Notebook Interativo de Unix Shell

Relatório do trabalho prático da disciplina de Sistemas Operativos

Grupo 33.
Gonçalo Rui Alves Faria
Departamento de Informática
Universidade do Minho

Conteúdo

Capítulo 1

Introdução

À Unidade Curricular de Sistemas Operativos, foi proposta a realização de um interpretador de comandos. Este interpretador, tem de possibilitar documentar os diferente conjuntos de comandos da shell, conjuntamente com texto do utilizador.

Embora não seja alvo de avaliação, todo o projeto foi elaborado tendo em conta os conhecimentos adquiridos relativos aos mecanismos de modularização, assim como tipos abstrato de dados, para a elaboração de programas em C.

Capítulo 2

Arquitetura da solução

O programa é essencialmente constituído por um módulo, que representa o notebook e um outro, responsável por interpretar cadeias de comandos e guardar o output da sua execução em *pipes* anônimos.

2.1 Notebook

A estrutura notebook foi implementada sobre uma outra, que é uma variação contingua do tradicional *Growing array*. Cada entrada deste array, é um tipo de dados que, ou é uma entrada de texto ou é uma entrada de código. Adicionalmente, às entradas de código, está associada uma entrada, que regista no caso de ser uma sequência de comandos válida, o output desta.

Adicionalmente, a estrutura contém métodos que possibilitam a leitura e escrita, das diferentes entradas de ficheiros notebook. Para este efeito e outros, em todo o projeto, foi implementado um *buffer* para reduzir o número de chamadas ao sistema. O *buffer*, lê e guarda em memória central, um grande número de caracteres de um dado descritor, que posteriormente podem ser sucessivamente consultados linha a linha.

Foi decido implementar em memória central, em alternativa ao uso de ficheiros, dado que o tempo de acesso à memória é significativamente inferior. O *Growing Array* foi escolhido, pois o acesso constante a qualquer entrada no notebook, torna trivial obter a saída de qualquer um dos comandos já executados no notebook.

Embora não fosse requisito, foi implementado o uso do notebook em modo de reutilização, no entanto, está disponível apenas usando como argumento na chamada do executável, a sequência de caracteres '-p++'. Este modo, para além de continuar a permitir re-escrever o ficheiro notebook, é também criado um outro, que poderá ser acedido em versões futuras, para reutilizar o outputs de comandos idênticos.

Esta capacidade, foi implementada pois melhora bastante o já curto tempo de processamento. No entanto, são disponibilizados ao utilizador, os riscos adjacentes ao uso desta opção. Dada a dependência dos diferentes comandos, assim como a possibilidade de redirecionamento de input e de output, os resultados a que o notebook chega podem nem sempre coincidir com os esperados.

2.2 Interpretador de comandos

Para o efeito de interpretação dos comandos, não foi simplesmente proporcionada uma imitação da função *system* da biblioteca padrão. Em alternativa, foi concebida uma estrutura de análise de linguagem geral, que pode ser usada para obter o resultado semântico, de qualquer tipo de expressão, numa dada linguagem.

Desde que indicados os símbolos dos operadores binários, as funções que calculam o valor semântico, a sua hierarquia e a função do cálculo das fórmulas atómicas, a estrutura consegue calcular o valor semântico de todas as expressões da dada linguagem.

Capítulo 3

Árvore sintática

A estrutura de análise semântica de expressões, é composta por uma árvore e um tabela de sintaxe. A árvore, facilita o acesso às diferentes expressões, assim como torna explícitas as suas relações. A tabela, não só faz a correspondência entre os símbolo e a função de cálculo semântico dos diferentes operadores, como também define a sua hierarquia.

A aplicação desta estrutura, para a tarefa de interpretar os comandos e operadores da unix shell, queriu a implementação das respetivas funções de cálculo semântico. Os símbolos, assim como a hierarquia escolhida, encontra-se na tabela embaixo, decrescente no grau de prioridade.

Operador sequencial	;
Operador paralelo	&
Operador conjunção lógica	&&
Operador pipe	|
Operador disjunção lógica	||

3.1 Construção

O processo de construção da árvore, é naturalmente recursivo devido à natureza indutiva do tipo de dados.

Inicialmente, procura-se palavra a palavra, a sequência de caracteres que representa o operador no cimo da hierarquia. Enquanto esta operação é efetuada, a expressão inicial fica naturalmente dividida em duas expressões mais pequenas. A expressão da frente, que contém todas as palavras que já foram analisadas, e a expressão detrás, que contêm as palavras ainda por analisar.

Na eventualidade do operador em questão ser identificado, à expressão da frente, que como sabemos não contêm o operador, é aplicada a análise do próximo operador da hierarquia. À expressão detrás, é novamente aplicada a função com o mesmo operador, visto que ainda não foi analisada e é inteiramente possível que este ainda nela se encontre.

