Publicação:	2021.09.27 (v2:2021.09.29)
Peso na nota final:	25% da nota final (mínimo de 5.0 em 20)
Informações:	Moodle
Data de entrega	**6 de novembro de 2021** **(data a aguardar confirmação do calendário de avaliação EI)**

Projeto 1S 2021-2022

`checkFile`

1 - Introdução

A extensão associada ao nome de um ficheiro traduz, usualmente, o tipo de ficheiro. Assim, é expectável que um ficheiro denominado "`a.png`" corresponda a um ficheiro do tipo PNG (Portable Network Graphic), e que um ficheiro de nome "`a.zip`" seja um arquivo ZIP. Contudo, nada impede que a extensão do nome de um ficheiro seja alterada, falseando a perceção do tipo do ficheiro baseada na extensão do nome do ficheiro. Por exemplo, é trivial alterar o nome do arquivo `a.zip` para `a.png`, dissimulando assim o arquivo `a.zip`.

No âmbito deste projeto, pretende-se que elabore, recorrendo à linguagem C, a aplicação `checkFile`, que consiste num verificador de ficheiros que determina o tipo de ficheiro e verifica se o mesmo corresponde ao anunciado pela extensão existente no nome do ficheiro.

Para determinar o tipo de ficheiro com base no respetivo conteúdo, o `checkFile` deve recorrer ao utilitário **`file`** do Linux (`man file`). Analisando o resultado devolvido pelo utilitário file, o `checkFile` determina se a extensão do nome do ficheiro é correta, e caso não seja, deve indicar quais são as extensões corretas.

2 - Funcionamento

O `checkFile` deve recorrer ao utilitário `file` (`man file`) de modo a determinar o tipo de conteúdo existente em cada ficheiro que deve analisar. Para o efeito deve fazer uso das primitivas `fork` e `exec` para execução do utilitário `file`, bem como capturar a saída padrão e a saída do canal de erro produzidas pela execução do utilitário `file`.

A aplicação `checkFile` deve ser capaz de detetar uma extensão errada nos seguintes tipos de ficheiros: PDF, GIF, JPG, PNG, MP4, ZIP, HTML. Obviamente, para determinar o verdadeiro tipo de um dado ficheiro, o `checkFile` deve sempre analisar o ficheiro. Assim, o `checkFile` analisa qualquer ficheiro que

lhe seja passado, sendo que no caso de ficheiros que não pertençam aos tipos acima indicados, o `checkFile` deve indicar que o tipo de ficheiro não é suportado (ver Exemplo 4).

3 - Sintaxe da linha de comandos e parâmetros

3.1 - Sintaxe da linha de comandos

O `checkFile` é uma aplicação de linha de comandos com a seguinte sintaxe de execução:

```
checkFile  [options] [filename]
```

3.2 - Parâmetros da linha de comandos

Opção	Explicação
`-f, --file <fich>`	Analisa o ficheiro `fich`, indicando se a extensão do nome do ficheiro corresponde ou não ao respetivo conteúdo. A aplicação deve ainda validar que `fich` corresponde ao nome de um ficheiro existente. O `checkFile` deve suportar múltiplas ocorrências da opção `-f/--file` na mesma linha de comando.
`-b, --batch <fich_with_filenames>`	O ficheiro `fich_with_filenames`, contém, em cada linha, o caminho/nome de um ficheiro cujo tipo se pretende validar com a extensão do respetivo nome. Assim, se `fich_with_filenames` tiver 20 linhas, o `checkFile` terá que processar cada um dos 20 ficheiros.
`-d,--dir <directory>`	Analisa cada um dos ficheiros existentes no diretório "`directory`", averiguando da validade da respetiva extensão. Apenas devem ser considerados os ficheiros que existem no diretório, ignorando eventuais subdiretórios.
`-h, --help`	Lista ajuda sucinta, incluindo o nome e número de estudante IPLeiria de cada autor e termina. Deve ainda ser mostrado a listagem dos tipos de ficheiros suportados pelo `checkFile`, nomeadamente: PDF, GIF, JPG, PNG, MP4, ZIP, HTML

3.3 - Tratamento de erros

A aplicação deve validar os parâmetros da linha de comando da seguinte forma:

a) Deve ser validada a existência do ficheiro indicado através do parâmetro `-f/--file <fich>`. Caso ocorra erro, a aplicação deve escrever, no canal de erro padrão, a seguinte mensagem, e terminar:

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

b) Na opção `-b/--batch <fich>`, cada ficheiro listado em `fich` deve ser validado de forma similar ao indicado na alínea a).

c) Deve ser validada a existência do diretório indicado através do parâmetro `-d/--dir <dir>`. Caso ocorra erro, a aplicação deve escrever, no canal de erro padrão, a mensagem que abaixo e terminar:

`ERROR: cannot open dir <dir> -- 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".

