# Universidade da Beira Interior

Departamento de Informática


Departamento de Informática

## Exercício 1 – Sistemas Operativos

Elaborado por:

Pedro Lourenço

Mateus Aleixo

Samuel Dias

**Professor Doutor: Paul Andrew Crocker**

# Agradecimentos

Enquanto grupo prestamos agradecimento a todo o Departamento de Informática porque sem o conhecimento que nos foi dado ao longo do curso não seria possível a realização deste trabalho. Dentro do Departamento de Informática prestamos um agradecimento especial ao Ex.mo Professor Doutor Paul Crocker .

# Resumo

O objetivo deste trabalho foi começar por desenvolver um programa usando linguagem de baixo nível para fazer as descodificações de um ficheiro encriptado usando comandos que realizavam certas e determinadas instruções, onde utilizamos a função *lseek* e *write* para permitir realizar esta parte do exercício. A principal restrição desta parte do exercício foi de facto não podermos usar as funções de output do *standard library*.

Após a realização da parte A, na parte B tivemos que criar um algoritmo de codificação de uma *string*. Apesar de agora podermos usar todas as funções do output do *standard library*, tentamos ao máximo manter-nos fieis à utilização de funções de baixo nível.

# Palavras-chave

Sistemas Operativos, Exercício 1, Parte A, Parte B

# Parte B

Para realizar a parte B, mantivemo-nos fiéis ao conteúdo dado nas aulas teórico-práticas da cadeira de Sistemas Operativos e usar a linguagem C para desenvolver o exercício pedido.

```
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#define BUFFER_SIZE 101
#define FILE_MODE (O_WRONLY | O_CREAT | O_APPEND)

void writeRandomLettersToFile(const char *filename, int numLetters)
{
   int fd = open(filename, FILE_MODE);

   if (fd < 0)
   {
      printf("Erro ao abrir %s\n", filename);
      return;
   }

   char letter;
   srand(time(NULL));

   for (int i = 0; i < numLetters; i++)
```

```
    {
        letter = 'a' + (rand() % 26);

        if (write(fd, &letter, 1) < 0)
        {
            perror("Erro ao escrever\n");
            close(fd);
            return;
        }

        fsync(fd);
    }
```

```
    close(fd);
}
void print_instructions(const char *string1, const char *string2)
{
    int len1 = strlen(string1);
    int len2 = strlen(string2);

    if (len2 > len1)
        return;

    const char *ptr = strstr(string1, string2);
    if (ptr == NULL)
    {
        printf("A string original não está contida na string encriptada\n");
        return;
    }

    int offset = ptr - string1;
    int remaining = len1 - offset - len2;
```

```
    printf("i %d\n", offset);
    printf("r %d\n", len2);

    if (remaining > 0)
    {
        printf("+ %d\n", remaining);
    }
}


int main(int argc, char *argv[])
{
    if (argc != 3)
    {
        printf("Uso: %s <.bin> <string>", argv[0]);
        return 1;
    }

    int file_input = open(argv[1], O_RDONLY);

    if (file_input < 0)
    {
        printf("Erro ao abrir <.bin>\n");
        return 1;
    }

    int file_output = open(argv[2], FILE_MODE);

    if (file_output < 0)
    {
```

```
    printf("Erro ao abrir <string>\n");
    close(file_input);
    return 1;
}

char buffer[BUFFER_SIZE];
int len;

while ((len = read(file_input, buffer, BUFFER_SIZE - 1)) > 0)
{
    writeRandomLettersToFile(argv[2], 5);

    for (int i = 0; i < len; i++)
    {
        if (buffer[i] == ' ')
        {
            writeRandomLettersToFile(argv[2], 10);
        }
        else
        {
            write(file_output, &buffer[i], 1);
        }
    }

    writeRandomLettersToFile(argv[2], 5);
    printf("Encriptação concluída com sucesso\n");
}

if (len < 0)
{
    perror("Erro ao ler <.bin>\n");
```

```
    close(file_input);
    close(file_output);
    return 1;
  }



if (lseek(file_output, 0, SEEK_SET) < 0 || lseek(file_input, 0, SEEK_SET) < 0)
{
    perror("Erro ao mover o indicador de posição\n");
    close(file_input);
    close(file_output);
    return 1;
  }

  close(file_input);
  close(file_output);

  int fi = open(argv[2], O_RDONLY);

  char str[BUFFER_SIZE];
  int str_len = read(fi, str, BUFFER_SIZE - 1);

  if (str_len < 0)
  {
    perror("Erro ao abrir ficheiro\n");
    close(fi);
    return 1;
  }

  str[str_len] = '\0';
```

```
    char *sub_str = strtok(buffer, " ");
    int i = 1;

    while (sub_str != NULL)
    {

        print_instructions(str, sub_str);
        sub_str = strtok(NULL, " ");
        i++;
    }
    printf("s 0\n");
    close(fi);

    return 0;
}
```

