Fundamentos de Sistemas de Operação

MIEI 2017/2018

Laboratory session 7

Overview

Resolução das questões 15 e 16 do teste.

Processos e pipes

Pretende-se implementar um programa de nome "parProc" que lança o processamento paralelo de um ficheiro aplicando-lhe dois comandos distintos, sem argumentos, mas lendo o ficheiro do disco apenas uma vez. Para tal, são criados os processos necessários para executar os comandos, de modo a que cada um receba no seu *standard input* o conteúdo do ficheiro. O processo pai deve ler o ficheiro e enviar o seu conteúdo a ambos os comandos. Cada comando afixado o resultado no *standard output*.
Por exemplo, dado o ficheiro "newsSurf.txt" o resultado desse programa deve ser:

```
$ cat newsSurf.txt
MEO Rip Curl Pro Portugal, Supertubos, 25 de Outubro de 2017 --
Gabriel Medina ganhou pela segunda vez consecutiva, derrotando Julian Wilson na final.
Frederico Morais, Kikas, foi derrotado pelo Brasileiro Filipe Toledo.
$ ./parProc newsSurf.txt wc sort
2   33    220
Frederico Morais, Kikas, foi derrotado pelo Brasileiro Filipe Toledo.
Gabriel Medina ganhou pela segunda vez consecutiva, derrotando Julian Wilson na final.
MEO Rip Curl Pro Portugal, Supertubos, 25 de Outubro de 2017 --
```

Ou seja, será afixado o que cada um dos comandos "wc" e "sort" escreva. Note que o programa deve terminar só depois de ambos os comandos terminarem. Complete o esqueleto do código que se segue, de acordo com as alíneas seguintes (assuma que não ocorrem erros durante a execução):

a) Implemente a leitura do ficheiro e a escrita do seu conteúdo para cada comando. Cada comando é lançado pela função "launch" que devolve o descritor do canal ligado ao standard input desse processo.

b) Implemente, na função "launch", a criação e a execução de cada comando. Implemente também o código no processo pai que espera que ambos os comandos terminem.

c) Implemente, na função "launch", a criação do meio de comunicação e redirecções que permitam a comunicação (i.e. o envio do ficheiro) entre o processo pai e o standard input do processo filho.

Programação com threads

Determinado programa recorre a múltiplos threads para cada um efetuar determinado cálculo (*worker*). Tem também um thread (*reducer*) para somar todos os cálculos intermédios dos threads produzindo o resultado final.

a) Complete os espaços marcados com a) por forma a criar os threads necessários para executar as funções *worker* e *reducer*, como indicado nos comentários ao longo do código, e também declarar e reservar memória para as variáveis assinaladas.

b) Complete os espaços marcados com b) por forma a garantir que o thread *reducer* só calcula a redução dos resultados computados pelos threads *worker* depois dos últimos terem efetivamente computado o dito resultado. O thread *reducer* não pode utilizar espera ativa.

c) Complete os espaços marcados com c) por forma a retornar ao thread principal (*main*) o resultado da computação efetuada pelo thread *reducer*.