Curso de Introdução ao Linux

Aula 1

Angelo Brito - asb
Adriano Melo - astm

Ementa

- Introdução a Unix e Linux
  - Conceitos Relacionados ao Linux
  - Distribuições Linux
  - Ambiente desktop gráficos
- Comandos básicos e utilitários
- Dispositivos, Sistema de arquivo Linux e Hierarquia padrão do sistema de arquivo.
- Documentação - Procura de Ajuda na rede (Faq, manuais)

Ementa

- Gerenciamento Administrativo
  - Gerenciamento de usuários e grupos
  - Usando log do sistema
  - Estratégia de backup
  - Permissões e posses de arquivos e diretórios
  - Gerenciamento de processos
- Instalação do Linux e Gerenciamento de pacote (programas)
- Kernel Linux
- Carregar driver no sistema
- Acesso Remoto, Configuração do PC na rede.
- Shells, Scripting e Makefile

Introdução a Unix e Linux

História

- O **Projeto GNU** é um Projeto de software livre e colaboração em massa anunciado em 1983 por Richard Stallman. Ele começou o Sistema Operacional GNU pelo desenvolvimento de softwares, que começou em 1984. O objetivo da fundação, pelas palavras de Stallman, era desenvolver "*a sufficient body of free software [...] to get along without any software that is not free*“[1], ou seja, desenvolver um sistema inteiramente composto de software livre e que fosse capaz de fazer tudo que um sistema proprietário era capaz de fazer.

- Para conseguir cumprir seu objetivo a fundação começou a trabalhar em um sistema chamado GNU, que era basicamente uma copia, composta de software livre e com algumas modificações, do sistema UNIX. GNU é uma anacronismo recursivo que significa "GNU's Not UNIX”. Quando o Kernel Linux foi lançado sob a Licença Publica Geral GNU em 1992, o Projeto GNU deixou de depender de softwares proprietários para executar.

Comandos básicos e utilitários

Linha de Comando

Linha de Comando do Unix

- Idéia: Aprender o essencial para trabalhar com linha de comando:
  - Variáveis de ambiente
  - Histórico de comando
  - Editar arquivos rapidamente
  - Invocar comandos
  - Executar comandos recursivamente

Linha de Comando do Unix

- Um administrador de Linux comumente prefere trabalhar com a interface de texto ao invés de interface gráfica.
- Esta interface possui um *prompt*, $ ou #, que significa que o sistemas está pronto para receber comandos.

Linha de Comando do Unix

Shell

- A tarefa do S.O. responsável por gerenciar a linha de comando é chamada de shell. Ele tem as seguintes funções:
  - prover o prompt
  - interpretar os comandos
  - Prover a interface entre o Linux kernel e o usuário
- O primeiro Shell para sistemas Unix foi escrito por Steve Bourne e foi chamado simplesmente de sh. O Shell padrão para o Linux é o bash, o Bourne-Again Shell, que foi derivado do sh e criado pelo Projeto GNU.
- O tcsh Shell também é bastante popular e foi derivado do csh (C shell)

Linha de Comando do Unix

Variável do Shell

- Prompt String 1
  - [\u@\h \W] \$
    - \u: nome do usuário
    - \h: nome do sistema
    - \w: diretório atual
    - \$: $
- PATH
  - Conter uma lista de todos os diretórios que contém comandos ou programas que você deseje executar

Linha de Comando do Unix

Variável do Shell

- Exemplo:
  - O comando less está em /usr/bin. Colocando este diretório na variável PATH, você poderá executá-lo simplesmente digitando o nome do programa, ao invés de digitar todo o caminho.

