Relatório

Rodrigo Ferreira Pintassilgo (2191190) e por Daniel Pires Patrício (2191195) 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.

Rodrigo Pintassilgo

Daniel Patrício

- **Opção -c, --compact:**
  - ✓ Totalmente operacional.
  - ✓ Para a sua realização foram utilizadas as funções mode_1_compact, mode_2_compact. O código implementado em cada uma delas é bastante semelhante ao implementado no mode_1, mode_2.
  - ✓ As funções mode_1_compact e mode_2_compact diferem apenas no número de bytes que cada um possui.
  - ✓ Nas funções mode_1_compact e mode_2_compact usou-se a função strtok para dividir a string dos ficheiros por vírgulas. Esta foi fornecida pelo argumento args_info.file_arg. Abriu-se um ficheiro (token) de forma a conseguir ler os bytes. Colocou-se o ponteiro a apontar para o fim do ficheiro, de forma a conseguir calcular-se o seu tamanho (size_of_file). CHUNK_LEN utilizou-se para ler ficheiro em porções, e de seguida colocou-se novamente o ponteiro a apontar para o início do ficheiro. Através do fread leu-se os bytes do ficheiro, e com a utilização do vetor seen permitiu verificar quais os bytes que se repetiam e somá-los. Se o número de ocorrências fosse superior a 0, somava-se o número de ocorrências com a variável sum. No final, fechou-se o ficheiro aberto no início.
- **Opção -d, --dir <DIR>:**
  - ✓ Totalmente operacional.
  - ✓ As funções implementadas foram a dir, a concat_dir e a dir_mode.
  - ✓ A função dir abre a string do diretório, depois verificou-se se os ficheiros que estão dentro do diretório colocado são regulares. A função dir_mode assegura todas as opções com que o --dir é compatível. A função concat_dir permite concatenar a string, reservando memória, de forma a escreve-la. Retorna a string concatenada, para ser utilizada na função dir, de forma a poder servir de parâmetro de entrada para a dir_mode.
- **Opção --discrete <value1, value2,...>:**
  - ✓ Implementado, mas com problemas. O código dá printf de todos os ficheiros introduzidos pelo utilizador, tal como pretendido, mas apenas conta os bytes de forma correta no primeiro ficheiro. Nos restantes que sejam introduzidos, o programa apenas consegue ler o primeiro byte inserido.
  - ✓ Foram utilizadas as funções mode_1_discrete e mode_2_discrete.
  - ✓ As funções não têm VLA, ou seja, não estão feitas de forma correta, mas o programa funciona parcialmente.

- **Opção -f, --file <file1,file2,...>:**
  - ✓ Totalmente operacional.
  - ✓ No main.c alocou-se memória através de um malloc que permitiu guardar a string dada por args_info.file_arg. Assim, utilizou-se a função strtok para dividi-la em strings diferentes, utilizando a vírgula como delimitador, de forma a conseguir obter o nome individual de cada ficheiro (token). No final do ficheiro, libertou-se a memória alocada e colocou-se o ponteiro a NULL. O parâmetro --file é obrigatório, mas não é compatível com o parâmetro --dir, ou seja, caso sejam colocados ao mesmo tempo o programa irá mostrar a respetiva mensagem de erro.
- **Opção -h, --help:**
  - ✓ Totalmente implementado.
  - ✓ É um menu de ajuda sucinta, ou seja, deu-se printf de todas as opções de comandos, de forma a mostrar ajuda para cada uma das opções de comando.
- **Opção -m, --mode <1 ou 2 ou 4>:**
  - ✓ Totalmente operacionais os comandos --mode 1 e --mode 2, exceto o comando --mode 4, que não foi implementado.
  - ✓ Para a sua realização foram utilizadas as funções mode_1, mode_2.
  - ✓ Os mode_1 e mode_2 diferem apenas no número de bytes que cada um possui.
  - ✓ Nos mode_1 e mode_2 usou-se a função strtok para dividir a string dos ficheiros por vírgulas. Esta foi fornecida pelo argumento args_info.file_arg. Abriu-se um ficheiro (token) de forma a conseguir ler os bytes. Colocou-se o ponteiro a apontar para o fim do ficheiro, de forma a conseguir calcular-se o seu tamanho (size_of_file). CHUNK_LEN utilizou-se para ler ficheiro em porções, e de seguida colocou-se novamente o ponteiro a apontar para o início do ficheiro. Através do fread leu-se os bytes do ficheiro, e com a utilização do vetor seen permitiu verificar quais os bytes que se repetiam e somá-los. Se o número de ocorrências fosse superior a 0, somava-se o número de ocorrências com a variável sum. No final, fechou-se o ficheiro aberto no início.
- **Opção -o, --output <file>:**
  - ✓ Totalmente operacional.
  - ✓ No main.c, aloca-se memória para guardar a string do ficheiro em que vai ser escrito, de seguida copia-se para o ponteiro string_output e por fim abre-se um ficheiro para escrita e leitura, w+, (este é criado se não existir). Assim, através das funções passa-se como parâmetro de entrada o ponteiro do tipo FILE de

forma a que dentro de cada uma se possa fazer uma verificação. Se existir output (args_info.output_given) escreve no ficheiro, se não escreve no terminal.

- **Opção -s, --search <PadraoEmHex>:**
  - ✓ Não implementado.
- **Opção --time:**
  - ✓ Totalmente implementada.
  - ✓ Na função main.c, a função clock retorna o número de tiques do relógio desde o início do programa, em caso de falha, retorna -1. Assim, o tempo é calculado subtraindo o tempo final pelo tempo inicial e dividindo por CLOCKS_PER_SEC (está definido na biblioteca <time.h> e define quantos tiques do relógio é um segundo.