Manipulador de imagens PPM - com fork

Introdução

No semestre de 2015.2 os alunos Nalbert Gabriel Melo Leal e Juliana Barbosa implementaram um manipulador de imagens PPM como trabalho de comclusão das disciplinas PTP e ITP do IMD/UFRN. Na epoca foi perceptivel que mesmo escrito na linguagem C a manipulação das imagens com uma largura e altura grandes, como por exemplo imagens HD, não eram tão rapidas podendo levar alguns segundos, por conta disso no semestre de 2016.2, um ano após a finalização das disciplinas PTP e ITP foi decidido pela dupla que o projeto deveria ser refeito para que a aplicação de filtros em imagens PPM sejam mais rapidos e eficientes.

O projeto

O manipulador de imagens PPM é focado em manipular as carracteristicas de uma imagem do formato PPM com o uso do fork() na aplicação de filtros. Funcionalidades como rotacionar ou dar zoom em imagens ppm não apresentaram problemas quando colocadas para rodar em imagens PPM de grande dimensão como HD, entretanto as funcionalidades de aplicação de filtros apresentave grande perca de performance quando utilizado em imagens HD ou superiores, sendo assim foi definido pelo grupo que seria uma boa ideia fazer uma implementação do programa de manipulação de imagens PPM para que funcionasse com o uso de mais de um processo, assim averia um ganho significativo na velocidade em que o computador computa os dados da imagem modificada.

A implementação

Na implementação do codigo a dupla passou por 3 linguagens de programação (C++, python e C), decidindo no final pela linguagem C. Inicialmente o plano era de que 4 processo simultaneos funconassem, entretanto ouve problemas como controle e comunicação entre os processos. Por conta disso decidimos que averia apenas um filho trabalhando en conjunto com um pai, isso não aumentaria a performance como desejado, entratanto já é um começo. O processo filho é criado e faz as modificações na parte superior da imagem, enquanto isso o pai faz as modificaçções na parte inferior da imagem. O pai espera o filho terminar de escrever no arquivo a parte que computou e de usar a função kill(getpid(), SIGKILL) para poder escrever sua propria parte, como pode ser notado existe uma

organização facilmente estabelecida que impede que os dados da imagem sejam escritos de forma errada no arquivo final.

Como compilar

Para compilar o programa basta usar o makefile com o seguinte comando:

```
make
```

Se quiser compilar sem o make basta usar o GCC:

```
gcc main.c modificacao.c struct.h lerEscreverPPM.c -o mai
n
```

Comisso basta executar o executavel ./main gerado.

Testes realizados

Os testes realizados foram feitos executando as funcionalidades na imagem de testes padrão “lena.ppm” que pode ser emcontrada no diretorio do projeto “test”

Posiveis bugs

Os bugs conhecidos são:

- Quando se executa a funcionalidade “binarização” que binariza a imagem por algum motivo as vezes o programa não consegue executar outra funcionalidade sendo necessario reiniciar o programa para executar outra funcionalidade.
- Quando se executa a funcionalidade “negativo” que aplica o filtro negativo na imagem a metate superior da imagem fica com o filtro e a metade inferior fica com um negativo de cor marrom

Sobre a dupla

- **Aluno**: Nalbert Gabriel Melo Leal;
- **GitHub**: github.com/nalbertg ;
- **Bitbucket**: bitbucket.org/nalbertg/gremlins
- **email pessoal**: nalbertrn@yahoo.com.br;
- **email acadêmico/profissional**: nalbertg@outlook.com;
- **Aluna**: Juliana Barbosa;
- **GitHub**: github.com/julianaabs ;