Universidade do Minho

Escola de Engenharia

Departamento de Informática

LEI, 3º ano – Sistemas Operativos, 2014/2015

Grupo 10

CloudShell

Resumo

Este documento tem como objetivo esclarecer todo o processo de implementação da aplicação CloudShell, desafio proposto pelos docentes da unidade curricular de Sistemas Operativos. Ao longo deste relatório mesmo serão abordados diversos temas como a arquitetura da aplicação, as diversas decisões e os problemas enfrentados durante o desenvolvimento da solução.

Índice

Introdução

Este documento aborda a especificação de uma aplicação, CloudShell, que pretende imitar o comportamento de uma shell mas, contabilizando recursos gastos aquando a execução de processos por parte do utilizador.

O objetivo deste trabalho é fornecer uma aplicação deste género usando conhecimentos lecionados nas aulas práticas e teóricas desta unidade curricular para prática e ganho de experiência por parte da equipa que o implementa.

Este relatório estrutura-se da seguinte forma, numa primeira parte é apresentada a estrutura da aplicação, de seguida são explicadas todas as implementações e, por fim são apresentadas as conclusões deste trabalho.

Arquitetura da Aplicação

A arquitetura inicial da aplicação sofreu bastantes alterações ao longo do desenvolvimento do programa face a ideias erradas e alterações das decisões.

Assim sendo a arquitetural final é constituída por:

- Servidor: Responsável pela estrutura de dados que armazena a informação sobre o gasto do utilizador e, pela resposta de OK ou KO conforme ainda exista ou não saldo.

- Shell: Responsável pela execução de processos, monitorização dos seus gastos e envio dessa informação ao servidor, bem como, término dos processos caso seja recebida a informação de KO.

- Estruturas auxiliares: array de pid_t com os pids dos processos filhos a decorrer, hashtable com o gasto atual de cada processo (chave é o pid do processo).

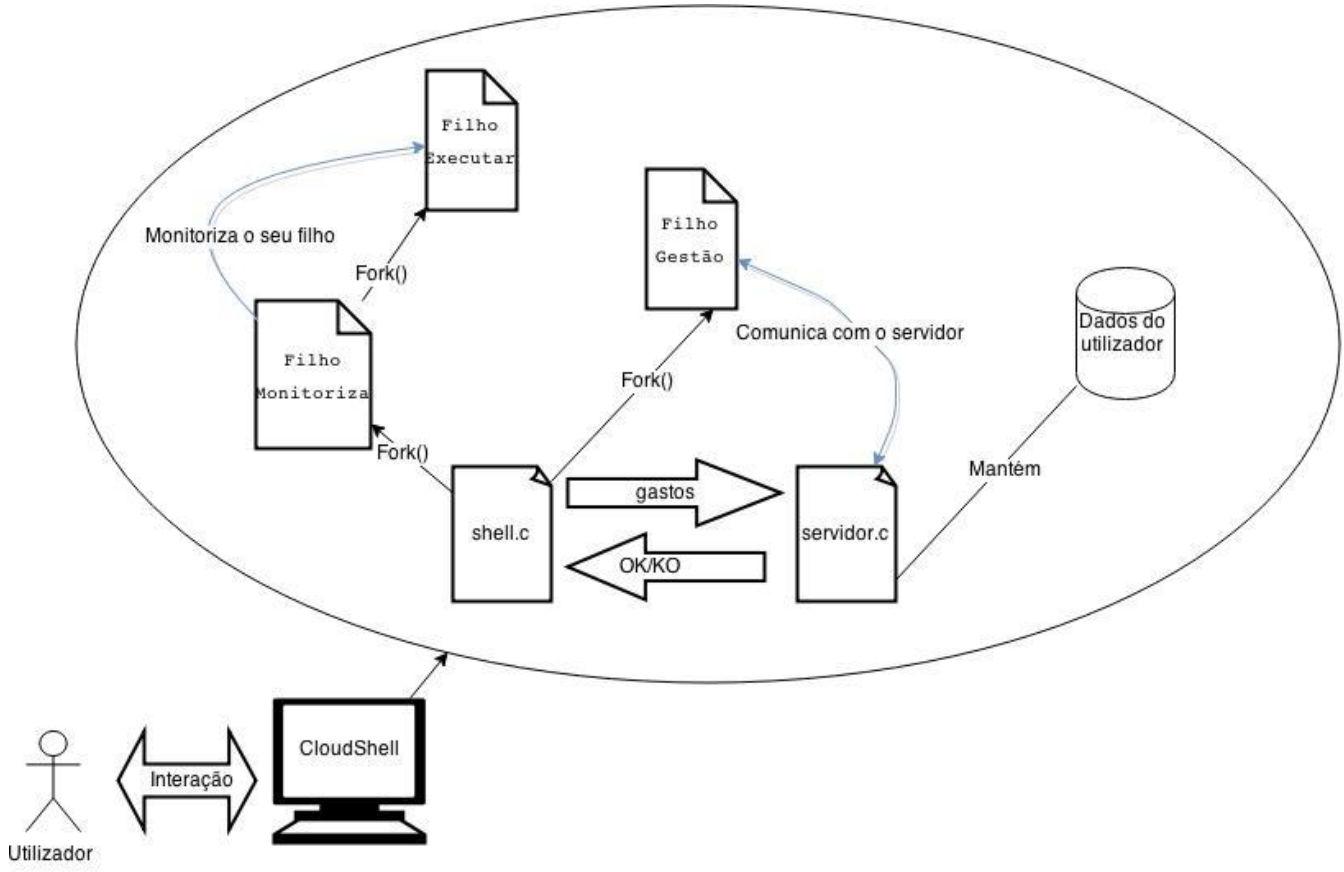

Figura 1 - Arquitetura da aplicação

Shell

A Shell logo de início cria um filho a que vamos chamar de Gestor, para de 10 em 10 segundos informar o servidor que deverá decrementar em x os recursos, de seguida cria ainda um outro filho que será encarregue da execução e monitorização dos gastos desse processo, o Monitorizador, este último é criado de cada vez que é recebido um comando para que seja feito o controlo a todos os comandos do utilizador sem bloqueios de funcionamento.

Gestor

Este filho acede às duas estruturas auxiliares, vai ao array de pids retirar os identificadores dos processos a decorrer e correspondentemente à hashtable verificar qual o seu gasto, depois de feita a soma destes valores, essa informação ao servidor por um named pipe e é recebida a resposta do servidor, caso esta seja OK nada é feito e o filho é posto “a dormir” fazendo um sleep(10), caso esta seja KO é lançado um SIGSTOP para que todos os processos sejam terminados.

Monitorizador

Por sua vez este filho cria um seu filho para a execução do comando e, mantendo o seu pid, através do uso de popen para invocação de um pidstat –p pid 15 e redirecionando o output deste para um ficheiro com o nome correspondente ao pid, está constantemente a analisar este ficheiro e a retirar os gastos do comando atualizando a estrutura auxiliar (hashtable). Note-se que o pid do processo que executa o comando é mantido e introduzido no array de pids.

Conclusão

Apesar de não termos conseguido cumprir tudo o que queríamos, por exemplo, aplicar o problema para vários utilizadores, é de notar que conseguimos utilizar adequadamente todos os conhecimentos obtidos e aprender na realização deste trabalho.

O facto de alguns pontos considerados extras não terem sido implementados prendeu-se com o curto espaço de tempo disponível para a sua especificação. Conseguimos, no entanto, aplicar quase todos os conceitos da matéria, foram utilizados pipes com nome, sinais, fork's, etc…