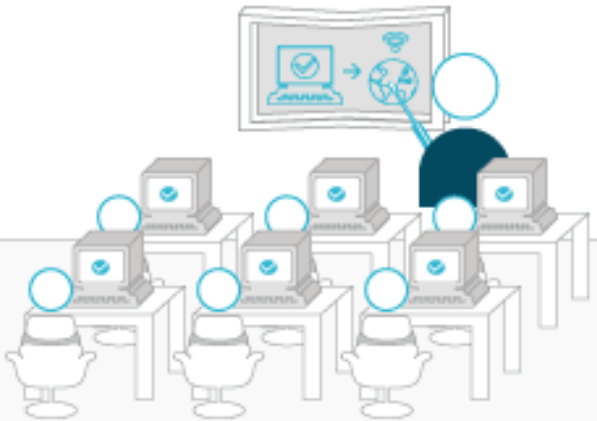
Título do Curso:

Administração de Sistemas GNU/Linux

Conteúdo do curso

ADMINISTRAÇÃO DE SISTEMAS GNU/LINUX

- Tópico 1: Introdução ao sistema operacional GNU/Linux.
- Tópico 2: Introdução ao Shell e comandos básicos.
- **Tópico 3: Manipulação de conteúdos com comandos no Shell.**
- Tópico 4: Comandos para gerenciamento do sistema e do Hardware.
- Tópico 5: Editor de Texto VI.
- Tópico 6: Administração de usuários e grupos.
- Tópico 7: Gerenciamento de permissões.
- Tópico 8: Gerenciamento de processos.
- Tópico 9: Sistemas de arquivos e particionamento.
- Tópico 10: Expressões regulares.
- Tópico 11: Introdução ao Shell Script.
- Tópico 12: Gerenciamento de Pacotes.
- Tópico 13: Agendamento de tarefas (cron) e Backup.



Tópico 3

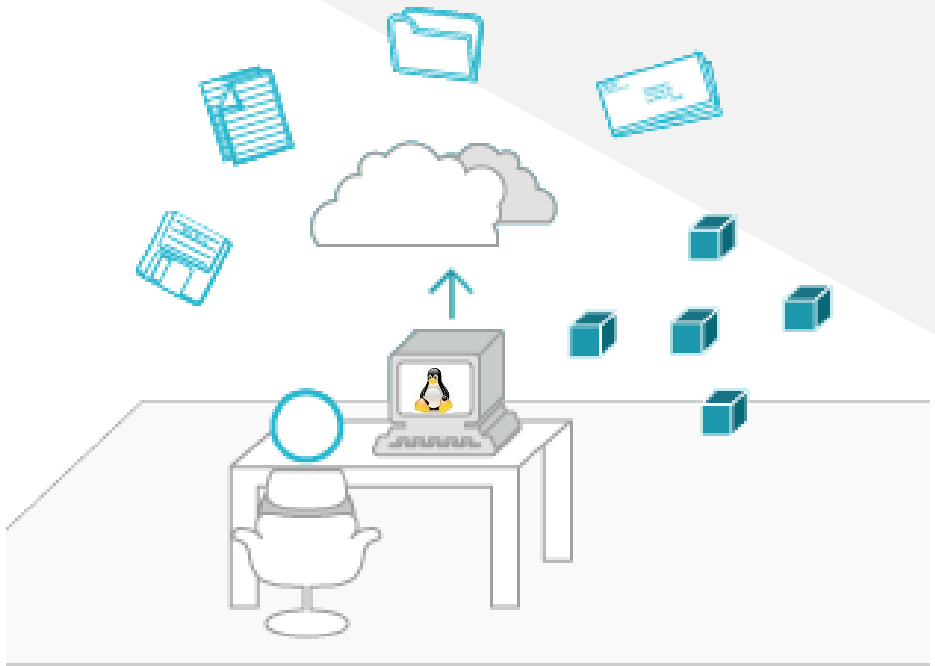
Manipulação de conteúdos com comandos Shell

```
[root@localhost ~]# tail -n 50 /var/log/messages | grep kernel >> log-kernel.txt
[root@localhost ~]#
```

Manipulação de conteúdos

com comandos no Shell

- Neste slide teremos comandos para visualização e manipulação de textos e alguns recursos do Shell, conforme listado abaixo:
  - **Visualizadores de texto;**
  - **Redirecionadores;**
  - **Concatenação de comandos;**
  - **Conectores e Operadores;**



Visualizadores de texto

Tópico 3: Manipulação de conteúdos com comandos Shell.