Linha de Comando do Unix

Variáveis de ambiente

- Algumas outras variáveis são necessárias durante uma execução de um programa executado a partir do shell. Nesse caso essas variáveis precisam ser exportadas.
  - $ export ECLIPSE_HOME=/opt/eclipse
  - $ export GAME_SHOOT=/home/rgo/jogos

Linha de Comando do Unix

Executando comandos no prompt

- Para os comandos serem executados no shell, geralmente eles precisam de quatro componentes:
  - Um programa válido (que esteja definido no PATH, através de variável de ambiente ou que seu caminho tenha sido explícito).
  - Opções do comando, geralmente precedido de traço '-'
  - Argumentos
  - Linha de aceitação (tecla Enter)

Linha de Comando do Unix

Executando comandos no prompt

- A maioria dos programas têm sua própria sintaxe, mas pelo menos o nome do programa se torna necessário. Ex.:
  - $ ls
  - $ ls –l
  - $ ls -l –a ou $ ls –la ou $ ls -l --all

Linha de Comando do Unix

Executando comandos no prompt

- As opções com traços duplos e palavras completas geralmente são encontrados nos programas do projeto GNU.
- Elas não podem ser combinadas como as opções de um traço.
- A vantagem é o fato de serem fáceis de lembrar e é simples de entender a sua funcionalidade ao lê o comando.

Linha de Comando do Unix

Variáveis de ambiente

- Você ainda pode refinar a sua busca usando argumentos.
  - $ ls -l *.txt
  - $ ls --all *.c -l

Linha de Comando do Unix

Variáveis de ambiente

- Alguns comandos como o *tar* e o *ps* não requerem que a opção tenha um traço, pois é necessário ou esperado pelo menos uma opção.
  - $ tar cf mytarfile file1 file2 file3
  - $ tar -cf mytarfile file1 file2 file3
  - $ tar xvf mytarfile.tar

Linha de Comando do Unix

Executando uma seqüência de comandos

- Para adicionar mais de um comando da linha de comando você pode utilizar o separador de comandos ‘;’
  - $ ls; ps

Linha de Comando do Unix

Histórico de Comandos

- As vezes é necessário executar um comando várias vezes, ou então o mesmo comando com algumas variações:
  - Para acessar comandos já digitados você pode apertar a tecla com seta para cima.
  - Acessando o arquivo ~/.bash_history
  - $!string
  - $!?string

Linha de Comando do Unix

Executando um comando recursivamente numa árvore de diretório

- Pode ser necessário executar um comando recursivamente
- Isto é uma poderosa ferramenta, mas também pode se tornar perigoso.
  - $ chmod o+w *.doc
  - $ chmod -R o+w Desktop

Linha de Comando do Unix

Executando um comando recursivamente numa árvore de diretório

- Uma abordagem de execução recursiva é o find:
  - $ find /usr/bin : arquivos e diretórios são listados recursivamente
  - $ find /usr/bin –name “*less*” : arquivos que contém a substring less são listados recursivamente

Processamento de Texto Utilizando Filtros

Processamento de Texto Utilizando Filtros

Há vários pacotes e bibliotecas do GNU textutils package que podem ser utilizados para manipulação de texto.

Processamento de Texto Utilizando Filtros

expand

- Sintaxe
  - expand [options] files
- Descrição
  - Converte tab em espaços. Este comando elimina a tab substituindo eles por um número de espaços equivalentes. Por padrão, tab são 8 espaços.
- Opções mais frequentes
  - –t tabs
    - Especifica o número de espaços
  - –i
    - Inicial; converte somente o início das linhas

Processamento de Texto Utilizando Filtros

expand

- [rgo@g1c06 rgo]$ expand test.nano
  - a b c

    d r f
- rgo@g1c06 rgo]$ expand -t 1 test.nano
  - a b c

    d r f
- [rgo@g1c06 rgo]$ expand -it 4 test.nano
  - a b c

    d r f

test.nano:

```
        a       b       c
d       r       f
```

Processamento de Texto Utilizando Filtros

fmt

- Sintaxe
  - fmt [options] [files]
- Descrição
  - Formata o texto especificando a largura da linha e remove linhas novas.
- Opção frequentemente usada
  - –u
    - Deixa espaços uniformes: um espaço entre palavras e dois espaços entre sentenças.

Processamento de Texto Utilizando Filtros

fmt

- $ fmt test.nano
  - a b c. e f g.
    h i j. l m n. o p q.
- $ fmt -u test.nano
  - a b c. e f g.
    h i j. l m n. o p q.
- $ fmt -u test.nano test2.nano
  - a b c. e f g.
    h i j. l m n. o p q.
    r g o =)

- $ cat test2.nano
  - r
    g
    o =)

Processamento de Texto Utilizando Filtros

head

- Sintaxe
  - head [options] [files]
- Descrição
  - Imprimi as primeiras linhas de um ou mais arquivos. Se mais de um arquivo for especificado, é impresso o início de cada arquivo.
- Opções freqüentemente usadas
  - –c n
    - Imprime os primeiros n bytes. Se n for seguido de k ou de m, imprime os primeiros kilobytes e megabytes.
  - –l n
    - Imprimi as primeiras n linhas, o padrão é 10.

