Linux

Introduction to Linux

Module 10: Some Advanced Linux commands

Introduction to Linux

In this module, we will go through some more advanced Linux commands that will allow us to manipulate file output and see differences between files.

These commands are extremely useful during releases (to see changes to the environment) as well as manipulating outputs of queries so that they can be presented to users.

Substituting values

It is fairly common that you will need to substitute values at some point in an output of a file – usually to turn something into a csv file so that it then can be displayed in Excel. For the examples in this module, we will create a file called fix.log that needs to have the following in it:

```
8=FIX4.4; 35=D; 34=55; 49=MTHREE; 56=MS; 52=2020-04-24-20:24:44; 55=AMZN; 40=2; 38=20925; 21=2; 11=algo20200424202444; 60=2020-04-24-20:24:44; 54=1; 44=133.47; 10=0252;

8=FIX4.4; 35=8; 34=59; 49=MS; 56=MTHREE; 52=2020-04-24-20:24:44; 55=AMZN; 40=2; 11=algo20200424202444; 31=133.47; 150=0; 39=0; 54=1; 44=133.47; 32=0; 17=exec20200424202444; 38=20925;
60=2020-04-24-20:24:44; 6=0; 14=0; 37=algo20200424202444; 10=200;

8=FIX4.4; 35=F; 34=55; 49=MTHREE; 56=MS; 52=2020-04-24-20:24:44; 41=algo20200424202444; 55=AMZN; 38=20925; 11=C_algo20200424202444; 60=2020-04-24-20:24:44; 10=060;

8=FIX4.4; 35=8; 34=59; 49=MS; 56=MTHREE; 52=2020-04-24-20:24:44; 55=AMZN; 11=C_algo20200424202444; 31=0; 150=6; 39=6; 54=1; 44=133.47; 17=exec20200424202444; 32=0;
41=algo20200424202444; 38=20925; 60=2020-04-24-20:24:44; 6=0.0; 14=0; 37=algo20200424202444; 10=252;

8=FIX4.4; 35=8; 34=60; 49=MS; 56=MTHREE; 52=2020-04-24-20:24:44; 151=15694; 55=AMZN; 11=C_algo20200424202444; 31=0; 150=4; 39=4; 54=1; 17=exec20200424202444; 32=0;
41=algo20200424202444; 38=20925; 60=2020-04-24-20:24:44; 6=0.0; 14=0; 37=algo20200424202444; 10=252;

8=FIX4.4; 35=D; 34=57; 49=MTHREE; 56=MS; 52=2020-04-24-20:24:46; 55=FB; 40=2; 38=3373; 21=2; 11=algo20200424202446; 60=2020-04-24-20:24:46; 54=1; 44=430.62; 10=0252;

8=FIX4.4; 35=8; 34=62; 49=MS; 56=MTHREE; 52=2020-04-24-20:24:46; 55=FB; 40=2; 11=algo20200424202446; 31=430.62; 150=0; 39=0; 54=1; 44=430.62; 32=0; 17=exec20200424202446; 38=3373;
60=2020-04-24-20:24:46; 6=0; 14=0; 37=algo20200424202446; 10=200;

8=FIX4.4; 35=8; 34=63; 49=MS; 56=MTHREE; 52=2020-04-24-20:24:46; 55=FB; 40=2; 11=algo20200424202446; 31=430.62; 150=1; 39=1; 54=1; 44=430.62; 32=1686; 17=exec20200424202446; 38=3373;
60=2020-04-24-20:24:46; 6=430.62; 14=1686; 37=algo20200424202446; 10=240;

8=FIX4.4; 35=8; 34=64; 49=MS; 56=MTHREE; 52=2020-04-24-20:24:48; 55=FB; 40=2; 11=algo20200424202446; 31=430.62; 150=2; 39=2; 54=1; 44=430.62; 32=1687; 17=exec20200424202448; 38=3373;
60=2020-04-24-20:24:48; 6=430.62; 14=3373; 37=algo20200424202446; 10=246;
```

sed command

sed is a powerful text stream editor. Its most common use is substitution (i.e., find and replace). You can do it outside of an editor.

For the below examples we will use the fix.log file created in the previous slide.

Command	Comments
sed ‘s/AMZN/AMAZON/’ fix.log	This will replace every occurrence of AMZN with AMAZON; it will output the result to your screen but will not change the original file
sed ‘s/AMZN/AMAZON/2’ fix.log	This will replace the 2nd occurrence of AMZN in every line – you can change the number 2 to whatever number you want to replace
sed ‘s/AMZN/AMAZON/g’ fix.log	Replacing every instance on every line requires the g (global option) to be added at the end of the command
sed ‘s/New\ York/NY/g’ file.txt	Here we show that if we want to replace something with a space, we have to escape the character using \
sed ‘s/;\ /,/g’ fix.log > fixOutput.csv	Here is a practical example of us turning this fix log into a csv file. We are replacing the “; “ with a comma globally and then redirecting the output to be stored into another file

Sed command continued

We can actually change the original file with sed as well. Do this with caution: it is better practice to output the results into a new file in case you make a mistake.

Command	Comments
sed -i s/AMZN/AMAZON/g fix.log	The –i option here will do a replace through the original file
sed ‘5d’ fix.log	This will delete the 5th line of the fix.log
sed ‘$d’ fix.log	This will delete the last line of the fix.log file
sed ‘3,6d’ fix.log	This will delete lines 3-6
sed ‘/pattern/d’ fix.log	This will delete pattern matching lines out of the file

awk

mthree

awk is a scripting language used for manipulating data and generating reports.   It is mostly used for pattern scanning and processing.

`awk '{print}' fix.log`	By default as no pattern is given, awk will print every line of fix.log
`awk '/FB/ {print}' fix.log`	This will print all lines that contain FB
`awk '{print $1, $4}' fix.log`	For each record (i.e. line), the awk command splits the record delimited by whitespace characters and stores it in the $n variable. If the line has 4 words, they will be stored in $1 $2 $3 $4 respectively, with the whole line being stored in $0
`ps -eo pid,stat|tail -1|awk '{print $1}'`	This will print the pid column only of your ps command – useful if you need that variable to pass into another command

Comparing the difference

mthree

At some point you may want to be able to compare two files. This can be pretty common when you want to check after a configuration change has been made to ensure only the changes you wanted to make have happened (assuming you would have backed up the original file in the first place).

Command	Comments
diff fix.log fixOutput.csv	This will show the output, but it can be challenging to read
diff –c fix.log fixOutput.csv	The –c provides some more context: lines starting with – means lines missing in the second file, lines starting with + means not in the first file, and lines starting with ! means changes between the two files
diff –b fix.log fixOutput.csv	This will ignore all changes in whitespace
diff –ui fix.log fixOutput.csv	This will produce a smaller output than the –c option; the –i will ignore case