Visualizadores de texto

- cat → Exibe na saída padrão (tela) o conteúdo de um arquivo.
  - cat [arquivo]
  - Ex.: cat /etc/passwd

- more → Exibe na saída padrão (tela) o conteúdo de um arquivo, exibindo uma tela de cada vez (ideal para arquivos extensos).
  - more [arquivo]
  - Ex.: more /etc/services
    - OBS.1: No canto inferior esquerdo podemos visualizar o percentual do arquivo já exibido.
    - OBS.2: Para sair da visualização temos 3 opções:
      - Visualizar o arquivo por completo;
      - Atalho [CTRL +C] que envia um “sinal de controle de interrupção”;
      - Pressionando a tecla [q];

Visualizadores de texto

- less → Exibe na saída padrão (tela) o conteúdo de um arquivo, permitindo a paginação através de comandos.
  - less [arquivo]
  - Ex.: less /etc/services
    - OBS.1: No canto inferior esquerdo podemos visualizar as linhas exibidas, a quantidade total de linhas e o percentual do arquivo já exibido.
    - OBS.2: Podemos “navegar” pelo arquivo através das setas ou pelas teclas [Page Up] / [Page Down];
    - OBS.3: Para sair da visualização temos as mesmas opções que o “more”:

Visualizadores de texto

- head → Exibe na saída padrão (tela) as dez primeiras linhas de um arquivo (por padrão, porém, a quantidade pode ser especificada).
  - head [arquivo]
  - Ex.: head /etc/services

- tail → Exibe na saída padrão (tela) as dez últimas linhas de um arquivo (por padrão, porém, a quantidade pode ser especificada).
  - tail [arquivo]
  - Ex.: tail /etc/services
    - OBS.: Este comando é muito utilizado para a visualização de Logs em tempo real, através do parâmetro “-f”. (Logs de sistema, acesso a internet, monitoramento, suspeita de invasões ou ataques de força bruta, etc...)
  - Ex.: tail -f /var/log/messages

Redirecionadores

Tópico 3: Manipulação de conteúdos com comandos Shell.

Redirecionadores

- Antes de citar sobre recursos de redirecionamento e concatenação de comandos, devemos conhecer os “descritores” e I/O padrão.

- Todo programa (seja na plataforma POSIX ou Micro$oft), realiza operações de entrada e saída de dados (I/O = Input/Output).

- As operações de entrada geralmente estão vinculadas aos dispositivos de entrada de dados (teclado/mouse) porém, podemos enviar informações de entrada para um comando a ser executado.

- Já a saída de dados geralmente é exibida em dispositivos de saída (monitor, impressora, etc), porém, podemos redirecionar a saída de um comando para um arquivo ou dispositivo, ao invés da saída padrão, que seria a tela do seu terminal...

Redirecionadores

- Estas operações de I/O possuem três variações, que são chamadas de **descritores**, sendo:
  - **Entrada padrão *(stdin)*:** Geralmente vinculado ao teclado, ou seja, tudo o que está sendo digitado é enviado para a entrada padrão.
    - Descritor de arquivos = 0 (zero) – Vamos entender esta numeração a seguir.
  - **Saída padrão *(stdout)*:** Ao executarmos um comando, o resultado/saída deste comando é enviado para a saída padrão, ou seja, na tela do terminal.
    - Descritor de arquivos = 1;
  - **Saída de erro padrão *(stderr)*:** Semelhante ao descritor 1, porém, recebe apenas sinais de erro que um comando gerar. Muito útil para filtrar e separar apenas os erros gerados por um processo ou comando.
    - Descritor de arquivos = 2;

Redirecionadores

- OK, após entendermos sobre os descritores, podemos concluir que a saída padrão de todo comando é exibida na tela e que esta saída pode ser filtrada, possibilitando visualizar apenas os erros ou êxitos gerados.
  - **Porém, como aplicar isso na prática?**

- Simples, basta utilizar os redirecionadores, representados pelos sinais “maior que” e “menor que” (Sinais: “>” e “<“ respectivamente).
  - Ex.: cat /etc/passwd **>** /root/lista_users.txt

  - No exemplo acima, temos o comando “cat” que tem a função de mostrar o conteúdo de um arquivo na saída padrão *(stdout)*, ou seja, na tela do terminal. Porém, com o redirecionador “>” podemos enviar a saída do comando para outro local, que no caso foi o **novo** arquivo de nome “lista_users.txt”. (**Teste o comando em sua VM**).

