PUC Minas

Pontifícia Universidade Católica de Minas Gerais
Campus de Poços de Caldas

Departamento de Ciência da Computação
Curso de Ciência da Computação
Disciplina de Sistemas Operacionais

TRABALHO DE PIPES

Lucas Omar Andrade Leal
Aluno

Prof. Dr. Joao Carlos de Moraes Morselli Junior
Professor

Poços de Caldas – MG

Junho de 2022

TRABALHO DE SISTEMAS OPERACIONAIS - PIPES

TRABALHO ACADÊMICO

Trabalho referente à DISCIPLINA DE SISTEMAS OPERACIONAIS, 1º Semestre de 2022.

Poços de Caldas – MG

Junho de 2022

Sumário

1 INTRODUÇÃO

Este documento tem como objetivo apresentar a descrição do trabalho acadêmico, notadamente no campo das ***Ciências da Computação*** (CC), em atendimento à tarefa solicitada da disciplina de Sistemas Operacionais – TRABALHO DE PIPES (30 PTS) 25/06, da Pontifícia Universidade Católica de Minas Gerais (PUC Minas). O trabalho aqui reportado está inserido no contexto do uso de ferramentas de ***Inter Process Communication (IPC) e Canais de Comunicação de Processos Utilizando PIPES***, como parte do estudo do assunto previsto na ementa da referida disciplina, na Pontifícia Universidade Católica de Minas Gerais (PUC Minas), *campus* de Poços de Caldas, enquadrando-se num cenário de desenvolvimento e implementação de um software que realiza o uso do sistema de intercomunicação de processos. Pretende-se apresentar com o **Código Fonte Desenvolvido em Linguagem C**, os cenários: 1) compreensão do funcionamento de Ferramentas de Intercomunicação de Processos; 2) estudo sobre o gerenciamento, troca de dados e manipulação entre arquivos; 3) desenvolvimento de uma aplicação que realiza o uso de PIPES para controle de dados enviados entre dois ou mais métodos em um sistema de criptografia e descriptografia de arquivos.

2 DESCRIÇÃO DA FERRAMENTA

Os PIPES são definidos como arquivos específicos para conexão entre dois processos, sendo assim, canais de comunicação para troca de dados para fins de escrita e leitura. De tal modo, se um processo A deseja enviar uma informação para B, o primeiro escreve sua mensagem no pipe que conecta os dois, e por conseguinte, o segundo faz a leitura do conteúdo contido no próprio arquivo de transmissão [1].

A ferramenta para Intercomunicação de Processos é de extrema importância para sincronização e integridade dos dados na ocorrência do software, uma vez que o pipe, em sua

etapa de leitura, é capaz de bloquear a escrita em seu arquivo. E da mesma maneira, impedir que dados sejam gravados enquanto há a execução de leitura [1].

De tal maneira, o código fonte apresentado faz a utilização dos recursos mencionados, a fim de sincronizar a execução de seus processos, sendo *pipe1* e *pipe2*, responsáveis pela comunicação do processo pai ao filho, criado no momento da seleção da opção 1. Estes, realizam a troca de informação da nomenclatura do ficheiro, fornecida pelo usuário. Neste ponto, fecha-se a leitura de *pipe1* e escrita de *pipe2*, para que seja possível que a função de escolha do arquivo (*choosefilecript*) envie seus dados ao *pipe1.* Após feito, a escrita em *pipe1* é encerrada junto à leitura de *pipe2.* O processo inverso ocorre na função de criptografia (*cript*), sendo inicialmente fechado o *pipe1* quanto à escrita e *pipe2* quanto à leitura, para que o processo atualmente em execução consiga realizar a leitura do *pipe1*.

O processo mencionado anteriormente irá ocorrer de forma semelhante na opção 2. Neste caso, os pipes são denominados *pipe3* e *pipe4*. Aqui, é fechada a leitura de *pipe3* e escrita de *pipe4*, para que seja possível que a função de escolha do arquivo (*choosefiledcript*) envie seus dados ao *pipe3. Em seguida*, a escrita em *pipe3* é encerrada, bem como a leitura de *pipe4.* De forma inversa, ocorre na função de descriptografia (*dcript*), sendo inicialmente fechado o *pipe3* quanto à escrita e *pipe4* quanto à leitura, para que o processo atualmente em execução consiga realizar a leitura do *pipe4*.

De maneira complementar o *pipe5*, utilizado somente para escrita, uma vez que a leitura de seu valor será somente apresentada na tela, expõe o status de execução das duas funções mencionadas anteriormente. Este, recebe os dados ao final de cada função e apresenta para o usuário identificando se o ficheiro se trata de um processo de criptografia ou descriptografia.

Vale ressaltar que a posição 0 dos vetores dos *pipes* se refere ao processo descritor de leitura, enquanto a posição 1, escrita.

3 DESCRIÇÃO DA APLICAÇÃO

Esta aplicação pretende estudar, de forma avançada, a estrutura de *Intercomunicação de Processos* e implementar, como artefato entregável, um software que seja capaz de coletar um arquivo especificado pelo usuário, permitindo que este ficheiro seja encriptado e decriptado, através do uso da técnica de criptografia por meio de *Marca D'água*, impossibilitando sua leitura através de aplicativos convencionais. De forma complementar, é realizado o uso da ferramenta de *PIPES* para envio das informações fornecidas pelo usuário entre os processos de escolha, criptografia e descriptografia, contribuindo para a compreensão dos temas abordados em aula.

