



RELATÓRIO

Trabalho Prático da Unidade Curricular de Sistemas Operativos

Engenharia de Sistemas Informáticos

30 de Abril de 2022

Projeto realizado por:

Henrique Neto
(16626)

João Moreira
(23522)

João Araújo
(23103)

Bruna Paço
(24073)

Rogério Filho
(21868)

Rodrigo Monteiro
(23514)

Índice

Introdução	2
Curta demonstração do conteúdo do programa	3
Código das respetivas funcionalidades	5
Testes de Funcionalidades	13
Implementação de um conjunto de comandos para manipular ficheiros.....	21
Conclusão	26
Bibliografia	27

Introdução

O seguinte relatório da unidade curricular de Sistemas Operativos do curso de Engenharia de Sistemas Informáticos do Instituto Politécnico do Cávado e do Ave, têm como objetivo dar a conhecer o trabalho prático sobre os conceitos de gestão de processos e de ficheiros, assim como a aplicação da comunicação entre processos.

Para isso, realizou-se um programa na linguagem C através da implementação de comandos baseados nas funções de chamada ao sistema, em inglês, system calls.

Tendo em conta o programa desenvolvido através da linguagem referida anteriormente, constam as seguintes funcionalidades:

1. Mostrar Ficheiro
2. Copiar Ficheiro
3. Acrescentar Origem Destino
4. Contar Linhas
5. Apagar Ficheiro
6. Informar Ficheiro
7. Lista de Ficheiros e Pastas de uma diretoria especifica

Curta demonstração do conteúdo do programa

➤ Funções no Header File

```
void printMenu();
void mostraFicheiro(char *fileName);
int tamanhoString(char *name);
bool verificarFicheiroExiste(char *fileName);
int bytesFicheiro(char *fileName);
void deletarFicheiro(char *fileName);
void copiarFicheiro(char *fileName);
void concatenarString(char *str1, char *str2, char *newString);
int tamanhoString(char *name);
void acrescentaDestino(char *fileOrigem, char *fileDestino);
void contarLinhas(char *fileName);
void listarDiretorio(char *diretoria);
```

➤ Menu

```
void printMenu() {
    printf("-----\n");
    printf("----- MENU ----- \n");
    printf("-----\n");
    printf(" [1] -> Mostrar Ficheiro\n");
    printf(" [2] -> Copiar Ficheiro\n");
    printf(" [3] -> Acrescenta Origem Destino\n");
    printf(" [4] -> Conta Linhas Ficheiro\n");
    printf(" [5] -> Apaga Ficheiro\n");
    printf(" [6] -> Informa Ficheiro\n");
    printf(" [7] -> Lista Diretoria\n");
    printf(" [0] -> SAIR\n");
    printf("opcao: ");
}
```

```
-----
----- MENU -----
-----
[1] -> Mostrar Ficheiro
[2] -> Copiar Ficheiro
[3] -> Acrescenta Origem Destino
[4] -> Conta Linhas Ficheiro
[5] -> Apaga Ficheiro
[6] -> Informa Ficheiro
[7] -> Lista Diretoria
[0] -> SAIR
```

➤ Makefile

```
M makefile
1  all: build run
2
3  ✓ build:
4      gcc ./src/*.c -o main
5
6  ✓ run:
7      ./main
8
9  ✓ clean:
10     rm main
```

Código das respetivas funcionalidades

1. Mostrar Ficheiro

Assim que o utilizador clicar na tecla 1 no menu que surgirá assim que o projeto for executado, aparecerá todo o conteúdo existente que consta no ficheiro de texto que foi referido como parâmetro.

É importante reconhecer que caso o ficheiro não exista, o utilizador será avisado do sucedido.

A porção de código escrito tendo em base uma função para mostrar determinado ficheiro segue-se na imagem:

```
void mostraFicheiro(char *fileName)
{
    char opcao;
    int file;

    printf("-----\n");

    //Abrir arquivo
    char readBuffer[bytesFicheiro(fileName)];
    file = open(fileName, O_RDONLY);

    read(file, readBuffer, sizeof(readBuffer));
    write(STDIN_FILENO, readBuffer, sizeof(readBuffer));
    close(file);

    printf("\n-----\n");
}
```

2. Copiar Ficheiro

Esta funcionalidade tem como finalidade fazer a cópia de um ficheiro onde por conseguinte será criado um novo ficheiro designado por "ficheiro.copia". Os dados do ficheiro são copiados na sua totalidade tendo como referência o ficheiro passado como parâmetro.

