Universidade Federal de Campina Grande – UFCG
Centro de Engenharia Elétrica e Informática – CEEI
Departamento de Sistemas e Computação – DSC
Professor: Reinaldo Gomes
Disciplina: Administração de Sistemas

Prática 02 – Comandos Básicos

Essa prática é destinada ao entendimento de alguns dos principais comandos básicos do Linux. Faça a leitura do texto referente aos comandos, execute em um terminal Linux a sequência de comandos apresentados e **responda** as perguntas destacadas em verde.

1. Comando find

Este comando procura um ou mais arquivos ou diretórios dentro do sistema de arquivos. Ele é flexível para pesquisar por diversos atributos. Execute os exemplos a seguir.

Exemplo 1:

```
$ find /usr/share -name "*.html"
```

Este exemplo busca em **/usr/share** todos os arquivos com extensão html.

Exemplo 2:

```
$ find /etc/ -name "rc*" -type f
```

Este exemplo procura no diretório **/etc** todos os arquivos cujos nomes começam com a expressão **rc**.

```
$ find /etc/ -name "rc" -type d
```

O que fez o comando acima?

O que fez o comando acima? Em sua opinião, o que mudou quando trocamos os o argumento do parâmetro type? Ou seja, o que significa os argumentos **f** e **d**?

2. Comando cat

Concatena e/ou exibe arquivos na saída padrão (tela). Também pode exibir o conteúdo de vários arquivos em sucessão.

As principais opções são:

- -n : Numera as linhas.
- -E : Exibe $ ao final de cada linha.
- -A : Exibe todo o conteúdo incluindo caracteres especiais, como acentos e espaços na forma de códigos.

Exemplo 1: Exibir o conteúdo do arquivo com as linhas numeradas.

Execute o que se pede a seguir.

1. Acione o editor nano ou pico e digite o seguinte texto:

```
"Software Livre" é uma questão
 de liberdade, não de preço.
 Para entender o conceito,
 você deve pensar em
"liberdade de expressão",
 não em "cerveja grátis".
```

2. Salve o arquivo com nome sl.txt
3. Acione o editor nano ou pico e digite o seguinte texto:

```
=> A liberdade de executar o programa, para qualquer
   propósito (liberdade no. 0)
=> A liberdade de estudar como o programa funciona,
   e adaptá-lo para as suas necessidades
   (liberdade no. 1). Aceso ao código-fonte
   é um pré-requisito para esta liberdade.
=> A liberdade de redistribuir cópias de modo
   que você possa ajudar ao seu próximo
   (liberdade no. 2).
=> A liberdade de aperfeiçoar o programa e
   liberar os seus aperfeiçoamentos, de modo
   que toda a comunidade se beneficie.
   (liberdade no. 3). Acesso ao código-fonte
   é um pré-requisito para esta liberdade.
```

4. Salve o arquivo com nome liberdades.txt e execute os comandos a seguir.
5. `$ cat –n sl.txt`
6. `$ cat sl.txt liberdades.txt`
7. `$ tac sl.txt`
8. `$ tac sl.txt liberdades.txt`

Você compreende o significado das saídas exibidas pelos **comandos 6, 7 e 8**? Qual a diferença entre esses comandos?

3. Redirecionamento de entrada e saída

Os sistemas Linux/UNIX trata todos os periféricos conectados ao sistema como arquivos. O teclado, por exemplo, é um "arquivo"de entrada; o vídeo é um arquivo de saída, assim como a impressora. O arquivo padrão de saída (stdout) é o dispositivo no qual o UNIX deseja os resultados por padrão. Esse dispositivo geralmente é o vídeo. Praticamente todos os programas componentes do sistema que apresentam dados, o fazem como stdout, como por exemplo, o comando ls. Assim como o dispositivo padrão de entrada (stdin) de dados geralmente é o teclado. Há, também, uma saída padrão para os erros (stderr).

Redirecionando saída padrão para um arquivo

Utilizando o símbolo > podemos redirecionar a saída que seria para o vídeo para, por exemplo, um arquivo em disco.

Abaixo, a saída do comando date é gravada para um arquivo de nome datas.

Exemplo: (Execute os comandos abaixo)

$ date > datas

$ cat datas

Acrescentando dados em um arquivo

O operador > cria um novo arquivo, caso ele não exista. Se ele já existe, o arquivo é sobreposto. Usando >>, a saída é acrescentada ao final do arquivo se o mesmo já existe ou é criado um novo arquivo, caso ele não exista.

Exemplo: (Execute os comandos abaixo)

$date > datas

$cat datas

$echo FIM DO ARQUIVO >> datas

$ cat datas

Qual a diferença dos comandos executados com **>** e **>>** ?Qual a diferença entre esses comandos que usam **>** e **>>**?

4. Comando diff

Compara o conteúdo de dois arquivos e lista quaisquer diferenças.

Exemplo1: Execute os comandos em sequência.

Criando os arquivo chamado a.txt e b.txt

```
1. $ cat > a.txt
   agua
   pedra
   vida
   ^C (Significa ctrl+c)
2. cat a.txt
3. $ cat > b.txt
   agua
   pedra
   vidro
   ^D
4. $ diff a.txt b.txt
5. $ diff -u a.txt b.txt
```

Você compreende o significado das saídas exibidas pelos **comandos 4 e 5**? Como fazer para jogar a saída do **comando 5** para um arquivo chamado saída.txt? Acrescente a palavra terra no arquivo a.txt e execute os **comandos 4 e 5** novamente, observando as saídas geradas.

