Sed por exemplos - Parte 1

Conheça o poderoso editor UNIX

Daniel Robbins
President and CEO, Gentoo Technologies, Inc.
Setembro de 2000
http://www-106.ibm.com/developerworks/linux/library/l-sed1.html

Nesta série de artigos, Daniel Robbins irá mostrar a você como usar o poderosíssimo (e quase sempre esquecido) editor de stream, o sed. O sed é uma ferramenta ideal para editar arquivos em lote ou para criar scripts shell que modificam arquivos de formas poderosas.

Conteúdo

Escolha um editor

No mundo UNIX, temos muitas opções quando o assunto é editar arquivos. Pense nisto -- vi, emacs, e jed vêm à lembrança, bem como muitos outros. Todos temos nosso editor favorito (junto com nossos atalhos de teclado prediletos) que conhecemos e gostamos. Com nosso editor de confiança, estamos prontos para lidar com qualquer tarefa de administração Unix ou de programação com facilidade.

Apesar dos editores interativos serem ótimos, eles tem limitações. Apesar de sua natureza interativa ser um ponto forte, ela também pode ser uma fraqueza. Considere uma situação em que você precisa executar alterações similares em um grupo de arquivos. Você pode instintivamente inicar seu editor favorito e executar uma batelada de edições mundanas, repetitivas, e demoradas à mão. Mas há uma forma melhor de fazer isto.

Entra o sed

Seria legal se pudéssemos automatizar o processo de fazer edições em arquivos, de forma a poder fazer edições em "lote", ou mesmo escrever scripts que tenham a habilidade de executar alterações sofisticadas a arquivos existentes. Para nossa sorte, para este tipo de situação, existe uma forma melhor -- e esta forma melhor é chamada "sed".

O sed é um editor de stream bastante leve que está incluído em quase todos os sabores de UNIX, incluindo o Linux. O sed tem muitas funcionalidades legais. Primeiro, ele é bastante leve, tipicamente muitas vezes menor que sua linguagem de script favorita. Segundo, como o sed é um editor de *stream*, ele pode fazer edições nos dados que recebe de stdin, como o que vem de um pipeline. Assim, você não precisa que os dados a serem editados estejam armazenados em um arquivo em disco. Como os dados podem ser assim facilmente "pipados" para o sed, é

muito fácil usar o sed como parte de uma pipeline longa e complexa em um shell script poderoso. Tente fazer isto com seu editor favorito.

GNU sed

Para a sorte nossa, usuários Linux, uma das versões mais legais do sed é o GNU sed, atualmente na versão 3.02. Todas distribuições do Linux têm o GNU sed, ou pelo menos deveriam. O GNU sed é popular não só por que seu código fonte pode ser livremente distribuído, mas por que ele tem muitas extensões úteis e que economizam tempo ao padrão sed POSIX. O GNU sed também não sofre as muitas limitações que versões anteriores e proprietárias do sed tem, como um comprimento de linha limitado -- o GNU sed consegue tratar linhas de qualquer comprimento com facilidade.

O GNU sed mais novo

Enquanto estava fazendo a pesquisa para este artigo, eu notei que vários aficcionados no sed online fazem referência ao GNU sed 3.02a. Estranhamente, eu não conseguia encontrar o sed 3.02 em **ftp.gnu.org** (veja Recursos para estes links), de forma que precisei procurar por ele em outro lugar. Eu encontrei o mesmo em **alpha.gnu.org**, em /pub/sed. Eu alegremente fiz o download, compilei, e instalei, só para descobrir minutos mais tarde que a versão mais recente do sed é a 3.02.80 -- e os fontes podem ser encontrados bem ao lado dos fontes da versão 3.02a, em **alpha.gnu.org**. Depois de baixar o GNU sed 3.02.80 e instalar o mesmo, eu finalmente estava pronto para seguir adiante.

alpha.gnu.org

O **alpha.gnu.org** (veja Recursos) é o lar de código fonte GNU novo e experimental. Entretanto, você irá encontrar muito código fonte legal e estável, também. Por alguma razão, muitos desenvolvedores GNU esquecem de mover código fonte estável para o ftp.gnu.org, ou eles tem períodos beta incrivelmente longos (2 anos!). Por exemplo, o sed 3.02a tem dois anos de idade, e mesmo o sed 3.02.80 tem um ano de idade, mas até agora (este artigo foi escrito em agosto de 2000) não estavam disponíveis em ftp.gnu.org!