4 - Tratamento de signals

A aplicação `checkFile` deve reagir à receção dos *signals* abaixo indicados da seguinte forma:

SIGQUIT: mostrar a mensagem "`Captured SIGQUIT signal (sent by PID: XX). Use SIGINT to terminate application.`", em que XX corresponde ao PID do processo que enviou o *signal* à aplicação `checkFile`.

SIGUSR1: no modo BATCH (`-b/--batch`), a aplicação deve indicar na saída padrão que a) iniciou o processamento na data/hora (formato: YYYY.MM.DD_HHhMIN:SEG, e.g., 2021.09.17_19h37:15) e b) que se encontra a processar o ficheiro **nº XX/nome do ficheiro** da lista. Nos outros modos, o *signal* SIGUSR1 deve ser ignorado.

4.1 - 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-Template.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 aplicação.

1) Ficheiro `a.pdf` é um ficheiro do tipo PDF

```
./checkfile -f a.pdf
[OK] 'a.pdf': extension 'pdf' matches file type 'pdf'
```

2) Ficheiro `a.png` não é um ficheiro do tipo PNG, mas sim um ficheiro do tipo PDF

```
./checkfile -f a.png
[MISMATCH] 'a.png': extension is 'png', file type is 'pdf'
```

3) ./checkFile -f a.docx

```
[MISMATCH] 'a.docx': extension is 'docx', file type is 'pdf'
```

4) ./checkFile -f true.docx

```
[INFO] 'true.docx': type 'application/vnd.openxmlformats-
officedocument.wordprocessingml.document' is not supported by checkFile
```

5) ./checkFile -f fich1.png -f fich2.png

```
[OK] 'fich1.png': extension 'png' matches file type 'png'
[OK] 'fich2.png': extension 'png' matches file type 'png'
```

6) ./checkfile --batch list_of_files.txt

```
[INFO] analyzing files listed in ‘list_of_files.txt’
[OK] '/tmp/ok.png’: extension ‘png’ matches file type ‘png’
[OK] '/tmp/ok.pdf’: extension ‘pdf’ matches file type ‘pdf’
[MISMATCH] '/tmp/false.jpg': extension is ‘jpg’, file type is ‘png’
[ERROR] cannot open file ‘/tmp/not_a_file’ – No such file or directory
[SUMMARY] files analyzed:4; files OK:2; files MISMATCH:1; errors:1
```

7) ./checkfile -d /tmp/dirWithFiles

```
[INFO] analyzing files of directory '/tmp/dirWithFiles'
[MISMATCH] '/tmp/dirWithFiles/false.jpg': extension is 'jpg', file type is 'png'
[ERROR] cannot open file '/tmp/dirWithFiles/lock.dat' – permission denied
[OK] '/tmp/ok.gif': extension 'gif' matches file type 'gif'
[OK] '/tmp/ok.html': extension 'html' matches file type 'html'
[SUMMARY] files analyzed: 3; files OK: 2; Mismatch: 0; errors: 1
```

8) ./checkfile -f /tmp/fileDoesNotExist.gif

```
[ERROR] cannot open file '/tmp/fileDoesNotExist.gif' -- No such file or directory
```

9) ./checkfile --batch list_of_files.txt

```
[ERROR] cannot open file 'list_of_files.txt' -- No such file or directory
```

10) ./checkfile -d /tmp/NonExistingDir

```
[ERROR] cannot open dir '/tmp/NonExistingDir' --  No such file or directory
```

5 - 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%**

6 - Relatório

- O projeto deve ser acompanhado de um relatório com um máximo de **seis** 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_PA_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.

7 - 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-2021-2022.n1-n2**"[1] 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 nº4) deve ser entregue através do mecanismo de entrega disponibilizado no moodle da unidade curricular. Caso o trabalho seja realizado em grupo, a entrega deve ser realizada pelo elemento do grupo que tem o menor número de estudante IPLeiria. Em caso de dúvidas deve consultar os docentes. Apenas é permitida **uma** entrega do projeto.

6 - Não serão consideradas tentativas de entrega realizadas 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 solicitada 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:

- 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

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