Redirecionadores “>” e “>>”

- Estes redirecionadores enviam a saída de um programa/comando para um arquivo ou dispositivo, ao invés de exibir na saída padrão (tela), porém com uma diferença:

- Redirecionador “>”: Envia a saída para um arquivo ou dispositivo, porém, se o arquivo já existir, o seu conteúdo é **sobrescrito**.
  - Ex.: ls -l /dev/ > /root/dispositivos.txt

- Redirecionador “>>”: Envia a saída para um arquivo ou dispositivo, porém, se o arquivo já existir, a saída do comando é **ADICIONADA** no final do arquivo especificado.
  - Ex.: ls -l /dev/ >> /root/dispositivos.txt
  - OBS.: Se executarmos “ls -l /dev/ 1>> /root/dispositivos.txt” teremos o mesmo resultado do exemplo acima (stdout = descritor 1, logo “1>>”).

Redirecionadores “<” e “<<”

- Estes redirecionadores ao invés de “enviar” dados como o anterior, possuem a função de “receber”.

- Redirecionador “<”: Faz com que determinado comando receba algo como entrada.
  - Ex.: more < /etc/services (Ou seja, o mesmo que “more /etc/services”).
  - Ex.2: < /etc/services less (Ou seja, uma forma bem diferente de visualizar o arquivo, mas possui o mesmo retorno que “less /etc/services”).

    - ***Bom, na prática ele realmente é usado?*** SIM. Geralmente é utilizado quando alguém deseja adicionar um “patch” (com alguma correção) antes da instalação de algum software através do código fonte. (Linux Avançado).

  - Ex.: patch -p0 < /software/patches/patch01 (Exemplo da resposta acima).
  - Ex.: script-migra.sh < /home/lista.txt (Exemplo da resposta acima).

Redirecionadores “<” e “<<”

- Redirecionador “<”: Exemplo de utilização em um Shell Script:
  - Conteúdo do Shell Script:

```
#!/bin/bash
arquivo=/etc/passwd
{
read linha1
read linha2
} < $arquivo

echo "A primeira linha do arquivo $arquivo é:"
echo "$linha1"
echo

echo "A segunda linha do arquivo $arquivo é:"
echo "$linha2"
```

  - Saída do Shell Script:

```
root@ninja:~# ./script.sh
A primeira linha do arquivo /etc/passwd é:
root:x:0:0::/root:/bin/bash

A segunda linha do arquivo /etc/passwd é:
bin:x:1:1:bin:/bin:/bin/false
```

Redirecionadores “<” e “<<”

- Redirecionador “<<”: Utilizado para determinar o final de um “bloco” de dados.
  - Ex.: cat >> arq.txt << EOF (Um conteúdo será adicionado no arquivo “arq.txt”, caso o mesmo exista, até a *String* **EOF** ser digitada).

Inserindo o conteúdo no arquivo:

```
root@ninja:~# cat >> arq.txt << EOF
> Primeira linha de texto
> Segunda linha...
> Vou finalizar o arquivo com a string EOF...
> EOF
root@ninja:~#
```

Visualizando o conteúdo do arquivo:

```
root@ninja:~# cat arq.txt
Primeira linha de texto
Segunda linha...
Vou finalizar o arquivo com a string EOF...
root@ninja:~#
```

- ***Bom, na prática ele realmente é usado?*** SIM, porém, pouco usual. Este exemplo acima pode ser aplicado caso tenhamos problemas com algum editor de texto... Mas não se esqueça que não podemos editar as linhas digitadas anteriormente ou o conteúdo já existente do arquivo.

