Publicação:	2020.05.06 (última atualização: 2020.05.15/12h00)
Peso na nota final:	5 valores em 20
Informações:	Moodle
Data de entrega	**27 de maio de 2020 / 23h59'**

Sistemas Operativos

Projeto 2 - 2S 2019-2020

Programação em C – ambiente Linux

freqCounter

1 - Introdução

Como o nome sugere, um contador de ocorrências tem por objetivo contar o número de ocorrências de determinados elementos considerando um determinado critério de classificação. Por exemplo, um contador de ocorrência de octetos (*bytes*) para ficheiro recebe o nome de um qualquer ficheiro, e determina quantas vezes é que aparece o byte com o valor 0 (zero), 1, 2, 3, …, 255 no ficheiro. No final, o resultado é algo similar ao seguidamente indicado:

```
byte 000: 23
byte 001: 12
(…)
byte 255: 14
```

No âmbito deste projeto, pretende-se a implementação, em linguagem C, da aplicação **`freqCounter`** capaz de determinar a frequência de ocorrência de determinados elementos em ficheiros. Os elementos a considerar podem ser octetos (8 bits), duo--octetos (dois octetos) ou quad-octetos (quatro octetos). Por exemplo, no caso de se pretender a contagem por duo-octetos (par de octetos), a aplicação deve ler do ficheiro um bloco de dois octetos e classificá-lo consoante o respetivo valor, que estará necessariamente compreendido no intervalo [0, 65535], dado que os valores são sempre interpretados como inteiros **sem** sinal, qualquer que seja o modo (octeto, duo-octeto ou quad-octeto) . Assim, o resultado será algo similar ao seguidamente indicado:

```
byte 00000: 9
(…)
byte 00255: 11
(…)
byte 65535: 21
```

2 - Funcionamento

A aplicação `freqCounter` funciona somente a partir da linha de comando. As opções da linha de comando que deve suportar são as seguintes:

Opção	Explicação
`-c, --compact`	Processa os ficheiros indicados por -f/--file ou -d/--dir produzindo o resultado em modo compacto. Os valores são separados por vírgula e **sem** espaço. **Não** é compatível com as opções --discrete e -s/--search, pelo que se detetadas na mesma linha de comando, a aplicação deve terminar com apropriada mensagem de erro. Parâmetro opcional.
`-d, --dir <DIR>`	Processa todos os ficheiros regulares existentes no diretório DIR, **não** considerando eventuais subdiretórios. Parâmetro obrigatório (exceto se existir o parâmetro -f/--file).
`--discrete <value1,value2,…>`	Apenas deve ser mostrada a contagem dos valores indicados na lista de valores (value1,value2,…). Os valores são separados por vírgula e **sem** espaço. Parâmetro opcional.
`-f, --file <file1,file2,…>`	Processa os ficheiros indicados na lista (nomes separados por vírgula **sem** espaço, e.g. file1,file2,file3). Parâmetro obrigatório (exceto se existir o parâmetro -d/--dir).
`-h, --help`	Lista ajuda sucinta, o nome e número IPLeiria de cada autor e termina.
`-m, --mode<1 ou 2 ou 4>`	Contagem orientada ao octeto (-m 1), bi-octeto (-m 2) ou quad-octeto (-m 4). Se opção não for especificada, deve ser considerado `-m 1`. Parâmetro opcional.
`-o, --output <file>`	Resultados da aplicação devem ser escritos para o ficheiro `file.` Parâmetro opcional.
`-s, --search <PadraoEmHex>`	Indica todas as ocorrências do padrão de octetos *padraoEmHex*. O valor de *bytesEmHex* é especificado através de i) uma *string* iniciada por `0x` e ii) que representa entre 1 a 32 *bytes* em hexadecimal (e.g., `0x34` ou `0x4589` ou `0x19460602`). A aplicação deve reportar todas as ocorrências do padrão (ver exemplo). A aplicação deve recusar pesquisas de padrões que não tenham um número completo de bytes (e.g, 0x3 ou 0x213). A opção `-s/--search` apenas pode ser combinada com as opções `-f/--file`, `-o/--output`, `-d/--dir` e `--time`. Parâmetro opcional.
`--time`	Indica em segundos, com precisão até aos microssegundos, o tempo de execução da aplicação (e.g., `1.223344 seconds`). Parâmetro opcional.

2.1 - Exemplos

São apresentados de seguida alguns exemplos que ilustram o funcionamento da aplicação.

