6º Exercício Prático
Trabalho para o Recesso de Final de Ano

Objetivo

O objetivo desta atividade é utilizar técnicas já aplicadas anteriormente nesta disciplina para otimizar um programa que extrai textos ocultos de uma imagem.

Materiais

1. Compilador GCC
2. Arquivos Esteganografia_V1.cpp e Esteganografia_V2.cpp
3. Arquivo com textos: LapisComTexto.ppm.

Desenvolvimento

Esteganografia é o termo usado quando se esconde textos em imagens.
As imagens coloridas, que tem pixels com 256 níveis em cada canal RGB, podem ter os bits menos significativos (LSB) alterados aleatoriamente sem afetar significativamente a imagem ao olhos dos seres humanos. Estes bits podem ser alterados para carregar outras informações em uma imagem. No caso do experimento de hoje, esses bits compõem textos previamente salvos na imagem LapisComTexto.ppm. Esta imagem tem três textos inseridos, um texto em cada canal. Os textos têm caracteres de 8 bits cada e são formados pelo bit menos significativo de 8 pixels consecutivos do respectivo canal para formar uma letra. Isso se repete até o primeiro caractere *null*. Na primeira linha da imagem, encontra-se o nome do arquivo e nas linhas seguintes o texto do codificado bit a bit. Estudo o código dos programas Esteganografia_V1.cpp e Esteganografia_V2.cpp para entender melhor.
No experimento de hoje, o aluno deve aplicar as técnicas de otimização que foram experimentadas nas aulas anteriores para reduzir o tempo de execução do programa Esteganografia_V1.cpp ou Esteganografia_V2.cpp.

Entendendo Operadores de bits

Antes de iniciar as otimizações é necessário entender alguns operadores que nos permitem testar, ligar ou desligar um bit em um inteiro.

Faça uma pesquisa na Internet sobre os operadores de bits da linguagem C e C++. **Explique o que os operadores “|”, “&”, “<<”, “>>” e “~” fazem.** Note que o operador “|” é diferente do operador “||” e o operador “&” é diferente do operador “&&”.

Abra o arquivo Esteganografia_V1.cpp em um editor e procure as funções ligaBit(), desligaBit() e LSBPixelEstahLigado(). **Explique o que as expressões “c |= 1 << nBit;”, “c &= ~(1 << nBit);” e “return ((p.r & 1) != 0);” fazem.** Use valores para exemplificar os funcionamentos dos operadores de bits.

Executando o programa sem otimização

Dois programas sem otimização que extraem os textos da imagem estão disponíveis. O Esteganografia_V2.cpp tem uma forma ligeiramente diferente que pode dar uma dica de um tipo de otimização aplicar.

1. Compile o programa Esteganografia_V1.cpp
    - gcc Esteganografia_V1.cpp -o Esteganografia_V1
2. Execute o programa.
    - ./Esteganografia_V1
3. Observe a saída do programa Esteganografia_V1 e responda:

   Quais arquivos foram gerados?

   Nome dos arquivos: _____, _____, _____

   Os textos contidos nos arquivos estão legíveis?

   ( ) Sim ( ) Não

   Qual foi o tempo observado na execução[1]?

   Tempo 1: _____ s

Aplicando as técnicas de otimização

Tente usar as técnicas de otimização vista até agora, uma de cada vez, mas juntando com as otimizações anteriores a cada passo que você tiver sucesso. Use sempre o Valgrind para identificar os *hotspots* e "ataque" primeiro os pontos de maior potencial de otimização. Sempre que uma das técnicas trouxer bons resultados, indique-os separadamente. Não esqueça de verificar sempre se os arquivos gerados estão corretos.

Obs.: o código de abertura da imagem, de extração e gravação dos textos estão repetidos 10 vezes apenas para que os valores de tempos medidos fiquem mais estáveis. Baixar de 10 para 1 essa repetição não é uma otimização e não deve ser feita.

Algumas perguntas pela ordem de importância para orientar suas otimizações:

1. Existe alguma função das bibliotecas do compilador que está gastando muito tempo de execução do programa? Qual é a rotina e qual a porcentagem do tempo ela gasta? Tem como substituir a chamada dessa função para fazer de uma forma mais otimizada?
2. Existe alguma função codificada no programa que está gastando muito tempo?
3. É possível desenrolar algum código?
4. Será que mudar o conceito de testagem de separação dos canais feitos para cada pixel para uma rotina específica para cada canal pode gerar algum resultado importante? Imagine substituir no programa principal o código:

1 **Todos os tempos deste trabalho devem ser medidos sem o Valgring**, mas o Valgrind pode ser útil para identificar os *hotspots* a otimizar.

```
for (j = 0; j < 10; j++) {
  lePPm("LapisComTexto.ppm", imagem);
  processa(imagem, 'R');
  processa(imagem, 'G');
  processa(imagem, 'B');
}
```

pelo código:

```
for (j = 0; j < 10; j++) {
  lePPm("LapisComTexto.ppm", imagem);
  processaR(imagem);
  processaG(imagem);
  processaB(imagem);
}
```

Note que esta substituição implicará que a rotina processa() seja refeita três vezes uma para cada canal RGB. Note ainda que se a função LSBPixelEstahLigado() continuar sendo usada da mesma forma nada mudará.

Avaliando os resultados

Envie a avaliação dos resultados como descrito no arquivo “Avaliacao Dos Resultados.pdf”.

Conclusão

Este trabalho oferece a oportunidade do aluno experimentar a aplicação das técnicas de otimização de uma forma mais autônoma.