CURSO DE INTRODUÇÃO AO LINUX

Distribuição openSUSE®

AULA 7 – Manipulação de texto

PETEC
ENGENHARIA DE COMPUTAÇÃO

Objetivos dessa aula

- Se familiarizar com editores de texto:
  - **nano**, editor simples em modo texto;
  - **kwrite**, editor simples gráfico;
  - **vi** e **emacs**, editores avançados com interfaces de modo texto e gráfica.
- Mostrar e adicionar conteúdo a arquivos com **cat** e **echo**;
- Buscar padrões com **grep**;

Editores básicos

Visão geral

Em algum ponto você precisará editar manualmente **arquivos de texto**. Você talvez esteja escrevendo um email estando offline, escrevendo um script para usar no **bash** ou outros interpretadores, alterando um arquivo de configuração do sistema ou de algum programa, ou ainda desenvolvendo código fonte de alguma linguagem de programação.

Administradores Linux muitas vezes ignoram os editores de texto, ao usar ferramentas gráficas para criar e modificar configurações do sistema. Porém, isso pode ser muito mais trabalhoso do que usar diretamente um editor.

Editores básicos

Visão geral

Aplicações de processamento de palavras, tal qual o **Notepad** ou aqueles que fazem parte de suítes de escritório (e.g. MS Word ou Libreoffice Writer), não são realmente editores básicos de texto pois adicional muita informação de formatação adicional (geralmente invisível) que provavelmente tornará arquivos de configuração e administração do sistema inutilizáveis para seu propósito desejado.

Como o Linux é repleto de escolhas, quando se trata de editores de texto você terá uma vasta gama de aplicações disponíveis, das mais simples às mais complexas, como:

- **nano**;
- **kwrite**;
- **vi**;
- **emacs**.

Editores básicos

Criando arquivos sem um Editor

Às vezes você terá vontade de criar um arquivo curto e não pretende abrir um editor de texto completo. Adicionalmente, fazer isso pode ser útil quando executado a partir de scripts, mesmo ao criar arquivos longos.

Se você quiser criar um arquivo sem usar um editor, há duas maneiras padrão para fazê-lo a partir da linha de comando e preenchê-lo de conteúdo.

Editores básicos

Criando arquivos sem um Editor

A primeira maneira é usando repetidamente o comando **echo**:

- $ echo linha um > arquivo
- $ echo linha dois >> arquivo
- $ echo linha três >> arquivo

Vimos anteriormente que um sinal maior-que (>) envia a saída de um comando para um arquivo. Dois sinais maior-que (>>) adiciona nova saída a um arquivo existente.

Editores básicos

Criando arquivos sem um Editor

A segunda maneira é através do uso de **cat** combinado com redirecionamento:

- $ cat << EOF > arquivo
- > linha um
- > linha dois
- > linha três
- > EOF
- $

Ambos os métodos anteriores, que são extremamente úteis quando aplicados em scripts, produzirão um arquivo com as seguintes linhas:

- linha um
- linha dois
- linha três

Editores básicos

Introdução aos editores nano e kwrite

Há alguns editores de texto que são bem óbvios; a curva de experiência é suave e são bastante capazes. Um editor particularmente fácil de ser usado é o **nano**, editor baseado em modo terminal de texto. Apenas invoque o **nano** dando um nome de arquivo por argumento. Toda ajuda que você precisar será mostrada na parte de baixo da tela, e você deve ser capaz de proceder sem problemas.

Como editor com interface gráfica, o **kwrite** é associado ao sistema de desktop KDE (**gedit** para o GNOME). Tanto o kwrite quanto o gedit são muito fáceis de usar e são extremamente capazes e altamente configuráveis. O KDE pode vir por padrão com a variante **kate** instalada.

Editores básicos

nano

O **nano** é fácil de usar, e requer pouco esforço para aprender. Para abrir um arquivo, digite *nano <arquivo>* e tecle ENTER. Se o arquivo não existir, ele será criado.