Exemplo 1

```
freqCounter –mode 1 -f input1.png,input2.png
```

```
freqCounter:'input.png':2345 bytes
byte 000:2
byte 001:12
(…)
byte 255:3
sum:2345
---------
freqCounter:'input2.png':4783 bytes
byte 000:18
byte 001:19
(…)
byte 255:25
sum:4783
```

NOTA: qualquer que seja o modo (-m 1 ou -m 2 ou -m 4), apenas devem ser mostrados valores cujo número de ocorrências seja superior a zero.

NOTA: as várias cores empregues no exemplo servem apenas para facilitar a compreensão do exemplo 2. A aplicação deve apresentar o resultado de forma tradicional.

NOTA: a linha *sum* corresponde ao somatório do número de ocorrências. No modo octeto (-m 1), o valor de *sum* deve corresponder ao tamanho do ficheiro.

Exemplo 2

```
freqCounter --mode 1 -f input1.png,input2.png --compact
```

```
input1.png:2345bytes:212…3:2345
input2.png:4783bytes:1819…25:4783
```

NOTA: com a opção "`-c/--compact`", os valores da contagem de octeto/bi-octeto/quad-octeto são escritos numa única linha, sem espaço entre si e terminado pelo somatório do número de ocorrência (sum). Este formato de saída permite uma rápida comparação de resultados.

Exemplo 3

```
freqCounter --mode 2 -f input1.png --discrete 0,23,61999
```

```
freqCounter:'input.png':2345 bytes
bi-byte 00000:1
bi-byte 00023:2
bi-byte 61999:0
```

NOTA: com a opção "`--discrete 0,23,61999`" apenas é mostrada a contagem referente aos valores `0, 23 e 61999`. O exemplo considera bi-octetos pois foi indicado a opção `--mode 2`.

NOTA: quando o tamanho do ficheiro não é múltiplo do tamanho unitário do modo (neste caso 2 octetos, dado ser `-m 2`), não são considerados os últimos octetos (neste caso, o último octeto do ficheiro).

Exemplo 4

```
freqCounter --mode 4 -f large.bin
```

```
freqCounter:'large.bin':178219 bytes
quad-byte 0000000000:1
quad-byte 0000000001:2
(…)
quad-byte 4294967295:1
```

NOTA: quando o tamanho do ficheiro não é múltiplo do tamanho unitário do modo (neste caso 4 octetos, dado ser `-m 4`), não são considerados os últimos octetos (neste caso, os últimos três octetos do ficheiro).

Exemplo 5

```
freqCounter -m 1 -d dir_with_files
```

```
DIR:'dir_with_files'
freqCounter:'file1.png':7123 bytes
byte 000:11
byte 001:21
(…)
byte 255:8
sum:7123
----------
freqCounter:'file2.txt':1223 bytes
byte 000:5
```

```
byte 003:2
(…)
byte 255:3
sum:1223
```

NOTA: a opção "`--dir DIR`" processa todos os ficheiros regulares do diretório DIR.

Exemplo 6

```
freqCounter -m 1 -f fileX.txt,fileY.png --time
```

```
freqCounter:'any_file.png':726217 bytes
byte 000:1122
(…)
byte 255:876
sum:726217

freqCounter:'fileY.png':217 bytes
byte 000:1
(…)
byte 255:3
sum:217
time: 0.0112312 seconds
```

NOTA: a opção "`--time`" mostra o tempo total de execução da aplicação em segundos com precisão até aos microssegundos.

Exemplo 7

```
freqCounter -m 1 -f ../in.txt --output /tmp/out.txt
```

```
INFO:output written to "/tmp/out.txt"
```

O conteúdo do ficheiro "/tmp/`out.txt`" é exatamente o que seria escrito na saída padrão (*stdout*). Com a opção -o/--output, deixa de ser escrito conteúdo na saída padrão.

```
freqCounter:'in.txt':25121 bytes
byte 000:12
(…)
byte 254:9
byte 255:3
sum:25121
```

Exemplo 8

```
freqCounter -m 1 -f 1.txt,../2.png,/tmp/3.zip
```

```
freqCounter:'1.txt':25 bytes
byte 000:1
(…)
byte 104:1
sum:25
-----
ERROR:'../2.png': CANNOT PROCESS FILE
-----
freqCounter:'/tmp/3.zip':1890 bytes
byte 000:121
(…)
byte 253:9
sum:1890
```

NOTA: quando ocorre um erro no acesso a um ficheiro (abertura falhou, leitura falhou, etc.), deve ser apresentado a mensagem ERROR acima indica. O processamento deve continuar.

