Comandos Linux para Filtros e Desvios

Conceitos Gerais

1. Introdução

- Em diversas situações é preciso aplicar comandos no Linux apenas em determinados arquivos. É neste ponto que se torna necessário utilizar os filtros.

- Também pode ser necessário aplicar uma cadeia de comandos, onde o resultado de um deve ser enviado para o outro. Esse processo é conhecido como pipe no Linux.

- A seguir serão analisados os principais comandos e exemplos para aplicação de filtros e desvios.

2. Pipes

- O **pipe** (simbolizado por |) permite que o resultado de um comando seja passado para outro comando, para que este o processe.

- Como exemplo, vamos visualizar um arquivo com o comando "cat" e passá-lo para o comando "tail", que exibe apenas as últimas 10 linhas do arquivo:

```
#visualizando o arquivo e passando com | para o coando tail
cat /etc/profile | tail
```

```
paulo@notebook1:~$ cat /etc/profile | tail
fi

if [ -d /etc/profile.d ]; then
  for i in $(run-parts --list --regex '^[a-zA-Z0-9_]
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
```

3. grep

- O comando **grep** no Linux é usado para **procurar padrões de texto dentro de arquivos ou saídas de outros comandos**. Ele é extremamente útil para filtrar informações, especialmente em logs, códigos e grandes volumes de dados.

- O nome vem de "**g**lobal **r**egular **e**xpression **p**rint", ou seja, ele busca por **expressões regulares (regex)** e imprime os resultados. A imagem abaixo mostra alguns parâmetros:

Parâmetro	Descrição
`-i`	Ignora maiúsculas/minúsculas
`-n`	Mostra número da linha
`-c`	Conta ocorrências
`-v`	Mostra linhas que **NÃO** contêm o padrão
`-r` ou `-R`	Busca recursivamente em diretórios
`-l`	Mostra apenas nomes dos arquivos
`-w`	Busca palavra exata
`-o`	Mostra apenas o trecho encontrado
`-E`	Permite expressões regulares avançadas

3. grep

- O quadro abaixo mostra alguns exemplos:

```
#Encontra erro, Erro, ERR0, etc.
grep -i "erro" log.txt

#mostra número das linhas que tem a palavra erro
grep -n "erro" log.txt

#busca recursivamente no diretório /etc por um arquivo com a palavra senha
grep -r "senha" /etc/

#buscar palavra exata
grep -w "root" usuarios.txt

#busca números com 3 digitos
grep -E "[0-9]{3}" arquivo.txt
```

3. grep

- O grep é muito usado juntamente com o pipe, para fazer um encadeamento de comandos. O quadro abaixo mostra alguns exemplos:

```
#mostra apenas processos relacionados com firefox
ps aux | grep firefox

#mostra conexões na porta 80
netstat -tuln | grep 80

#procurar erro em um arquivo de log, olhando apenas as últimas linhas
tail -f log.txt | grep erro

#buscar comandos relacionados com ssh já usados no histórico
history | grep ssh

#litar processos do python que não são do usuário root (-v exclui root)
ps aux | grep python | grep -v root
```

4. Redirecionamento >

- O desvio ou redirecionamento tem como função desviar o resultado de algum comando para um arquivo ou dispositivo. Não desvia mensagens de erro, apenas de operações bem-sucedidas.

```
#grava no arquivo.txt o resultado do ls.
ls /home/paulo/Documentos > arquivo.txt

#grava o resultado do free no arquivo.txt dentro do diretório /tmp
free > /tmp/memo.txt

#concatena os arquivos e o resultado vai para resultado.txt
cat arquivo1.txt arquivo2.txt > resultado.txt

#verifica a diferença entre os arquivos e grava
diff arquivo1.txt arquivo2.txt > resultado.txt

#o resultado do desvio criará um arquivo caso ele não exista.
#se existir, o arquivo será substituído
#cuidado com comandos como cat a.txt b.txt > a.txt
#nesse caso, a.txt será apagado e criado um novo antes de ler completamente seus dados
```

5. Direcionamento de Erro 2> e 2>>

- Grava no arquivo algum resultado anormal:

```
#como arquivo1.txt não existe grava erro no teste.txt
cat arquivo1.txt 2> teste.txt

#como arquivo1.txt não existe grava erro no final do teste.txt
cat arquivo1.txt 2>> teste.txt
```