Como artefato entregável, eis as funcionalidades realizadas pelo código em sua execução:

1. Apresentação de um *menu* de opções para o usuário que permite realizar a de Criptografar, Descriptografar ou Encerrar a execução do software. (FIGURA 01);

2. Ao selecionar a opção referente à encriptação, é solicitado que seja informado o nome de um arquivo para ser codificado. (FIGURA 01);

    2.1. Este, caso não presente no mesmo diretório, alerta ao cliente que houve uma falha de seleção e o pedido é novamente realizado. (FIGURA 01);

    2.2. No sucesso da seleção do arquivo, o software realiza a criação de um uma cópia, de prefixo *‘crpt_’,* implementando a inserção da marca d’água no código binário do mesmo. Assim, tornando-o ilegível aos seus programas de leitura convencionais. (FIGURA 02);

3. Caso seja escolhida a opção de decriptação, assim como na etapa anterior, um nome é

requerido ao usuário. (FIGURA 03);

3.1. Novamente, um alerta será enviado ao usuário atual, caso um ficheiro com o mesmo nome não seja encontrado na pasta atual do código fonte, e então seja solicitada a inserção do substantivo. (FIGURA 03);

3.2. Ao encontrar o arquivo, é realizada a criação de uma cópia com o prefixo *'dcrpt_'*, a qual irá remover a marca d'água presente no código binário. Desta maneira, a recuperação das informações contidas nele, é realizada. (FIGURA 04);

