Tarefa: criação de threads

Grupo: Rafael Augusto de Oliveira Guedes (20210097627)
Sthefania Fernandes Silva (20210072430)

A tarefa consiste em criar um programa que utiliza *threads* para processar uma imagem em nível de cinza e calcular as imagens de borda nas direções x e y.

Para a criação das *threads*, o código foi desenvolvido na linguagem de programação C++, utilizando três bibliotecas: **pthreads**, **stdio** e **string**. A **Pthreads**, é uma biblioteca para programação concorrente (*multithreading*), ou seja, uma função será executada concorrentemente com as outras pthreads/processos que existirem no sistema operacional. A biblioteca **stdio** foi usada para ser feita escrita e leitura dos arquivos PGM, enquanto a *string* para a manipulação de *strings*.

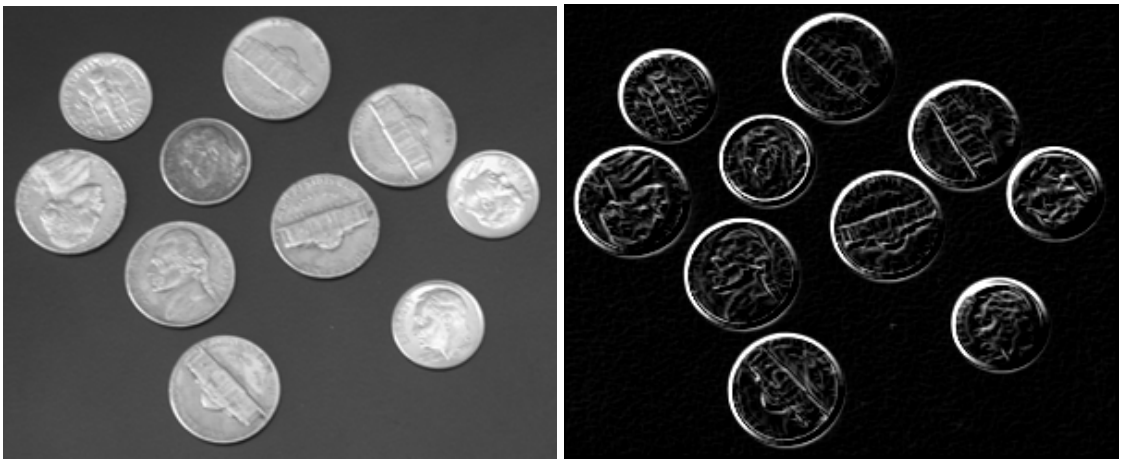
Para assegurar que as *threads* tenham acesso aos *arrays* associados às imagens, foram definidos 4 *arrays* globais: image, Gx, Gy, G. O tamanho dos *arrays* é igual ao tamanho da imagem de entrada fornecida para a tarefa, intitulada “coins.ascii.PGM”.

Na função principal (*main*) começamos lendo o arquivo PGM. Dado o formato do arquivo, primeiro realizamos a leitura do tipo de PGM, que é indicado como binário e marcado como "P2". Na linha seguinte, a imagem contém um comentário que deve ser ignorado. Logo após o comentário, encontramos informações cruciais: o número de colunas e linhas e o valor máximo do pixel. Após essas informações, os pixels que compõem a imagem são listados, e todos eles são armazenados no *array* nomeado image. Vale ressaltar que no arquivo PGM original, existia um espaçamento a mais entre os valores que indicavam o número de linhas e o de colunas, então foi necessário apagar esse espaçamento extra para que os valores corretos fossem pegos.

Com todas as informações da imagem guardadas, duas *threads* são criadas: uma para calcular a imagem de borda na direção x (Gx) e outra na direção y (Gy). Após a conclusão das tarefas das *threads* filhas, a *thread* principal lê as imagens resultantes (arrays Gx e Gy) e combina essas informações para calcular a imagem de saída G. Foi preciso sincronizar a *thread* principal com as *threads* filhas usando funções de junção (join) para garantir que o cálculo seja concluído corretamente e que não haja conflitos de acesso aos dados compartilhados.

Com a imagem de saída gerada, um novo arquivo foi criado para armazenar a imagem resultante em formato PGM. Abaixo, temos a imagem original e a imagem resultante após o processamento.

Figura 1 - a) Imagem original b) Imagem processada.



a) b)

O programa foi feito em computadores com o sistema operacional Linux, dessa forma, a biblioteca de *threads* escolhida para realizar a atividade foi a **pthreads**. Essa biblioteca é um padrão conhecido como **POSIX threads** criada pelo IEEE que possui diversas funções para criar e gerenciar *threads*, permitindo realizar compartilhamento de recursos e sincronização de operações. Para se trabalhar com essa biblioteca, não houve muitas complicações pois foi seguido exemplos mostrados em sala de aula e disponibilizados para a turma.

