
  2) Seguidamente, copie a máquina virtual da UC para a pasta EI_PA_Recurso 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_Recurso do ambiente de trabalho e descompactá-lo.
- **Antes de iniciar a prova:**
  - Execute os seguintes comandos:

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

    ```
    cd ~/ERecurso/R_NUMERO/;  tar cvf ERecurso_YYYYMMDD_R_NUMERO.tar *
    ```

    (em que YYYYMMDD corresponde à data corrente (e.g., 20250203) e R_NUMERO obedece ao formato acima indicado);
  - Verifique que o arquivo ".tar" que criou não está vazio, através da execução de:

    ```
    tar tvf ERecurso_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 1 [10 valores]

(Escreva as suas respostas a esta pergunta no diretório **"~/ProvaP/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 (e.g., /bin/sh).
NOTA 2: a solução deve ser implementada com recurso aos ficheiros/makefile do arquivo EmptyProject-Template.zip **(usar a versão disponível no Moodle)**
NOTA 3: 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.

Utilizando a linguagem C e a norma Pthreads, desenvolva a aplicação **temp_monitor**, cuja finalidade é simular o funcionamento de vários sensores de temperatura, a trabalhar em paralelo, para monitorizar um determinado ambiente. Cada sensor é implementado com uma *thread* que realiza leituras de temperatura, enquanto uma *thread* de monitorização processa essas leituras, calculando a média e apresentando o resultado no final de cada rodada de recolha de dados.

Cada *thread* sensor gera uma leitura de temperatura aleatória que consiste num valor inteiro compreendido no intervalo fechado [5°C, 30°C]. Em cada rodada, cada *thread* sensor regista o valor num *buffer* partilhado, sendo que cada sensor só pode realizar uma leitura por rodada.

A *thread* de monitorização é responsável por calcular e exibir a média das leituras de temperatura após o término de cada rodada. Depois de apresentar os resultados, a *thread* de monitorização aguarda a conclusão da próxima rodada de leituras.
O número de sensores e rodadas de leituras é determinado pelos seguintes argumentos da linha de comando:

- **-s/--sensors <int>**: parâmetro obrigatório que indica o número de sensores e que deve estar compreendido no intervalo inteiro [2,10]
- **-r/--rounds<int>**: parâmetro obrigatório que deve estar compreendido no intervalo inteiro [1,10]

O tratamento dos parâmetros da linha de comandos deve ser feito com recurso à ferramenta `gengetopt`.
Quando é atingido o número de rodadas indicado pelo parâmetro **-r/--rounds**, a aplicação deve terminar.

A aplicação deve ainda enviar a informação apropriada através dos canais de saída padrão (*stdout* e *stderr)*, seguindo o modelo apresentado no Exemplo 1.

Exemplo 1

```
./temp_monitor --sensors 3 --rounds 2
Round #01: [sensor-thread 03] temperature 18°C
Round #01: [sensor-thread 02] temperature 10°C
Round #01: [sensor-thread 01] temperature 8°C

Round #01: [monitoring-thread] average = 12.0°C
-------------------------------------------------

Round #02: [sensor-thread 02] temperature 7°C
Round #02: [sensor-thread 03] temperature 19°C
Round #02: [sensor-thread 01] temperature 20°C

Round #02: [monitoring-thread] average = 15.3°C
-------------------------------------------------
```

Exemplo 2

```
./temp_monitor -s 15 -r 5
ERROR - Total number of sensors must be within [2,10]
```

Exemplo 3

```
./temp_monitor -s 5 -r 20
ERROR - Total number of rounds must be within [1,10]
```

Pergunta 2 [10 valores]

(Escreva as suas respostas a esta pergunta no diretório **"~/EEspecial/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

Recorrendo à linguagem C, pretende-se que implemente o serviço cliente/servidor TCP concorrente transfFile. Este serviço permite a um cliente solicitar e obter um ficheiro existente no servidor. O ficheiro é indicado através do nome.

O protocolo aplicacional é o seguinte: o cliente – **`transFile_C`** — envia para a aplicação servidora – **`transFile_S`** – o nome do ficheiro que pretende receber, indicando se relevante, o caminho (e.g., "**PA.txt**", etc.). O servidor responde com o conteúdo do ficheiro, caso esse exista. Caso contrário, o servidor envia a string "404 NOT FOUND".

Dado tratar-se de um serviço concorrente, o servidor deve ter a capacidade de atender vários clientes simultaneamente. Por uma razão de simplicidade, o servidor procura o ficheiro solicitado pelo cliente apenas na pasta corrente do servidor. Em termos de protocolo aplicacional, o servidor envia o ficheiro ao cliente em blocos de 256 bytes. Quando tiver transmitido a totalidade do ficheiro, o servidor deve encerrar a ligação com o cliente, o mesmo sucedendo aquando da notificação "404 NOT FOUND" indicadora da inexistência do ficheiro. Em termos de saída padrão, o servidor deve ainda documentar o pedido recebido pelo cliente, i.e., o nome do ficheiro solicitado pelo cliente. As mensagens do servidor na saída padrão devem indicar o PID do processo que as produz (**exemplo:** `[SERVER][49999] Listening on port 9999).` Por sua vez, o cliente deve receber o ficheiro fazendo uso de um buffer de receção de 256 bytes, gravando esse ficheiro com o nome solicitado na pasta /tmp, usando para o efeito o nome do ficheiro (e.g., "/tmp/PA.txt"), indicando na saída padrão o número de bytes do ficheiro. Tenha em atenção que todas as mensagens de erro devem ser enviadas para o canal de saída padrão (`stderr`).

Servidor – `transfile_S`

O programa servidor deve implementar o suporte para os seguintes parâmetros da linha de comandos:
`--port/-p <int>:` porto de escuta do servidor. Deve estar compreendido no intervalo [**1024,65535]**. Parâmetro obrigatório.

Cliente – `transfile_C`

O programa cliente deve disponibilizar 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.
`--file/-f <string>`: ficheiro a solicitar ao servidor.

Sugestão: código para leitura/escrita de ficheiro com API de baixo nível.

`// Abrir um ficheiro para leitura`
`int f = open(filename, O_RDONLY);`
`// Leitura de bloco SIZE do ficheiro`
`char buff[SIZE];`
`… = read(f, buff, SIZE)`
`(…)`
	`// Abrir um ficheiro para escrita`
	`f = open(filename,O_WRONLY|O_CREAT|O_TRUNC,0644);`
	`// Escrita de conteúdo existente em buff para o ficheiro`
	`…= write(f, buff, num_bytes_to_write);`

Considere os seguintes exemplos de execução das aplicações cliente e servidor.
Exemplo 1

`./transFile_S -p 9999`
`----------`
`[SERVER](2398) Server listening on port 9999`
`[SERVER](2401) request for file 'a.txt'`
`[SERVER](2401) file 'a.txt' sent to remote client`
`----------`
`[SERVER](2398) Server listening on port 9999`
`[SERVER](2403) request for 'not_exist.txt'`
`[SERVER](2403) file 'not_exist.txt' not found`
	`./transFile_C -i 127.0.0.254 -p 9999 -f a.txt`
	`[CLIENT] 22056 bytes saved to /tmp/a.txt`
	
	`./transFile_C -i 127.0.0.254 -p 9999 -f not_exist.txt`
	`[CLIENT] Server responded: '404 NOT FOUND'`

Exemplos 2

./transFile_S -p 123456 [SERVER] invalid port	**./transFile_C -p 4444 -i AAA.12.0.1 -f a.txt** ERROR - Invalid Network Address