# Universidade da Beira Interior

# Sistemas Operativos
# 2º Semestre – 2022/2023
# V2 02/05/2023

- O projeto este ano consiste em dois exercícios. O primeiro relacionado com a utilização das funções e ficheiros de baixo nível para leitura e escrita e o segundo com a comunicação entre processos (IPC)
- Cada grupo do trabalho pode ter 2 ou 3 alunos do mesmo turno pratico (de preferência)
- Inscrição do Grupo - https://forms.office.com/e/MsAHS6XxrS até dia 27 de Março

## Exercício 1 (1.5 Valores)

Este exercício tem como objetivo utilizar as funções do I/O de ficheiros de baixo-nível, as chamadas ao sistema de open, write e lseek. O objetivo dos programas a escrever é "esteganografia" ou a arte de esconder uma mensagem e ocultar a sua existência dentro dum outro objecto.

## PARTE A

O primeiro programa é para pesquisar dentro dum ficheiro binário para encontrar uma string escondido para quem sabe a existência duma sequência de instruções especiais. Podem usar as funções de input do standard library, por exemplo scanf() para ler as instruções que vem pelo Standard Input, mas não podem usar as funções de output do standard library (printf, putchar etc) (nem podem ter estas palavras em comentários)

Ao programa ser dado uma sequência de instruções que tem de seguir para obter a string escondido. Há seis tipos de instrução (i) ir para o início dum ficheiro e avançar n bytes, (ii) (i) ir para o fim dum ficheiro e recuar n bytes (iii) avançar/recuar da posição atual n bytes e (iv) ler n bytes para um buffer (v) limpar o buffer (vi) terminar ou Stop.

### Input

O nome do ficheiro binário para abrir em modo leitura é passado no primeiro argumento ao programa (argv[1]). Se a abertura do ficheiro dar um erro o seu programa deverá terminar logo. O input (standard input) contêm uma sequência de linhas. Cada linha contêm um carácter e um inteiro separado por um espaço. A sequência termine com o par 's' 0(zero). Os caracteres e os seus significados são:
+ n : avançar n bytes
- n : recuar n bytes
i n : ir para o inicio do ficheiro e avançar n bytes
f n : ir para o fim do ficheiro e recuar n bytes
r n : read/ler n bytes e append (acrescentar) a um buffer.
l 0 : limpar o buffer
s 0 : stop

### Output

O output é o conteúdo valido do buffer escrito para o standard output usando apenas o comando write seguido por um byte (a nova linha).
Para escrever uma nova linha, o caracter '\n', poderá definir uma variável ou constante e utilizar write(), por exemplo
const char novalinha='\n';
write(STDOUT_FILENO,&novalinha,1);

### Condicionantes

O buffer terá um máximo de 101 bytes, portanto a string a encontrar não terá mais de 100 caracteres.
Neste programa **não pode** usar printf - a palavra printf não poderá aparecer em parte algum do seu programa - em vez do printf utilizar write(STDOUT FILENO,..) para escrever para o ecrã.

Pode ver se a palavra printf está no seu programa usando o comando grep.

## Exemplo

Considere o ficheiro de **Exatamente 22** bytes test1.bin que contêm a string seguinte (sem fim de linha)
ola one two three four

Nota que são 22 bytes visto que são 18 caracteres (a-z) e 4 espaços. Deverá criar este ficheiro para efetuar testes (exemplo echo – n "ola one two three four" > test1.bin .

Para saber o número de bytes dum ficheiro usar o comando ls -l e para ver os bytes individuais interpretados como caracteres usar o comando octal dump "od -bc"
Considere os inputs nos ficheiros input1 e input2 como em baixo. Deverá correr o seu programa com os seus testes da seguinte maneira: ./revelar test1.bin < input1

## Sample Input 1

```
i 2
+ 2
r 3
+ 5
r 5
s 0
```

## Sample Output 1

```
onethree
```

## Sample Input 2

```
i 3
r 3
f 2
r 1
l 0
i 2
i 0
r 3
f 2
- 2
r 4
s 0
```

## Sample Output 2

```
olafour
```

Entregar código fonte e mais 3 exemplos criados pelo equipa.

