Publicação:	2020.09.26 (v3)
Peso na nota final:	25% da nota final (mínimo de 5.0 em 20)
Informações:	Moodle
Data de entrega	**Consultar calendário de avaliação EI**

Projeto 1S 2020-2021

nanoShell

1 - Introdução

Como sabe, no contexto da informática, uma *shell* designa a aplicação que recebe e interpreta comandos passados através da linha de comando. Exemplos de *shells* incluem, entre outras, a BASH, a ZSH e CSH.

No âmbito deste projeto pretende-se que elabore, recorrendo à linguagem C, uma *shell* denominada de **`nanoShell`**. O `nanoShell` deve ser capaz de:

a) Executar uma aplicação com múltiplos parâmetros de entrada

b) Capturar *signals* que sejam enviados para a nanoShell

c) Suportar redireccionamento da saída padrão (símbolo `>` e `>>`) e do canal de erro padrão (símbolo `2>` e `2>>`)

d) Mostrar uma ajuda sucinta

2 - Funcionamento

O `nanoShell` é uma aplicação de linha de comando que corre por cima da *shell* empregue pelo utilizador (e.g., `/bin/bash` no caso da máquina virtual Linux de PA). Ao ser executada sem parâmetros da linha de comando (./nanoShell), o `nanoShell` entra em modo de espera, aguardando por comandos do utilizador:

```
nanoShell$
```
[1]

O utilizador pode então executar comandos/aplicações dentro da `nanoShell`, tais como:

```
   1)
nanoShell$ date +%Y%m%d_%A -d19870527
```

[1] O uso da cor vermelha nos exemplos serve apenas para ilustrar a forma como a nanoShell aguarda por comandos.

```
19870527_Wednesday
```

2)

```
nanoShell$ file /bin/top
bin/top: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=0d351155e6a1cf2f096fc412246da411738ff7ec, for GNU/Linux 3.2.0,
stripped
```

O término do `nanoShell` é feito através da escrita da `bye`.

```
nanoShell$ bye
[INFO] bye command detected. Terminating nanoShell
```

2.1 - Funcionalidades

- O `nanoShell` deve suportar redireccionamento do canal de saída padrão (*stdout*) e do canal de erro padrão (*stderr*). Para o efeito o *nanoShell* deve detetar a presença dos símbolos ">", ">>", "2>" e "2>>" interpretando-os de forma similar ao que é feito por uma *shell* normal (e.g., BASH). **Sugestão:** `freopen`

- O `nanoShell` deve capturar os *signals* que lhe sejam enviados, adotando o seguinte comportamento:

- SIGINT: término do `nanoShell` com a indicação do PID do processo que enviou o *signal* ao `nanoShell`

- SIGUSR1: escrita de mensagem para o canal de saída padrão (*stdout*) a indicar data/hora em que o `nanoShell` foi lançado (e.g., 2020-09-11T18:00:27+01:00)

- SIGUSR2: escrita de mensagem para o ficheiro `nanoShell_status_dataHora.txt` (e.g., `nanoshell_status_2020.09.11_11h20.30.txt`), a indicar o número de aplicações/comandos que já foram executados pela atual instância da `nanoShell`, bem como quantas execuções solicitaram redireccionamento do canal de saída padrão e quantas solicitaram redireccionamento do canal de erro padrão. O ficheiro deve ser criado no diretório corrente. Considere o seguinte exemplo:

```
24 executions of applications
2 executions with STDOUT redir
1 execution with STDERR redir
```

(…)

- O `nanoShell` **não** tem suporte para *metacarateres*, tais como o ***?*** e o ***. Assim, por exemplo, se o utilizador solicitar a execução de `ls *.c`, a execução que é solicitada é efetivamente `ls a *.c`, **não** existindo expansão de **`*.c`** para eventuais nomes de ficheiros compatíveis com o padrão. Ao contrário, numa *shell* tradicional (e.g., BASH), o **`*.c`** é substituído pelos nomes dos ficheiros (nesse caso do diretório corrente) que sejam compatíveis com o padrão, isto é, cujo nome de ficheiro termine por "**.c**".

- O `nanoShell` **não** tem suporte para "*pipes*", isto é, não suporta múltiplas aplicações/comandos interligados pelo símbolo | (barra vertical). Por exemplo, o pedido `ps ax | wc -l` deve ser considerado pedido inadequado.

- O `nanoShell` **não** tem suporte para elementos entre aspas e entre plicas. Assim, sempre que o pedido do utilizador tenha uma ou mais aspas e/ou uma mais plica, o pedido deve ser considerado como inadequado. Por exemplo, `ls "a.txt"` deve ser considerado inadequado devido à presença de aspas, o mesmo sucedendo com `ls a.txt'`.(o número de plicas é irrelevante).