*Script 1 - Parte B*

# Objetivo da Parte B

O código acima é um programa em linguagem C que recebe dois argumentos na linha de comando: o nome de um ficheiro binário com a string necessária e um ficheiro (que precisa de já estar criado). O programa lê o conteúdo do ficheiro binário, substitui os espaços por sequências de letras aleatórias e escreve o resultado em um novo ficheiro. Em seguida, o programa lê a nova string, a divide em substrings separadas por espaços e imprime as instruções para desencriptar a string original.

O programa começa com a definição de algumas constantes e a declaração de algumas funções. A função writeRandomLettersToFile gera uma sequência de letras aleatórias e as escreve em um ficheiro. A função print_instructions recebe duas strings e imprime as instruções necessárias para descriptografar a segunda string a partir da primeira.

Em seguida, a função principal main é definida. Ela começa verificando se o número correto de argumentos foi passado na linha de comando. Caso isso não aconteça, o programa exibe uma mensagem de erro o programa termina. Em seguida, o programa abre o ficheiro binário para leitura e o ficheiro de saída para escrita. Sempre que não for possível abrir um dos ficheiros, uma mensagem de erro é exibida.

Se passar pela parte das verificações do programa, lê o conteúdo do ficheiro binário em blocos de tamanho definido pela constante BUFFER_SIZE. A cada bloco lido, o programa chama a função writeRandomLettersToFile para gerar uma sequência aleatória de letras e escrevê-las no ficheiro de output. Em seguida, o programa percorre o bloco lido caracter por caracter. Se o caracter for um espaço em branco, o programa chama novamente a função writeRandomLettersToFile para criar outra sequência aleatória de letras e escrevê-las no ficheiro de saída. Se o caracter não for um espaço em branco, ele escreve no ficheiro de saída. Finalmente, após processar todo o bloco, o programa chama novamente a função writeRandomLettersToFile para gerar uma sequência aleatória de letras.

Após processar todo o ficheiro binário, o programa verifica se ocorreu algum erro na leitura. Se sim, devolve uma mensagem de erro. Se não, o programa move o indicador de posição para o início dos ficheiros de entrada e saída e continua a execução.

Em seguida, o programa abre novamente o ficheiro de saída e lê o conteúdo do buffer.

Após a leitura, o programa vai dividir a nova string em substrings separadas por espaços usando a função strtok e para cada substring, chama a função print_instructions para imprimir as instruções necessárias para desencriptar a substring original. Finalmente, o programa imprime a instrução “s 0” indicando o final da encriptação

O programa termina após fechar todos os ficheiros abertos.

# Exemplos de execução

Para exemplificar a execução da parte B usamos 3 ficheiros bin para exemplificar, onde:

- T1.bin tem a string “ParteB”
- T2.bin tem a string “teste do teste”
- T3.bin tem a string “SO Engenaharia Informática”

