INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Licenciatura em Engenharia Eletrónica e Telecomunicações e de Computadores (LEETC)



**Sistemas Operativos**

**1º Trabalho**

**LT41D - Grupo 1**

Diogo Freixo Filipe (nº50387)

João Cardoso Santos (nº51009)

Rodrigo Roussado Viegas (nº50977)

1. Neste exercício foram, primeiramente, criadas as macros NUMBER\_OF\_CHILDS de forma a definir facilmente o número de processos filhos que vão ser criados pelo processo pai, N\_PRINTS para definir o número de vezes que os processos filhos vão fazer os *prints* e SLEEP\_TIME que dá o tempo de espera entre *prints* (em segundos).   
   De seguida foi usado um *loop for* para criar os vários processos através da chamada de sistema *fork*. Cada processo filho executa um *while* de forma a fazer os *prints*com um *sleep* de um segundo entre estes.
2. Este código inicia-se com um *fork* para criar o processo que vai executar o comando *date* (é necessário criar um processo filho para cada comando a executar porque a chamada de sistema *execlp* sobrepõe o código do processo com o código do comando a executar). Este processo segue para executar o comando enquanto o processo pai continua e cria o segundo processo filho (que vai executar o comando *ping* em seguida). Por fim, o processo pai espera que os outros finalizem com recurso à chamada de sistema *waitpid* antes de também este finalizar.

3.

Primeiramente, começa-se por abrir o *pipe* que vai permitir a comunicação entre os dois processos, parte essencial do código.

No contexto do processo filho, começa-se por fechar o descritor de ficheiro relativo ao *pipe* que não se vai utilizar (neste caso, o de escrita). De seguida temos uma leitura do *pipe*, enquanto este não for fechado pelo processo pai, que vai ler um inteiro, calcular o seu quadrado e fazer *printf* do valor resultante.

No contexto do processo pai, é fechado o descritor de ficheiro de leitura do *pipe*, prosseguindo para receber os valores inseridos pelo utilizador enquanto este não for EOF (ou não ser possível fazer a leitura) e passando-os (pelo *pipe*) ao processo filho.

4.

Neste programa, começamos por confirmar que foi passado pelo menos um ficheiro na linha de comando. Prosseguimos para abrir um *pipe* e fazer um *fork*.

No contexto do processo filho, fechamos o descritor de ficheiro do *pipe* que não vamos utilizar (neste caso, o de leitura) e duplicamos o descritor de ficheiro de escrita do *pipe* para o *standard output* para permitir que o resultado do comando *execlp* possa aí ser mostrado (visto que este vai sobrepor o código do processo com o código do comando “wc -w”).

No contexto do processo pai, é fechado o descritor de ficheiro de escrita do *pipe* e espera-se que o processo filho termine a sua execução de forma a garantir que os dados já estão no *pipe* no momento da leitura deste. Por fim, este faz *printf* do valor recebido através do pipe e fecha o descritor de ficheiro de leitura do *pipe*.

5.

Este código é semelhante ao apresentado no exercício 4. As diferenças centram-se no suporte a leitura de múltiplos ficheiros e contagens das palavras nestes presentes. É criado um processo filho através de um *for* para cada ficheiro passado como parâmetro possa ter as suas palavras contadas.

No contexto dos processos filhos a única mudança é que, agora, a seleção do ficheiro cujas palavras vão ser contadas é feita através de argv.

No contexto do processos pai, existe uma variável “*sum*” que vai acumular os valores das contagens realizadas pelos processos filhos e, posteriormente, apresentar o valor em *standard output*.

6.

7.

a) Verdadeiro

b) Falso

c) Falso

d) Verdadeiro (dúvida)

8.

a) Falso

b) Falso

c) Verdadeiro

d) (dúvida)