Processamento de Texto Utilizando Filtros

head

- $ head -c2 test.nano
  - a
- $ head -c2 test.nano test2.nano
  - ==> test.nano <==

    a

    ==> test2.nano <==

    r

Processamento de Texto Utilizando Filtros

join

- Sintaxe
  - join [options] file1 file2
- Descrição
  - Imprime uma linha para cada linha de dois arquivos que tenham os mesmos parâmetros de entrada. Esta função pode ser pensada como um banco de dados muito simples.
- Opção freqüente
  - –j1 field - Join no campo do arquivo1.

Processamento de Texto Utilizando Filtros

join

- Exemplos
  - Suponha que os seguintes arquivos:

Arquivo1:

```
1 one
2 two
3 three
```

Arquivo2:

```
1 11
2 22
3 33
```

$ join -j 1 file1 file2

Saída:

```
1 one 11
2 two 22
3 three 33
```

Processamento de Texto Utilizando Filtros

nl

- *Sintaxe*
  - nl [*options*] *[files]*
- *Descrição*
  - Numerar as linhas de um arquivo, que são concatenados na saída. *Este comando é usado para numerar as linhas no corpo de um texto, incluindo um cabeçalho especial e opções de roda-pé normalmente excluídas da numeração de paginas. A numeração é feita para cada pagina lógica, que deve conter um cabeçalho, um corpo e um rodapé. Onde cada parte é delimitada, respectivamente, por \:\:\: , \:\: e \:*

Processamento de Texto Utilizando Filtros

nl

- *Opções freqüentes:*
  - –b *style*
    - Seta o estilo de numeração do corpo para style, *t por definição.*
  - –f *style*
    - Seta o estilo de numeração do roda-pé para style, n *por definição.*
  - –h *style*
    - Seta o estilo de numeração do cabeçalho para style, n *por definição.*

Processamento de Texto Utilizando Filtros

nl

- Estilos podem ser da forma:
  - a – numera todas as linhas.
  - t – Só numera as linhas não vazias.
  - n – não numera linhas.
  - p***BRE –*** só numera as linhas que contenham a expressão regular BRE.

Processamento de Texto Utilizando Filtros

nl

- *Example*
- Se o seguinte comando for executado:

File1:

```
\:\:\:
header
\:\:
line1
line2
line3
\:
footer
\:\:\:
header
\:\:
line1
\:
footer
```

$ nl -h a file1

Output:

```
1 header
2 line1
3 line2
4 line3
footer
1 header
2 line1
3 line2
4 line3
footer
```

Processamento de Texto Utilizando Filtros

od

- *Sintaxe*
  - od [*options*] [*files*]
- *Descrição*
  - *Descarrega arquivos no formato octal ou outros. Este comando imprime uma representação não ambígua do conteúdo de um arquivo na forma octal, por definição, ou outra.*
  - *Podendo receber mais de um arquivo como entrada, os arquivos serão ordenados em fila e concatenados como uma única entrada. Sua principal utilização é para examinar o código binários de arquivos binários, mas pode ser utilizado em qualquer tipo de arquivo, para examinar uma stream por exemplo.*

Processamento de Texto Utilizando Filtros

od

- *Opções freqüentes:*
  - –t *type*
    - Especifica o tipo de saida*, que podem ser:*
      - a – caractere nomeado
      - c - caracteres ASCII ou backslash escape
      - o - Octal (O Padrão)
      - x - Hexadecimal

- *Exemplo*
  - Se *file1 contiver:*

    a1\n

    A1\n

(Onde \n é o caracter de nova linha.)