O programa providencia uma "barra de atalhos" de duas linhas embaixo na tela que lista os comandos disponíveis. Alguns desses comandos são:

- **CTRL-G** : Mostrar a tela de ajuda
- **CTRL-O** : Gravar para arquivo
- **CTRL-X** : Sair do arquivo
- **CTRL-R** : Inserir o conteúdo de outro arquivo
- **CTRL-C** : Cancelar comandos anteriores

Editores básicos

kwrite

O **kwrite** é um editor de modo gráfico simples de ser usado, que pode ser executado a partir de um ambiente de desktop gráfica. Visualmente, é bem parecido com o **Notepad** do Windows, mas é muito mais capaz, sendo muito configurável e possuindo uma gama de plugins disponíveis para estender ainda mais sua capacidade.

Para abrir um novo arquivo no kwrite, basta abrir o programa a partir da interface gráfica, ou pela linha de comando digitando *kwrite <arquivo>*. Se o arquivo não existir, será criado.

Usar o kwrite é bastante intuitivo, não necessitando muito treinamento. Sua interface é composta de elementos familiares.

Editores mais avançados

vi e emacs

Desenvolvedores e administradores mais versados em trabalhar com sistemas de base UNIX quase sempre usarão uma de duas opções de editores veneráveis: **vi** ou **emacs**. Ambos estão facilmente disponíveis em qualquer distribuição e são completamente compatíveis com versões disponíveis em outros sistemas operacionais.

Tanto **vi** quanto **emacs** possuem uma forma básica puramente baseada em texto que pode ser executada em um ambiente não-gráfico. Eles também possuem uma ou mais formas gráficas de base **X** com capacidades estendidas: essas podem ser mais amigáveis para usuários menos experientes.

Ao passo que vi e emacs possuem curvas de aprendizado significantemente íngrimes para novos usuários, eles são extremamente eficientes quando dominados.

Editores mais avançados

Introdução ao vi

Geralmente o programa realmente instalado em seu sistema é o **vim** (vi melhorado), que usa **vi** por alias.

Mesmo que você não queira usá-lo, é bom pelo menos familiarizar-se com ele: é uma ferramenta padrão instalada em quase toda distribuição Linux, e às vezes único editor de texto disponível imediatamente.

O KDE oferece (mas não mantém) a interface gráfica **kvim**. O GNOME estende o vi com a interface gráfica **gvim**.

Ao usar o vi, todos os comandos são realizados através do teclado – você não precisa mover suas mãos para um dispositivo apontador, a não ser que você queira fazê-lo em uma das versões gráficas do editor.

Editores mais avançados

vimtutor

Executando **vimtutor**, inicia-se um curto mas compreensivo tutorial para aqueles que desejam aprender seus primeiros comandos no vi. Esse tutorial é um bom início para aprender a usar o editor.

Mesmo que ele possua somente uma introdução e sete lições, ele tem material o suficiente para te tornar um usuário proficiente em vi pois cobre um grande número de comandos. Após aprender esse básico, você sempre pode procurar por novos truques para adicionar à sua lista de comandos do vi, pois sempre há maneiras otimizadas de trabalhar em vi digitando menos.

Editores mais avançados

Modos no vi

O vi trabalha em três **modos**, e é vital se manter atento a qual modo você está trabalhando. Vários comandos e teclas se comportam diferentemente dependendo do modo.

- **Comando**: Modo padrão, aperte ESC quando nos outros modos
  - Cada tecla é interpretada como um comando do editor que pode modificar o conteúdo de arquivos;
- **Inserção**: aperte i a partir do modo Comando
  - Usado para inserir texto, indicado pela inscrição "**? INSERT ?**" embaixo na tela;
- **Linha**: aperte : a partir do modo Comando
  - Cada caractere é um comando externo, incluindo operações de gravação e saída.