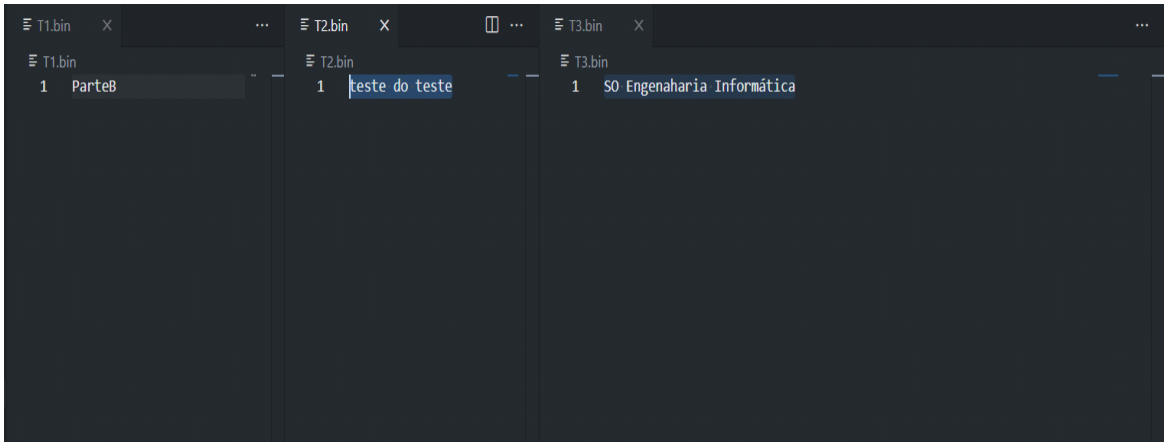


*Figura 1 - Ficheiros Bin para teste*

Ao executar a Parte B para todas os ficheiros do tipo .bin obtemos:

