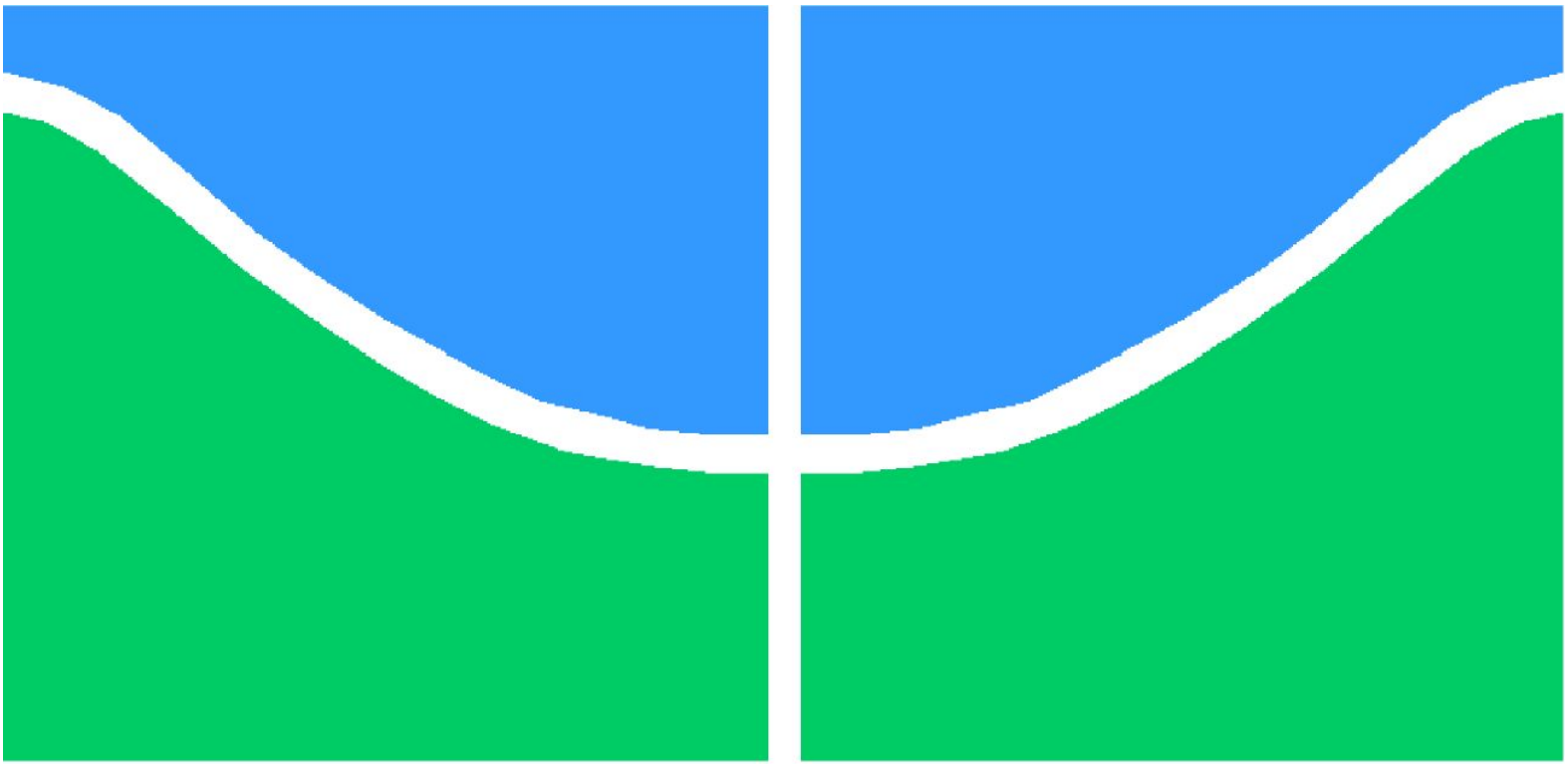


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Fundamentos de sistemas operacionais

Relatório do Trabalho 3

Brasília, DF
01 de Junho de 2017

LUCAS VILELA TAVEIRA BRILHANTE

FUNDAMENTOS DE SISTEMAS OPERACIONAIS | TRABALHO 04

Trabalho elaborado para disciplina de

Fundamentos de sistemas operacionais do curso de

Engenharia de Software da

Universidade de Brasília campus Gama.

Orientador: Prof. Thiago

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA

Brasília, DF
01 de Junho de 2017

1. Ambiente

1.1. Sistema operacional

O sistema operacional utilizado foi a distribuição Linux Mint 11, derivado do ubuntu.

1.2. Desenvolvimento

Para o desenvolvimento foi utilizado C, em conjunto com o compilador GCC. Para escrever as linhas de código foi usado o sublime editor, makefile para facilitar instalação.

