Ficha Prática 2 Sistemas Operativos 2023
I/O Baixo Nível

Parte 1 (1 Hora)

- Abrir Ficha 7 : http://www.di.ubi.pt/~operativos/praticos/praticos.html
- Implementar exemplo 7.1 e Fazer o exercício 7.1
- Implementar exemplo 7.2 e Fazer o exercício 7.2

Parte II (1 Hora)

Neste exercício vai alterar o programa (exercício 7.2) realizada em cima para copiar ficheiros em blocos de 128 bytes usando as funções de **`read()`** e **`write()`** de baixo nível. Recorde que o programa se executa da seguinte forma:

```
$./exercicio72 exercicio72.c backup.c
```

1. Faça as alterações necessárias ao seu programa para que este utilize um tamanho de leitura e escrita de blocos variável. Portanto aqui o importante é que a constante **`BUFFSIZE`** passe a ser uma variável:

```
BUFFSIZE = atoi(argv[3]);
char buf[BUFFSIZE] → char *buf=malloc(BUFFSIZE)
```

   Deverá chamar este novo programa solowlevelcp.c. Um **exemplo** da execução deste novo programa seria

```
$ ./solowlevelcp /etc/passwd passwdcpy 128
```

2. Faça as alterações necessárias ao seu programa para imprimir o tempo usado pelo CPU para efetuar a cópia do ficheiro. Deverá imprimir este valor no canal de **`stderr`**! Considere a sugestão seguinte:

```
clock_t inicio, fim; float tempoUsado;
inicio=clock();
.....FUNÇÂO DE COPIA....
fim=clock();
tempoUsado= (float)(fim-inicio)/ (float) CLOCKS_PER_SEC;
```

   Exemplo
```
$ ./solowlevelcp  /etc/passwd  passcopy  4
Copia Feita com BlockSize 4 Tempo do CPU 0.150000
```

3. Crie um ficheiro (16MB ou outro) e investigue o desempenho do programa copiando o ficheiro variando o tamanho de bloco. Poderá utilizar um *script* para automatizar o procedimento (ver anexo **sc1.sh**) que faz uso do comando dd para criar um ficheiro de valores aleatorios.

   PREENCHA A TABELA: Tamanho do Ficheiro _____

TAMANHO do BLOCO	TEMPO CPU lowlevel	TEMPO CPU High Level
4096		
1024		
256		
64		
16		
4		

4. O programa **sohighlevelcp.c `(em anexo)`** faz copiar um ficheiro usando as funções de I/O de alto-nível (as funções da biblioteca stdio.h) caracter a caracter (byte a byte) e também reporte o cpu usado. **Investigue** o tempo necessário para copiar o ficheiro de 100mb do script e introduzir na tabela em cima
   a. Pode utilizar o comando time do bash. i.e **time ./sohighlevelcp /tmp/bigFile /tmp/xxfile**

5. Pergunta Teórica : Explique os seus resultados !

ANEXOS e NOTAS PARTE

Ficheiros Disponiveis Online
VIA GIT: git clone https://gitlab.com/crkxcariy/soubi.git
VIA URL: usando um browser : https://gitlab.com/crkxcariy/soubi/-/tree/master/p2-2

SCRIPT sc1.sh

```
#!/bin/bash
# Script File sc1.sh
#
bigFile=/tmp/bigFile
if [ ! -e $bigFile ] ; then
    # create 16MB file in /tmp with random values
    dd if=/dev/urandom of=${bigFile} bs=16M count=1
fi
size=4096
tmpFile=$(mktemp)
while [ $size -gt 1 ]; do
   echo $size
   solowlevelcp $bigFile  $tmpFile   $size
   rm $tmpFile
   let size=size/4
done
```

Notas Para Executar:
i. solowlevelcp ou ./solowlevelcp conforme o valor da variável PATH
ii. Se tiver Mac usar 16000000 em vez de 16M
iii. Fazer chmod +x sc1.sh para que o script seja executavel
iv. Depois basta fazer sc.sh ou ./sc1.sh

PROGRAM sohighlevelcp.c

```
/*
   C Program to Copy a File using high level buffered IO
*/
#include <stdio.h>
#include <time.h>
int main (int argc, char **argv)
{
  clock_t inicio, fim;
  float tempoUsado;
  FILE *fp1, *fp2;
  int ch;

  if ((fp1 = fopen (argv[1], "r")) == NULL)
    {
      fprintf (stderr,"\n%s cannot be opened for reading\n",argv[1]);
      return 1;
    }
  if ((fp2 = fopen (argv[2], "w")) == NULL)
    {
      fprintf (stderr,"\n%s cannot be opened for Writing\n",argv[2]);
      return 1;
    }
  inicio = clock();
  while ( (ch = fgetc (fp1) ) != EOF )
    {
      fputc (ch, fp2); //copying file byte  by byte
    }
  fim = clock();
  tempoUsado = (float) (fim - inicio) / (float) CLOCKS_PER_SEC;
  fprintf (stderr, "Tempo %f\n", tempoUsado);
  return 0;
}
```

Notas

Bibliotecas Padrão

```
#include <stdio.h>
#include <stdlib.h>
//#include <io.h>    //windows
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h> //linux
#include <time.h>

//useful default file mode
#define FILE_MODE (S_IRUSR | S_IWUSR)

//useful constants
#define   STDIN_FILENO   0
#define  STDOUT_FILENO   1
#define  STDERR_FILENO   2
```

- Para desligar o "buffering" associado aos inputs/outputs padrão (stdin e stdout).
    - a função "setbuf" consultar "man setbuf"

Extra Aulas :

- Qual o mecanismo de Hardware/Sistema Operativo que permite copiar um ficheiro em blocos e não apenas byte a byte ?
- Existe um valor ótimo de I/O para cada ficheiro – este valor depende de que ?
- Este valor ótimo pode ser guardado pelo sistema operativo nos atributos do ficheiro – identifique como isto é feito (ver pagina 3 dos apontamentos 7 sobre Ficheiros e Diretorias)