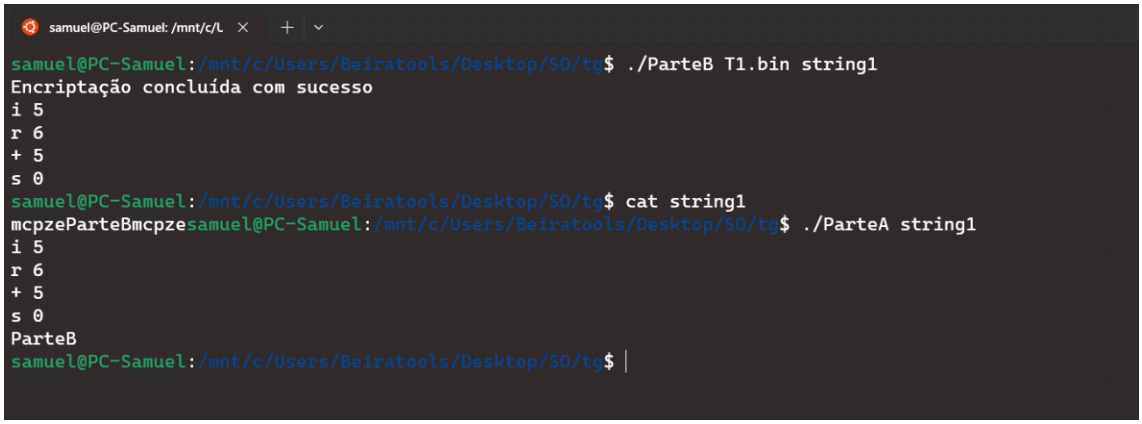


*Figura 2- Teste da parte B no file T1.bin*

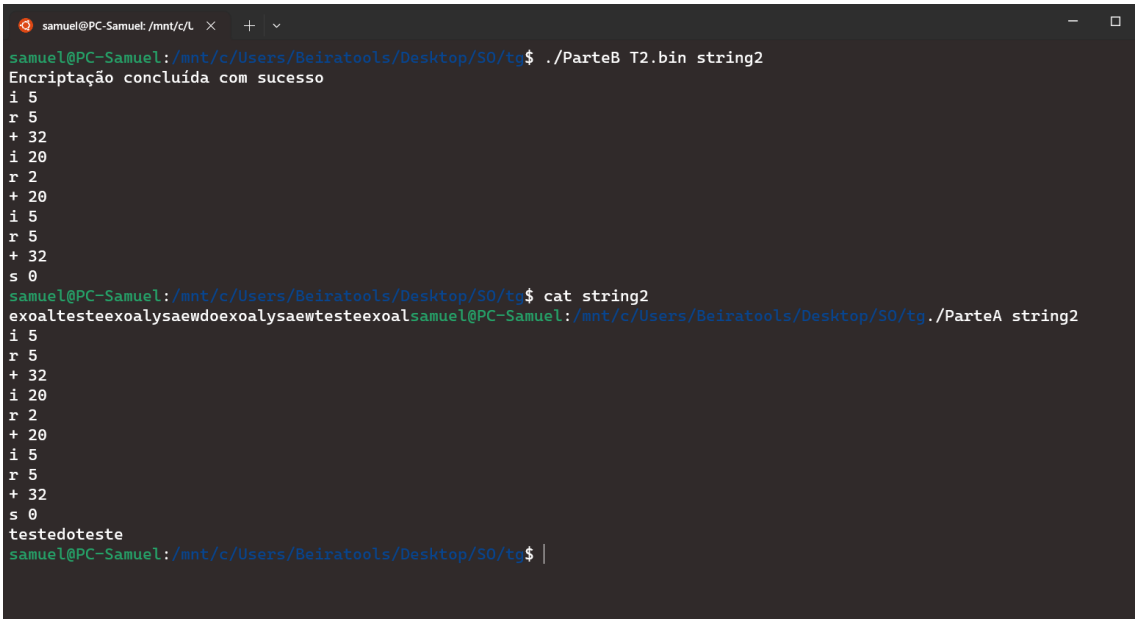


*Figura 3- Teste da parte B no file T2.bin*

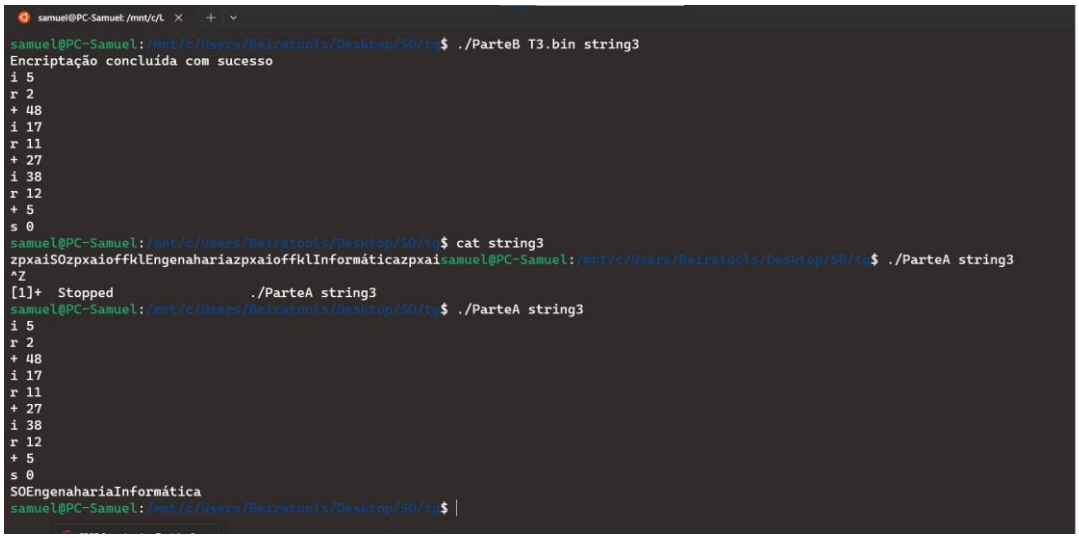


*Figura 4 - Teste da parte B no file T3.bin*

# Conclusões

Com a realização deste trabalho conseguimos pôr em prática todos os conteúdos lecionados ao longo das últimas semanas tanto a nível técnico como a nível teórico. No decorrer da execução do trabalho pratico damos destaque à aplicação da teoria lecionada, mas sem nunca esquecer

Na generalidade do trabalho passamos por ultrapassar alguns problemas com sucesso, o que é mais um ponto positivo a tirar da realização deste trabalho, porque nos ajudou a superar dificuldades e a enfrentar novos desafios.

Para concluir podemos tirar um balanço positivo após a realização de todo o trabalho visto conseguimos ultrapassar dificuldades, aplicar conhecimentos e trabalhar em grupo de forma eficiente.