Exemplo2: Execute os comandos em sequência.

Criando os arquivo chamado a.txt e b.txt

```
1. $ cat > c.txt
   agua
   pedra
   vida^C (Significa ctrl+c)
2. $ cat > d.txt
   agua
   pedra
   vidro^D
3. cat c.txt d.txt
4. $ diff a.txt b.txt
5. $ diff -u a.txt b.txt
```

5. Comando grep

Pesquisa palavras ou padrões dentro de arquivos textos ou a partir da saída de um comando. Execute os exemplos a seguir.

Opções importantes deste comando:

-i => filtra uma string (ou texto), independente do fato de ser maiúscula ou minúscula.

-v => filtra tudo da saída ou arquivo, menos a string informada.

Exemplo 1: Execute os comandos em sequência.

```
1. $ cat /etc/shells
2. $ grep bash /etc/shells
3. $ grep -v bash /etc/shells
```

Observe que o comando exibe na tela apenas as linhas do arquivo **shells** que contém a palavra **bash**. Qual a diferença da saída do **comando 2** para o **comando 3**?

Exemplo 2: Execute os comandos em sequência.

```
1. $ ls /etc
2. $ ls /etc | grep ^in
```

Observe que o **comando 2** filtra da saída apresenta pelo comando ls /etc, e exibe tudo que começa com a expressão **"in"**.

Exemplo 3:

```
$ ls /etc | grep init
```
O que fez o comando acima?

```
$ ls /etc | grep conf
$ ls /etc | grep conf$
```
O que fez o comando acima? Em sua opinião, o que significa a inclusão do caractere **$** ? Qual a diferença entre os dois comandos? E se executássemos o último comando dessa forma (faz diferença?): `$ ls /etc | grep ^conf`

Exemplo 4: Execute os comandos em sequência.
```
1. $ ls /etc | grep network
2. $ ls /etc | grep -i network
```

Qual a diferença entre as saídas apresentadas nos **comandos 1 e 2**?

5. Comando cut

Comando utilizado recortar caracteres de uma saída/arquivo ou para extrair campos/colunas de um arquivo texto. Execute o comando abaixo e verifique a saída.

```
$ echo administracao de sistemas | cut -c 1-13
```

Trabalhando agora com o cut para extrair colunas de arquivos. Verifique os exemplos a seguir. Execute os comandos no terminal, mas antes crie o arquivo agenda.txt com o conteúdo apresentado abaixo.

```
$ cat > agenda.txt
Ana Maria 333-3333
Jose Maria 444-4444
Paula Pereira 555-555
^D

$ cat agenda.txt | cut -c-5

$ cat agenda.txt | cut -f1 -d" "

$ cat agenda.txt | cut -f2 -d" "

$ cat /etc/passwd

$ cat /etc/passwd | cut -f1 -d:

$ cat /etc/passwd | cut -f1 -d/
```

De acordo com os comandos executados, qual o significado dos parâmetros -d e -f ?

6. Comando uniq

Remove linhas duplicadas dos arquivos. Mas há uma condição: a lista deve estar ordenada. A opção -d faz com que sejam listados somente os nomes repetidos. O comando uniq não aapresenta resultado quando a lista não está ordenada.

Verifique os exemplos a seguir. Execute os comandos no terminal, mas antes crie o arquivo lista com o conteúdo apresentado abaixo.

```
$ cat > lista
```

```
Alexandre
José
Felipe
Tatiana
Beatriz
Eduardo
Enrico
Aline
Erica
José
$ uniq lista
```
Qual resultado apresentado?

```
$ sort lista | uniq
```
O comando agora deu certo?

```
$ sort lista | uniq -d
```
Qual o significado o parâmetro -d?

7. Outros comandos importantes

wc : conta o número de linhas.
Principais opções:

- -l : Mostra apenas o número de linhas.
- -w : Apenas o número de palavras
- -c : Apenas o número de caracteres.

Execute os comandos abaixo.
$ wc sl.txt
$ wc -l sl.txt
$ wc -w sl.txt
$ wc -c sl.txt
Qual a diferença entre os comandos acima?
Explique a diferença entre os comandos executados anteriormente.

tr : Substitui caracteres.
Pode ser usado para trocar as letras de maiúsculas para minúsculas.
Sintaxe: tr [opções] <string1> <string2>
As opções principais são:

- -c : Realiza a troca de todos os caracteres, exceto da string1.
- -d : Elimina os caracteres especicados em string1, ignorando string2.
- -s : Comprime a seqüência de caracteres repetidos da string2.

Exemplo: Converte as letras minúsculas para maiúsculas.

Digite o seguinte texto utilizando o editor nano ou pico e salve com o nome cidades.txt:
campina grande
patos
recife
queimadas

Execute o comando abaixo:
$ cat cidades.txt | tr "[a-z]" "[A-Z]"
$ cat cidades.txt | tr "[a,b]" "[A,B]"

Qual o resultado? Explique o que aconteceu.