Também para esta função, caso o ficheiro seja inexistente, o utilizador será alertado do ocorrido.

```
do {  
    printf("> Insira o caminho do ficheiro a ser copiado: ");  
    scanf("%s", fileName);  
  
    if (verificarFicheiroExiste(fileName) == FALSE) {  
        perror("    > Erro ao ler ficheiro!");  
    }  
  
} while (verificarFicheiroExiste(fileName) == FALSE);
```

A parte do código desenvolvido referente à funcionalidade de copiar um determinado ficheiro pode ser vista na imagem:

```
void copiarFicheiro(char *fileName) {  
  
    int file, newFile;  
    char readBuffer[bytesFicheiro(fileName)];  
    char opcao;  
  
    char *aux = ".copia";  
    char newString[tamanhoString(fileName) + tamanhoString(aux)];  
  
    concatenarString(fileName, aux, newString);  
  
    file = open(fileName, O_RDONLY);  
    newFile = open(newString, O_RDWR|O_CREAT, S_IRUSR | S_IWUSR);  
  
    read(file, readBuffer, sizeof(readBuffer));  
    int result = write(newFile, readBuffer, sizeof(readBuffer));  
    close(file);  
    close(newFile);  
  
    if (result < 0) {  
        perror("> Nada foi escrito!");  
    } else {  
        printf("> Ficheiro copiado com sucesso!\n");  
        printf("    > Foram copiados: %d bytes para o ficheiro '%s'\n", result, newString);  
    }  
  
    printf("-----\n");  
}
```


3. Acrescentar Origem Destino

De facto, para que o utilizador tenha a possibilidade de acrescentar, no menu terá de pressionar a tecla 3 e de seguida já vai poder acrescentar a informação que consta em origem para o final, em destino.

Ainda em relação a esta função, se o utilizador for em busca de um ficheiro que é inexistente, o mesmo será avisado.

Para que fosse possível concretizar o funcionamento deste comando, foi escrita a seguinte porção de código:

```
void acrescentaDestino(char *fileOrigem, char *fileDestino) {  
  
    int file1, file2;  
    char readBuffer[bytesFicheiro(fileOrigem)];  
    char opcao;  
  
    file1 = open(fileOrigem, O_RDONLY);  
    file2 = open(fileDestino, O_WRONLY | O_APPEND);  
  
    read(file1, readBuffer, sizeof(readBuffer));  
    int result = write(file2, readBuffer, sizeof(readBuffer));  
    close(file1);  
    close(file2);  
  
    if (result == -1) {  
        perror("> Nada foi escrito!\n");  
    } else {  
        printf("> Ficheiro copiado com sucesso!\n");  
    }  
  
    printf("\n-----\n");  
}
```

4. Contar Linhas

A partir desta função, o utilizador terá a oportunidade de saber quantas linhas contém o ficheiro de texto. As linhas serão contadas e a quantidade das mesmas vai ser visível.

Se o ficheiro não estiver explicito, a função não é executada.

A função contém a seguinte informação:

```
void contarLinhas(char *fileName)
{
    int fd, count=0;
    int size = bytesFicheiro(fileName);
    char readBuffer[bytesFicheiro(fileName)];
    char *apont;
    char opcao;

    fd = open(fileName, O_RDONLY);

    read(fd, readBuffer, sizeof(readBuffer));

    apont = strtok(readBuffer, "\n");

    while(apont != NULL)
    {
        count ++;
        apont = strtok(NULL, "\n");
    }

    close(fd);

    printf("Numero de linhas: %d\n", count);

    printf("\n-----\n");
}
```

5. Apagar Ficheiro

Através desta funcionalidade, o utilizador vai definir e digitar o nome de um ficheiro que deseja eliminar após clicar na tecla 5 do menu.