4. Encerrar o processo. (FIGURA 05)

4 CÓDIGO

```
/**
 * Descrição:
 *  Projeto de implementação de pipes em um programa que contém mais de 1
processo
 *  para a disciplina de Sistemas Operacionais S1-2022
 *
 * Autoria:
 *  Lucas Omar Andrade Leal - lucasomarandradeleal@gmail.com
 *
 * Compilado da seguinte forma:
 *      $ gcc main.c -o main
 *      $ ./main
 **/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// VERIFICA O SISTEMA OPERACIONAL PARA SUBSTITUIR FUNÇÕES DE LIMPEZA DE TELA,
BUFFER E SLEEP
#if defined(__MINGW32__) || defined(_MSC_VER)
#define clean_input() fflush(stdin)
#define clear_screen() system("cls")
#define sleep() system("timeout 1")
#else
#include <stdio_ext.h>
#define clean_input() __fpurge(stdin)
#define clear_screen() system("clear")
#define sleep() system("sleep 1")
```

```
#endif

#define MAXBUFF 1024      // número de caract. do buffer
#define MAX_PWD_SIZE 1024 // tamanho máximo de nomenclatura de entrada

#define BYTE 8 // 1 byte =  8 bits para percorrer o arquivo binário
unsigned char byte;

// funções de controle de cor para visualização do usuário
char print_pink(char *s);
char print_yellow(char *s);
char print_blue(char *s);
char print_green(char *s);
char print_green(char *s);
char print_red(char *s);
char print_white(char *s);
char print_cyan(char *s);

// variáveis globais para leitura e escrita para os pipes
int readfd, writefd;

/**
 * - Função main é responsável por apresentar o menu ao usuário e receber a
opção que
 * ele deseja... criptografar, descriptografar ou sair
 * - Por meio da variável option_valida é determinado o que é válido na
entrada de dados
 * - Opção 1 leva o usuário a entrar com o nome de um arquivo específico para
que ele seja criptografado
 * - Opção 2 leva o usuário a entrar com o nome de um arquivo específico para
que seja descriptografado
 * - Opção 3 sai do programa
 * */

/**
 * Processo de criptografar
 * Pai -> Filho
 * pipe1[0] - leitura no pipe 1
 * pipe1[1] - escrita no pipe 1
 * Filho -> Pai
 * pipe2[0] - leitura no pipe 2
 * pipe2[1] - escrita no pipe 2
 */
/**
 * Processo de descriptografar
 * Pai -> Filho
 * pipe3[0] - leitura no pipe 3
 * pipe3[1] - escrita no pipe 3
 * Filho -> Pai
 * pipe4[0] - leitura no pipe 4
 * pipe4[1] - escrita no pipe 4
 * */
```

```c
/**
 * Processo que somente recebe dados para pode apresentá-los na tela
 * pipe5[0] - leitura no pipe 5
 * pipe5[1] - escrita no pipe 5
 */

main()
{
    system("clear");

    char option, option_valida;
    int counter = 0,
        descritor, // usado para criar o processo filho pelo fork
        pipe1[2],  // comunicação, pai → filho processo de criptografia
        pipe2[2],  // comunicação, filho → pai processo de  criptografia
        pipe3[2],  // comunicação, pai → filho processo de  descriptografia
        pipe4[2],  // comunicação, filho → pai processo de  descriptografia
        pipe5[2];  // comunicação sequencial, filho → pai processo de status
do programa - pipe5 voltado somente para leitura de dados e informar estado
do processo

    print_blue("-------------------------------------");
    print_blue("\n| CR33PT3R - SISTEMAS OPERACIONAIS |\n");
    print_blue("-------------------------------------\n\n");

    if (pipe(pipe1) < 0 || pipe(pipe2) < 0 || pipe(pipe3) < 0 || pipe(pipe4)
< 0)
    {
        print_red("\nmain.c: Erro na chamada do pipe");
        printf("\nerror: pipe1 = %d pipe2 = %d pipe3 = %d pipe4 = %d",
pipe(pipe1), pipe(pipe2), pipe(pipe3), pipe(pipe4));
        exit(0);
    }

    do
    {
        menu();                // mostra o menu
        scanf("%c", &option); // recebe a opção
        getchar();             // recebe o enter
        counter++;

        // valida a opção
        option_valida = (option == '1' || option == '2' || option == '3');
        if (!option_valida)
        {
            clear_screen();
            fflush(stdin);
            print_red("\nOps... Opção ivalida\n\n");
            counter = 0;
        }
    } while (!option_valida);
```

```
    switch (option)
    {
    case '1':
        //   Fork para criar o processo filho
        if ((descritor = fork()) < 0)
        {
            print_red("\nmain.c: Erro na chamada do fork");
            printf("\nerror: fork = %d", descritor);
            exit(0);
        }
        else if (descritor > 0) // PAI
        {
            close(pipe1[0]); // fecha leitura no pipe1
            close(pipe2[1]); // fecha escrita no pipe2

            choosefilecript(pipe2[0], pipe1[1]); // Chama a escolha do
arquivo no pai

            close(pipe1[1]); // fecha escrita pipe1
            close(pipe2[0]); // fecha leitura pipe2
            exit(0);
        }
        else // (descritor > 0) // FILHO
        {
            close(pipe1[1]); // fecha escrita no pipe1
            close(pipe2[0]); // fecha leitura no pipe2

            cript(pipe1[0], pipe2[1]); // Chama a o processo de criptografia
do no filho

            close(pipe1[0]); // fecha leitura no pipe1
            close(pipe2[1]); // fecha escrita no pipe2

            mostraStatus(pipe5[0]);
            close(pipe5[0]); // fecha escrita no pipe5
            exit(0);
        }
        break;
    case '2':
        //   Fork para criar o processo filho
        if ((descritor = fork()) < 0)
        {
            print_red("\nmain.c: Erro na chamada do fork");
            printf("\nerror: fork = %d", descritor);
            exit(0);
        }
        else if (descritor > 0) // PAI
        {
            close(pipe3[0]); // fecha leitura no pipe3
            close(pipe4[1]); // fecha escrita no pipe4
```

```
            choosefiledcript(pipe4[0], pipe3[1]); // Chama a escolha do
arquivo no pai

            close(pipe3[1]); // fecha escrita pipe3
            close(pipe4[0]); // fecha leitura pipe4
            exit(0);
        }
        else // (descritor > 0) // FILHO
        {
            close(pipe3[1]); // fecha escrita no pipe3
            close(pipe4[0]); // fecha leitura no pipe4

            dcript(pipe3[0], pipe4[1]); // Chama a o processo de
descriptografia do no filho

            close(pipe3[0]); // fecha leitura no pipe3
            close(pipe4[1]); // fecha escrita no pipe4

            mostraStatus(pipe5[0]);
            close(pipe5[0]); // fecha escrita no pipe5
            exit(0);
        }
        break;
    case '3':
        print_pink("\nAdeus...");
        exit(0);
        break;
    }
}

/**
 * Processo 1 para seleção de um arquivo
 * - Tenta abrir em modo de leitura o arquivo passado pelo usuário, para
 * verificar sua existência
 * - O processo de solicitação do nome é feito até que o arquivo exista
 * - Ao encontrar o arquivo escolhido, ele envia através do pipe, o nome
para o processo 2 que irá criptografar
 */
choosefilecript(readfd, writefd)
{
    int i = 0;