PETeco
ENGENHARIA DE COMPUTAÇÃO

Editores mais avançados

Trabalhando com arquivos no vi

A lista a seguir mostra alguns comandos importantes. Tecle ENTER para realizar o comando desejado.

- $ vi <arquivo> – Iniciar o editor e editar o arquivo desejado;
- $ vi -r <arquivo> - Iniciar o vi e editar o arquivo desejado em modo recuperação;
- :r <outro> - Ler outro arquivo e inserir seu conteúdo na posição atual;
- :w – Gravar o arquivo;
- :w <nome> - Gravar arquivo com o nome dado;
- :w! <outro> - Sobrescrever outro arquivo;
- :x/:wq – Gravar arquivo e sair do vi;
- :q – Sair do vi;
- :q! - Sair do vi descartando alterações.

Editores mais avançados

Mudando a posição do cursor no vi

A lista a seguir mostra teclas importantes para mudar a posição do cursor. Comandos usando : requerem que ENTER seja pressionado depois.

- Teclas de seta – Mover o cursor;
- j – Mover o cursor uma linha abaixo;
- k – Mover o cursor uma linha acima;
- h – Mover o cursor um caractere à esquerda;
- l – Mover o cursor um caractere à direita;
- 0 – Mover para o início da linha;
- $ – Mover para o fim da linha;
- w – Mover para o início da próxima palavra;
- :0 – Mover para o início do arquivo;
- :n – Mover para a linha n;
- :$ – Mover para a última linha do arquivo;
- Page Down – Avançar uma página;
- Page Up – Voltar uma página.

Editores mais avançados

Procurando texto no vi

A seguir a lista de comandos e teclas mais importantes para procurar texto no vi:

- /padrão – Procurar por um padrão à frente;
- ?padrão – Procurar por um padrão atrás;
- n – Ir até a próxima ocorrência do padrão procurado;
- N – Ir até a ocorrência anterior do padrão procurado.

Editores mais avançados

Trabalhando texto no vi

- a – Adicionar texto após o cursor. Encerrar apertando ESC;
- A – Adicionar texto ao fim da linha atual. Encerrar apertando ESC;
- i – Inserir texto antes do cursor. Encerrar apertando ESC;
- I - Inserir texto no início da linha atual. Encerrar apertando ESC;
- o – Começar uma nova linha abaixo da atual, inserir texto então. Encerrar apertando ESC;
- O – Começar uma nova linha acima da atual, inserir texto então. Encerrar apertando ESC;
- r – Repor caractere na posição atual;
- R – Repor texto começando na posição atual. Encerrar apertando ESC;
- x – Deletar caractere na posição atual;

Editores mais avançados

Trabalhando texto no vi

- Nx – Deletar N caracteres, a partir da posição atual;
- dw – Deletar a palavra na posição atual;
- D – Deletar o resto da linha atual;
- dd – Deletar a linha atual;
- Ndd ou dNd – Deletar N linhas;
- u – Desfazer a última operação;
- yy – Copiar a linha atual;
- Nyy ou yNy – Copiar N linhas;
- p – Colar a(s) linha(s) copiada(s) na posição atual;

PET COMP
ENGENHARIA DE COMPUTAÇÃO

Editores mais avançados

Introdução ao emacs

O editor **emacs** é um competidor popular ao vi, mas ao contrário deste último, não trabalha em modos. O emacs é altamente personalizável e inclui um bom número de características. Foi inicialmente projetado para uso em console, mas foi logo adaptado para trabalhar também com uma GUI. O emacs possui várias outras capacidades além de editar texto: pode ser usado para emails, depuradores, etc.

Ao invés de possuir diferentes modos para comando e inserção, o emacs usa as teclas CTRL e Meta (Alt e Esc) para comandos especiais.

Editores mais avançados

Trabalhando com o emacs

A lista a seguir mostra algumas combinações de teclas importantes, usadas para iniciar, sair, abrir ou gravar arquivos com o emacs.

