Linux Academy

Hands-on Lab

Process Text Streams Using Filters - Working with the Textutils Package

Contents

Lab Connection Information

- Labs may take up to five minutes to build
- The IP address of your server is located on the Hands-on Lab page
- Username: linuxacademy
- Password: 123456
- Root Password: 123456

Related Courses

LPIC-1: System Administrator - Exam 101

Related Videos

Process Text Streams Using Filters

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

In this lab, we're going to learn to process text using a number of tools found in the `textutils` package.

To get started, log in to the server using the credentials provided on the Hands-on Lab page.

Download Template Files

First, we'll download some text files so that we have material on which to try out our filtering tools. Before we can download them, we'll need to install `git`:

```
[linuxacademy@ip] sudo yum install git
```

Create a temporary file to store the text files:

```
[linuxacademy@ip] mkdir tmp
[linuxacademy@ip] cd tmp
[linuxacademy@ip] git clone https://github.com/linuxacademy1/textfiles .
```

Once the files have download, we can list the files to see what we downloaded:

```
[linuxacademy@ip] ll
total 44
-rw-rw-r-- 1 linuxacademy linuxacademy   37 May  8 15:40 alpha.txt
-rw-rw-r-- 1 linuxacademy linuxacademy  162 May  8 15:40 columns.txt
-rw-rw-r-- 1 linuxacademy linuxacademy   28 May  8 15:40 file1.txt
-rw-rw-r-- 1 linuxacademy linuxacademy   29 May  8 15:40 file2.txt
-rw-rw-r-- 1 linuxacademy linuxacademy  339 May  8 15:40 nowisthetime.
txt
-rw-rw-r-- 1 linuxacademy linuxacademy   50 May  8 15:40 numbers.txt
-rw-rw-r-- 1 linuxacademy linuxacademy   47 May  8 15:40 README.md
-rw-rw-r-- 1 linuxacademy linuxacademy  332 May  8 15:40 sedexample.txt
-rw-rw-r-- 1 linuxacademy linuxacademy   64 May  8 15:40 tabs.txt
-rw-rw-r-- 1 linuxacademy linuxacademy  158 May  8 15:40 unique.txt
-rw-rw-r-- 1 linuxacademy linuxacademy 1538 May  8 15:40 users.txt
```

Use Text Processing Commands

Now that we've gotten some text samples to work with, we can start using different processing tools to see how they affect the text.

sort

The `sort` command allows us to put lines of text in order by alphabetic or numeric value. We'll use the `numbers.txt` file to explore its functionality.

```
[linuxacademy@ip] cat numbers.txt
```

To sort the numbers:

```
[linuxacademy@ip] sort numbers.txt
```

This might not be what we're expecting, however. By default, `sort` uses the first character in each line to arrange the result, rather than using actual numeric values. When we're working with numbers, the `-n` option provides a result more consistent with what we need:

```
[linuxacademy@ip] sort -n numbers.txt
```

This time, the numbers in the first column are sorted by actual value, not just their first digit.

We don't have to sort only by the first column's value, however. To sort the second column:

```
[linuxacademy@ip] sort -k2 -n numbers.txt
```

The `-k2` option specifies that the second delimited column should be used. Combined with `-n`, this sorts the second column by numeric value.

We can also use the `sort` command to sort alphabetically:

```
[linuxacademy@ip] cat alpha.txt
[linuxacademy@ip] sort alpha.txt
```

In the output, each line is listed alphabetically.

The `sort` command does not overwrite the source file. To create a permanent copy of the sorted text, we must redirect the output to a new file:

```
[linuxacademy@ip] sort alpha.txt > sortedalpha.txt
[linuxacademy@ip] cat sortedalpha.txt
```

When we view the new `sortedalpha.txt` file, we can see that it contains an alphabetical listing of the contents of the original.

nl

The `nl` command allows us to add line numbers to a file. This can be useful in situations where we need to send code or a configuration file to someone, while referencing a specific part of the file.

Let's first look at a file without line numbers:

```
[linuxacademy@ip] cat /etc/passwd
```

Supposed we wanted to show a team member one specific part of this file. Without line numbers, it may be difficult to explain exactly where in the file we're looking. To add line numbers to the file:

```
[linuxacademy@ip] nl /etc/passwd
```

This time, each line is prefixed with a number. This makes it easy to point someone to the exact line in a file that we want to review.

The default behavior is to assign numbers to only *non-blank* lines. In source code, for example, it may be useful to give numbers to *every* line, and we can do this with an option:

```
[linuxacademy@ip] nl -ba /etc/passwd
```

wc

The `wc` command allows us to count the number of items in a file, whether it be words, lines, or characters.

For example, to count the number of lines:

```
[linuxacademy@ip] wc -l /etc/passwd
22 /etc/passwd
```

We can also count the number of words:

```
[linuxacademy@ip] wc -w /etc/passwd
30 /etc/passwd
```

A string of characters separated by spaces is considered to be a single word, so keep this in mind when using the `wc` tool.