    unsigned char byte;
    FILE *arquivoEscolhido; // ponteiros para percorrer os bytes dos arquivos
    char arquivo[MAX_PWD_SIZE];

    print_green("\nProcesso 1...");

    do
    {
        print_green("\n1 - Digite o nome do arquivo a ser criptografado \nEx:
arquivo.txt\n.: ");
        scanf("%s", arquivo);
```

```
        arquivoEscolhido = fopen(arquivo, "r"); // tenta abrir o arquivo para
leitura, se não conseguir o arquivo não existe

        if (arquivoEscolhido == NULL)
        {
            print_red("\nATENÇÃO o arquivo '");
            printf("%s", arquivo);
            print_red("' nao foi encontrado!!!\n");
        }

    } while (arquivoEscolhido == NULL);

    printf("\nO arquivo '%s' foi selecionado com sucesso!!!\n", arquivo);
    write(writefd, arquivo, 1024); // <<<<<<<
    printf("...fim do processo 1\n\n");
}

/**
 * Processo 2 para criptografia do arquivo
 * - O nome recebido através do pipe é passado como parâmetro para a função
que irá abrir o arquivo em modo de leitura e escrita binária
 * - O processo de inserção de uma marca d'água o torna ilegível pelos demais
programas
 * */
cript(readfd, writefd)
{
    int i = 0;
    char arquivo[MAX_PWD_SIZE], arquivoCopia[MAX_PWD_SIZE] = "crpt_", key[21]
= "SistemasOperacionais";
    FILE *arquivoEscolhido, *arquivoEscolhidoCopia; // ponteiros para
percorrer os bytes dos arquivos

    read(readfd, arquivo, 1024);   // <<<<<<<
    strcat(arquivoCopia, arquivo); // add 'crpt_' na frente do nome do
arquivo

    arquivoEscolhido = fopen(arquivo, "rb");           // abre o arquivo em
modo de leitura (read)
    arquivoEscolhidoCopia = fopen(arquivoCopia, "wb"); // cria o arquivo para
receber a cópia dos dados

    clear_screen();

    print_green("\nProcesso 2...");
    print_green("\nIniciando criptografia do arquivo ");
    printf("%s\n", arquivo);

    for (int x = 0; x < 3; x++)
    {
        sleep();
        print_green(".\n");
```

```
    }

    while (!feof(arquivoEscolhido))
    {
        i++;
        fread(&byte, sizeof(unsigned char), 1, arquivoEscolhido);
        fwrite(&byte, sizeof(unsigned char), 1, arquivoEscolhidoCopia);
        if (i == 2)
            fwrite(&key, sizeof(char), 21, arquivoEscolhidoCopia);
    }

    fclose(arquivoEscolhido);      // fecha leitura do  original
    fclose(arquivoEscolhidoCopia); // fecha escrita do novo
}

/**
 * Processo 3 para seleção de um arquivo criptografado
 * - Tenta abrir em modo de leitura o arquivo passado pelo usuário, para
 * verificar sua existência
 * - O processo de solicitação do nome é feito até que o arquivo exista
 * - Ao encontrar o arquivo escolhido, ele envia através do pipe,  o nome
para o processo 4 que irá descriptografar
 */
choosefiledcript(readfd, writefd)
{
    int i = 0;
    char arquivoToDesc[MAX_PWD_SIZE];
    FILE *arquivoToDescEscolhido; // ponteiros para percorrer os bytes dos
arquivos

    print_yellow("\nProcesso 3...");
    do
    {
        print_yellow("\n2 - Digite o nome do arquivo a ser descriptografado
\nEx: arquivo.txt\n.: ");
        scanf("%s", arquivoToDesc);

        arquivoToDescEscolhido = fopen(arquivoToDesc, "r");

        if (arquivoToDescEscolhido == NULL)
            printf("\nO arquivo '%s' nao existe!!!\n\n", arquivoToDesc);

    } while (arquivoToDescEscolhido == NULL);

    printf("\nO arquivo '%s' foi selecionado com sucesso!!!\n",
arquivoToDesc);
    write(writefd, arquivoToDesc, 1024); // <<<<<<<
    printf("...fim do processo 3\n");
}

/**
 * Processo 4 para descriptografar o arquivo
```

```
 * - O nome recebido através do pipe é passado como parâmetro para a função
que irá abrir o arquivo em modo de leitura e escrita binária
 * - O processo retira a marca d'água reescrevendo um novo arquivo e
"pulando" os bytes específicos que contém a marca d'água
 * */
dcript(readfd, writefd)
{
    int i = 0;
    char arquivoToDesc[MAX_PWD_SIZE], arquivoToDescCopia[MAX_PWD_SIZE] =
"dcrpt_", key[21] = "SistemasOperacionais";
    FILE *arquivoToDescEscolhido, *arquivoToDescEscolhidoCopia; // ponteiros
para percorrer os bytes dos arquivos

    read(readfd, arquivoToDesc, 1024);
    strcat(arquivoToDescCopia, arquivoToDesc); // add 'crpt_' na frente do
nome do arquivo

    arquivoToDescEscolhido = fopen(arquivoToDesc, "rb");
    arquivoToDescEscolhidoCopia = fopen(arquivoToDescCopia, "wb");

    clear_screen();

    print_yellow("\nProcesso 4...");
    print_yellow("\nIniciando descriptografia do arquivo ");
    printf("%s\n", arquivoToDesc);

    for (int x = 0; x < 3; x++)
    {
        sleep();
        print_yellow(".\n");
    }

    while (!feof(arquivoToDescEscolhido))
    {
        i++;

        fread(&byte, sizeof(unsigned char), 1, arquivoToDescEscolhido);
        if (i > 2 && i <= 23) // pula os bytes que têm a marca d'água
            continue;
        fwrite(&byte, sizeof(unsigned char), 1, arquivoToDescEscolhidoCopia);
    }
    fclose(arquivoToDescEscolhido);
    fclose(arquivoToDescEscolhidoCopia);
}

// Apresenta o menu na tela
menu()
{
    print_green("1. Criptografar um arquivo\n");
    print_yellow("2. Descriptografar um arquivo");
    print_pink("\n3. Sair");
    printf("\n.: ");
```

```
}

/**
 * Processo 5 apresentar o status do processo de criptografia ou
descriptografia
 * - O processo recebe pelo pipe5 o status ocorrido durante a criptografia ou
descriptografia
 * - Ele identifica qual foi realizado através da função 'strstr' que
verifica se uma
 * string passada como parâmetro contém uma substring
 */
mostraStatus(readfd)
{
    print_cyan("\nProcesso 5...");

    char *verifica, resultado[MAX_PWD_SIZE];
    read(readfd, resultado, 1024);

    // verifica se o nome do arquivo contém  "dcrpt_"
    verifica = strstr(resultado, "dcrpt_");

    if (verifica)
    {
        print_cyan("\n>> O arquivo foi descriptografado gerando o arquivo
'");
        printf("%s", resultado);
        print_cyan("' <<\n\n");
    }
    else
    {
        print_cyan("\n>> O arquivo foi criptografado gerando o arquivo '");
        printf("%s", resultado);
        print_cyan("' <<\n\n");
    }
    print_pink("\nPress enter to exit...");
}

// Funções de controle de cor para visualização do usuário
// Recebem uma string e formatam sua cor para saída no terminal bash
char print_pink(char *s)
{
    printf("\033[1;35m%s\033[0m", s);
}

char print_cyan(char *s)
{
    printf("\033[1;36m%s\033[0m", s);
}

char print_yellow(char *s)
{
    printf("\033[1;33m%s\033[0m", s);
```

```
}

char print_blue(char *s)
{
    printf("\033[1;34m%s\033[0m", s);
}

char print_green(char *s)
{
    printf("\033[1;32m%s\033[0m", s);
}

char print_red(char *s)
{
    printf("\033[1;31m%s\033[0m", s);
}

char print_white(char *s)
{
    printf("\033[1;29m%s\033[0m", s);
}

char print_reset(char *s)
{
    printf("\033[0m%s\033[0m", s);
}
```

