INSTITUTO POLITÉCNICO DE lEIRIA

Engenharia Informática – Programação Avançada

2.º Ano – 1.º Semestre

PROJETO PRÁTICO

---

paCodec Tool – Etapa 2

Paulo Penicheiro nº2130628 e por Ruben Bernardo 2130664 declaram sob compromisso de honra que o presente trabalho (código, relatórios e afins) foi integralmente realizado por nós, sendo que as contribuições externas se encontra claramente e inequivocamente identificadas no relatório e no próprio código fonte. Mais se declara que os estudantes acima identificados não disponibilizaram o código ou partes dele a terceiros.

PAULO PENICHEIRO

N 2130628

ENHENHARIA INFORMATICA

Prof. PATRÍCIO DOMINGUES

LEIRIA, NOVEMBRO 2015

RUBEN BERNARDO

N º 2130664

ÍNDICE

1 Introdução

O relatório descreve as atividades realizadas no âmbito da execução do trabalho prático do curso de Engenharia Informática na disciplina de Programação Avançada – 2.ª Etapa.

Consiste, na implementação de um utilitário em modo consola chamado paCodec cujo objetivo é permitir efetuar a Codificação/descodificação com perda, de imagens do tipo .pgm.

Nas páginas seguintes é possível encontrar as opções tomadas na criação do programa, os objetivos cumpridos e não cumpridos nesta 2ª Etapa do trabalho.

2 Funções Implementadas

Funcionalidades comuns às funções

Uma vez que optámos usar a versão base da etapa 1 fornecida pelos professores, mantivemos todas as funções do utilitário paCodec incluídas no ficheiro *utils.c,* estando os respetivos protótipos declarados no ficheiro *utils.h.*
Passamos a descrever essas mesmas funções:

Função about

Esta função é desencadeada caso seja dado o argumento *--about* (forma longa), e mostra a informação dos alunos que desenvolveram o utilitário paCodec.

Função install_signal_handler

Esta função tem como finalidade analisar um sinal de sistema ativado pelo utilizador e atribuir um valor à respetiva *flag*. No caso desta primeira etapa apenas será avaliado o sinal SIGINT e caso este tenha sido ativado a *flag sigint_flag* ficará com o valor a "1".
Esta *flag* trata-se de uma varável global irá permitir a qualquer função verificar a ativação deste sinal.

Função process_signal

Esta função tem como objetivo processar o sinal ativado pelo utilizador e agir em conformidade fechando também todos os ficheiros abertos e informando o utilizador que a aplicação foi terminada.

Função encodePGM

Esta função é desencadeada caso seja dado o argumento *--encode* e é responsável pela codificação de ficheiros de imagem do tipo .pgm transformando-as num ficheiro do tipo .cod, com perdas de qualidade.

A função recebe a estrutura do ficheiro .pgm e a estrutura do respetivo dicionário, mais o nome do ficheiro original assim como o nome do respetivo dicionário.

Como algoritmo de descodificação, analisa cada valor do ficheiro .cod, atribuindo à matriz o valor de acordo com o índice do dicionário, preenchendo também o cabeçalho da estrutura do ficheiro descodificado, codifica e escreve a imagem num ficheiro codificado do tipo .cod utilizando as funções (*write_z2_cod_to_file e write_z5_cod_to_file).*

Função parallelEncode

Esta função é desencadeada caso seja dado o argumento *–paralel-encode* e é responsável pela codificação em paralelo recorrendo às threads, de ficheiros de imagem do tipo .pgm transformando-as num ficheiro do tipo .cod, com perdas de qualidade.

A função recebe a estrutura do ficheiro .pgm e a estrutura do respetivo dicionário, mais o nome do ficheiro original e o número de threads a utilizar.

Como algoritmo utiliza o número de threads dado pelo utilizador, caso contrário utiliza o número de threads do processador, onde cada *thread* processa um bloco da imagem (*função process_block*), preenchendo o cabeçalho da imagem e escrevendo o ficheiro codificado de acordo com o seu tipo, utilizando as funções (*write_z2_cod_to_file e write_z5_cod_to_file)*

Função quadError

Esta função vai calcular o erro quadrático da imagem, onde para cada bloco da imagem de entrada é efetuada uma pesquisa no dicionário de padrões por forma a ser encontrado o bloco do dicionário que mais se assemelha ao bloco da imagem de entrada em processamento.

O grau de semelhança entre um bloco da imagem e um bloco do dicionário é avaliado através do erro quadrático que corresponde à soma do quadrado das diferenças entre as intensidades dos pixéis do bloco de imagem em processamento e as intensidades dos pixéis correspondentes do bloco de dicionário em avaliação.

Esta função retorna o índice do bloco com menor distorção.

Função encodeBlockimgX

A função recebe o índice do bloco da imagem a codificar, a estrutura da imagem pgm, a estrutura do dicionário e o número de blocos da imagem por linha.

Como algoritmo, com base no índice do bloco calcula as coordenadas x,y do pixel de partida, e de seguida chama a função quadError que calcula e devolve o índice do bloco com menor distorção.

Função process_block

A função recebe um ponteiro genérico com o parâmetro da função.

Como algoritmo esta função chama a função getNewTask, para obter a tarefa (bloco) a codificar, de seguida chama função encodingBlockimgX descrita mais acima, com o intuito de receber o bloco do dicionário com menor distorção, e assim vai construindo a estrutura do ficheiro codificado.

