Relatório do Projecto de Sistemas Operativos II

Implementação do Interpretador de comandos minha shell(msh)

INTRODUÇÃO

Uma shell é um programa compilado que disponibiliza uma interface para o utilizador poder fazer invocações ao sistema operativo (kernel). Esta interface consegue interpretar comandos e invocar outros programas através de uma linha de comando, disponibilizando uma prompt para o utilizador pode inserir os comados.

Pretendemos desenvolver uma shell simples cujos o nome é minha shell (msh), que irá fornecer uma interface ao utilizador para o sistema operativo. A shell irá permitir a execução de qualquer comando do sistema operativo Linux, em como possibilitar o uso desses comandos com redirecionamento da saída desses comandos para arquivos ou mesmo redirecionar o conteúdo de um arquivo como entrada para o comando. Também possibilitar a execução desses comandos quer bem background quer em foreground.

Esquema genérico da implementação do shell

```
Inicio algoritmo

criar arquivo vazio
ler comando
se comando igual exit  sai do shell

faça
        se comando esta vazio escreva “comando vazio”
        senão faça
                se comando está em background faça
                        executar em segundo plano
                verificar redirecionamento do comando lido
                escolha
                        caso não tem redirecionamento
                                executa comando simples
                        caso redirecionamento de entrada
                                separar comando e ficheiro
                        caso redirecionamento de saída
                                separar comando e ficheiro
                fimescolha

                se comando igual a historia
                        ler arquivo msh.hist
                senão se comando igual !!
                        executa comando anterior
                senão
                        executa o comando

                limpar variáveis
                ler comando


até comando diferente de exit

fim algoritmo
```

Shell básica(comandos básicos)

Para fazer a minha shell executar qualquer comando básico precisamos passar pelos seguintes processos:

1º Ler o comando

Onde utilizamos principalmente a função fgets para ler o comando lido e colocar ele num vetor de caracteres (strings).

2º Criar um processo filho

Onde criamos um processo filho com o uso da função **fork()** e atribuímos o seu pid numa variável. Caso o pid for maior que 0 estamos na fase de execução do processo pai e neste caso colocamos ele em espera até o filho terminar a sua execução. Mas caso ele seja igual a zero estamos na fase do processo filho e neste ca

Shell básica(comandos básicos)

3º Fazer o processo filho executar o comando lido

Para fazer o filho executar um comando especifico utilizamos principalmente a função nos permite copiar o código de um programa especifico para o código do nosso programa corrente alterando assim a sua execução. Ele possui a seguinte sintaxe:

Onde filename é o campo onde entra o nome do comando que queremos executar e argv é um vector que contém tanto o nome do comando bem como os seus parâmetros digitados pelo usuário.

Assim, para executar o comando lido no passo 1 com o execvp precisamos usar o novo processo que foi criado no passo 2, após o processo estar criado podemos executar o comando com o execvp colocando o comando e os argumentos do comando lido em seus respectivos lugares no vector e chamando o vector dentro da função para pegar o nome do comando e a seguir pegar os seus parâmetros digitados.

Redirecionamento(Entrada/Saída)

O redirecionamento permite fazer com que a minha shell canalize a saída de um comando para um arquivo (**redirecionamento de saída**) utilizando o simbolo **>** ou canalize o conteúdo de um arquivo como entrada para um comando(**redirecionamento de entrada**) utilizando o símbolo **<**.

Para conseguirmos fazer com que a minha shell execute comandos com redirecionamento tanto de saída como de entrada seguimos os seguintes passos:

1º ler comando

2º Verificar comando lido possui

Usamos a **função strchr** que verifica se uma dada string possui um caracter específico em sua composição. Fizemos:

Se string comando lido tem **>** então comando possui redirecionamento de saída

Senão se string comando lido tem **<** então comando possui redirecionamento de entrada

Senão se string comando lido não tem nem < nem > então comando não possui redirecionamento e deve ser executado de forma simples.

