Projeto Prático de Sistemas Operativos

LEI

Relatório do Projeto Cloud Shell

2014/15

Grupo 34

Trabalho realizado por:

- Alexandre Lopes Mandim da Silva (A73674)
- Francisco José Torres Ribeiro (A70712)
- Rafael Mota Oliveira (A72307)

Índice:

Introdução:

Este trabalho foi realizado no âmbito da unidade curricular de Sistemas

Operativos. O principal objetivo do trabalho foi o de colocar em prática a matéria e os conhecimentos leccionados ao longo deste semestre.

Com este trabalho esperamos consolidar todos estes conteúdos, obter uma boa preparação para o teste de avaliação e fomentar o trabalho em equipa, aptidão fundamental no mundo do trabalho.

De modo a facilitar a implementação inicial do projeto e também devido ao tempo disponibilizado para a realização do mesmo optamos primeiramente por uma abordagem mais simples, que consistiu em ter apenas uma *CloudShell* (um utilizador). Numa fase mais avançada e após a nossa satisfação com o estado de funcionamento do projeto decidimos extender a solução para várias *CloudShell's.*

Nos capítulos seguintes procede-se uma explicação mais detalhada do trabalho, dos processos que o constituem e da maneira como os mesmos se relacionam.

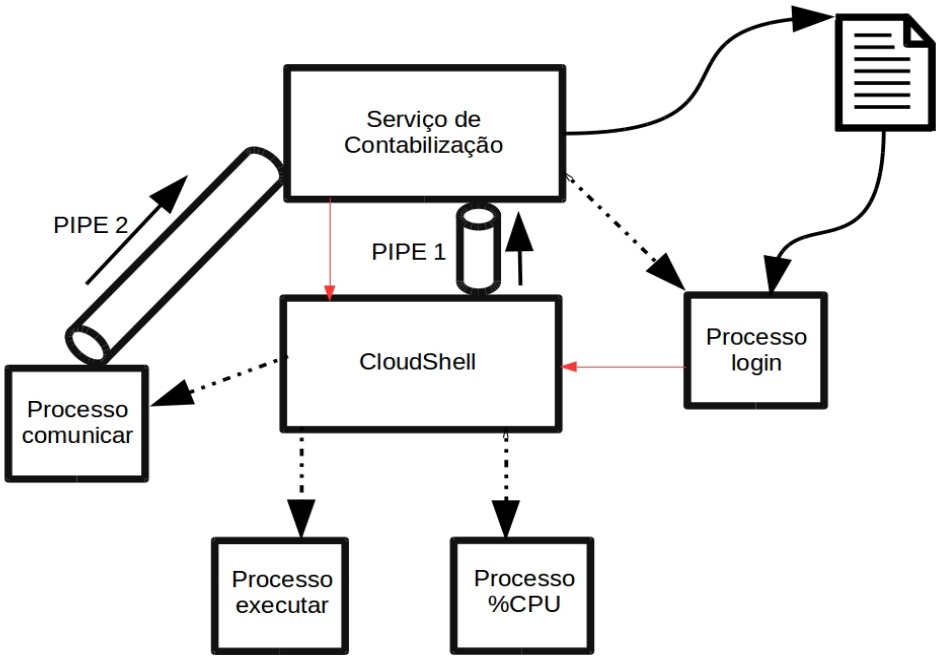
Estruturação do projeto:

Esta secção procura esclarecer sucintamente a arquitetura do projeto. Todos os processos, *pipes, signal's* e restante funcionamento do projeto será explicado com maior minuciosidade mais adiante.

O projeto é composto por dois processos principais: Servidor de Contabilização e o *CloudShell.*

O serviço de contabilização é responsável pela monotorização das *CloudShell's* dos utilizadores e respetivo processamento e armazenamento desses dados. Este serviço é também responsável pela criação de um processo auxiliar *login* que permite aos utilizadores iniciar sessão.

Em segundo lugar, o serviço de *CloudShell* comunica inicialmente com o processo de *login* e espera por um *signal* de aprovação/reprovação. Se este for aprovado a *CloudShell* cria um processo que vai ser responsável pela comunicação dos recursos consumidos, pelos processos a serem executados, com o Serviço de Contabilização. Após a criação do processo de comunicação, espera que o utilizador insira um comando para execução. Após este comando são criados dois processos no qual um executa o pretendido e o outro monitoriza os recursos gastos por este último.



CloudShell

Este processo é o responsável por permitir ao utilizador executar comandos.

- Processo *Login*

Quando executado é necessário a autenticação por parte do utilizador. Esta

autenticação é realizada pelo processo da *CloudShell* enviando através de um *pipe* com nome ao processo de *login* do servidor o nome e respetiva *password.* Após a verificação dos dados é enviado um sinal à *CloudShell* permitindo ou não o inicio de sessão do utilizador. Este início pode ser negado por não existir esse nome/*password* ou por não existir saldo suficiente.

- Processo Comunicador

Se a autenticação for bem sucedida, este processo (*CloudShell)* cria um processo filho (Comunicador) que vai servir para enviar os dados dos processos executados nesta *bash* ao serviço de contabilização.