Redirecionadores de Erro“2>” e “2>>”

- Estes redirecionadores enviam apenas os **sinais de erro** *(stderr)* da saída de um programa/comando para um arquivo ou dispositivo, ao invés de exibir na saída padrão (tela):

- Redirecionador “2>”: Envia apenas os **sinais de erro** para um arquivo ou dispositivo, porém, se o arquivo já existir, o seu conteúdo é **sobrescrito**.
  - Ex.: `ls -l /nao_existe 2> /root/log_erro.txt`

- Redirecionador “2>>”: Envia apenas os **sinais de erro** para um arquivo ou dispositivo, porém, se o arquivo já existir, a saída (apenas os erros) será **ADICIONADA** no final do arquivo.
  - Ex.: `ls -l /nao_tem 2>> /root/log_erro.txt`

Redirecionadores de Erro“2>” e “2>>”

- Este redirecionador é muito utilizado quando desejamos filtrar apenas os erros, ou visualizar apenas as informações desejadas descartando os erros, ou quando desejamos visualizar ambos posteriormente.
  - Ex.: `find / -name group 2> /root/erros.txt > /root/lista.txt`

- ***Bom, na prática ele realmente é usado?***
  - SIM e muito, principalmente quando estamos executando um programa com uma saída extensa. Exemplos:
    - Migração de dados;
    - Instalação de um Software através do código fonte;
    - Ao compactar ou descompactar muitos arquivos (como tarefas de backup).

Buraco negro do Linux... Quem é “/dev/null”

- E se o comando gerar muitos erros que eu não desejo visualizar ou gastar espaço em disco conforme o exemplo anterior?
  - Ex.: find / -name group 2**> /dev/null** > /root/lista.txt

- **QUEM É “/dev/null”??**
  - O caminho “/dev/null” é um *“bit bucket”* (dispositivo nulo), ou seja, lugar nenhum. O local é chamado por muitos de “buraco negro”.
  - O “/dev/null” é um arquivo especial que descarta toda informação enviada para ele, além de não retornar nenhuma informação caso seja acessado.

```
root@ninja:~# cat /dev/null
root@ninja:~#
```

Concatenação de comandos

Tópico 3: Manipulação de conteúdos com comandos Shell.

Concatenação (o famoso PIPE) Comandos Compostos

- Concatenação de comandos nada mais é do que “pegar” a saída de um comando e utilizá-la como “entrada” para o comando seguinte.

- Para realizar a concatenação, devemos utilizar dutos (duto = PIPE = |).

- Exemplo: vamos supor que desejamos listar um diretório com muitos arquivos e gostaríamos de analisar calmamente a saída do comando.
  - Ex.: ls -l /etc | less (Neste exemplo estamos listando o diretório “/etc” e seus atributos, posteriormente, o uso do PIPE “|” possibilita utilizar a saída deste comando como entrada do comando seguinte, no caso o “less”, de forma a controlar a saída com as setas do teclado).

- O PIPE é **MUITO** utilizado no Linux (e no Windows também). Entretanto, vamos ver possibilidades muito úteis nos próximos slides/vídeos.

Conectores e Operadores

Tópico 3: Manipulação de conteúdos com comandos Shell.

Conectores de comandos
Comandos Compostos

- Podemos solicitar a execução de diversos comandos em sequência no Linux.
- Esta execução em sequência é diferente do processo de concatenação.
- A execução de comandos em sequência não utiliza a saída do comando anterior como dados de entrada no comando seguinte.
- Conectores e operadores são geralmente utilizados em scripts.

Conectores de comandos
Comandos Compostos

- Para executar comandos em sequência, temos os seguintes conectores:
  - Ponto e vírgula “;” para execução de comandos em sequência:
    - Ex.: `clear ; cd /etc ; cp -a services /root`
  - Operador “&&” (AND), caso o primeiro comando execute com sucesso (código de retorno = 0), o segundo comando também será executado:
    - Ex.: `cd /Teste && rm -rf *`
  - Operador “||” (OR), caso o primeiro comando não execute com sucesso (código de retorno ≠ 0), o segundo comando será executado. RESUMINDO, quando um dos comandos for executado com sucesso, os comandos seguintes não são verificados:
    - Ex.: `cd /home/Teste || mkdir /home/Teste`

No próximo slide…

- Manipulação de conteúdos com comandos Shell:
  - Filtros de conteúdo:
    - grep
    - wc
    - sort
    - cut
    - awk
    - diff
  - Empacotadores e Compactadores:
    - zip
    - unzip
    - tar

Referências

- BONAN, Adilson Rodrigues. **LINUX – Fundamentos, Prática & Certificação LPI**. Editora: Alta Books. RJ. 2010;

- PEREIRA, Guilherme Rodrigues. **Slides para aula expositiva**. Centro Universitário UNA.

- SILVA, Gleydson Mazioli. **Guia Foca GNU/Linux**. Disponível em: https://guiafoca.org/

DGP

Tecnologia da Informação

Obrigado!

Guilherme Rodrigues