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

Prova Prática – Exame Recurso – Diurno – Enunciado G

2025.02.03 / 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.
- 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**
  2) Seguidamente, copie a máquina virtual da UC para a pasta EI_PA 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 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 **"~/ERecurso/R_NUMERO/Pergunta1"**)
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 ficheiro EmptyProject-Template.zip **(usar a última 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 e à norma Pthreads, implemente a aplicação *airport*, cujo propósito é o de simular a atividade num aeroporto que tem somente uma pista de aterragem/descolagem, focando em três operações distintas: i) Aterragens; ii) Descolagens; iii) Inspeção da pista. Cada uma das operações é implementada por uma *thread* distinta.

As operações de aterragem e descolagem são alternadas. Isto é, a seguir a uma descolagem, ocorre uma aterragem e assim sucessivamente. Periodicamente, ocorre a operação de inspeção, na qual, como o nome sugere, a pista é inspecionada para detetar eventuais objetos estranhos. Obviamente, qualquer uma dessas atividades é exclusiva, i.e., enquanto decorre uma delas, nenhuma das outras pode ocorrer.
Para efeitos da simulação, que deve recorrer a três threads – aterragem, descolagem, inspeção – considere ainda que cada operação ocupa a pista em **um** segundo. Isto é, tanto uma aterragem, como uma descolagem, ou uma inspeção, levam **um** segundo a serem realizadas. Considere ainda que a inspeção da pista ocorre a cada **cinco** segundos.

O número total de operações a serem realizadas, que corresponde à soma do nº de descolagens, aterragens e inspeções de pista, é controlado através do seguinte parâmetro da linha de comandos:

> **-n/--num_operations <int>**: parâmetro obrigatório que deve estar compreendido no intervalo inteiro [5,15].

Quando é atingido o valor indicado pelo parâmetro **-n/--num_operations**, a aplicação deve terminar. A aplicação deve ainda emitir informação apropriada no canal de saída padrão (*stdout)*, seguindo o modelo apresentado no Exemplo 1.

Exemplo 1

```
./airport -n 7
[0001] Landing started.
[0001] Landing completed.
========================================
[0001] Takeoff started.
[0001] Takeoff completed.
========================================
[0002] Landing started.
[0002] Landing completed.
========================================
[0002] Takeoff started.
[0002] Takeoff completed.
========================================
[0003] Landing started.
[0003] Landing completed.
========================================
[0001] INSPECTION started.
[0001] INSPECTION completed.
========================================
[0003] Takeoff started.
[0003] Takeoff completed.
========================================
[LANDING #3] reached max number of operations (7)
[TAKEOFF #3] reached max number of operations (7)
[INSPECTION #1] reached max number of operations (7)
```

Exemplo 2

```
./airport -n 21
ERROR: total number of operations must be within [5,15]
```

Pergunta 2 [10 valores]

(Escreva as suas respostas a esta pergunta no diretório **"~/ERecurso/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/makefile do ficheiro **EmptyProject-client-server-template.zip (usar a última 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, pretende-se que implemente o serviço **cliente/servidor TCP concorrente** `randChars`. Esse serviço permite a um cliente solicitar e obter N caracteres aleatórios do servidor. Os caracteres aleatórios devem estar compreendidos no intervalo ['a', 'z'] e ['A', 'Z'].
O protocolo aplicacional é o seguinte: o cliente – **`C_randChars`** — envia para o servidor – **`S_randChars`** – um valor inteiro de 16 bits sem sinal que corresponde ao número de caracteres aleatórios que devem ser devolvidos pelo servidor. Recebido o pedido do cliente, o servidor envia o número de caracteres aleatórios solicitado, usando para o efeito blocos com **o tamanho máximo de 64 caracteres**. O cliente deve mostrar, na saída padrão (`stdout`), os caracteres recebidos. Depois de ter recebido todos os caracteres, o cliente deve ainda mostrar o número de operações de receção (`read/recv`) que teve de efetuar para receber todos os caracteres solicitados.
Dado tratar-se de um serviço concorrente, o servidor deve ter a capacidade de atender vários clientes simultaneamente. Em termos de protocolo aplicacional, o servidor deve encerrar a ligação com o cliente logo que tenha enviado o número de caracteres solicitados pelo cliente. O servidor deve ainda documentar na saída padrão o pedido recebido pelo cliente, i.e., o número de caracteres aleatórios solicitados pelo cliente. As mensagens do servidor na saída padrão devem indicar o PID do processo que as produz (**exemplo:** `"[SERVER][50260] Listening on port 9999"`, em que 50260 corresponde ao PID do processo que a produz)**.**

Servidor – S_randChars

O programa servidor deve recorrer ao gengetopt para 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 – C_randChars

O programa cliente deve recorrer ao gengetopt para implementar o suporte para 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.
`--nChars/-n <int>`: número de caracteres aleatórios a solicitar ao servidor. Valor deve estar compreendido entre **[1,65535]**. Parâmetro obrigatório.

Sugestão: funções **`int rand(unsigned int *seedp);`** e **`void srand(unsigned int seed);`**

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

Exemplo 1

```
./S_randChars -p 9999
[SERVER][50260] Listening on port 9999...
[SERVER][50262] Request for 66 random chars
[SERVER][50262] Sent 66 random chars to remote client
[SERVER][50262] Connection closed with remote client
```

```
./C_randChars -i 127.0.0.1 -p 9999 --nChars 66
[CLIENT] Requested 66 random bytes from server
cFPuFKKqIeTkIAztFMNUOZqv(…)
[CLIENT] Received 66 bytes in total (2 recv operations)
```

Exemplo 2

```
./S_randChars -p 9999
[SERVER][50266] Listening on port 9999...
[SERVER][50268] Request for 60 random chars
[SERVER][50268] Sent 60 random chars to remote client
[SERVER][50268] Connection closed with remote client
[SERVER][50270] Request for 30 random chars
[SERVER][50270] Sent 30 random chars to remote client
[SERVER][50270] Connection closed with remote client
```

```
./C_randChars -i 127.0.0.1 -p 9999 --nChars 60
[CLIENT] Requested 60 random bytes from server
zrZjGNYElzezFZPbpQx(…)
[CLIENT] Received 60 bytes in total (1 recv operations)

./C_randChars -i 127.0.0.1 -p 9999 --nChars 30
[CLIENT] Requested 30 random bytes from server
zrZjGNYElzezFZP(…)
[CLIENT] Received 30 bytes in total (1 recv operations)
```

Exemplos 3

```
./S_randChars -p 99999999
ERROR - invalid port
```

```
./C_randChars -p 4444 -i 999.0.0.1 -nChars 120
ERROR - Invalid Network Address
```

Exemplo 4

```
./C_randChars -p 3344 -i 127.0.0.1 -n -120
ERROR - Invalid number of random chars '-120'
```