- *Signal's* recebidos

Na *CloudShell* "apanhámos" três sinais que o Servidor de Contabilização envia à *CloudShell:*

- O *signal* de matar todos os processos criados devido a não existir saldo positivo;
- O *signal* de *login* incorreto que avisa o utilizador que o *login* está errado e termina o processo;
- O *signal* de l*ogin* correto que envia uma mensagem ao ecrã de boas-vindas e permite ao utilizador executar comandos.

- Processo Executar e Monitorizar %CPU

Finalmente a *CloudShell* fica á espera que o utilizador lhe envie comandos através do teclado. Quando recebidos a *CloudShell* cria um processo que executa o comando inserido e outro que monitoriza a CPU gasta por esse processo. Este último filho além de monitorizar esse recurso também é o responsável por enviar (de segundo em segundo) esses dados ao processo Comunicador através de um *pipe* anónimo. Este de seguida envia ao Servidor de Contabilização.

Servidor de Contabilização

Este processo é o responsável por receber a informação relativa aos gastos dos processos de todas as *CloudShell's,* processar essa informação, armazenando-a na *Heap* e num *File* e envio de sinais ás *CloudShell's.*

- Estrutura de dados e ficheiro

Para guardar a informação relativa aos utilizadores implementá-mos uma lista ligada de *USER* em que cada um destes contém um nome, uma *password* e um saldo e guardámos essa informação, periodicamente, num ficheiro.

Quando o Servidor de Contabilização é iniciado carregamos em memória central a informação contida no ficheiro.

- FIFO's

Quando o Servidor de Contabilização é iniciado, após carregar os dados dos *USER'S* em memória central são criados dois *pipes* com nome em que um espera que uma *CloudShell* envie um pedido de *login* e o outro recebe a informação dos processos que estão a ser executados nas *CloudShell's.* São criados também dois processos para evitar que os *FIFO's* fiquem sem "entradas" abertas e consequentemente terminem o Servidor de Contabilização.

- Processo *Login*

É criado um processo que espera por autenticações por parte das *CloudShell's*. Quando algum pedido é recebido faz-se uma nova leitura dos dados no ficheiro e é verificado se o utilizador existe, se a *password* está correta e se o mesmo tem saldo. Para cada um destes casos é enviado à *CloudShell* que fez o pedido, um *signal* a indicar qual das situações ocorreu.

- Leitura das percentagens da utilizacação de *CPU*

Após todos os passos indicados anteriormente o servidor fica à espera que alguma *CloudShell* lhe envie dados de CPU. Após a leitura destes dados atualiza o saldo do *USER* e guarda em memória central e no ficheiro os novods dados para que, caso exista um *crash* do Servidor de Contabilização, estes dados não sejam perdidos. Caso este seja nulo ou negativo é enviado um *signal* à *CloudShell* respetiva a indicar que não pode executar comandos até carregar o saldo.

Dificuldades encontradas

A primeira dificuldade com que nos deparámos foi com o uso do *pidstat*. Não foi uma dificuldade na interpretação e execução do comando em si, mas sim na complexidade em fazer *parsing* do *output* deste mesmo comando. Como este problema em nada refletia os conceitos abordados nesta unidade curricular mas sim no trabalho de fazer *parsing* de uma *string*, contando que o tempo era um bem escasso e o problema facilmente era contornado com a execução do comando *ps -p <pid> -o %cpu* %mem … que fornece um *output* muito mais simples optámos por esta solução visto que o resultado obtido era o mesmo.

Por fim, a segunda e última dificuldade foi na implementação de várias *CloudShell's* a comunicar com o Servidor de Contabilização. Inicialmente cada *CloudShell* fazia três *writes* em que enviava ao Servidor o *pid* da *CloudShell* em questão, a % de *CPU* gasta e o nome do utilizador. Isto poderia permitir que dois utilizadores tentassem comunicar com o Servidor de Contabilização ao mesmo tempo

e existir um conflito na receção dos *writes* das duas *CloudShell's* o que iria provocar uma interpretação incorreta de tipo de dados, por exemplo.

Para solucionar este problema optámos por definir uma organização diferente no envio dos dados ao Servidor de Contabilização, fazendo assim apenas um *write* por parte de cada *CloudShell*, de um *buffer* que no inicio contém um *float* da percentagem gasta pelo *cpu*, um *int* com o *pid* da *CloudShell* e por fim o nome do *USER.* Para isso usámos a função de *memcpy* disponibilizada na biblioteca *string.h.*

MakeFile

A *makefile* compila os dois ficheiros com o código em C.
make compila e cria os executavéis;
make iniciarServidorECliente inicia o servidor e uma *CloudShell;*

Conclusão

Para concluir, este trabalho correspondeu às expetativas mencionadas na introdução. Melhoramos os nossos conhecimentos relativos ao acesso de ficheiros (*read's* e *write's*), á gestão de processos (*fork's*, *wait's*, *exit's*, *pause*, ...) , execução de programas (família *exec*), redirecionamento de descritores (*dup2*), *pipes* anónimos (*pipe*), *fifo's* (*mkfifo*) e por fim sinais (*signal*, *kill*). Todos estes conhecimentos foram implementados no trabalho prático.

Finalmente é de salientar que o aprofundamento da matéria no projeto é uma mais valia para a preparação do teste final e consequente sucesso nesta unidade curricular.