5 APRESENTAÇÃO GRÁFICA

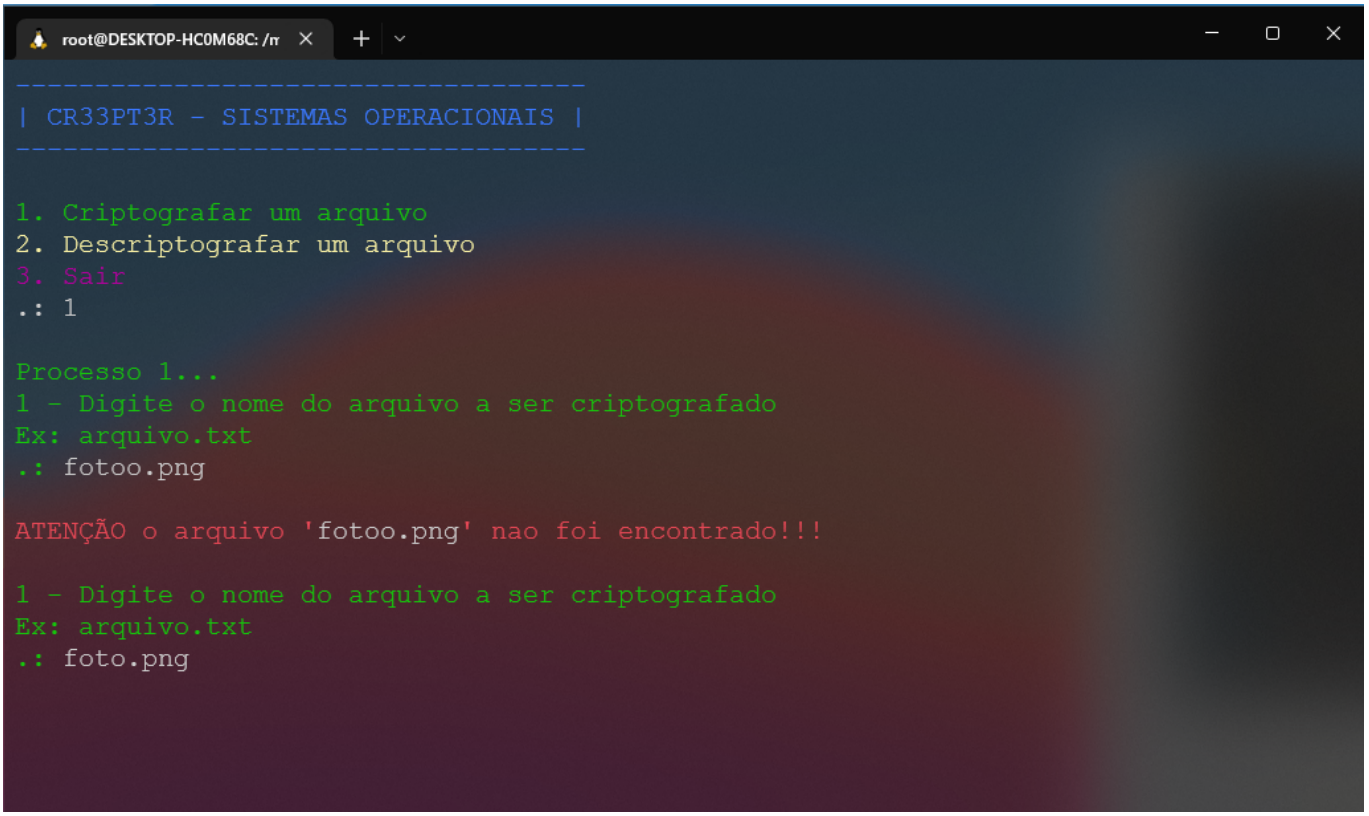

Figura 01: Menu, Solicitação do arquivo para criptografar e Alerta de erro

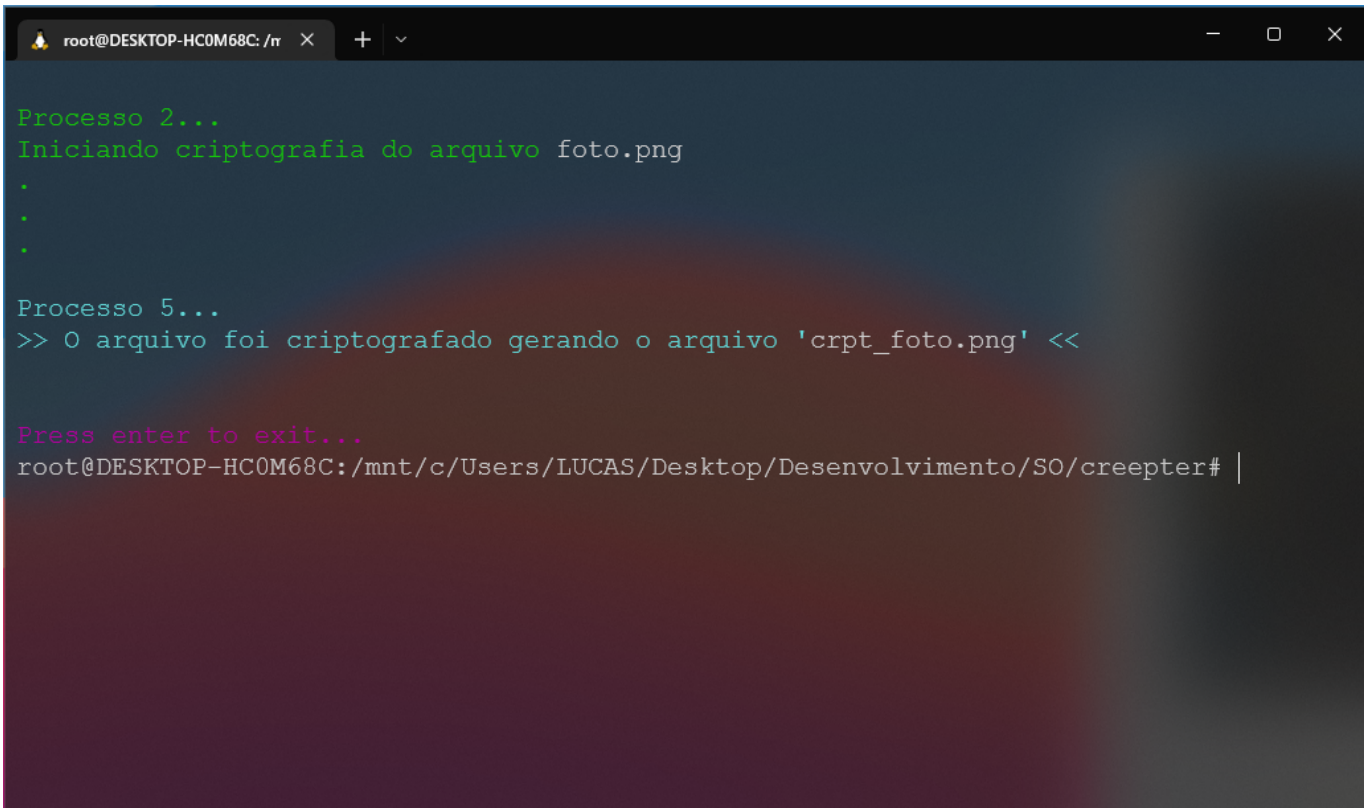

Figura 02: Criptografia do arquivo e Mensagem de status

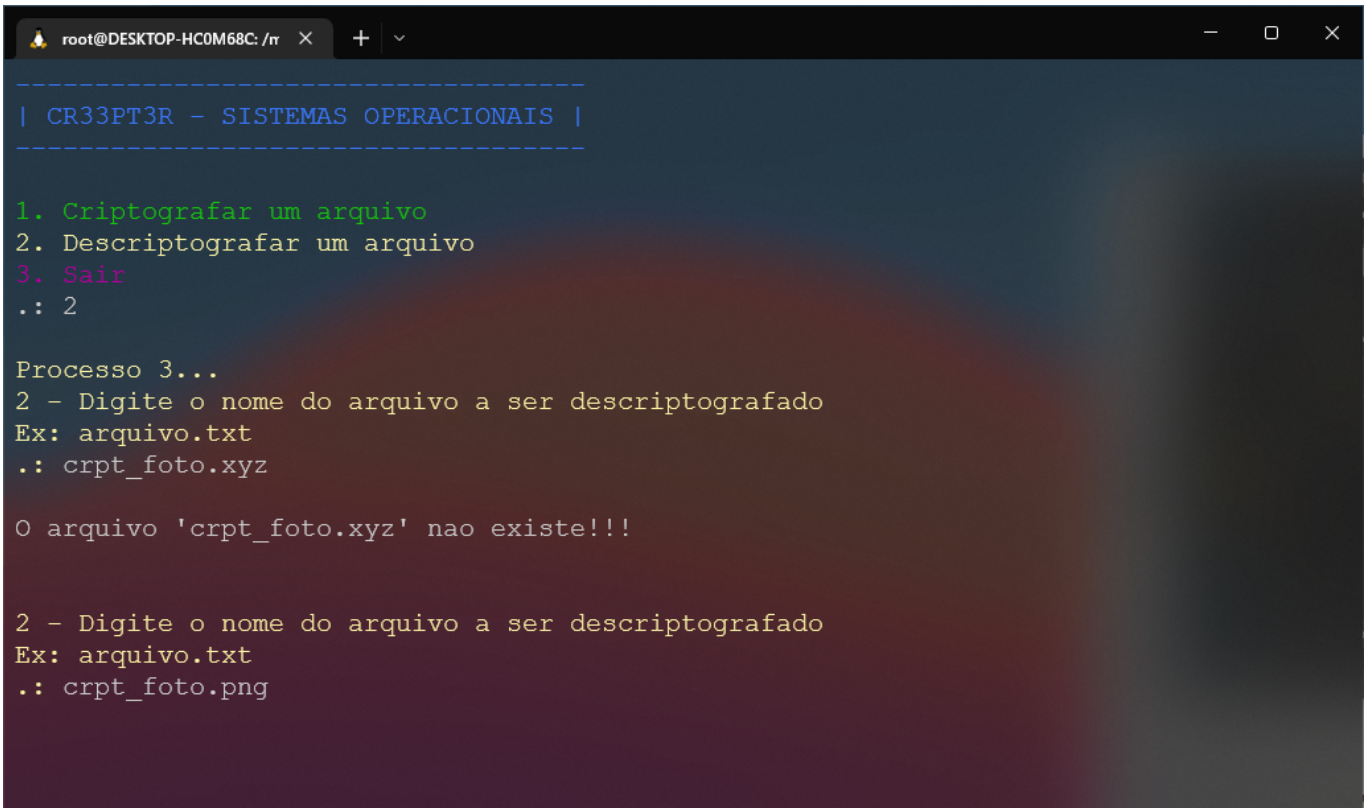