We can also count the number of characters:

```
[linuxacademy@ip] wc -c /etc/passwd
1012 /etc/passwd
```

The `wc` tool may also be used to count items in a directory, not only a file:

```
[linuxacademy@ip] ls -al /var | wc -l
20
```

In fact, we can use `wc` with any command that normally returns values to the standard output or standard

error by using a pipe:

```
[linuxacademy@ip] sudo cat /var/log/messages | wc -l
2404
```

expand

The `expand` command allows us to regulate text that contains columns delimited by different amounts of space. For example, let's look at the `tabs.txt` file:

```
[linuxacademy@ip] cat tabs.txt
column1 column2 column3
mycolumn1       mycolumn2       mycolumn3
c1      c2      c3
```

We can see there are clearly three columns, but the spacing between columns is different on each row. The `expand` tool allows us to create a consistent amount of space sizes between columns.

```
[linuxacademy@ip] expand -t 10 tabs.txt
column1   column2   column3
mycolumn1 mycolumn2 mycolumn3
c1        c2        c3
```

Now we'll see that the beginning of each element matches with the others, creating a layout that is much easier to read.

cut

The `cut` command allows us to extract characters at a specific point from each line of text. Let's take a look at our `columns.txt` file:

```
[linuxacademy@ip] cat columns.txt
```

We can get the character at position *5* of each line by using `cut` with the `-c` option:

```
[linuxacademy@ip] cut -c 5 columns.txt
t
k
e
y
```

Without context, this output isn't very useful. Instead, we can specify a range to give us something more readable:

```
[linuxacademy@ip] cut -c 1-5 columns.txt
first
clark
bruce
barry
```

Since each first name is exactly 5 characters, this allows us to filter out the first names exactly as written. However, let's see what happens when we try to get the last names:

```
[linuxacademy@ip] cut -c 7-11 columns.txt
last:
kent:
wayne
allen
```

The first two values are shorter than the others, so extra characters are included. Fortunately, the text is organized into colon (:) delimited columns, which we can use to get cleaner output:

```
[linuxacademy@ip] cut -d: -f 1 columns.txt
first
clark
bruce
barry
```

The `-d:` option specifies the delimiter is a colon. We can also use `-d" "` for a space, or use just `-d` by itself to specify tabs. The `-f` option allows us to specify a field. In this example, we get the first name field by using *1*, but we can get the last name field by using *2*, and so on.

We can also cut out multiple fields at a time:

```
[linuxacademy@ip] cut -d: -f 1,2,4 columns.txt
```

This returns the first name, last name, and email address.

Like other commands, we can redirect these results to a new text file for further processing or reference:

```
[linuxacademy@ip] cut -d: -f 1,2,4 columns.txt > newcolumns.txt
```

paste

The `paste` command allows us to concatenate two files.

We've got two sample files, `file1.txt` and `file2.txt`, which contain a few lines each:

```
[linuxacademy@ip] cat file1.txt
```

```
value1
value2
value3
value4


[linuxacademy@ip] cat file2.txt
value1
value2
value3
value45
```

If we `paste` them together, the values will be combined side by side:

```
[linuxacademy@ip] paste file1.txt file2.txt
value1  value1
value2  value2
value3  value3
value4  value45
```

This is useful in creating text columns that we can process further using other tools from this lab.

join

The `join` command is similar to `paste`, but it combines elements with a matching pair. If you've previously worked with relational databases, this may seem familiar. Let's start by joining our two files from the previous section:

```
[linuxacademy@ip] join file1.txt file2.txt
value1
value2
value3
```

We only see the first three values because those values have matching pairs. The result is that these pairs are joined together in one column. The fourth values, `value4` and `value45` are in the same row, but don't match, so they're not included.

uniq

The `uniq` command allows us to extract unique values from text. Let's look at the `unique.txt` file:

```
[linuxacademy@ip] cat unique.txt
This is a line
This is a line
This is a different line
This is a different line
This is the same line
This is the same line
```

```
Different Line 1
Different Line 2
```

Notice many of the lines are duplicates. We can use the `uniq` command on its own to filter out the duplicate lines:

```
[linuxacademy@ip] uniq unique.txt
This is a line
This is a different line
This is the same line
Different Line 1
Different Line 2
```

We can also print *only* the lines with duplicate with the `-d` option:

```
[linuxacademy@ip] uniq -d unique.txt
This is a line
This is a different line
This is the same line
```

This time, we only see the lines that have duplicate entries. If we want to print _*all instances*_ of duplicate lines, we can use `-D`:

```
[linuxacademy@ip] uniq -D unique.txt
This is a line
This is a line
This is a different line
This is a different line
This is the same line
This is the same line
```

head

The `head` command allows us to see the first lines of a given text. It's similar to `cat`, but only prints a specified number of lines from the beginning of the file. To see how this might be useful, let's look at the a large file.

```
[linuxacademy@ip] sudo cat /var/log/messages
```

This outputs an enormous amount of information, which may make it difficult to find what we're looking for. Instead of using a pager like `more` or a filter like `grep`, we can use `head` to see only the beginning of the file:

```
[linuxacademy@ip] sudo head /var/log/messages
```

By default, `head` shows the first *10* lines. We can change this number using the `-n` option. For example, to show the first *15* lines instead:

```
[linuxacademy@ip] sudo head -n 15 /var/log/messages
```

tail

The `tail` command is similar to `head`, but shows the end of a file instead of the beginning.

```
[linuxacademy@ip] sudo tail /var/log/messages
```

This will show the last *10* lines of the file. Like with `head`, we can use the `-n` option to change this behavior:

```
[linuxacademy@ip] sudo tail -n 15 /var/log/messages
```

We can also follow a log file to monitor its changes in real time. First, we'll need to become the *root* user:

```
[linuxacademy@ip] sudo su -
[linuxacademy@ip] tail -f /var/log/messages
```

Note the output and open a new console tab. Connect to the same lab server and install a package:

```
[linuxacademy@ip] sudo yum install vsftpd
```

Once it's been installed, check the first tab again. We should see a line letting us know that the `vsftpd` package was installed. Likewise, we can remove the package in our second tab:

```
[linuxacademy@ip] sudo yum remove vsftpd
```

The output from `tail` in the first tab will be updated accordingly.

Review

Now that we've reviewed some of the tools in the `textutils` package, you're well on your way to becoming the best system administrator you can be.

Congratulations, you've completed the lab on processing text streams using filters!