- $ emacs <arquivo> - Inicia o emacs e edita o arquivo desejado;
- CTRL-x i – Inserir arquivo na posição atual;
- CTRL-x s – Salvar todos os arquivos;
- CTRL-x CTRL-w – Gravar arquivo, dando um novo nome quando solicitado;
- CTRL-x CTRL-s – Salvar arquivo atual;
- CTRL-x CTRL-c – Sair após perguntar para salvar qualquer arquivo modificado.

O tutorial do emacs é um bom começo para aprender comandos básicos do editor. Ele está disponível a qualquer momento no emacs usando o comando **CTRL-h t**.

PETeco
ENGENHARIA DE COMPUTAÇÃO

Editores mais avançados

Mudando a posição do cursor no emacs

- Teclas de seta – Mover o cursor;
- CTRL-n – Descer uma linha;
- CTRL-p – Subir uma linha;
- CTRL-f – Mover o cursor um caractere à direita;
- CTRL-b – Mover o cursor um caractere à esquerda;
- CTRL-a – Mover para o início da linha;
- CTRL-e – Mover para o fim da linha;
- Meta-f – Mover para o início da próxima palavra;
- Meta-b – Mover para o início da palavra anterior;
- Meta-< – Mover para o início do arquivo;
- Meta-g-g-n – Mover para a linha n (igual a Esc-x Goto-line n);
- Meta-> – Mover para o fim do arquivo;
- Page Down – Avançar uma página;
- Page Up – Voltar uma página.

Editores mais avançados

Procurando texto no emacs

- CTRL-s – Procurar à frente pelo padrão inserido, ou pela próxima ocorrência;
- CTRL-r – Procurar atrás pelo padrão inserido, ou pela ocorrência anterior.

Editores mais avançados

Trabalhando texto no emacs

- CTRL-o – Inserir linha em branco;
- CTRL-d – Deletar caractere na posição atual;
- CTRL-k – Deletar o resto da linha atual;
- CTRL-_ - Desfazer a operação anterior;
- CTRL-Espaço – Marcar o começo de uma região selecionada;
- CTRL-w – Deletar o texto marcado e escrever no buffer;
- CTRL-y – Inserir na posição atual do cursor o que quer que tenha sido deletado mais recentemente.

Manipulação de texto

Ferramentas de Linha de Comando

Independente de seu papel com o Linux (administrador, desenvolvedor ou usuário) você constantemente precisará vasculhar e analisar arquivos de texto e/ou dados deles. Essas são operações de **manipulação de arquivos**. Logo, é essencial que o usuário Linux se torne adepto de realizar certas operações.

Na maior parte do tempo tal manipulação é feita na linha de comando, que permite aos usuários realizar tarefas mais eficientemente do que usando uma GUI. Além disso, a CLI é mais adequada para automatizar tarefas frequentes.

cat e echo

cat

cat é a abreviação de concatenar e é uma das utilidades de linha de comando mais usadas no Linux. É frequentemente usado para ler e mostrar arquivos, além de seus conteúdos. Para ver um arquivo, basta usar o seguinte comando:

- $ cat <arquivo>

O principal propósito do cat, porém, é combinar (concatenar) múltiplos arquivos juntos. É possível realizar as ações da lista a seguir usando cat:

cat e echo

cat

- cat arquivo1 arquivo2 – Concatenar múltiplos arquivos e mostrar a saída, i.e, o conteúdo completo do primeiro arquivo seguido do conteúdo do segundo;
- cat arquivo1 arquivo2 > novoarquivo – Combinar múltiplos arquivos e salvar a saída em um novo;
- cat arquivo >> outro – Adiciona o conteúdo de um arquivo ao fim de outro já existente;
- cat > arquivo – Toda linha subsequente digitada irá ao arquivo até que CTRL-D seja digitado.
- cat >> arquivo – Toda linha subsequente será adicionada ao fim do arquivo até que CTRL-D seja digitado.