Figura 03: Menu, Solicitação do arquivo para descriptografar e Alerta de erro

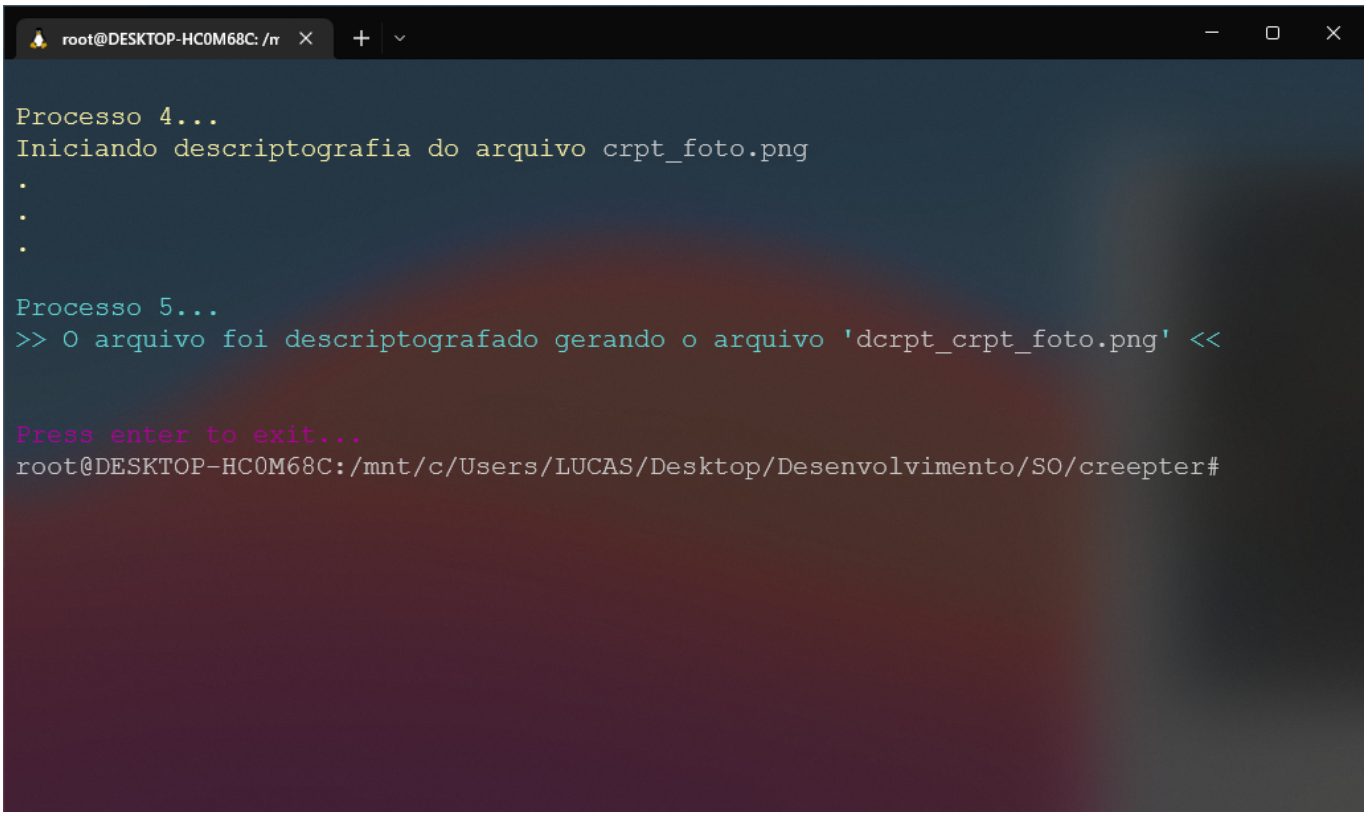

Figura 04: Criptografia do arquivo e Mensagem de status

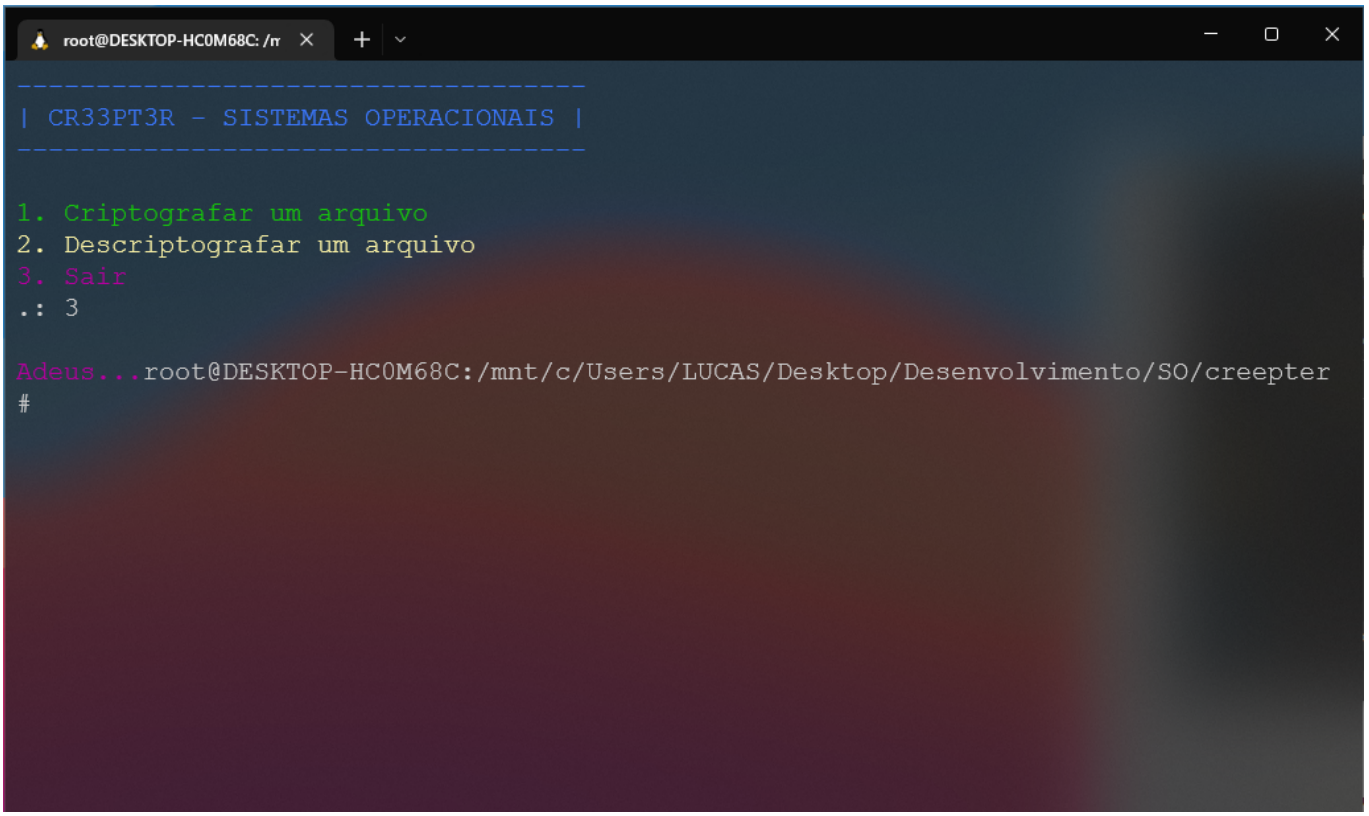

Figura 05: Encerramento da aplicação

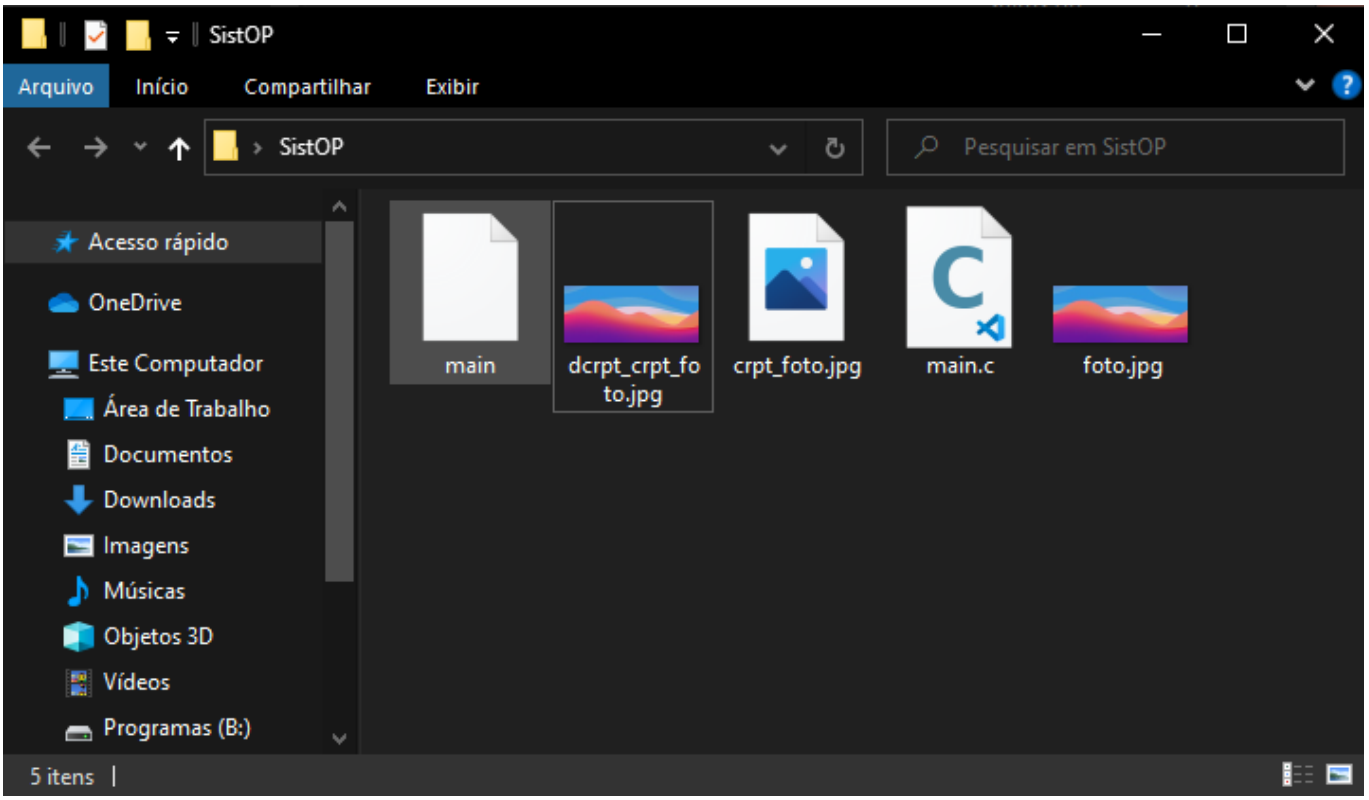

Figura 06: Arquivos manipulados

7 REFERÊNCIAS

TANENBAUM, Andrew S.; BOS, Herbert. Sistemas Operacionais Modernos. 4. ed. rev. São Paulo, Brasil: Pearson Education do Brasil, 2016. 757 p. ISBN 978-85-4301-818-8.

JÚNIOR, João Benedito dos Santos. Linguagens de Programação - Linguagem C. Poços de Caldas, Minas Gerais, Pontifícia Universidade Católica – PUC Minas, 2º semestre de 2020. Disponibilizado pela disciplina de Algoritmos e Estruturas de Dados II.

LEAL, Lucas Omar Andrade. Caderno de Algoritmos e Estruturas de Dados II, 2020. Disponível em:
<https://docs.google.com/document/d/1xKh8IVPibrBhTyvyclD2P-Sn8VW5WNak/edit?usp=sharing&ouid=113631318397658761493&rtpof=true&sd=true>