O ficheiro será removido a partir da função “unlink”, isto é, uma entrada de diretório que se refere a um ficheiro

Caso o ficheiro não exista, nada acontecerá.

```
void deletarFicheiro(char *filename)
{
    char opcao;

    unlink(filename);
    printf("Ficheiro Deletado com sucesso!!\n");

    printf("-----\n");
}
```

6. Informações do Ficheiro

Através desta funcionalidade, o utilizador vai ter acesso a informações do sistema de ficheiros em relação ao ficheiro indicado, tipo de ficheiro (normal, diretoria, link, etc.), i-node, utilizador dono em formato textual e datas de criação, leitura e modificação em formato textual.

```
void informa(char *filename)
{
    struct stat sfile;
    struct tm dt;
    char opcao;

    stat(filename,&sfile);

    struct passwd *pw = getpwuid(sfile.st_uid);
    struct group *gr = getgrgid(sfile.st_gid);

    //Accessing data members of stat struct
    if(S_ISDIR(sfile.st_mode))
    {
        printf("\nTipo: Diretoria\n");
    }
    else if(S_ISREG(sfile.st_mode))
    {
        printf("\nTipo: Ficheiro Regular\n");
    }
    else if(S_ISLNK(sfile.st_mode))
    {
        printf("\nTipo: Link\n");
    }
    printf("\nI-node: %ld\n",sfile.st_ino);
    printf("\nUid: (      %d/      %s)\n",sfile.st_uid,pw->pw_name);
    printf("\nGrupo: %s\n", gr->gr_name);
    dt = *(gmtime(&sfile.st_ctime));
    printf("\nCriado em: %d-%d-%d %d:%d:%d\n", dt.tm_mday, dt.tm_mon + 1, dt.tm_year + 1900, dt.tm_hour + 1, dt.tm_min, dt.tm_sec);
    dt = *(gmtime(&sfile.st_mtime));
    printf("\nUltima modificação: %d-%d-%d %d:%d:%d\n", dt.tm_mday, dt.tm_mon + 1, dt.tm_year + 1900, dt.tm_hour + 1, dt.tm_min, dt.tm_sec);
    dt = *(gmtime(&sfile.st_atime));
    printf("\nUltimo acesso: %d-%d-%d %d:%d:%d\n", dt.tm_mday, dt.tm_mon + 1, dt.tm_year + 1900, dt.tm_hour + 1, dt.tm_min, dt.tm_sec);
    printf("\n");
}
```

7. Listar diretorias

Através desta funcionalidade, o utilizador terá acesso a uma lista de todas as pastas e ficheiros existentes na diretoria indicada ou na diretoria atual (no caso de não ser especificado).

```
void listarDiretorio(char *diretoria)
{
    DIR *dir;
    struct dirent *entry;
    struct stat filestat;
    char opcao;

    if ((dir = opendir(diretoria)) == NULL) {
        perror("Erro ao abrir diretório");
    } else {
        printf("> Conteúdo de: %s: \n", diretoria);

        while((entry = readdir(dir))) {
            stat(entry->d_name, &filestat);

            if (S_ISDIR(filestat.st_mode)) {
                printf("%s: %s\n", "Dir", entry->d_name);
            } else {
                printf("%s: %s\n", "File", entry->d_name);
            }
        }

        closedir(dir);
    }

    printf("\n-----\n");

    do {
        printf("Pressione 'v' para voltar:");
        scanf(" %c", &opcao);
    } while (opcao != 'v' && opcao != 'V');
}
```


Testes de Funcionalidades

1. Mostrar Ficheiro

Ao clicar na tecla 1 do menu, vai aparecer uma pergunta para o utilizador para digitar o caminho do ficheiro.

```
-----  
----- MENU -----  
-----  
[1] -> Mostrar Ficheiro  
[2] -> Copiar Ficheiro  
[3] -> Acrescenta Origem Destino  
[4] -> Apaga Ficheiro  
[5] -> Informa Ficheiro  
[6] -> Informa Ficheiro  
[7] -> Lista Diretoria  
[0] -> SAIR  
opcao: 1
```