Sempre que a `nanoShell` classifique um pedido como inadequado, deve produzir a seguinte saída:

```
[ERROR] Wrong request 'PEDIDO_DO_UTILIZADOR'
```

Considere o seguinte exemplo:

```
nanoShell$ ls "a.txt"
[ERROR] Wrong request 'ls "a.txt"'
```

2.2 - Parâmetros da linha de comando

Opção	Explicação
`-f, --file <fich>`	Executa as linhas de comando indicadas no ficheiro de texto **fich**. As linhas em branco e as linhas iniciadas por "#" (linhas de comentários) devem ser ignoradas. Esta opção não é compatível com mais nenhuma opção da linha de comandos. Exemplo de ficheiro de linhas de comandos: **`#ficheiro "test_commands.txt"`** **`whoami`** **`date`** **`#fim do ficheiro`**
`-h, --help`	Lista ajuda sucinta, o nome e número IPLeiria de cada autor e termina.
`-m, --max <int>`	Número máximo de execuções de comandos/aplicações levadas a cabo pela `nanoShell`, sendo que quando alcance esse limite, a `nanoShell` termina. Esta opção não é compatível com mais nenhuma opção da linha de comandos.
`-s, --signalfile`	Quando inicia a execução, o `nanoShell` cria, no diretório corrente, o ficheiro **signal.txt** que contém os comandos que possibilitem o envio de *signals* ao `nanoShell`. Exemplo: se o PID do processo `nanoShell` for 2020, o conteúdo do ficheiro **signal.txt** será: **`kill -SIGINT 2020`** **`kill -SIGUSR1 2020`** **`kill -SIGIUSR2 2020`**

2.3 - Tratamento de erros

A aplicação deve validar os parâmetros da linha de comando.

a) Por exemplo, o parâmetro `-m/--max` <int> só é considerado válido para valores de <int> inteiros maiores do que zero. Caso seja indicado um valor não conforme, deve ser produzida a seguinte mensagem de erro:

`ERROR: invalid value ‘<int>’ for -m/--max.`

b) Deve ser validado a existência do ficheiro indicado através do parâmetro -f/--file <fich>.

`ERROR: cannot open file <fich> -- MENSAGEM_DE_ERRO_DO_SISTEMA`, em que MENSAGEM_DE_ERRO_DO_SISTEMA corresponde à mensagem de erro do sistema.

NOTA: A aplicação pode efetuar outras validações que os estudantes considerarem relevantes. Essas validações devem ser indicadas no relatório da aplicação numa secção denominada por “Validações adicionais”.

2.4 - Regras

1) A aplicação deve ser desenvolvida em linguagem C para o ambiente Linux da máquina virtual da UC. Apenas podem ser empregues os recursos existentes na máquina virtual da UC.
2) Não é permitida o uso da função `system` ou de qualquer chamada à `shell`.
3) Deve ser empregue o template EmptyProject-Templatev3.05.zip disponibilizado no moodle da UC
4) O projeto deve compilar, através da seguinte linha de comando: `make`
5) O projeto é avaliado na máquina virtual disponibilizada para a UC.
6) Projetos entregues que não compilem na máquina virtual da UC usando o *makefile* submetido com o projeto são avaliados com a classificação de 0 (zero).
7) As opções da linha de comando devem ser implementadas através do utilitário *gengetopt*.

Exemplos

Considere os seguintes exemplos de execução da `nanoShell`.

1) Execução com redireccionamento do canal de saída padrão

```
nanoShell$ cat /proc/cpuinfo > /tmp/out.txt
[INFO] stdout redirected to '/tmp/out.txt'
```

2) Execução com redireccionamento do canal de erro padrão

```
nanoShell$ cat /proc/cpuinfo 2> /tmp/err.txt
[INFO] stderr redirected to '/tmp/err.txt'
processor       : 0
vendor_id       : GenuineIntel
      (…)
```

3) Execução com uso da opção `-f/--file`

```
nanoShell -f some_commands.txt
[INFO] executing from file 'some_commands.txt'
[command #1]: whoami
user
[command #2]: uname -a > out.txt
[INFO] stdout redirected to 'out.txt'
name -r -v
```

5.4.0-40-generic #44-Ubuntu SMP Tue Jun 23 00:01:04 UTC 2020
(…)

4) Execução com uso da opção `-m/--max <int>`

```
nanoShell -m 2
[INFO] terminates after 2 commands
nanoShell$ whoami
user
nanoShell$ gcc -V
Using built-in specs.
COLLECT_GCC=gcc
(…)
Thread model: posix
gcc version 9.3.0 (Ubuntu 9.3.0-10ubuntu2)

[END] Executed 2 commands (-m 2).
```

5) Execução com a opção `-s/--signalfile`

```
nanoShell -s
[INFO] created file 'signals.txt'
nanoShell$
```

3 - Avaliação

A avaliação do projeto é distribuída da seguinte forma:

i) Funcionamento e eficiência: **80%**

ii) Implementação, organização e qualidade do código: **10%**

Este item abrange os seguintes elementos: comentários, estrutura de dados, nome dos identificadores [variáveis, funções], organização em funções, pertinência das mensagens de erro, simplicidade e elegância do código.

iii) Relatório: **10%**

4 - Relatório

- O projeto deve ser acompanhado de um relatório com um máximo de **cinco** páginas.

