Exercício 4

Sistemas Operativos 2014-15
LEIC-A / LEIC-T / LETI
IST

Resumo

Este exercício consiste em fazer uma versão paralela do programa LEITOR, desenvolvido nos exercícios anteriores.

1 LEITOR paralelo 1

Os alunos devem desenvolver uma versão paralela do programa LEITOR, que é capaz de verificar a coerência de ficheiros de forma concorrente. Esta versão deve ser concretizada de forma semelhante ao ESCRITOR paralelo: existe um programa pai que lança 3 processos filhos, cada um dos quais fará a verificação de um ficheiro.

Pormenorizadamente:

1. O programa pai, chamado LEITOR-PAI, terá que lançar 3 processos filhos através da system call *fork* para verificar 3 ficheiros seleccionados aleatoriamente entre os 5 ficheiros pré-definidos ("SO2014-0.txt","SO2014-1.txt",...).

2. Os processos filhos devem usar a system call *exec*[1] para lançar o programa LEITOR. Pretende-se que o LEITOR execute a mesma verificação efectuada nos exercícios anteriores, mas que desta vez receba como parâmetro de input (através do "argv") o índice do ficheiro para verificar. Por exemplo, executando de shell:

   ```
   ./LEITOR 1
   ```

   o através da *system call execl*():

   ```
   execl("LEITOR","LEITOR","1",NULL)
   ```

   o programa LEITOR deverá verificar o ficheiro "SO-2014-1.txt".

3. Os ficheiros a serem lidos pelos filhos são escolhidos aleatoriamente pelo pai, que passa esta informação aos filhos através do método referido acima.

4. O programa pai terminará apenas depois dos seus 3 filhos se executarem, e terá que imprimir os valores que eles devolveram.

[1] Uma qualquer das variantes da *exec* será aceitável.

2 LEITOR paralelo 2

Os alunos devem desenvolver uma segunda versão paralela do programa LEITOR, que é capaz de verificar a coerência de 3 ficheiros de forma concorrente. Ao contrário da versão anterior, esta versão deve criar 3 fios de execução (*threads*) que se executam no mesmo espaço de endereçamento e que analisam ficheiros diferentes. O nome do ficheiro a analizar por cada thread terá que ser especificado como parâmetro de input na função `pthread_create`.

O programa deve terminar, depois dos seus 3 threads se executarem, e imprimir para cada thread o valor devolvido (ou seja -1 em caso de erros durante a verificação, e 0 caso contrário).

3 LEITOR paralelo 3

Nesta versão um leitor deve usar K *threads* paralelos para analisar o mesmo ficheiro — cujo nome deverá ser passado como parâmetro de input ao programa, como nas variantes anteriores.

Considere que L é o número de linhas do ficheiro. O objectivo é ter cada *thread* a verificar uma porção diferente do ficheiro, de tamanho L/K linhas (arredondado por defeito caso L não for múltiplo de K). Ou seja, o primeiro *thread* verifica as primeiras L/K linhas, o segundo *thread* as L/K linhas seguintes, e assim sucessivamente. Note que, se o número de linhas do ficheiro não for múltiplo de K, o último *thread* poderá ter mais linhas para verificar que os restantes.

O programa só deve terminar depois de todas as linhas serem verificadas.

4 Experimente

- Compare o desempenho do leitor paralelo 1 e do leitor paralelo 2.
- Compare o desempenho do leitor paralelo 3 com o desempenho de um leitor sequencial que verifique o mesmo ficheiro (pode ser interessante experimentar para diferentes tamanhos de ficheiros).

5 Submissão

Os alunos devem submeter um ficheiro no formato zip com o código fonte, uma Makefile, e um executável através do sistema Fénix. O exercício deve obrigatóriamente compilar e executar nos computadores dos laboratórios.

A submissão pode ser feita até às 23:59 do dia 14 de Novembro 2014.

6 Cooperação entre Grupos

Os alunos são livres de discutir com outros colegas soluçcoes laternativas para o exercício. No entanto, *em caso algum*, os alunos podem copiar ou deixar copiar o código do exercício. Caso duas soluções sejam cópias, ambos os grupos reprovarão à disciplina.