Redirecionamento(Entrada/Saída)

3º Separar comando do ficheiro

Aqui caso o comando lido tenha algum tipo de redirecionamento precisamos separar o comando lido nas suas componentes comando e ficheiro onde será redirecionado o comando. Para tal usamos a função **strtok** que nos permite separar campos de uma dada string. Depois de terminada está etapa teremos o comando a ser executado na variável comando e na variável ficheiro o nome do ficheiro a que se deve redirecionar a saída ou entrada do comando.

4º Executar o comando com base no tipo de redirecionamento

Com base no tipo de redirecionamento escolhemos se precisamos fazer a abertura de um ficheiro para escrita (redirecionamento de saída) ou fazer a abertura do ficheiro para leitura (redirecionamento de entrada). Desta forma fizemos:

Caso comando possui redirecionamento de saída

1º Abrimos o ficheiro com o nome descrito pelo usuário em modo de escrita

2º Executamos o comando descrito pelo usuário

3º Usamos a função **dup2** para fazer com que o comando executado não seja imprimido na tela mais sim no ficheiro aberto no passo 1;

História

A solução que encontramos para a História foi muito simples, para guardar os comandos fizemos:

Para Guardar um Comando no ficheiro msh.hist:

Ler comando

Se comando é diferente historia executa o comando

Abrir ficheiro msh.hist em modo de escrita

Escrever comando lido no final do ficheiro msh.hist

Fechar o ficheiro msh.hist

Ler comando

Para Ler o conteúdo do ficheiro msh.hist seguiu-se os passos:

Ler comando

Se comando igual a historia

- Abrir o ficheiro msh.hist em modo leitura
- Ler conteúdo de msh.hist e imprimir na tela
- Fechar o ficheiro msh.hist

Ler comando

Execução de comandos(Background/Foreground)

Um comando pode ser executado em dois planos numa shell normal, em foreground(plano principal) ou em background (plano secundário).

A execução normal de um comando e quase sempre em foreground, sendo que ele se caracteriza pelo falta do uso do operador & no final do comando.

Por default a msh que criamos sempre executa os comandos em foreground, a não ser que o usuário digite & no final do comando fazendo que seja interpretado assim que o comando deve ser executado em background ou segundo plano.

Para executar um comando em foreground seguimos os passos seguintes:

Ler comando

Se comando possui & então tipo de plano igual a background

Senão tipo de plano igual a foreground

Criar novo processo filho

Se pid filho > 0 então estamos no pai

Se plano igual a foreground

Pai espera processo filho terminar sua execução

Senão se plano igual a background

Pai envia sinal de stop no filho

Se próximo comando lido igual a fg pai envia sinal de continuar no último filho

Se pid == 0 então estamos no filho

Filho executa o comando lido

Execução de comandos(Background/Foreground)

Está estratégia resume-se a quando em foreground: criar um processo filho e colocar o pai a esperar enquanto o filho ainda não terminou sua execução. Filho executa o comando solicitado por meio da função execvp e finaliza a sua execução voltando ao pai que deixa de esperar e vai ler o próximo comando digitado.

Já na execução em background, o msh cria um processo filho, mas tão logo o filho é criado o processo pai envia um sinal de stop para ele, colocando-o em background, assim o pai continua rodando e o filho é pausado até que seja digitado o comando fg(foreground) que retira o último processo filho da lista de processos em background.

Conclusão

Este trabalho revelou-se bastante desafiador, pois criar o nosso próprio interpretador de comando exigiu ter forte domínio de áreas como criação de processos em sistemas Linux, entender o modo como os comandos são executados no sistema operativos, entender a família exec sendo que a ele junto da função fork se deve quase toda a flexibilidade que o sistema oferece para executar programa. Entender também como funciona o redirecionamento de comandos no sistema para que nos conseguíssemos recriar o mesmo no msh, perceber o uso de funções do C como strtok, strchr, e funções do Linux como o dup2, bem como as funções de manuseio de ficheiros que nós estudamos no decorrer do semestre como a open, a write , read e etc.