- A primeira página identifica os estudantes do grupo com nome completo, número de estudante, fotografia de rosto atualizada e a seguinte declaração: "*Nome_Estudante_1 (numero_estudante_1) e por Nome_Estudante_2 (numero_estudante_2) declaram sob compromisso de honra que o presente trabalho (código, relatórios e afins) foi integralmente realizado por nós, sendo que as contribuições externas se encontram claramente e inequivocamente identificadas no próprio código. Mais se declara que os estudantes acima identificados não disponibilizaram o código ou partes dele a terceiros.*".

- As restantes páginas do relatório devem descrever **para cada opção**:

a) Estado de funcionamento: *totalmente operacional*; *não implementado*; *implementado, mas com problemas* (neste caso indicar os problemas)

b) Como foi implementada a funcionalidade, nomeadamente o(s) algoritmo(s) empregue(s), quais são as principais estruturas de dados/funções empregues.

- O relatório deve ser entregue em formato PDF. Relatórios entregues num formato que não seja o formato PDF **não** serão considerados. O nome do ficheiro PDF do relatório deve ser: **`relatorio_proj_nanoShell_n1-n2.pdf`**, em que **n1** representa o número de estudante do 1º elemento do grupo e **n2** o número de estudante do 2º elemento do grupo.

5 - Regras

1 - O trabalho será realizado **individualmente** ou em grupo (máximo de **dois** estudantes, que podem ser de turnos práticos distintos).

2 - O trabalho deve estar claramente identificado, com o **nome completo e** respetivo **número** de cada estudante escrito como comentário no início de cada ficheiro de código C do projeto.

3 - Os comentários e os identificadores presentes no código fonte (nome de variáveis, funções, etc.) devem fazer uso da língua inglesa. As mensagens da aplicação devem também elas ser em língua inglesa.

4 - Todos os ficheiros do projeto (código C, ficheiros .h, relatório) devem ser reunidos, através de um utilitário de arquivo e compressão (zip, 7z, tar.gz, ou tar.bz2), num único ficheiro denominado "**PA.proj_nanoShell_2020-2021.n1-n2**"[2] em que **n1** representa o número de estudante do 1º elemento do grupo e **n2** o número de estudante do 2º elemento do grupo.

5 - O ficheiro relativo ao ponto anterior (regra nº 5) deve ser entregue através do mecanismo de entrega disponibilizado no moodle da unidade curricular. Em caso de dúvidas deve consultar os docentes. Apenas é permitida **uma** entrega do projeto.

6 - Não serão consideradas tentativas de entrega feitas após o prazo.

7 - Fraudes ou tentativas de fraudes originam uma classificação **nula** no presente trabalho para os prevaricadores, bem como o relato do sucedido às instâncias superiores.

8 - Após a entrega do projeto, poderá ser necessária uma apresentação oral do mesmo através de teleconferência, sendo esta agendada pelo docente. A apresentação é individual, sendo que a nota percentual na apresentação (de 0% a 100%) é multiplicada pela nota resultante da correção para efeitos de cálculo da nota final do projeto.

9 - Caso faça uso do correio eletrónico para o esclarecimento de dúvidas, deve sempre iniciar o assunto da mensagem por **[EI_PA][Projeto]** (caso contrário, a mensagem corre o risco de não ser corretamente identificada pelo filtro *anti-spam*). Para além disso, deve identificar-se com o nome, número, regime e turno prático que frequenta.

9.1 - Para que o esclarecimento de **dúvidas de programação** seja mais efetivo, qualquer comunicação deve incluir os seguintes elementos:

[2] A extensão do arquivo (.zip, .7z, .tar.gz, tar.bz2) depende do utilitário empregue para a compactação.

- Deve incluir código que reproduz o problema: **1)** Incluir uma versão do código como texto (não como captura de ecrã) tão simples quanto possível, mas que consegue reproduzir o problema; **2)** Eventualmente, colocar uma ou outra captura de ecrã para ilustrar o problema (falha na execução, *segmentation fault*, etc.); **3)** Sempre que possível enviar um ficheiro *main.c* contendo apenas o código que causa o problema e que pode ser compilado individualmente; **4)** Explicar com clareza e de forma sucinta o que está a acontecer e o que parece estar errado; **5)** Explicar o que já foi tentado fazer para resolver a situação.

Dúvidas não percetíveis ou com ausência dos elementos acima mencionados correm o risco de não serem respondidas.

Bibliografia

- Slides das aulas teórico-práticas de Programação Avançada
- Fichas das aulas práticas de Programação Avançada
- Páginas do manual eletrónico do Unix (utilitário `man`)
- Bibliografia recomendada para a UC