## PARTE B

No segundo parte deste exercício deverá criar um programa que dado uma string (segredo) e opcionalmente um ficheiro, aleatoriamente partilhe e esconde o string dentro do ficheiro criando aleatoriamente também a sequência de instruções ncessaria para revelar a a string usando o programa da parte A. Portanto o ouput do programa da parte B são os inputs do programa da parte A

- PB( S ) -> F, L ou PB ( S, F ) -> L ou PB ( S, Fin) -> F,L
- onde
- PA( F, L ) -> S

Não é necessário aqui considerar a operação de limpar o buffer visto podemos criar sequências alectórios seguidos por "l 0" e acrescentar uma última sequencia que tenha a verdadeira string. Aqui está livre para criar um ou mais mecanismos para a criação duma sequência automática e aleatório!

**Entregar no moodle** um ficheiro pdf e dois ficheiros .c: PDF com breve descrição do seu algoritmo (parte B) e exemplos de execução, e código fonte.

Prazo 23.59 Horas 16 de Abril de 2023.

## Exercício 2 (3 Valores)

Neste exercício o grupo irá escrever duas versões dum programa chamado "tokenring" que recebe 4 inteiros - n, p, t, max - na linha de comando.

- O programa deverá criar **n** processos ligados entre si por canais de comunicação geridos pelo sistema operativo do tipo FIFOs (pipetas com nome ou named pipes".
- As "named pipes" devem ter nomes pipe1to2, pipe2to3, ..., pipento1 e devem permitir a comunicação unidirecional entre um processo i e o processo i+1.
- A última "named pipe" fecha o anel permitindo ao processo n comunicar com o processo 1.
- Depois de criado este anel de processos, p1 deverá enviar uma "token"(uma mensagem com um inteiro com valor inicial 0) para o processo seguinte (p1 > p2) e por aí em diante (p2 > p3 > ... > pn > p1 > ...).
- De cada vez que um processo recebe a "token"deve incrementar o seu valor e reenviá-la de imediato para o processo seguinte ou, com uma probabilidade de p, bloquear o seu envio durante t segundos, imprimindo, no caso de bloqueio, uma mensagem assinalando esse facto (ver o exemplo que se segue).
- A "token"deve circular entre os processos, incrementando o seu valor em cada "hop", até que o sistema recebe a ordem para terminar.

$ tokenring 5 0.25 2 1500
[p2] blocked on token (val = 1000)
[p5] blocked on token (val = 1003)

**Versão 1 do programa** - O sistema deve terminar quando n>=max.

**Versão 2 do programa** - Criar/Inventar outro sistema de terminação que não esteja baseado no valor do token. (Cuidado para não deixar processos num estado de starvation/deadlock.)

Por exemplo - O sistema podia começar o processo de terminação quando um processo deteta que o valor num ficheiro "tokenctl.txt" muda de valor 0 para 1. Outra ideia será o envio dum sinal para um dos processos para iniciar o processo de terminação.

Nota que poderá enviar informação suplementar no "token". Quer dizer, o token pode ser apenas um inteiro ou qualquer estrutura. P.ex (123,-1->continuar) (123,2->stop e 2 o processo que detetou a mudança de 0 para 1)

Ver https://man7.org/linux/man-pages/man3/lockf.3.html

**References**


- *UNIX Network Programming, Volume 2, Second Edition: Interprocess Communications*, 1999
- https://www.di.ubi.pt/~operativos/praticos/html/11-ipc.html
- https://www.di.ubi.pt/~operativos/praticos/html/7-fileIO.html


Entregar: breve descrição do seu algoritmo, código fonte e exemplos de execução

Prazo 23.59 Horas 4 de Junho de 2023. Apresentação dia 6 (terça) e 7 (quarta) de junho