Universidade Federal de Minas Gerais

DCC605: Sistemas Operacionais

Trabalho Prático 0

Execução

Execução: individual
Valor: 5 pontos

Introdução

Neste trabalho você se familiarizará com a interface de *chamadas de sistema* do Linux implementando algumas funcionalidades num shell simples. Para que você foque apenas na parte de chamadas de sistema, baixe o esqueleto do shell e o estude. O esqueleto do shell contém duas partes: um processador de linhas de comando e código para execução dos comandos. Você não precisa modificar o processador de linhas de comando (a não ser que queira implementar algumas das atividades extra abaixo), mas deve completar o código para execução dos comandos. O processador de linhas só reconhece comandos simples como:

```
ls > y
cat < y | sort | uniq | wc > y1
cat y1
rm y1
ls | sort | uniq | wc
rm y
```

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`) 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
```

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:

```
$ ./a.out
```

Para sair do shell simplificado aperte `ctrl+d` (fim de arquivo). Teste o shell executando os comandos no arquivo `teste.sh`:

```
$ ./a.out < teste.sh
```

Essa execução irá falhar pois você ainda não implementou várias funcionalidades do shell. É isso que você fará nesse trabalho.

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 execução de programas simples como:

```
ls
cat sh.c
```

Nota: Você não precisa implementar o código do programa `ls`; você deve simplesmente implementar as funções no esqueleto do shell simplificado para permitir que ele execute comandos existentes no sistema, como acima.

Se ainda não conhecê-la, dê uma olhada no manual da função `exec` (`$ man 3 exec`). **Importante: *não* use a função `system` para implementar as funções do seu shell.**

Redirecionamento de entrada/saída

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

```
echo "DCC605 is cool" > 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.

Dica: Dê uma olhada no manual das funções `open` e `close` (`man 2 open`). Se você não conhece o esquema de entrada e saída padrão de programas, dê uma olhada no artigo da Wikipedia aqui.

Sequenciamento de comandos

Implemente *pipes* para que você consiga rodar comandos tipo

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

O processador de linhas já reconhece `'|'` e constrói uma estrutura `pipecmd` pra você. A única coisa que você precisa fazer é completar o código para o case `'|'` na função `runcmd`. Teste sua implementação para o comando acima. Se precisar, leia a documentação das funções `pipe`, `fork` e `close`.

Esclarecimentos

1. Não esqueça de responder à pergunta na tarefa 1 presente no esqueleto do shell.
2. **Não use a função `system` na sua implementação.** Use `fork` e `exec`.

Extras

Cada um dos extras abaixo valem um ponto extra:

1. Implemente sub-shells instanciadas com parênteses.
2. Implemente suporte a `&` (execução em background) e `wait`.
3. *Pull requests* com eventual *merge* no repositorio Git valem ponto.

Correção

Seu shell será testado com o script `grade.sh` disponibilizado no repositório. 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.

Seu código será inspecionado manualmente. Para facilitar o trabalho do professor, seu código será extraído do código do shell usando o script `getmarks.py` que retira o texto entre os marcadores no texto (`MARK START` e `MARK END`). Por causa disso, entregue o código em um arquivo único e não retire os marcadores de sua solução final. Verifique se todo o seu código está sendo extraído rodando o script no seu código.

Entrega

Cada grupo deve imprimir, preencher, assinar e entregar o relatório de execução do trabalho contendo o termo de compromisso. Seu grupo deve submeter no Moodle *apenas* o arquivo `sh.c`, em um (único) arquivo chamado `sh.c` (o nome deve ser exatamente `sh.c` para que seu shell possa ser testado automaticamente).