O caminho do ficheiro terá de conter o nome da pasta, por exemplo “src” e o nome do ficheiro complementado com a extensão, ou seja, no exemplo que veremos a seguir: sosd.txt




```
> Insira o caminho do ficheiro: src/sosd.txt
```

De seguida, o utilizador verá a informação que está contida no terminal que pertence ao ficheiro mencionado anteriormente.

```
-----  
TESTE  
TESTE  
TESTE  
ADICIONADO  
ADICIONADO  
ADICIONADO  
-----
```

Nota: Sempre que o utilizador não escrever no terminal o nome do caminho do ficheiro corretamente, surgirá uma mensagem de erro.



```
> Insira o caminho do ficheiro: src/sosd11.txt
  > Erro ao ler ficheiro!: No such file or directory
> Insira o caminho do ficheiro: 
```

2. Copiar Ficheiro

Para fazer a cópia de um determinado ficheiro, o utilizador pressiona na tecla 2 do menu, será lhe questionado novamente sobre o caminho do ficheiro que irá ser copiado

```
> Insira o caminho do ficheiro a ser copiado: src/sosd.txt
```

O formato do novo ficheiro é: nomeficheiro.txt.copia

A seguir, o utilizador, tendo em conta que digitou corretamente o caminho do ficheiro a ser copiado, receberá uma mensagem no terminal a informar que o ficheiro foi copiado com sucesso, porém também, a quantidade de bytes que foram passados para o novo ficheiro.

```
> Ficheiro copiado com sucesso!  
  > Foram copiados: 54 bytes para o ficheiro 'src/sosd.txt.copia'  
-----  
Pression 'v' para voltar:█
```

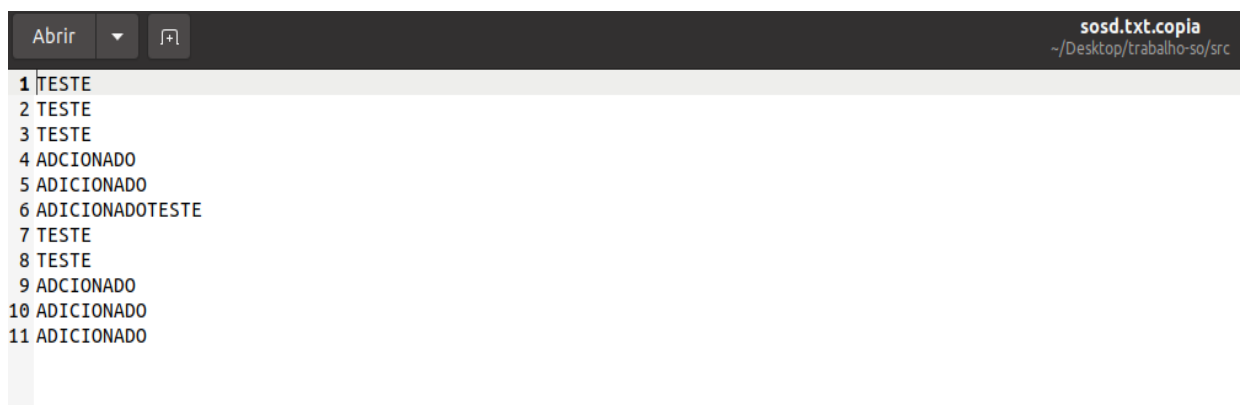

3. Acrescentar Origem Destino

Para esta funcionalidade ser bem-sucedida, o utilizador vai digitar dois elementos: o caminho do ficheiro 1 e o caminho do ficheiro 2.

```
> Insira o caminho do ficheiro 1: src/sosd.txt  
> Insira o caminho do ficheiro 2: src/sosd.txt.copia
```

```
> Ficheiro copiado com sucesso!
```


Para comprovar que a função está funcional, a imagem abaixo demonstra exatamente que foi adicionado no ficheiro sosd.txt.copia duplicadamente a informação que consta no arquivo sosd.txt



```
Abrir  sosd.txt.copia  
~/Desktop/trabalho-so/src  
1 |TESTE  
2 |TESTE  
3 |TESTE  
4 |ADICIONADO  
5 |ADICIONADO  
6 |ADICIONADOTESTE  
7 |TESTE  
8 |TESTE  
9 |ADICIONADO  
10 |ADICIONADO  
11 |ADICIONADO
```