Função getNewTask

A função recebe um ponteiro para a estrutura com os parâmetros a utilizar nas threads.

Como algoritmo é definido um bloco de código para evitar a concorrência do usando um *mutex (pthread_mutex_lock e pthread_mutex_unlock)*.

No final devolve a identificação da tarefa a ser executada.

Função build_cod

A função recebe um ponteiro para a estrutura do ficheiro codificado .cod, a estrutura da imagem pgm, a estrutura do dicionário e o nome do ficheiro original (pgm).
Como algoritmo esta função constrói o ficheiro codificado, preenchendo o cabeçalho e calculando o índice máximo do dicionário a ser utilizado.

Função Write_Z2_cod_to_file

A função recebe a estrutura do ficheiro codificado .cod, e o nome do ficheiro original (pgm).
Como algoritmo escreve o ficheiro codificado do tipo Z2(texto) para o diretório do ficheiro original.

Função validate_dic_pgm

A função recebe a estrutura do ficheiro original e a estrutura do dicionário
Como algoritmo verifica se a largura e altura da imagem original é múltipla da largura e altura do dicionário, retornando o valor -1 se o dicionário for inválido para a imagem dada, e retorna o valor 0 caso o dicionário seja válido para a imagem dada.

Função Write_Z5_cod_to_file

A função recebe a estrutura do ficheiro codificado .cod, e o nome do ficheiro original (pgm).
Como algoritmo escreve o ficheiro codificado do tipo Z5(binário) para o diretório do ficheiro original, byte a byte caso o índice do dicionário seja menor que 256 ou dois em dois bytes caso o índice do dicionário seja maior ou igual a 256.

Funções Principais – ficheiro main.c

Função main

A função *main* consiste na função principal que irá invocar as principais funcionalidades da aplicação, nomeadamente a conversão dos ficheiros de imagem, conforme enunciado apresentados pelos professores.

Esta função trata o *parsing* dos parâmetros passados pela linha de comandos.

Este processamento dos parâmetros do utilizador será realizado por intermédio da biblioteca <*cmdline.h*> criada pelo ficheiro de configuração *gengetopt pagegengetopt.ggo*.

Esta função é também responsável pela contabilização do tempo despendido em cada operação, recorrendo ao uso da biblioteca <*time.h*> e da função *clock()*.

3 - Conclusão

Este relatório tem como objetivo principal a documentação do código realizado e implementado demonstrando quais das funcionalidades do paCodec estão funcionais e a validação dos argumentos permitidos e/ou obrigatórios solicitados no enunciado.

Os exercícios realizados nas aulas foram fundamentais para a resolução desta segunda etapa do projeto, tornando cada vez mais aliciante para alcançar todos os objetivos propostos.

Todos os objetivos foram atingidos para as várias funcionalidades, exigidas no enunciado do projeto, como se pode comprovar pelos testes fornecidos pelos professores.

No entanto, entendemos que a solução atual apresentada, possa não ser a mais correta, uma vez que nesta área, várias opções possam ser tomadas para o mesmo fim, reiteramos mais uma vez que o projeto envolveu um algum grau de complexidade e a nossa pouca experiência na linguagem C tornou o desenvolvimento desta etapa uma tarefa árdua de concretizar.

As principais dificuldades que encontrámos para a realização desta etapa consistiram na programação do algoritmo da codificação das imagens em formato .pgm, tanto para que o algoritmo codificasse bem as imagens mas também que ao ser paralelizado em threads pudesse continuar a atingir os objetivos a que se propôs.

A elaboração desta etapa contribuiu para melhorar os nossos conhecimentos como estudantes e programadores, concluindo que a linguagem de programação C é bastante versátil e focalizada na programação procedimental.

4 - Objetivos

Objetivos Atingidos

- Funcionalidade de codificar imagens em formato *.cod* do tipo Z2 a partir de ficheiros do tipo .pgm, implementada com sucesso.
- Funcionalidade de codificar imagens em formato *.cod* do tipo Z5 a partir de ficheiros do tipo .pgm, implementada com sucesso.
- Funcionalidade de codificar imagens em paralelo (usando threads) de ficheiros de formato *.cod* do tipo Z2 implementada com sucesso.
- Funcionalidade de codificar imagens em paralelo (usando threads) de ficheiros de formato *.cod* do tipo Z5 implementada com sucesso.
- Criação e gravação do ficheiro Z2 codificado *.cod* para a diretoria do ficheiro original *.pgm* , implementada com sucesso.
- Criação e gravação do ficheiro Z5 codificado *.cod* para a diretoria do ficheiro original *.pgm* , implementada com sucesso.

Objetivos Não Atingidos

- Nesta etapa todos os objetivos foram concluídos com sucesso

5 - Bibliografia

As referências bibliográficas de seguida, foram de especial auxílio para a criação das caraterísticas do descodificador de imagens necessárias á 1ª etapa.

- Apresentações das aulas teóricas/práticas de Programação Avançada
- Exercícios práticos realizados nas Aulas práticas
- Exercícios/Fichas práticas das aulas de Programação Avançada
- Comando *man* (manual de instruções) usado nas funções passadas na linha de comandos de *Linux*
- *Livro C por "Luís Damas*