Enunciado do Projecto de Sistemas Operativos 2015-16
SHELL PARALELA - EXERCÍCIO 5

LEIC-A / LEIC-T / LETI
IST

1 Acesso remoto e redireção de saídas no par-shell

O 5º exercício pretende suportar o acesso à par-shell através de terminais remotos e a redireção do *output* dos comandos.

O desenvolvimento da solução deste exercício deve assentar no código que entregaram anteriormente (ou solução publicada pelo corpo docente caso optem por isso), sem a modificação da alínea extra realizada durante a aula de avaliação.

1.1 Redireção do *output* dos comandos

Pretende-se que o *standard output (stdout)* dos processos filho lançados pelo par-shell passe a ser redirecionado para ficheiros individuais na diretoria onde o par-shell foi lançado. Cada processo filho lançado deve ter o seu *standard output* redirecionado para um ficheiro chamado `par-shell-out-PID.txt`, em que PID se refere ao pid do processo filho.

1.2 Terminais remotos

Pretende-se permitir que diferentes utilizadores na mesma máquina introduzam comandos na par-shell através de diferentes terminais. Para tal, a par-shell deixa de ler comandos diretamente a partir do *standard input (stdin)*.

Em alternativa, a par-shell passa a ter um *pipe* com nome por onde recebe comandos. O pipe deve ter o nome `par-shell-in` e estar localizado na diretoria onde a par-shell foi lançada.

Devem ser implementado os seguintes pontos:

1. Deve ser criado um progama `par-shell-terminal` que, correndo num processo autónomo, lê ordens do teclado e as envia para o named pipe alvo. O *pathname* do *named pipe* é indicado como argumento na linha de comando do `par-shell-terminal`. Em cada momento poderá haver múltiplos processos a correr este programa e a enviar concorrentemente comandos para o mesmo par-shell.

2. O programa par-shell-terminal deve aceitar o comando `stats` que imprime, no ecrã onde corre o par-shell-terminal, o número de processos filho da par-shell que estão atualmente em execução e o tempo total de todos os processos que já foram executados na par-shell.

De notar que o número de processos e o tempo total são relativos aos comandos enviados por todos os terminais.

3. O programa par-shell-terminal deve aceitar o comando *exit* que permite sair do programa par-shell-terminal.

4. Quando a par-shell recebe o signal SIGINT, deve terminar todos os programas par-shell-terminal em execução. A terminação de cada process terminal deve ser feita por envio direto de *signal*, causando a terminação abrupta de cada processo terminal.

5. O programa par-shell-terminal deve aceitar o comando *exit-global* que permite terminar a par-shell e todos os terminais. A terminação dos processos deve ser feita ordeiramente. Isto é, o processo par-shell deve esperar que os comandos pendentes e em execução sejam completados e imprimir adequadamente o ficheiro `log.txt` antes de terminar.

2 Funções do Unix/Linux

Para este exercício sugerimos que sejam usadas as seguintes funções da API do Unix/Linux:

`unlink, mkinfo, open, close, dup, read, write, signal, kill`

3 Experimente

Para verificar a implementação dos requisitos siga o seguinte procedimento:

1. Abrir 4 terminais.

2. No terminal 1 lançamos o comando par-shell.

3. No terminal 2 verificamos que foi criado o ficheiro `par-shell-in`.

4. Em seguida no terminal 3 lançamos o comando `par-shell-terminal par-shell-in` e escrevemos `fibonacci 1000`.

5. No terminal 2 verificamos que foi criado um ficheiro `par-shell-out-PID.txt` em que PID é o identificador do processo que foi criado pelo comando `par-shell`. Verificamos que o conteúdo desse ficheiro é o output do comando `fibonacci`.

6. Em seguida no terminal 4 lançamos o comando `par-shell-terminal par-shell-in` e escrevemos `fibonacci 2000`.

7. Voltamos a verificar o ponto 5.

8. Finalmente, verificamos que por cada comando que introduzirmos no terminal 3 ou 4 será criado o correspondente ficheiro `par-shell-out-PID.txt`.

9. Voltar a executar um processo `par-shell` e dois processos `par-shell-terminal`.

10. Num dos terminais dar comandos para executar vários processos fibonacci que demorem alguns segundos.

11. Escrever stats num dos terminais que executa `par-shell-terminal`. Verificar que é imprimido o número de processos em execução e o tempo total (que inclue o tempo total da execução anterior).

12. Voltar a executar um processo par-shell e dois processos `par-shell-terminal`.

13. Escrever exit num dos terminais que executa par-shell-terminal. Verificar que todos os processos terminam.

14. Voltar a executar um processo par-shell e dois processos `par-shell-terminal`.

15. Fazer CTRL-C no terminal que corre o `par-shell`. Verificar que todos os processos terminam.

16. Voltar a executar um processo par-shell e dois processos `par-shell-terminal`.

17. Escrever exit-global num dos terminais que executa `par-shell-terminal`. Verificar que todos os processos terminam.

4 Entrega e avaliação

Os alunos devem submeter um ficheiro no formato zip com o código fonte e uma Makefile. O exercício deve obrigatoriamente compilar e executar nos computadores dos laboratórios.

A data limite para a entrega do exercício é 4 de Dezembro até às 23h59m do dia indicado.

A avaliação do exercício será feita na discussão final do projeto, na qual terão que estar presentes todos os elementos do grupo e onde será atribuida uma nota individual a cada elemento. A data da discussão será anunciada no site da cadeira.

5 Cooperação entre Grupos

Os alunos são livres de discutir com outros colegas soluções alternativas 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.