- O comando **$ od –t a file1** vai gerar a seguinte saída:

  00000000 a 1 nl A 1 nl

  00000006 (essa linha é a contagem dos bytes no arquivo)

Processamento de Texto Utilizando Filtros

paste

- *Sintaxe*
  - paste [*options] files*
- *Descrição*
  - Cola juntas as linhas correspondentes de um ou mais arquivos em colunas verticais.
- *Opções freqüente usadas:*
  - –d'*n*‘ – *se*para as colunas com caracteres *n no lugar da tabulação padrão.*
  - –s – Mistura as linhas de um arquivo em uma única linha. Quando múltiplos arquivos são especificados, seus conteúdos são dispostos individualmente em cada linha da saída, uma linha por arquivo.

Processamento de Texto Utilizando Filtros

paste

- Exemplo

file1:

```
1
2
3
```

file2:

```
A
B
C
```

- *Exemplo 1*
  - Um simples paste cria colunas para cada arquivo na saída

    $ **paste file1 file2**

saida1:

```
1 A
2 B
3 C
```

- *Exemplo 2*
  - A opção separadora de colunas tem suas colunas separadas pelo caracteres especifico que foi definido

    $ **paste -d'@' file1 file2**

saida2:

```
1@A
2@B
3@C
```

- *Exemplo 3*
  - A opção de uma linha por arquivo (–s) gera:

    $ **paste -s file1 file2**

saida3:

```
1 2 3
A B C
```

Processamento de Texto Utilizando Filtros

split

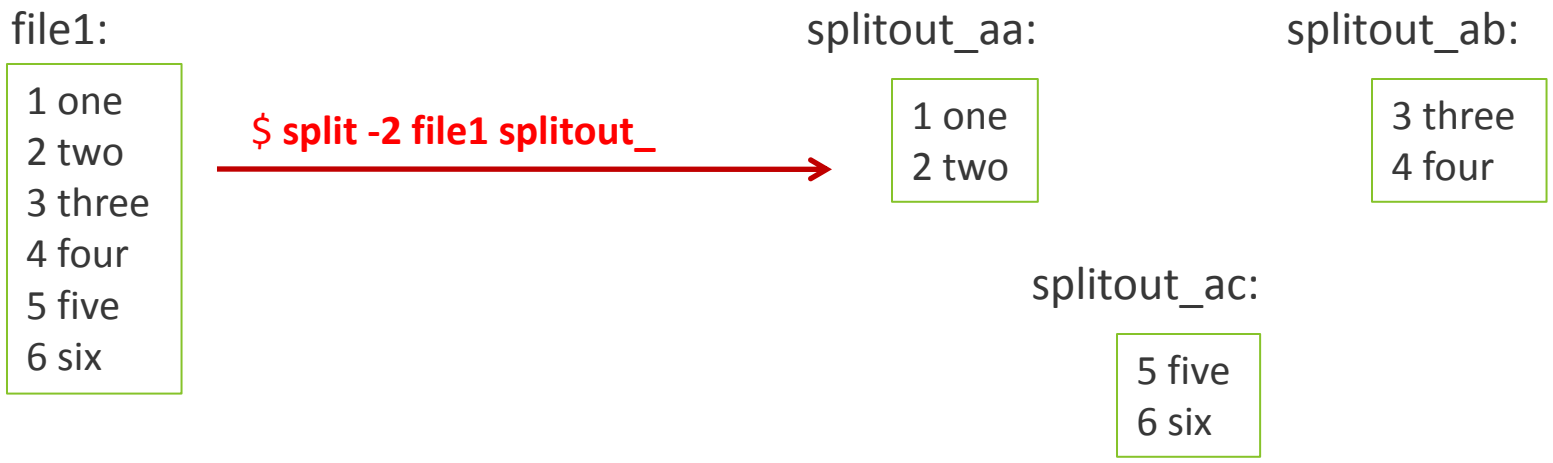
- *Sintaxe*
  - split [*option] [infile] [outfile]*
- *Descrição*
  - O Split comando split divide um arquivo de entrada em arquivos de saida de tamanhos fixos. Ou seja, é muito utilizado para mandar arquivos grandes pela internet. Onde, os arquivos de saida tem nomes predefinidos: PREFIX (x por padrão) seguido de um grupo de caracteres como aa, ab, …

Processamento de Texto Utilizando Filtros

split

- *Opções freqüentes*
  - –n – Divide o arquivo em seguimentos de n linhas.
- *Exemplo*

