Programação em *shell*

Entrada, saída e *pipelines*

André Grégio, Fabiano Silva, Luiz Albini e Marcos Castilho

Departamento de Informática – UFPR, Curitiba/PR

14 de julho de 2020

Objetivos

- Apresentar o conceito de entrada e saída em *bash*
- Apresentar o conceito de *pipelines*

Entrada e saída no UNIX

- Ideia simples e brilhante com enorme implicação
  - Entrada e saída padrão
  - Redirecionamento de entrada e saída
  - Pipelines

Entrada e saída padrão

- Entrada padrão: por default é o teclado
- Saída padrão: por default é o monitor de vídeo
- Saída padrão de erros: por default é o monitor de vídeo

Exemplo

O comando `cat` lê linhas da entrada padrão e imprime na saída padrão até que seja digitado um ^D (control-D).

```
ci1001@fradim:~/tmp$ cat
Alo mamae!
Alo mamae!
Hello world
Hello world
^D
ci1001@fradim:~/tmp$
```

Redirecionamento de entrada e saída

- <: redireciona a entrada padrão
- >: redireciona a saída padrão
- 2>: redireciona a saída padrão de erros
- &>: redireciona a saída padrão e a saída padrão de erros
- >>: redireciona a saída padrão, mas apensa, ao invéz de sobrescrever

Exemplo

Considere que existe um arquivo cujo conteúdo são duas linhas: Alo mamae! e Hello world:

```
ci1001@fradim:~/tmp$ cat arquivo.txt
Alo mamae!
Hello world
ci1001@fradim:~/tmp$
```

Podemos copiar este arquivo em outro sem usar o comando cp

```
ci1001@fradim:~/tmp$ cat < arquivo.txt > arquivo.copia
ci1001@fradim:~/tmp$ cat arquivo.copia
Alo mamae!
Hello world
ci1001@fradim:~/tmp$
```

Um exemplo bacana

Aqui tentamos listar três arquivos, um deles não existe e o `ls` acusa erro na tela:

```
ci1001@fradim:~/tmp$ ls teste{,.pub,.old}
ls: não é possível acessar 'teste.old': Arquivo ou diretório não encontrado
teste  teste.pub
```

Aqui direcionamos a saída padrão de erros para um arquivo erro.txt. O `ls` exibe os arquivos existentes e cria um novo arquivo contendo a mensagem de erro.

```
ci1001@fradim:~/tmp$ ls teste{,.pub,.old} 2> erro.txt
teste  teste.pub
ci1001@fradim:~/tmp$ cat erro.txt
ls: não é possível acessar 'teste.old': Arquivo ou diretório não encontrado
```

Pipelines

- Um *pipe*, denotado |, pode ser traduzido como *tubo*
- É uma maneira de ligar a saída de um programa para a entrada padrão de outro programa.
- Dois ou mais programas conectados por *pipes* é um *pipeline*
- É um conceito muito elegante no UNIX, permite, por exemplo, aplicação de filtros variados até produzir a saída desejada pelo usuário.
- Um exemplo para três comandos:
  `comando1 | comando2 | comando3`

Exemplo

- `grep` é um programa que filtra linhas de arquivos
- `wc` é um programa que conta linhas de um arquivo quando usada a opção `-l`
- Suponha que queiramos contar quantas funções foram definidas em um programa em *Pascal*.

O comando *grep*

Este grep basicamente filtra as linhas que contém um padrão

```
aula28$ grep write floodfill.pas
writeln (nmov,'/',MAX_MOVIMENTOS);
write (m[i,j]:3);
writeln;
writeln ('Parabens, voce ganhou em ',n_mov,' movimentos!!!')
writeln ('Tente novamente...');
```

Exemplo

Este grep imprime apenas as linhas que contém a palavra function no início de uma linha

```
aula28$ grep ^function floodfill.pas
function ler_cor (jogo: tipo_jogo): integer;
function acabou (jogo: tipo_jogo): integer;
function inunda_vizinho (var jogo: tipo_jogo;cor_velha,cor_nova,x,y: integer): b
function testa_vitoria (jogo: tipo_jogo): boolean;
function distancia (e1,e2: elemento): real;
function sorteia_cor (jogo: tipo_jogo;cor_velha: integer): integer;
function escolhe_cor (jogo: tipo_jogo): integer;
aula28$
```

Exemplo

Poderíamos direcionar esta saída para um arquivo e depois usar o `wc`:

```
aula28$ grep ^function floodfill.pas > saida.txt
aula28$ wc -l saida.txt
7 saida.txt
```

Exemplo

É desnecessário ter que criar o arquivo temporário, basta usar o *pipe* logo após a saída do `grep`:

```
aula28$ grep ^function floodfill.pas | wc -l
7
```

Outro exemplo

- O comando `cut` seleciona colunas em um arquivo
- O comando `sort` ordena arquivos
- Suponha que queiramos imprimir os nomes completos, de forma ordenada, que aparecem na segunda coluna deste arquivo chamado `dados.txt`:

```
fulano: Fulano de tal: 1968
beltrano: Beltrano da silva: 1934
sicrano: Sicrano de Souza: 1945
```

Solução

Usamos o `cut` com as opções `-d:` (define o separador como `:`) e `-f2` (imprime a segunda coluna), mas a saída está fora de ordem.

```
ci1001@fradim:~/tmp$ cut -d: -f2 dados.txt
 Fulano de tal
 Beltrano da silva
 Sicrano de Souza
```

Ligando esta saída com um *pipe* para o comando `sort`:

```
ci1001@fradim:~/tmp$ cut -d: -f2 dados.txt | sort
 Beltrano da silva
 Fulano de tal
 Sicrano de Souza
```

Conclusão

- Apresentamos os conceitos de entrada e saída
- Incluindo *pipelines*, conceito muito importante
- Na próxima videoaula veremos processos, *foreground* e *background*

Licença

- Slides feitos em LaTeX usando beamer
- Licença

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.http://creativecommons.org/licenses/by-nc-nd/2.5/br/

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.http://creativecommons.org/licenses/by-nc-nd/2.5/br/