L:EIC / SO2223
Trabalho Prático

Q1. Escreva um programa `samples` que dado um ficheiro de texto e dois inteiros, `n` e `m`, na linha de comando imprime no terminal um texto constituído por `n` fragmentos, cada um com `m` caracteres, do ficheiro original. Os fragmentos devem ser obtidos em `n` posições diferentes no ficheiro de forma aleatória e ser escritos sequencialmente no `stdout` sem qualquer separador ou mudança de linha entre caracteres `>` e `<`.

```
$ samples
usage: samples file numberfrags maxfragsize
$ cat quote.txt
Look again at that dot. That's here. That's home. That's us. On it everyone
you love, everyone you know, everyone you ever heard of, every human
being who ever was, lived out their lives. The aggregate of our joy and
suffering, thousands of confident religions, ideologies, and economic
doctrines, every hunter and forager, every hero and coward, every creator
and destroyer of civilization, every king and peasant, every young couple
in love, every mother and father, hopeful child, inventor and explorer,
every teacher of morals, every corrupt politician, every "superstar," every
"supreme leader," every saint and sinner in the history of our species
lived there - on a mote of dust suspended in a sunbeam.
$ samples quote.txt 5 7
>ry king<
>yone yo<
>unbeam.<
>king an<
>y teach<
```

Sugestão: use as funções `fseek()`, `srandom()` e `random()` da `libc`. Como semente para a geração dos números aleatórios deve usar o valor `0`.

Q2. Escreva um programa `txt2epub` que dada uma lista de `n` ficheiros em texto - `f1.txt`, `f2.txt`, ..., `fn.txt` - na linha de comando, aplique o programa `pandoc` a cada um dos ficheiros gerando versões EPUB dos mesmos:

```
pandoc f1.txt -o f1.epub
...
pandoc fn.txt -o fn.epub
```

A conversão dos ficheiros `.txt` nos respectivos `.epub` deve ser feita em paralelo por `n` processos criados para o efeito. Cada processo obtém um nome de um ficheiro de `argv[]` e converte-o como descrito. Depois dos processos terminarem de converter todos os ficheiros o último passo do programa `txt2epub` será a geração de um ficheiro `.zip` como os `n` ficheiros em formato EPUB.

```
zip ebooks.zip f1.epub ... fn.epub
```

Por exemplo:

```
$ txt2epub iliada.txt odisseia.txt eneida.txt ... metamorfoses.txt
[pid2751] converting iliada.txt ...
[pid2749] converting metamorfoses.txt ...
[pid2752] converting odisseia.txt ...
...
$ ls
ebooks.zip
eneida.epub
iliada.epub
metamorfoses.epub
odisseia.epub
```

Sugestão: obtenha um conjunto de livros em formato `.txt` a partir da Internet. Use apenas cópias legais de livros. Veja aqui por exemplo: Projecto Gutenberg.

Q3. Escreva um programa `tokenring` que recebe 3 inteiros - `n`, `p` e `t` - na linha de comando. Quando executado, cria `n` processos ligados entre si por "named pipes". As "named pipes" devem ter nomes `pipe1to2`, `pipe2to3`, ..., `pipento1` e, como o nome indica, permitem 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 > ...`). A "token"deve circular entre os processos ininterruptamente, incrementando o seu valor em cada "hop". De cada vez que um processo recebe a "token"deve reenviá-la de imediato para o processo seguinte ou, com uma probabilidade de `p`, bloquear o seu envio durante `t` segundos, imprimindo uma mensagem assinalando esse facto (ver o exemplo que se segue). Em ambos os casos, o valor da "token"deve ser incrementado.

```
$ tokenring 5 0.01 10
[p2] lock on token (val = 2867)
[p2] unlock token
[p5] lock on token (val = 9213)
[p5] unlock token
...
```

Sugestão: utilize a função da API do kernel `mkfifo()` para criar as "named pipes".

Considerações Gerais. Para garantir que os resultados que obtiveram podem ser reproduzidos pelos docentes, devem fazer os testes finais de todos os programas na máquina `gnomo.fe.up.pt`.

Quando o trabalho estiver terminado, cada grupo deve enviar por e-mail, para o docente da turma prática respectiva, um arquivo `.zip` cujo nome deve conter o número do grupo e o número da turma PL a que pertencem os membros do grupo, e.g., `G2PL6.zip`. O arquivo deve conter um directório com o mesmo nome - `G2PL6` - dentro do qual estarão três sub-directórios - `Q1`, `Q2` e `Q3` - cada um contendo o código fonte C para o problema respectivo bem como um ficheiro `makefile` com regras que permitam compilar o programa e limpar ficheiros temporários e binários. A pasta `G2PL6` deve incluir também um ficheiro de texto com os nomes completos e os números mecanográficos dos elementos do grupo.

Para além do envio do código fonte com a resolução dos problemas, como descrito acima, cada grupo deverá fazer uma apresentação do trabalho prático para o docente da turma prática respectiva em data a combinar. Na apresentação é obrigatória a presença de todos os membros do grupo.

Bom trabalho!

A equipa da unidade curricular "Sistemas Operativos"