- Suponha *file1:*



Processamento de Texto Utilizando Filtros

tac

- *Sintaxe*
  - tac [*file*]
- *Descrição*
  - Este comando é nomeado ao contrario do comando cat, pois simplesmente imprime arquivos de texto com as linhas em ordem reversa, que é o contrario do comando cat.
- *Exemplo*

Processamento de Texto Utilizando Filtros

tail

- *Sintaxe*
  - tail [*options*] *[files]*
- *Descrição*
  - Imprime as ultimas linhas de um ou mais arquivos (“tail” = calda). Quando mais de um arquivo é especificado, um cabeçalho é impresso no começo de cada arquivo e são listadas em sucessão.

Processamento de Texto Utilizando Filtros

tail

- *Opções Freqüentes*
  - –c *n – esta opção imprime os últimos n bytes, ou se n for seguido por um k ou m, imprime as ultimas n kilobytes ou n megabytes, respectivamente.*
  - –f – imprime o arquivo continuamente enquanto outro processo escreve no arquivo. Esta opção é útil para acompanhar arquivos de log enquanto o sistema ainda está em execução.
  - –n *m – imprime as últimas m linhas. O padrão é 10.*

Processamento de Texto Utilizando Filtros

tail

- Exemplo

file1:

```
1 one
2 two
3 three
4 four
5 five
6 six
7 seven
8 eight
9 nine
10 ten
11 eleven
```

$ tail file1 →

saida1:

```
2 two
3 three
4 four
5 five
6 six
7 seven
8 eight
9 nine
10 ten
11 eleven
```

$ tail –n 4 file1 →

saida2:

```
8 eight
9 nine
10 ten
11 eleven
```

Processamento de Texto Utilizando Filtros

tr

- *Sintaxe*
  - tr [*options] [[string1 [string2]]*
- *Descrição*
  - Traduz os caracteres da string1 para os caracteres correspondentes da string2. tr não tem arquivos como argumentos e portanto deve usar um padrão para a entrada e saída. Se a string 1 e a string2 estiverem no range (a-z ou A-Z), elas deverão representar o mesmo numero de caracteres.
- *Opções freqüentes*
  - –d – Deleta os caracteres da string1 da saída.
  - –s – ignora caracteres repetidos na string1.

Processamento de Texto Utilizando Filtros

tr

- *Exemplo 1*
  - Para mudar todos os caracteres minúsculos para maiúsculos do arquivo 1 use os seguintes comandos:
    - $ **cat arquivo1| tr a-z A-Z ou** $ **tr a-z A-Z < arquivo1**
- *Exemplo 2*
  - Para suprimir caracteres “a” repetidos no arquivo1:
    - $ **cat arquivo1| tr -s a**
- *Example 3*
  - Para remover todos os “a”, “b” e “c” do arquivo1:
    - $ **cat arquivo1| tr -d abc**

Processamento de Texto Utilizando Filtros

wc

- *Sintaxe*
  - wc [*options*] *[files]*
- *Descrição*
  - Imprime a contagem de caracteres, palavras e linhas para arquivos. Quando múltiplos arquivos são listados são usadas uma linha para cada arquivo e uma linha acumulativa no final.
- Opções freqüentes
  - –c – imprime só a contagem de caracteres.
  - –l – imprime só a contagem de linhas.
  - –w – imprime só a contagem de palavras.

Processamento de Texto Utilizando Filtros

wc

- *Exemplo 1*
  - Mostra todas as contagens dos arquivos *file1, file2, and file3:*
    - $ **wc file[123]**
- *Exemplo 2*
  - Conta o numero de linhas no arquivo1:
    - $ **wc -l arquivo1**

Processamento de Texto Utilizando Filtros

sort

- *Sintaxe*
  - sort [*options*] [*files*]
- *Descrição*
  - Imprime uma concatenação ordenada de todos os arquivos na saída padrão.
- Opções freqüentes
  - –n – Compara de acordo com o valor numerico da string
  - –r – Reverte o resultado da ordenação
- Exemplo
  - Ordena reversamente dois arquivos file1 e file2
    - $ sort –r file1 file2

Curso de Introdução ao Linux

Aula 1

Angelo Brito - asb
Adriano Melo - astm