O sed correto

Nesta série, iremos usar o GNU sed 3.02.80. Alguns (mas bem poucos) dos exemplos mais avançados que você verá no próximo artido desta série não irão funcionar no GNU sed 3.02 ou 3.02a. Se você está usando um sed não-GNU, os resultados podem variar. Por que não separar um tempo agora para instalar o GNU sed 3.02.80? Assim, não só você estará pronto para o resto da série, mas poderá usar o melhor sed em existência!

Exemplos sed

O sed trabalha executando qualquer número de operações de edição especificadas pelo usuário ("comandos") nos dados de entrada. O sed é baseado em linhas, assim os comandos são executados em cada linha, em ordem. E, o sed escreve o resultado na saída padrão (stdout) -- ele não modifica nenhum arquivo de entrada.

Vamos ver alguns exemplos. Os primeiros serão um pouco estranhos por que estarei usando-os para ilustrar como o sed funciona em vez de fazer alguma tarefa útil. Entretanto, se você é novo no sed, é muito importante que você os entenda. Aqui está nosso primeiro exemplo:

```
$ sed -e 'd' /etc/services
```

Se você escrever este comando, você não verá absolutamente nenhuma saída. Agora, o que aconteceu? Neste exemplo, chamamos o sed com um comando de edição, 'd'. O sed abriu o arquivo /etc/services, leu uma linha para seu buffer de padrões, executou nosso comando de edição ("apague linha"), e então escreveu o buffer de padrões (que estava vazio). Ele repetiu estes passos para cada linha sucessiva. Este comando não produziu saída, por que o comando "d" apagou todas as linhas no buffer de padrões!

Existem algumas coisas que devem ser notadas neste exemplo. Primeiro, o arquivo /etc/services não foi modificado. Isto é por que, novamente, o sed somente lê o arquivo especificado na linha de comando, usando o mesmo como entrada -- ele não tenta modificar o arquivo. A segunda coisa a notar é que o sed é orientado a linhas. O comando 'd' não diz simplesmente ao sed para apagar todos os dados de entrada de uma só vez. Ao invés disto, o sed lê cada linha de /etc/services, uma de cada vez, para seu buffer interno, chamado de buffer de padrões. Uma vez que uma linha foi lida para o buffer de padrões, ele executou o comando 'd' e escreveu o conteúdo do buffer de padrões (nada, neste exemplo). Mais tarde, irei mostrar como usar intervalos de endereços para controlar a quais linhas um comando é aplicado -- mas na ausência de endereços, um comando é aplicado a *todas as linhas*.

A terceira coisa a notar é o uso de apóstrofos em torno do comando 'd'. É uma boa idéia habituar-se a usar apóstrofos para delimitar seus comandos sed, pois desta forma a expansão shell é desabilitada.

Outro exemplo sed

Segue outro exemplo de como usar o sed para remover a primeira linha do arquivo /etc/services de sua stream de saída:

```
$ sed -e '1d' /etc/services | more
```

Como pode ser visto, este comando é bastante similar a nosso primeiro comando 'd', exceto que ele é precedido por um '1'. Se você adivinhou que o '1' refere-se à linha de número um, você está certo. Enquanto no nosso primeiro exemplo usamos o 'd' sozinho, desta vez usamos o comando 'd' precedido por um endereço numérico opcional. Usando endereços, você pode informar ao sed para executar edições somente em uma linha particular ou algumas linhas particulares.

Intervalos de endereços

Vamos, agora, ver como deve ser especificado um *intervalo* de endereços. Neste exemplo, o sed irá apagar as linhas 1-10 da entrada:

```
$ sed -e '1,10d' /etc/services | more
```

Quando separamos dois endereços por uma vírgula, o sed irá aplicar o comando que segue ao intervalo que começa no primeiro endereço, e termina no segundo endereço. Neste exemplo, o comando 'd' foi aplicado às linhas 1-10, inclusive. Todas as outras linhas foram ignoradas.