O comando **tac** exibe as linhas de um arquivo em ordem reversa, mantendo a mesma sintaxe que o **cat**.

cat e echo

Usando cat interativamente

cat pode ser usado para ler da entrada padrão (tal como a janela de terminal) se nenhum arquivo for especificado. Você pode usar o operador > para criar e adicionar linhas a um novo arquivo, e o operador >> para adicionar linhas (ou arquivos) a um já existente.

Para criar um novo arquivo, no prompt de comando digite *cat <arquivo>* e tecle ENTER. Esse comando cria um novo arquivo e espera que o usuário edite/entre texto. Após digitar o texto necessário, tecle CTRL-D ao início da próxima linha para salvar e sair da edição.

Outra maneira para criar um arquivo a partir do terminal é *cat > <arquivo> << EOF*. Um novo arquivo é criado e você pode digitar todo texto necessário. Para sair, digite EOF (sensível a maiúsculas e minúsculas) ao início de uma nova linha.

cat e echo

echo

echo simplesmente mostra (ecoa) texto. É usado simplesmente, como em:

- $ echo <string>

echo pode ser usado para mostrar uma string na **saída padrão** ou para adicionar a um novo arquivo (com o operador >) ou a um arquivo existente (com o operador >>).

A opção **-e** junto das seguintes switches é usada para habilitar sequências especiais de caracteres, tais como o caractere **nova linha** ou **tabulação horizontal**.

- **\n** representa nova linha
- **\t** representa tabulação horizontal.

cat e echo

echo

O **echo** é particularmente útil para visualizar o valor de variáveis de ambiente (como visto anteriormente). A lista a seguir mostra alguns comandos e utilizações do echo:

- $ echo <string> > novoarquivo – Coloca a string especificada em um novo arquivo;
- $ echo <string> >> arquivo – Adiciona a string especificada ao fim de um arquivo já existente;
- $ echo $VARIAVEL – Mostra o conteúdo da variável de ambiente especificada.

Trabalhando com arquivos grandes

Administradores de sistema precisam trabalhar com arquivos de configuração, de texto, de documantação e de registro. Alguns destes arquivos podem ser grandes ou se tornar grandes conforme acumularem dados com o tempo. Estes arquivos vão precisar tanto de visualização e atualização administrativa.

Para visualizar um arquivo grande sem sobrecarregar a memória ao abrir um editor de texto, podem ser usados os seguintes comandos:

- $ less <arquivo>
- $ cat <arquivo> | less

head

head lê algumas primeiras linhas (10 por padrão) de cada arquivo especificado e mostra na saída padrão. É possível especificar um número diferente de linhas com a opção **-n**.

Por exemplo, se você quiser exibir as 5 primeiras linhas do arquivo *lista.txt*, você usaria o seguinte comando:

- $ head -n 5 lista.txt

tail

tail exibe algumas últimas linhas (10 por padrão) de cada arquivo especificado na saída padrão. É possível especificar um número diferente de linhas com a opção **-n**.

Por exemplo, se você quiser exibir as 15 últimas linhas do arquivo *lista.txt*, você usaria o seguinte comando:

- $ tail -n 15 lista.txt

O comando tail é especialmente útil para avaliar as últimas entradas em arquivos de log, de modo a resolver problemas recentes.

Ferramentas de manipulação de arquivos

Ao gerenciar seus arquivos você pode precisar realizar várias tarefas, como ordenar dados e copiar dados de um lugar a outro. O Linux te providencia várias ferramentas de manipulação de arquivos que você pode usar enquanto trabalha com arquivos de texto, como:

- sort;
- uniq;
- paste;
- join;
- split.

sort

sort é usado para reorganizar as linhas de um arquivo de texto em ordem ascendente ou descentende. Você também pode organizar por campos particulares de um arquivo. A organização padrão é pela ordem de caracteres ASCII (alfabeticamente, essencialmente).

