Sistemas Operativos
(2º ano de LCC)

Trabalho Prático

Relatório de Desenvolvimento

Bernardo Filipe Alves Rodrigues
(A79008)

Francisco Domingos Martins Oliveira
(A82066)

José Pedro Neto Faria
(A82725)

25 de Julho de 2021

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização

O presente relatório foi elaborado no âmbito do Trabalho Prático da Unidade Curricular de Sistemas Operativos, que se insere no 2º semestre do 2º ano de Ciências da Computação.

Neste Trabalho prático foi proposto a implementação de um serviço de monitorização de execução e de comunicação entre processos.

1.2 Resumo

Ao longo do relatório vamos explicar as funcionalidades do nosso serviço, tal como as estratégias que tomamos para resolver os problemas que nos foram apresentados.

Capítulo 2

Funcionamento

Para representar a estrutura Cliente-Servidor o nosso projecto tem dois processos que comunicam entre si via pipes com nome. De um lado **argus** , ou o cliente , que produz linhas comandos para o pipe **client to server** . Do outro lado temos o **argusb** a consumir o conteúdo do mesmo pipe , executa-lo e se necessário responder ao client pelo segunto pipe **server to client**. Segue-se uma lista de comandos suportados assim como exemplos da sua utilização :

- -m "10"
- -e "c1 — c2 — ... — cn"
- -l
- -t "1"
- -r
- -h

2.0.1 Cliente - argus

Modo bash

Quando invocado sem argumentos este programa cria dois processos :

- Um que lê linhas do stdin do terminal onde estiver a correr e escreve o resultado do parsing correspondente no pipe de comunicação com o servidor
- O outro lê as respostas do servidor do pipe cuja comunicação se dá no sentido oposto do ponto anterior e escreve no stdout do client se o comando que o cliente escolher assim o exigir.

  Note-se assincronia que existe entre os dois processos na medida em que não é preciso que haja uma resposta do servidor para o cliente enviar mais comandos e vice-versa. Para sair o utilizador deve fazer **Ctrl+C**.

2.0.2 Invocação com argumentos

Uma vez invocado com as extensões do enunciado como por exemplo **-e** ou **-l** após a execução do comando correspondente o cliente é imediatamente devolvido ao prompt normal.

2.0.3 Servidor - argusd

O servidor reage linhas com a sintaxe **-tag ["args"]** , qualquer comando passado que não obedeça a este formato é ignorado a nível de execução relembrasse o cliente do modo de utilização .

Capítulo 3

Comandos

3.1 Opção -e

No contexto deste comando fazemos dois **forks** ficando com um total de 4 processos para gerir a execução de um comando .

No fundo da hierarquia está o processo que que executa o ou os comandos que foram passados como argumento. Fazemos **exec** de um programa desenvolvido por nos que executa uma cadeia de tamanho arbitrário (maior ou igual a um) de comandos intercalados com pipes usando redireccionamento do output do comando anterior para o stdin do comando seguinte . Para mais detalhes encorajamos a consulta do código fonte do ficheiro **bashpipe.c**.

No nível seguinte temos o processo que faz a gestão das condicionantes á execução do seu filho. Após registar do **pid** da sua descendência num array global **pids** e contar o tempo que o seu filho está a demorar a decorrer enquanto ele não termina. Se o tempo de execução for excedido é enviado ao filho o sinal **SIGTERM** caso contrario ele encerra-se. Em ambos os cenários de terminação é acrescentada uma entrada no ficheiro **terminadas** com a informação adicional de como o processo terminou .

Neste nível , é onde é gerida a informação e registada nos ficheiros que o utilizador pode aceder com outras opções. Aqui é acrescentada uma entrada no ficheiro **decorrendo** onde se mantém indexadas os processos a decorrer criados pelo utilizador .

Por fim no topo de hierarquia temos o processo que prossegue com o trabalho do servidor e avança para a iteração seguinte do ciclo que lê comandos do pipe **client to argus** .

3.2 Opção -m

O comando **-m** apenas irá alterar a variável global **MAXTIME** pelo valor escolhido. Esta variável vai ser usada na execução de tarefas (**-e**) que, como previamente mencionado, irá contar o tempo enquanto até atingir o valor escolhido, se for esse o caso , irá terminar a tarefa.

3.3 Opção -l

O comando **-l** abre o ficheiro **decorrendo** que contém todas as tarefas a decorrer e escreve para o cliente.

3.4 Opção -t

O comando **-t** abre o ficheiro **decorrendo** e o ficheiro **temp** copiando todas as linhas do ficheiro decorrendo para o ficheiro temporário excepto a execução que se pretende terminar. Guarda essa execução no ficheiro **terminadas** e coloca-a a **0** no array global dos pids e por fim muda o nome do ficheiro temp para **decorrendo**.

3.5 Opção -r

Todas as execuções terminadas porque excederam o tempo , executaram ou foram terminadas pelo cliente são colocadas num ficheiro chamado **terminadas** . O comando **-r** abre o ficheiro **terminadas** que contém todas as tarefas terminadas e escreve para o cliente.

3.6 Opção -h

O comando **-h** apenas lista os comandos e como eles podem vir a ser utilizados .

Capítulo 4

Conclusão

A realização deste trabalho permitiu-nos perceber melhor o funcionamento e a aplicação dos temas abordados nas aulas de sistemas operativos e assim conciliar melhor a matéria, principalmente no que toca a definições formais de estruturas como pipes e relações entre processos pai-filho. De um modo geral podemos afirmar que fazemos um balanço positivo do nosso trabalho e que conseguimos concretizar os objectivos pretendido excepto a opção -i"que , embora toda a pesquisa feita pelo grupo, não foi possível realizar .