A seguir, o código completo utilizado para o processamento da imagem.

```
#include <pthread.h>
#include <stdio.h>
#include <string.h>

#define linha 246
#define coluna 300
int image[linha][coluna] = {};
int Gx[linha][coluna]={}, Gy[linha][coluna]={}, G[linha][coluna]={};

void *GdeX(void *arg){
    int i, j;
    for (i=1; i<linha-2; i++){
            for (j=1; j<coluna-2; j++){
                //Cálculo de Gx*/
                Gx[i][j] = (image[i+1][j-1] + image[i+1][j] +
image[i+1][j+1]) - (image[i-1][j-1] + image[i-1][j] + image[i-1][j+1]);

                //Saturando*/
                if(Gx[i][j] < 0){
                    Gx[i][j] = 0;
```

```
                }
                if(Gx[i][j] > 255){
                    Gx[i][j] = 255;
                }
            }
        }
}

void *GdeY(void *arg){
    int i, j;
    for (i=1; i<linha-2; i++){
            for (j=1; j<coluna-2; j++){
                //Cálculo de Gy*/
                Gy[i][j] = (image[i-1][j+1] + image[i][j+1] +
image[i+1][j+1]) - (image[i-1][j-1] + image[i][j-1] + image[i+1][j-1]);

                //Saturando*/
                if(Gy[i][j] < 0){
                    Gy[i][j] = 0;
                }
                if(Gy[i][j] > 255){
                    Gy[i][j] = 255;
                }
            }
        }
}

int main() {
    FILE *file = fopen("coins.ascii.pgm", "r");
    if (!file) {
        printf("Erro ao abrir o arquivo da imagem.\n");
        return 1;
    }

    // Lê o cabeçalho do arquivo PGM
    char type[3], line[1024];
    int colunas, linhas, maxval;

    fscanf(file, "%s\n", type);
    fscanf(file, "%[^\n]\n", line);
    fscanf(file, "%d %d %d", &colunas, &linhas, &maxval);

    if (strcmp(type, "P2") != 0) {
        printf("O arquivo não está no formato PGM P2.\n");
        fclose(file);
        return 1;
```

```
    }

    // Lê os valores dos pixels da imagem
    for (int i = 0; i < linhas; i++) {
        for (int j = 0; j < colunas; j++) {
            fscanf(file, "%d", &image[i][j]);
        }
    }

    fclose(file);

    int num_threads = 2; // Número de threads
    pthread_t threads[num_threads];

    for (int i = 0; i < num_threads; i++) {
        if(i==0)
        {
            pthread_create(&threads[i], NULL, GdeX, NULL);
        }
        else
        {
            pthread_create(&threads[i], NULL, GdeY, NULL);
        }
    }

    for (int i = 0; i < num_threads; i++) {
        pthread_join(threads[i], NULL);
    }

    // Agora Gx e Gy contêm os resultados desejados

    printf("Threads terminaram.\n");

    /* Gerando imagem de saída... */
    for (int i = 1; i < linhas - 1; i++) {
        for (int j = 1; j < colunas - 1; j++) {
            G[i][j] = Gx[i][j] + Gy[i][j];
            if (G[i][j] > 255) {
                G[i][j] = 255;
            }
        }
    }

    // Exibe a imagem de saída (você pode salvar em um arquivo PGM se
desejar)
    printf("P2\n%d %d\n%d\n", colunas, linhas, maxval);
```

```
    // Abre um novo arquivo para escrita
    FILE *output_file = fopen("imagem_saida.pgm", "w");

    if (!output_file) {
        printf("Erro ao abrir o arquivo de saída.\n");
        return 1;
    }

    // Escreve o cabeçalho PGM no arquivo
    fprintf(output_file, "P2\n%d %d\n%d\n", colunas, linhas, maxval);

    // Escreve os valores dos pixels da imagem resultante no arquivo
    for (int i = 0; i < linhas; i++) {
        for (int j = 0; j < colunas; j++) {
            fprintf(output_file, "%d ", G[i][j]);
        }
        fprintf(output_file, "\n");
    }

    // Fecha o arquivo de saída
    fclose(output_file);

    printf("Imagem resultante salva em imagem_saida.pgm\n");

    return 0;
}
```

Referências Consultadas

SILBERSCHATZ, Abraham. **Operating System Concepts**. 8. ed. John Wiley & Sons, 2008.

GARCIA, Fernando Deluno. **Threads POSIX**. 2017. Disponível em: https://embarcados.com.br/threads-posix/. Acesso em: 17 out. 2023.