r/sbin/
nologi

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

TEFC: Shell Scripting - Engenharia da Computação

DECOM

Lista de Exercícios 05

Entradas, Saídas e Redirecionamentos

Wescley Júnior Gonçalves Navarro

Exercício 01) Descreva em detalhes o que cada um dos comandos a seguir está fazendo indicando claramente o efeito do redirecionamento.

a) **`$ date > date`**

Redireciona a saída padrão do primeiro date(a data atual) para um arquivo chamado date.

b) **`$ wc -l < /usr/share/dict/words`**

Faz a entrada padrão do comando wc -l ser o conteúdo do arquivo /usr/share/dict/words mostrando assim o numero de linhas do arquivo.

c) **`$ cat /etc/shadow 2>shadow`**

Redireciona a saída de erro da tentativa de acessar o arquivo shadow sem ser super usuário para um arquivo chamado shadow.

d) **`$ find /var -type f 1>/dev/null`**

Redireciona a saída da busca para o arquivo null, mostrando apenas as saídas de erro no terminal.

Exercício 02) O comando de copiar (`cp`), se executado sem parâmetros, exibe mensagem informando que faltam operandos:

```
$ cp
cp: missing file operand
Try 'cp --help' for more information.
```

a) Essa mensagem é exibida na saída padrão ou na de erro?

Por meio do strace podemos ver que foi escrita no fd 2 ou seja saída de erro padrão.

b) Mostre, com um comando (sem usar `strace` ou comandos similares de depuração), o porquê da resposta do item `(a)`.

Podemos usar redirecionamento de saída padrão ou de erro:
cp > padrão ou cp 2> erro

Exercício 03) Mostre passo a passo o que cada comando encadeado pelos *pipes* (`|`) está fazendo. Depois, mostre qual o **propósito** desse comando completo, ou seja, indique para que ele serve.

a) **`$ cat /etc/passwd | cut -f7 -d':' | sort | uniq`**

cat /etc/passwd adquire os usuários do sistema, manda para cut -f7 -d':' que adquire do arquivo a 7 coluna separada por :, manda para sort que organiza em ordem alfabética e por ultimo manda para uniq que mostra apenas os nomes únicos.

b) **`$ ps aux | sort -k3 -n | tail -n 1`**

(ps aux) mostra com riqueza de detalhes todos os processos em execução no sistema naquele exato instante, organiza por ordem crescente com base na terceira coluna(% de uso da CPU)(sort -k3 -n) e pega o ultimo(tail -n 1), ou seja pega o processo que esta mais usando a CPU nesse momento.

c) **`$ cat /etc/resolv.conf | grep nameserver | wc –l`**

Mostra quantas linhas que o comando cat pegou que tem a palavra nameserver.

d) **`$ ls -l /tmp | tail -n +2 | awk '{print $3}' \ | sort | uniq –c`**

Lista o conteúdo do diretório /tmp em formato longo, remove a primeira linha da saída do ls -l, que é a linha "total", de cada linha restante, extrai e imprime apenas a terceira coluna, que corresponde ao nome do usuário dono do arquivo/diretório, organiza a lista de nomes de usuários em ordem alfabética, agrupa as linhas idênticas e conta quantas vezes cada uma aparece.
O comando serve para contar quantos arquivos e diretórios cada usuário possui dentro do diretório /tmp

Exercício 04) Construa comandos usando *pipes* para resolver os seguintes problemas.

a) O arquivo `/proc/cpuinfo` lista informações sobre os processadores do computador. Consulte esse arquivo e conte quantos processadores tem o computador. Dica: use a linha que possui a entrada `processor`.

sudo cat /proc/cpuinfo | grep processor | wc -l

b) O arquivo `/etc/services` possui uma lista de nomes de protocolos com suas respectivas portas padrão e se é `tcp` ou `udp`. Consulte esse arquivo para obter somente o número da porta do protocolo `ftp-data` (saída esperada: `20`).

cat /etc/services | grep ftp-data | cut -f2 | cut -f1 -d'/'

c) Obter o maior `UID` (*User ID*) do arquivo `/etc/passwd`.

```
sudo cat /etc/passwd | cut -f3 -d':' | sort -n | tail -1
```

d) Obter a quantidade de memória total (primária + secundária) que o sistema está consumindo no momento em *GBytes*. Dica: use o comando `free -tg`.

```
free -tg | tail -1 | awk '{print $3}'
```

Exercício 05) Comandos que não aceitam dados da entrada padrão, somente por parâmetros, limitam sua aplicação com *pipes*. Existe um utilitário muito útil capaz de transformar a saída padrão em argumentos para outro programa chamado `xargs`. Reescreva os comandos a seguir usando `xargs` para contornar essa limitação.

a) Identificar o nome base do diretório atual.

```
$ pwd | basename
basename: missing operand
Try 'basename --help' for more information.

$ pwd | xargs basename
```

b) Remover arquivos temporários.

```
$ find /tmp -type f -name '*.tmp' | rm
usage: rm [-f | -i] [-dPRrvW] file ...
       unlink file

$ find /tmp -type f -name '*.tmp' | xargs rm
```