Exemplo 9

```
freqCounter -s 0x504B -f Politecnico_png.zip[1]
```

```
freqCounter:looking for '0x504B' in 'Politecnico_png.zip'
#1: offset: 0x0
#2: offset: 0x57AF
#3: offset: 0xE461
#4: offset: 0xE4BB
#5: offset: 0xE515
```

NOTA: são mostrados os offsets (em formato hexadecimal) onde foi encontrada a sequência de bytes `0x504B`.

Exemplo 10

```
freqCounter -s 0x123 -f some_file.jpg
```

```
freqCounter:invalid value ‘0x123’ for -s/--search (needs to be an integer value of bytes)
```

NOTA: 0x123 é um parâmetro inválido, pois não representa um número completo de bytes. Se pretender 0x0123, o utilizador da aplicação tem mesmo que escrever o 0x0123.

Exemplo 11

```
freqCounter -s 123 -f some_file.jpg
```

```
freqCounter:invalid value ‘123’ for -s/--search (needs to be specified in HEX format)
```

NOTA: 123 não está apresentado em formato hexadecimal (deveria ser 0x0123).

2.2 - Tratamento de erros

A aplicação deve validar os parâmetros da linha de comando.

a) Por exemplo, o parâmetro `-m/--mode` apenas suporta os valores 1, 2 e 4. Um valor fora dessa gama leva ao reportar da situação e ao término da aplicação com a seguinte mensagem:

```
ERROR: invalid value ‘VALOR’ for -m/--mode.
```

b) Deve ser validado a existência do diretório indicado através do parâmetro -d/--dir <DIR>.

```
ERROR: cannot access directory ‘DIR’ em que DIR representa o nome do diretório
```

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

3 - Implementação

a) 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.

b) Não é permitida o uso da função *system* ou de qualquer chamada à *shell*.

c) Deve ser empregue o *template* `EmptyProject-Templatev3.05.zip` disponibilizado no moodle da UC

[1] Ficheiro disponível em: https://www.ipleiria.pt/normasgraficas/wp-content/uploads/sites/80/2016/07/Politecnico_png.zip

d) O projeto deve compilar, através da seguinte linha de comando: `make`

e) 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).

NOTA: o modo `-m 4 / --mode 4` pode abranger até 2^{32} elementos distintos, i.e., valores entre 0 e 4294967295. Deste modo, é expectável que a implementação do modo *quad-byte* possa ser distinta dos modos byte e bi-byte.

4 - 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%**

5 - 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 da linha de comando**:

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_proj2_freqCounter_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.

6 - Regras

1 - O trabalho será realizado **individualmente** ou em grupo (máximo de **dois** estudantes, que podem ser de turnos práticos distintos). Devem manter-se os grupos do projeto 1. Qualquer alteração à composição do grupo deve ser solicitada ao docente responsável da UC com email para patricio.domingues@ipleiria.pt e com o assunto iniciado por [EI_SO][Proj2] e a identificação de todos os estudantes envolvidos (nome + número de estudante). Os pedidos devem ser realizados até ao dia 20 de maio de 2020 e devem estar devidamente fundamentados. Pedidos de alteração de composição de grupo sem fundamentação apropriada ou fora do prazo não serão aceites.

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 "**SO.proj2_freqCounter_2019-2020.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_SO][Projeto_2]** (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:

9.2 - Deve incluir código que reproduz o problema:

- Incluir uma versão do código como texto (não como captura de ecrã) tão simples quanto possível,

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

mas que consegue reproduzir o problema;

- Eventualmente, colocar uma ou outra captura de ecrã para ilustrar o problema (falha na execução, *segmentation fault*, etc.);
- Sempre que possível enviar um ficheiro *main.c* contendo apenas o código que causa o problema e que pode ser compilado individualmente;
- Explicar com clareza e de forma sucinta o que está a acontecer e o que parece estar errado;
- 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 e vídeos das aulas teóricas de Sistemas Operativos
- Fichas das aulas práticas de Sistemas Operativos
- Páginas do manual eletrónico do Unix (utilitário `man`)
- Love, Robert. *Linux system programming: talking directly to the kernel and C library*. O'Reilly Media, Inc., 2013.
- Stevens, W. Richard; Rago, Stephen A. Advanced Programming in the UNIX Environment, Third Edition, 2013.