Tiago Júlio Santos – 2201755

Rodrigo Filipe Capitão - 2201741

Rodrigo Filipe Capitão (2201741) e por Tiago Júlio Santos (2201755) 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.

Funcionalidades

File:

- **Estado:** Totalmente Operacional
- **Implementação:** O ficheiro enviado através dos argumentos é aberto para leitura.
  Depois da verificação da existência do ficheiro é criado um fork() no qual é executado o comando "file *ficheiro* --mime" através do comando execlp. O output do comando execlp é guardado num ficheiro "resultado.txt" através da função dup2.
  O ficheiro "resultado.txt" é depois aberto para análise, com recurso a funções de manipulação de strings é extraído o tipo e extensão do ficheiro. Estes dois são comparados, caso sejam iguais é mostrado o resultado [OK], caso contrário é mostrado [MISMATCH], se o tipo de ficheiro não for suportado essa informação é também mostrada.

Batch:

- **Estado:** Totalmente Operacional
- **Implementação:** O ficheiro com o nome dos ficheiros a analisar enviado através dos argumentos é aberto para leitura. O ficheiro é lido linha a linha e os nomes dos ficheiros são guardados num vetor do tipo char. Cada ficheiro é depois sujeito a verificação como feito na opção File. Depois da verificação da existência do ficheiro o procedimento é idêntico ao do modo File. Com recurso a um ciclo for() são analisados todos os ficheiros guardados no vetor seguindo o mesmo processo da opção File.
  O resultado é também guardado e analisado de forma idêntica à opção File.
  Na opção batch são contados o total de ficheiros analisados, o número de ficheiros cujo resultado foi [OK] ou [MISMATCH], e o número de erros. Esta informação é mostrada no final da execução do programa.
  Foi usado uma estrutura de dados para guardar cada linha num índice diferente.

Directory:

- **Estado:** Totalmente Operacional
- **Implementação:** Com recurso a estruturas e funções da library "dirent.h" é aberta a diretoria e os ficheiros são analisados. Com recurso a funções de manipulação de strings os conteúdos da diretoria são analisados, separando subdiretorias de ficheiros. Os nomes dos ficheiros são guardados num vetor de forma idêntica à opção Batch.
  Depois de guardados no vetor, os ficheiros são analisados da mesma forma que as duas opções anteriores e os resultados guardados e mostrados também de forma igual.
  Na opção Directory são também contados o total de ficheiros analisados, o número de ficheiros cujo resultado foi [OK] ou [MISMATCH], e o número de erros, sendo esta informação mostrada no final da execução do programa.
  Foi usado uma estrutura de dados para guardar cada ficheiro lido no directory num índice diferente.

Help:

- **Estado:** Totalmente Operacional
- **Implementação:** A opção help apresenta no ecrã o nome dos autores do programa, as diferentes opções para execução do programa e o tipo de ficheiros suportado pelo mesmo.

Signals:

- **Estado:** Implementado, mas com problemas
  O programa está preparado para receber dois sinais, o SIGUSR1 (apenas na opção Batch) e o SIGQUIT.
  Quando é enviado o sinal SIGQUIT o processo é terminado e é mostrada a mensagem "Captured SIGQUIT signal (sent by PID: XX). Use SIGINT to terminate application.", o utilizador poderá depois enviar o sinal SIGINT para terminar o programa.
  Na opção Batch, quando recebido o sinal SIGUSR1 é apresentada a data e hora de início do processamento dos ficheiros.

- **Problemas:** No SIGQUIT, devido à forma como foi feito o fork() dentro de um loop, quando é enviado o sinal SIGQUIT, o print da mensagem fica em loop infinito e só termina quando o utilizador envia o sinal SIGINT, e no

sinal SIGUSR1, não foi implementada a mensagem que mostra o ficheiro atual a ser processado.