Endereços com expressões regulares

Agora é hora de um exemplo mais útil. Digamos que você quer ver o conteúdo do seu arquivo /etc/services, mas não está interessado em ver qualquer um do comentários incluídos. Como você sabe, comentários podem ser colocados no arquivo /etc/services começando a linha com o caracter '#'. Para evitar os comentários, queremos que o sed apague as linhas que começam com um '#'. Aqui está como fazer isto:

```
$ sed -e '/^#/d' /etc/services | more
```

Tente este exemplo e veja o que acontece. Você irá notar que o sed executa a tarefa desejada muito bem. Agora, vamos ver o que aconteceu.

Para entender o comando '/^#/d', primeiro precisamos dissecar o mesmo. Primeiro, vamos remover o 'd' -- estamos usando o mesmo comando de exclusão de linhas que usamos previamente. A nova adição é a parte '/^#/', que é um novo tipo de endereço de *expressão regular*. Endereços de expressão regular sempre estão entre barras. Eles especificam um *padrão*, e o comando que segue imediatamente um endereço de expressão regular somente será aplicado a uma linha se ela combinar com este padrão em particular.

Então, o '/^#/' é uma expressão regular. Mas o que ela faz? Mas o que ela faz? Obviamente é uma boa hora para relembrar as expressões regulares.

Relembrando expressões regulares

Podemos usar expressões regulares para expressar padrões que queremos encontrar no texto. Se você alguma vez usou o caracter "*" na linha de comando, voc~e usou algo que é similar, mas não idêntico, a expressões regulares. Aqui estão os caracteres especiais que você pode usar em expressões regulares.

Caracter	Descrição
^	Combina com o início de uma linha
$	Combina com o fim da linha
.	Combina com um único caracter
*	Combina com zero ou mais ocorrências do caracter *anterior*
[ ]	Combina todos os caracteres dentro dos [ ]

Provavelmente a melhor forma de tomar pé com expressões regulares é ver alguns exemplos. Todos os exemplos serão aceitos pelo sed como endereços válidos que podem aparecer no lado esquerdo de um comando. Aqui estão alguns:

Expressão regular	Descrição
/./	Combina qualquer linha que contenha pelo menos um caracter
/../	Combina qualquer linha que contenha pelo menos dois caracteres
/^#/	combina com qualquer linha que comece com um '#'
/^$/	Combina com todas as linhas em branco
/}$/	Combina com qualquer linha que termine com "}" (sem espaços)
/} *$/	Combina com qualquer linha que termine com um "}", seguido por zero ou mais espaços
/[abc]/	Combina com qualquer linha que contenha um 'a', 'b' ou 'c' minúsculos
/^[abc]/	Irá combinar com qualquer linha que *começar* com os caracteres 'a', 'b' ou 'c'

Eu encorajo você a tentar vários destes exemplos. Gaste algum tempo para ganhar familiaridade com expressões regulares, e tente algumas expressões regulares de sua própria criação. Você pdoe usar uma regexp desta forma:

```
$ sed -n -e '/regexp/p' /path/to/my/test/file | more
```

Note a opção '-n', que informa ao sed para não imprimir o espaço de padrões a menos que for explicitamente comandado para tal. Você também notará que trocamos o comando 'd' pelo comando 'p', que, como você pode estar adivinhando, comanda explicitamente o sed para escrever o espaço de padrões. Voila, agora somente as linhas que combinarem com o padrão serão escritas.

Mais sobre endereços

Até agora olhamos em endereçamento de linhas, endereços de intervalos de linhas, e endereços regexp. Mas ainda há mais possibilidades. Podemos especificar duas expressões regulares, separadas por uma vírgula, e o sed irá combinar todas as linhas a partir da primeira linha que combina com a primeira expressão regular, até e incluindo a linha que combina a segunda expressão regular. Por exemplo, o seguinte comando irá imprimir um bloco de texto que comece com uma linha contendo "BEGIN", e terminando com uma linha que contém "END":

```
$ sed -n -e '/BEGIN/,/END/p' /my/test/file | more
```

Se o "BEGIN" não for encontrado, nenhum dado será impresso. E, se o "BEGIN" for encontrado, mas nenhum "END" for encontrado em nenhuma linha que o siga, todas as linhas a seguir serão impressas. Isto acontece por que o sed é orientado a stream -- ele não sabe se um "END" irá ou não aparecer.