4. Contar Linhas do Ficheiro

Assim que o utilizador premir a tecla 4 do menu, necessitará de escrever o caminho do arquivo de forma correta.





```
> Insira o caminho do ficheiro: src/sosd.txt
```

O utilizador pode ver pelo terminal e pelo próprio ficheiro de texto que, para este caso, o número de linhas é seis.



```
Numero de linhas: 6
```



```
1 TESTE
2 TESTE
3 TESTE
4 ADICIONADO
5 ADICIONADO
6 ADICIONADO
```

sosd.txt
~/Transferências/trabalho-so-master/src

5. Apagar Ficheiro

Para remover um ficheiro, o utilizador precisa de inserir o caminho do arquivo.

```
> Insira o caminho do ficheiro:    src/sosd.txt.copia
```

De seguida, o ficheiro é apagado com sucesso, se e só se, o caminho do arquivo foi escrito de forma correta.

```
Ficheiro Deletado com sucesso!!
```

```
-----  
Pression 'v' para voltar:
```

6. Informação do Ficheiro

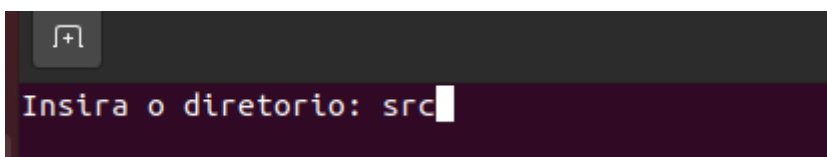
Através desta função, é possível ver os detalhes de um determinado ficheiro.

O procedimento que têm de se ter passa por digitar o caminho do ficheiro e por conseguinte, o utilizador verá as seguintes informações relativa a um determinado ficheiro, neste caso, menciono o sosd.txt.

```
Tipo: Ficheiro Regular
I-node: 16122201
Uid: (      1000/      joaoaraujo)
Grupo: joaoaraujo
Criado em: 1-5-2022 21:14:47
Ultima modifcação: 1-5-2022 21:11:0
Ultmo acesso: 1-5-2022 21:17:37
Pressione 'v' para voltar:
```

7. Listar Diretoria

Para que seja possível concretizar a experiência de se poder ver a listagem de conteúdo de uma diretoria ao acaso, o utilizador terá primeiramente de inserir o diretório desejado, por exemplo src.



Para finalizar, o mesmo terá ao seu dispor a devida lista.



Implementação de um conjunto de comandos para manipular ficheiros

a) Criação de uma nova partição

Neste primeiro passo é apresentada a listagem de todas as partições no disco /dev/sda

```
root@henrique:/home/henrique# fdisk -l /dev/sdb
Disk /dev/sdb: 20.13 GiB, 21613379584 bytes, 42213632 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

1. Para esta fase, foi criada uma partição nova de 10GB.

```
Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xb776d16f.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-42213631, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-42213631, default 42213631): +10G

Created a new partition 1 of type 'Linux' and of size 10 GiB.
```

2. Aplicação do comando “w” para ser possível gravar as alterações.

```
Created a new partition 1 of type 'Linux' and of size 10 GiB.  
Command (m for help): w  
The partition table has been altered.  
Calling ioctl() to re-read partition table.  
Syncing disks.
```

b) Criação de um novo volume

1. Criação de um novo volume na nova partição ocupando o espaço todo do disco virtual

```
root@henrique:/home/henrique# sudo pvcreate /dev/sdb1  
Physical volume "/dev/sdb1" successfully created.
```

2. Criação de um novo volume group na partição /dev/sdb1

```
root@henrique:/home/henrique# sudo vgcreate bruxos /dev/sdb1  
Volume group "bruxos" successfully created  
root@henrique:/home/henrique# _
```

3. Adição de dois volumes lógicos de 5GB no disco /dev/sdb1