Foi usado também o git para controle de versão, para poder continuar o desenvolvimento de qualquer computador.

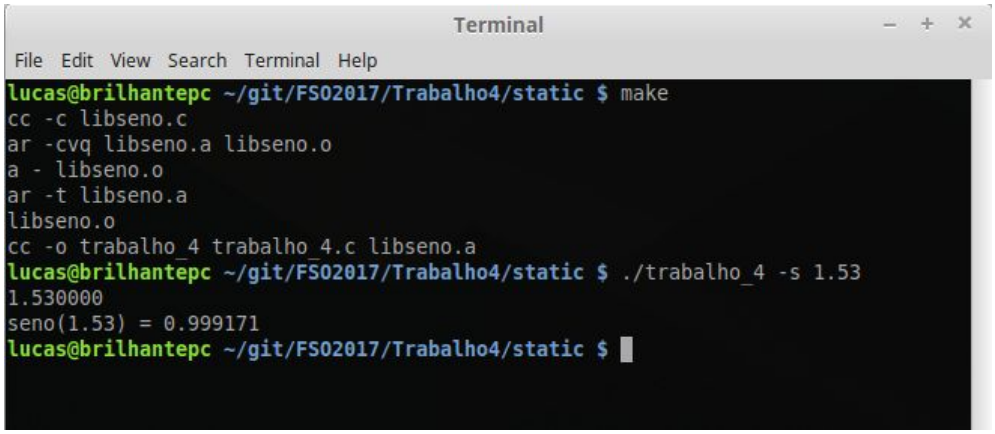
2. O programa

2.1. Dependencias

- **libs**
  #include <stdio.h>
  #include <stdlib.h>
  #include <dlfcn.h>
- **Makefile**

2.2. Telas

Após a execução, dado os parâmetros corretos, a seguinte tela irá aparecer:



Brasília, DF
01 de Junho de 2017

2.3. Compilação

2.3.1. MakeFiles

2.3.1.1. Estático

Os comandos dentro do makefile que fazem a compilação e o link são:

cc -c libseno.c << Compila .c da biblioteca
ar -cvq libseno.a libseno.o << transforma .o da biblioteca em .a
ar -t libseno.a
cc -o trabalho_4 trabalho_4.c libseno.a << Linka lib com executável

2.3.1.2. Dinámico

gcc -c -fpic libseno.c << Compila .c da biblioteca
gcc -shared -o libseno.so libseno.o << gera .so da biblioteca
gcc -I. -L. -Wl,-rpath=. -o trabalho_4 trabalho_4.c -lseno << Link dinamico com biblioteca e gera executável.

2.3.1.3. Dinamicamente carregado

gcc -c -fpic libseno.c << Compila .c da biblioteca
gcc -shared -o libseno.so libseno.o << Gera .so da biblioteca
gcc -I. -L. -rdynamic -o trabalho_4 trabalho_4.c -ldl << Link que força aplicação a dar load na biblioteca em tempo de execução. Gera executável.

2.3.2. Compilando

Para compilar e executar a aplicação deve-se rodar no terminal:

make
./trabalho_4 -a 1.53

Caso já queira compilar novamente deve-se rodar antes:

make clean

Esta sequência de comandos vale para os três exemplos.

2.4. Limitações

- Não há validação dos dados, caso seja entrado algum dado não válido, não há como prever o comportamento do programa.

2.5. Questões teóricas

- **Em relação ao programa que contempla os itens a) e b), quais foram as alterações de códigos-fonte necessárias para a solução (se houverem)?**

  A única alteração que teve que ser feita foi no makefile, criando uma biblioteca estática (.a) ou dinâmica (.so), e a linkagem.

- **Dados os conhecimentos adquiridos em função desse trabalho, indique vantagens e problemas decorrentes da utilização de bibliotecas dinâmicas.**

  Vantagem: Não aumenta o tamanho do executável binário.

  Problema: Como bibliotecas dinâmicas são compartilhadas entre processo, eles vão compartilhar variáveis e funções estáticas, podendo ter interferências entre processos dentro da lib.

- **Em relação às dependências do binário das aplicações criadas como soluções aos itens anteriores (a, b e c):**
  - **Quais são as dependências necessárias para a execução da aplicação? (Uma resposta para cada item).**

    Estático: Para a execução do binário não é necessário nenhuma outra dependência externa. (./trabalho_4)

    Dinâmico: Caso o arquivo lib .so não esteja na pasta, o binário não executará.

    Dinamicamente carregado: Caso o arquivo lib .so não esteja na pasta, ele continuará a executar, mas avisará que não foi possível carregar lib e finalizaram execução.

  - **Em relação ao posicionamento em memória, entre as instanciações da sua aplicação houve alguma alteração de endereçamento das dependências na memória?**

    Sim, houve mudança dos endereço de memória a para cada uma das compilações.

Brasília, DF
01 de Junho de 2017