Adicionalmente, um nó é criado e é lhe associado o índice na tabela de sintaxe do operador que dividiu as expressões. À subárvore esquerda, é atribuido o resultado da árvore sintática da expressão da frente, e à subárvore da direita, o resultado da árvore sintática da expressão detrás.

Este processo vai se repetindo recursivamente, para todos os operadores da hierarquia, tornando as expressões cada vez mais pequenas, até que fiquem compostas unicamente por fórmulas atómicas.

3.2 Avaliação semântica

O processo de avaliação semântica, depende na sua grande maioria, das funções de cálculo semântico introduzidas inicialmente na tabela sintática.

A função de avaliação semântica, recebe a raiz da árvore como parâmetro, e verifica se o nó desta é do tipo das fórmulas atómicas, ou se é um nó operador. Caso seja um nó operador, executa esse nó com o operador indicado na tabela de sintaxe, caso contrário, executa a função de avaliação de fórmulas atómicas.

3.2.1 Fórmulas atómicas

Para esta aplicação de interpretação dos comandos unix shell, para o cálculo das fórmulas atómicas, foi introduzida uma função, que não só usa as funções de chamada ao sistema da família *exec*, onde são indicados os executáveis seguidos dos seus argumentos, como também são avaliados os operadores de redirecionamento, indicado na tabela em baixo.

Redirecionamento do descritor 1	$>$
Redirecionamento do descritor 0	$<$
Redirecionamento do descritor 1 em apéndice	$>>$
Redirecionamento do descritor com o número n	$n>$

Os comandos de redirecionamento foram implementados usando as funções *fork* e *dup2* . Sendo que para o operador $>>$, em vez de proceder-se à abertura do ficheiro indicado no modo de truncamento, foi usado o modo de apêndice.

3.2.2 Operadores

Sequencial

Este operador, primeiro executa a função de avaliação semântica no nó esquerdo. De seguida, espera pelo sinalizador de sucesso ou insucesso e executa a função de avaliação semântica no nó direito. Após o sinalizador desta, termina a avaliação.

Paralelo

O operador paralelo, faz essencialmente o mesmo que o operador sequencial, no entanto, em vez de esperar pela avaliação do nó esquerdo, executa a avaliação de ambos os nós, e espera pelo resultado sinalizador dos dois no final.

Conjunção lógica

Este operador, executa a avaliação do nó esquerdo, e direciona o descritor 1 do processo para um *pipe* anónimo. Após a espera do resultado sinalizador, no caso de sucesso, procedesse com a execução da avaliação do nó direito da mesma forma, e para o mesmo *pipe*. Após a obtenção do resultado, e se ambos os nós foram processados com sucesso, o *pipe* que contêm os resultados de ambos, é escrito no descritor 1.

Pipe

O operador pipe, executa a função de avaliação semântica no nó da esquerda, no entanto, guarda toda a informação que o processo escreve no descritor 1, num *pipe* anónimo usando o comando *dup2*. De seguida, direciona o conteúdo deste *pipe*, para o descritor 0 do processo que executa a função de avaliação do nó direito. No final, espera o resultado de sinalização destas duas chamadas, e termina a avaliação.

Disjunção lógica

Este operador, executa a função de avaliação semântica no nó da esquerda e espera que esta termine. Se o resultado sinalizador indicar sucesso, a função termina a avaliação, caso contrário, é executado o nó da direita.

Capítulo 4

Manuseamento de interrupções e comandos errados

4.1 Comandos errados

Para tratamento de possíveis erros ao avaliar as expressões, foi introduzido um canal de comunicação com o processo principal.

Criado no início do programa, este canal implementado na forma de *pipe* com nome, é lhe dedicado um processo que o abre no modo de leitura.

Sendo que nenhum processo abriu o canal no modo de escrita, o processo entra em *deadlock* até que isso aconteça. O objetivo, é o de usar a remoção deste processo de *deadlock*, como forma de indicar que um erro ocorreu num dos comandos, pois assim que saia de *deadlock* o processo envia uma mensagem de interrupção ao programa principal.

Adicionalmente, devido aos operadores de disjunção e de conjunção, foi concebida uma forma de propagação de exceções, que indica se ao encontrar um erro, a função de avaliação semântica deve lidar com este, ou propaga-lo a chamadas subsequentes . Este mecanismo foi necessário, pois o operador de disjunção, possibilita a uma das suas sub expressões ser inválida, o que seria impossível, se assim que identificado um erro este o trata-se.

4.2 Interrupções do utilizador

Para lidar com possíveis interrupções do utilizador, foi usada a função *signal*, que associa a um dado sinal, uma função que opera sempre que este é chamado.

A função definida, é uma que remove o *pipe* com nome criado, no início do programa, e após enviar uma mensagem, termina sem escrever o ficheiro notebook que se encontra a ser interpretado.