Exemplo com fonte C

Se você quiser imprimir somente a função main() em um arquivo fonte C, pode escrever:

```
$ sed -n -e '/main[[:space:]]*(/,/^}/p' sourcefile.c | more
```

Este comando tem duas expressões regulares, '/main[[:space:]]*(/' e '/^}/', e um comando, 'p'. A primeira expressão regular irá combinar com a string "main" seguida por qualquer número de espaços ou tabulações, seguido por um abre-parêntesis. Isto deve combinar com o início de sua declaração ANSI C main() mais comum.

Nesta expressão regular particular, encontramos a classe de caracteres '[[:space:]]'. Esta é simplesmente uma palavra chave especial que diz ao sed para combinar com um TAB ou espaço. Se você quisesse, em vez de escrever "[[:space:]]", poderia ter escrito um '[', seguido por um espaço literal, então um Control-V, e uma tabulação literal, e um ']' -- O Control-V informa ao bash que você quer inserir uma tabulação "real" ao invés de executar uma expansão de comando. É mais claro, especialmente em scripts, usar a classe de comando '[[:space:]]'.

OK, agora quanto à segunda regexp, '/^}', ela vai combinar com um caracter '}' que apareça no início de uma nova linha. Se seu código está bem formatado, este será o fecha-chaves da função main(). Se não for, não irá combinar -- uma dos truques sobre usar combinação de padrões.

O comando 'p' faz o que sempre faz, informa explicitamente ao sed que este deve escrever a linha, já que estamos no modo silencioso, '-n'. Tente executar o comando em um arquivo fonte C -- ele deve apresentar todo o bloco main() {}, incluindo o "main()" inicial, e o último "}".

O próximo artigo

Agora que já vimos o básico, vamos aumentar o passo nos próximos artigos. Se você mal pode esperar para ter algum material com mai substância, seja paciente -- ele está chegando! Enquanto isto, você pode querer checar os seguintes links sobre sed e expressões regulares.

Recursos

Sobre o sed:

- Leia os outros artigos do Daniel sobre o Sed: Sed por exemplos, parte 2 e parte 3
- Cheque o excelente sed FAQ, de Eric Pement
- Os fontes do sed 3.02 podem ser encontrados em ftp://ftp.gnu.org/pub/gnu/sed
- O novo sed 3.02.80 pode ser encontrado em alpha.gnu.org
- Eric Pement também tem uma utilíssima lista de one-liners sed, que qualquer aspirante a guru definitivamente deve dar uma olhada
- Se você prefere um livro, sed & awk, 2nd Edition é uma excelente escolha
- Talvez você queira ler a 7a. edição da página man do sed do UNIX (de cerca de 1978!)
- Estude o tutorial do sed de Felix von Leitner
- Leia o artigo "Text processing in Python", no *developerWorks*

Sobre expressões regulares:

- Estude o tutorial exclusivo do dW, o using regular expressions, para encontrar e modificar padrões de texto
- Veja o how-to sobre expressões regulares em Python.org.
- Veja também o overview of regular expressions da University of Kentucky.

Sobre o autor

Residindo em Albuquerque, Novo México, Daniel Robbins é o Chief Architect do Gentoo Project, CEO da Gentoo Technologies, Inc., o criador do Gentoo Linux, um Linux avançado para o PC, e o sistema Portage, um sistema de ports para o Linux de última geração. Ele também é um autor-contribuinte dos livros da Macmillan *Caldera OpenLinux Unleashed*, *SuSE Linux Unleashed*, e *Samba Unleashed*. Daniel está envolvido com computadores de alguma forma desde o segundo grau, quando ele foi exposto pela primeira vez à linguagem de programação Logo, bem como a uma dose podencialmente perigosa de Pac Man. Isto provavelmente explica por que ele tem servido desde então como Lead Graphic Artist na SONY Eletronic Publishing/ Psygnosis. Daniel gosta de passar o

tempo com sua esposa, Mary, e sua nova filhinha, Hadassah. Ele pode ser encontrado no email drobbins@gentoo.org.