```
root@henrique:/home/henrique# sudo lvcreate -L 5G -n lvbruxos bruxos  
Logical volume "lvbruxos" created.  
root@henrique:/home/henrique# sudo lvcreate -L 5G -n lv2bruxos bruxos  
Volume group "bruxos" has insufficient free space (1279 extents): 1280 required.  
root@henrique:/home/henrique# sudo lvcreate -L 4.9G -n lv2bruxos bruxos  
Rounding up size to full physical extent 4.90 GiB  
Logical volume "lv2bruxos" created.  
root@henrique:/home/henrique#
```

c) Criação de um sistema de ficheiros ext4 em um deles e ext3 no outro

1. Criação de um sistema de ficheiro ext4 no volume “lvbruxos” e ext3 no volume “lv2bruxos”

```
root@henrique:/home/henrique# sudo mkfs.ext4 /dev/bruxos/lvbruxos
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: e72a5b69-6451-4c1c-b0e1-4d92a872153b
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
root@henrique:/home/henrique# sudo mkfs.ext3 /dev/bruxos/lv2bruxos
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 1285120 4k blocks and 321280 inodes
Filesystem UUID: 993e2fe5-46c8-4939-92ff-10c83edc2df2
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks):
```


d) Montagem de cada um dos sistemas de ficheiros nas diretorias – Persistente a reboots

1. Criação de pastas ext4 e ext3 em mnt

```
root@henrique:/home/henrique# mkdir /mnt/ext4
root@henrique:/home/henrique# mkdir /mnt/ext3
root@henrique:/home/henrique# ls /mnt
ext3  ext4
```

2. Montar os sistemas de ficheiros acabados de criar nas respetivas pastas

```
root@henrique:/home/henrique# mount /dev/bruxos/lvbruxos /mnt/ext4
root@henrique:/home/henrique# mount /dev/bruxos/lv2bruxos /mnt/ext3
root@henrique:/home/henrique# ls /mnt/ext4
lost+found
root@henrique:/home/henrique# ls /mnt/ext3
lost+found
```

```
/dev/mapper/bruxos-lvbruxos on /mnt/ext4 type ext4 (rw,relatime)
/dev/mapper/bruxos-lv2bruxos on /mnt/ext3 type ext3 (rw,relatime)
```

3. Persistente a reboot

```
GNU nano 4.8 /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/ef4f52f9-6a05-4918-91fb-99de0b1aacc4 / ext4 defaults 0 1
/swap.img none swap sw 0 0
/dev/bruxos/lvbruxos /mnt/ext4 ext4 defaults 1 1
/dev/bruxos/lv2bruxos /mnt/ext3 ext3 defaults 1 1
```

e) Criação de um ficheiro com o nome composto pelo grupo dos números de alunos

1. Criação de um ficheiro com o nome composto pelo grupo

```
root@henrique:/mnt/ext4# touch 16626-23103-21868-23514-23522-24073.txt
root@henrique:/mnt/ext4# ls
16626-23103-21868-23514-23522-24073.txt  lost+found
```

2. Alterar permissões do grupo para este não ter quaisquer permissões no ficheiro e restante com permissão somente de leitura

```
root@henrique:/mnt/ext4# chmod g-r 16626-23103-21868-23514-23522-24073.txt
root@henrique:/mnt/ext4# ls -l
total 16
-rw----r-- 1 root root 0 May 1 19:56 16626-23103-21868-23514-23522-24073.txt
drwx----- 2 root root 16384 May 1 19:36 lost+found
```

f) Permissões que o ficheiro /etc/shadow têm

```
root@henrique:/etc# ls -l shadow
-rw-r----- 1 root shadow 1029 May 1 19:14 shadow
```

- I. O Dono do ficheiro pode ler e escrever;
- II. O grupo pode somente ler;
- III. Os outros não têm quaisquer permissões no ficheiro.

Conclusão

Para concluir, foi uma mais-valia realizar este projeto pois foi relevante para termos a perceção acerca de como se realiza a gestão de processos e de ficheiros bem como a aplicação da comunicação entre processos através deste trabalho prático realizado tendo em base vários recursos como a linguagem C onde, também, através da mesma foi possível aprimorar os conhecimentos da realização de um programa.

Bibliografia

<https://social.microsoft.com/Forums/pt-BR/home>

<https://stackoverflow.com/>

Recursos da unidade curricular presentes no Moodle