Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Sistemas Operacionais
Trabalho Prático #1 – Shell Básico

1 Introdução

Nesse trabalho você irá explorar alguns conceitos da primeira parte da disciplina. Em particular, os conceitos de pipes e estruturas de processos de kernel, onde você se familiarizará com a interface de chamadas de sistema do Linux implementando algumas funcionalidades de um shell bem simples.

Para que você foque apenas na parte de chamadas de sistema, abra o arquivo fonte do shell `sh.c` e estude-o. O esqueleto do shell contém duas partes: um processador de linhas de comando e o código para execução dos comandos. Você não precisa modificar o processador de linhas de comando, mas deve completar o código para execução dos comandos. O processador de linhas só reconhece comandos simples como:

```
ps aux > log.txt
cat < log.txt | sort | uniq | wc > resultado.txt
cat resultado.txt
rm resultado.txt
ls | grep .c | wc
rm log.txt
```

Se você não entende o que esses comandos fazem, estude o manual de um shell do Linux (por exemplo, do bash) bem como o manual de cada um dos comandos acima (`ls`, `cat`, `rm`, `sort`, `uniq`, `wc`, `ps`, ...) para se familiarizar. Copie e cole esses comandos num arquivo, por exemplo, `teste.sh`.

Você pode compilar o esqueleto do shell rodando:

```
$ gcc sh.c -o myshell.out
```

Nota: Nesta especificação colocamos um sinal de dólar (`$`) antes das linhas que devem ser executadas no shell do sistema (por exemplo, o bash). As linhas de comando sem dólar devem ser executadas no shell simplificado que você está implementando.

Esse comando irá produzir um arquivo `a.out` que você pode rodar:

```
$ ./myshell.out
```

Alternativamente, você pode compilar e testar seu TP de forma automática rodando:

```
$ make
```

A princípio, ambos irão falhar, pois você ainda não implementou várias funcionalidades do shell. É isso que você fará nesse trabalho.

2 Correção da Função `fork1`

Antes de prosseguir para a implementação dos comandos do shell, é necessário corrigir a função `fork1` na Tarefa 1. Essa função é essencial para a criação de novos processos.

Após corrigir `fork1`, teste a função compilando e executando o shell para garantir que não há falhas antes de continuar para a próxima seção.

3 Executando Comandos Simples

Implemente comandos simples, como:

```
| ls
```

O processador de linhas já constrói uma estrutura `execcmd` para você, a única coisa que você precisa fazer é escrever o código do `case ' '` (espaço) na função `runcmd`. Depois de escrever o código, teste a execução de programas simples como:

```
ls
cat sh.c
```

Nota: Você não precisa implementar o código do programa `ls` e dos outros. O que você deve fazer é implementar as funções no esqueleto do shell simplificado para permitir que ele execute comandos já existentes no sistema, como acima.

Dica: dê uma olhada no manual da função `exec` (`$ man 3 exec`). Importante: não use a função `system` para implementar as funções do seu shell.

4 Redirecionamento de Entrada e Saída

Implemente comandos com redirecionamento de entrada e saída para que você possa rodar:

```
echo "Sistemas Operacionais é legal" > x.txt
cat < x.txt
```

O processador de linhas já reconhece `>` e `<` e constrói uma estrutura `redircmd` para você. Seu trabalho é apenas preencher o código na função `runcmd` para esses casos. Teste sua implementação com os comandos acima e outros comandos similares.

5 Sequenciamento de Comandos

Implemente pipes para suportar comandos como:

```
ls | sort | uniq | wc
```

O processador de linhas já reconhece o `|` e constrói uma estrutura `pipecmd` para você. Você precisará completar o código para o `case '|'` na função `runcmd`. Teste sua implementação para o comando acima.

Dica: Consulte o manual das funções pipe, fork e close.

6 Entrega e Esclarecimentos

1. Este trabalho poderá ser realizado em dupla.
2. Não use a função `system` na sua implementação! Use `fork` e `exec`.
3. Esse trabalho utiliza obrigatoriamente a linguagem C.

4. Seu grupo deverá submeter no Moodle somente o arquivo `sh.c`, garantindo que o nome seja exatamente `sh.c`, para que o shell possa ser testado automaticamente. Qualquer outra forma de submissão será desconsiderada.

5. Instruções sobre como gerar um relatório para o trabalho estão contidas no próprio arquivo `sh.c`.

6. Seu shell será testado com um script similar ao `grade.sh` disponibilizado na especificação. A saída será conferida automaticamente. Por causa disso, seu shell deve imprimir somente a saída dos programas em casos onde não ocorre erro. Use o script `grade.sh` disponibilizado para verificar a corretude de sua implementação.

7. Submissões serão aceitas tanto em português como em inglês.

8. Comece a fazer esse trabalho o antes!