- $ sort <arquivo> – Organiza as linhas no arquivo especificado;
- $ cat arquivo1 arquivo2 | sort – Concatena os dois arquivos, organiza as linhas e mostra a saída no terminal;
- $ sort -r <arquivo> – Organiza as linhas no arquivo em ordem reversa.

PET COMP
ENGENHARIA DE COMPUTAÇÃO

uniq

uniq é usado para remover linhas duplicadas em um arquivo de texto e é útil para simplificar a exibição de texto. O uniq requer que entradas duplicadas estejam consecutivas para serem removidas, logo é comum usar **sort** antes e canalizar a saída para **uniq**. Se usar a opção **-u** no comando **sort**, essa ação é unificada.

Para contar o número de entradas duplicadas, use o seguinte comando:

- $ uniq -c <arquivo>

paste

Suponha que você possua um arquivo que contém o nome completo de todos os empregados e outro arquivo que liste seus telefones e Ids. Você vai querer criar um novo arquivo que contenha todos os dados listados nas três colunas, nome, ID e telefone.

paste pode ser usado para criar um arquivo único contendo todas as três colunas. As colunas diferentes são identificadas com base em delimitadores (espaçamento usado para separar dois campos). Por exemplo, delimitadores podem ser espaços, tabulações ou um ENTER. O comando paste aceita as seguintes opções:

- **-d <delimitadores>** : Especificar uma lista de delimitadores para serem usados ao invés de tabulações;
- **-s** : Força o paste a adicionar dados serialmente, ao invés de paralelamente.

paste

paste pode ser usado para combinar campos de diferentes arquivos, assim como combinar linhas de múltiplos arquivos. Por exemplo, a linha 1 do arquivo1 pode ser combinada com a linha 1 do arquivo2.

Para colar conteúdos de dois arquivos:

- $ paste arquivo1 arquivo2

A sintaxe para usar um delimitador diferente é:

- $ paste -d ':' arquivo1 arquivo2

join

Suponha que você tenha dois arquivos com algumas colunas similares. Você salvou os telefones de seus empregados em dois arquivos, um com seus primeiros nomes, o outro com seus sobrenomes. Você quer combiná-los sem repedir a coluna comum.

Para isso, você usará a ferramenta **join**, que essencialmente é uma versão aprimorada do **paste**. O join primeiro confere se os arquivos possuem campos comuns, e então junta as linhas de dois arquivos baseado em um campo comum.

Para combinar dois arquivos em um campo comum, no terminal use o seguinte comando:

- $ join arquivo1 arquivo2

split

split é usado para quebrar um arquivo em segmentos de mesmo tamanho para facilitar a visualização e manipulação, e geralmente é usado apenas em arquivos relativamente grandes.

Por padrão, o split quebra um arquivo em segmentos de 1000 linhas. O arquivo original permanece inalterado, e um conjunto de novos arquivos com o mesmo nome sequido de um prefixo é criado. Por padrão, o prefixo **x** é adicionado.

Para dividir um arquivo em segmentos, alterando o prefixo, use o seguinte comando:

- $ split <arquivo> <prefixo>

grep

grep é extensivamente usado como uma ferramenta de busca de texto primária. Ele vasculha arquivos por padrões especificados e pode ser usado com as seguintes opções:

- $ grep [padrão] <arquivo> - Procura por um padrão no arquivo e imprime todas as linhas correspondentes;
- $ grep -v [padrão] <arquivo> - Imprime todas as linhas que **não** correspondem ao padrão;
- $ grep [0-9] <arquivo> - Imprime as linhas que contêm os números 0 até 9;
- $ grep -C n [padrão] <arquivo> - Imprime o contexto (número n de linhas acima e abaixo) das linhas correspondam ao padrão.