Australian National University

Tools for viewing and editing binary data

Eric McCreath

cat and less

The 'cat' command will concatenate a list of files and print the result to standard output. This gives you a quick way of viewing the contents of a single ASCII file.

In the below example (and other examples within these slides) the file called 'greeting' contains the string 'Hello World!\n'

```
$ cat greeting
Hello World!
```

The 'less' command gives you more control over viewing ASCII files as you can scroll and do simple searches within in the text (there is a program called 'more' which also enables file perusal, the 'less' command's name comes from 'more' as less is the opposite of more).

If you are working from the command line then these tools are often quicker that bringing up an editor like 'gedit'.

File redirection and pipes

When programs are executed in Unix operating systems the program has a *'standard input'* , *'standard ouput'* and *'standard error'* data streams.

This provides a great deal of fexibility as programs are not tied to particular files.

When you run a program from the command shell, like bash, the *standard out* and *standard error* streams are printed to the shell. The *standard in* stream comes from what you type (ctr-d tells bash there is no more key presses to read).

In the below example I run 'sort' and type 9, 3 and **ctl-d.**

```
$ sort
9
3
3
9
```

File redirection and pipes

The '<' symbol can be used to redirect input from a file. The '>' symbol can be used to redirect data to a file. Or '>>' to append to a file. '|' to redirect the data from the output of one program to the input of another (programs are normally executed concurrently).

In the below example I 'cat' the greeting twice and redirect the output to a new file called 'other'.

```
$ cat greeting greeting > other
$ cat other
Hello World!
Hello World!
```

echo and touch

'echo' prints to standard out the parameter the echo command is given.

```
$ echo Hello World!
Hello World!
```

Combined with redirection this give a quick way of creating a file with some content.

```
$ echo Hello World! > greeting
```

echo by default adds a new line character. If you don't want this use the -n option.

'touch' is use to change the timestamp on a file. However, it is also a quick way of creating an empty file.

```
$ touch newemptyfile
$ ls -l newemptyfile
-rw-rw-r-- 1 ericm ericm 0 Dec 13 16:05 newemptyfile
```

stat

'stat' will show information about a file.

```
$ stat PCP2012.png
  File: `PCP2012.png'
  Size: 36802           Blocks: 72         IO Block: 4096   regular file
Device: 801h/2049d      Inode: 15479095    Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/   ericm)   Gid: ( 1000/   ericm)
Access: 2012-12-13 10:34:21.912935929 +1100
Modify: 2012-12-07 14:16:47.870797693 +1100
Change: 2012-12-07 14:16:47.870797693 +1100
 Birth: -
```

The 'file' command will attempt to determine the type of a file.

```
$ file PCP2012.png
PCP2012.png: PNG image data, 180 x 234, 8-bit/color RGB, non-interlaced
```

od

'od' is like cat but from viewing files that have binary content. 'od' will show the contents of a file in octal, hex, decimal, character and other formats.

```
$ od PCP2012.png | head -n 1
0000000 050211 043516 005015 005032 000000 006400 044111 051104
$ od -c PCP2012.png | head -n 1
0000000 211   P   N   G  \r  \n 032  \n  \0  \0  \0  \r   I   H   D   R
$ od -x PCP2012.png | head -n 1
0000000 5089 474e 0a0d 0a1a 0000 0d00 4849 5244
```

I use 'head -n 1' in the above example to just show the first line of the content of the file.

hexedit

'hexedit' is a simple program for editing the binary data within a file.

'hexedit' can also be used for viewing binary data as it provide the hex with the ascii character in a column on the side.

In the below example I ran hexedit on the PCP2012.png file.

```
00000000   89 50 4E 47  0D 0A 1A 0A  00 00 00 0D  49 48 44 52  .PNG........IHDR
00000010   00 00 00 B4  00 00 00 EA  08 02 00 00  00 84 2B F8  ..............+.
00000020   E2 00 00 00  03 73 42 49  54 08 08 08  DB E1 4F E0  .....sBIT.....O.
00000030   00 00 00 19  74 45 58 74  53 6F 66 74  77 61 72 65  ....tEXtSoftware
00000040   00 67 6E 6F  6D 65 2D 73  63 72 65 65  6E 73 68 6F  .gnome-screensho
00000050   74 EF 03 BF  3E 00 00 20  00 49 44 41  54 78 9C EC  t...>.. .IDATx..
00000060   9D 77 94 24  57 7D EF 6F  57 AE EA AE  D4 39 4D 9E  .w.$W}.oW....9M.
```

dd

'dd' provides a simple program for copying one file (or parts of one file) to another file.

'dd' give you a lot more control over what and how a file is copied compared to the 'cp' command.

In the below example the first 5 bytes of the 'greeting' file is copied to the 'shortgreeting' file.

```
$ echo Hello World! > greeting
$ dd if=greeting bs=1 count=5 of=shortgreeting
5+0 records in
5+0 records out
5 bytes (5 B) copied, 4.1967e-05 s, 119 kB/s
$ cat shortgreeting
Hello
```

Handy 'files' in Linux

/dev/null - this is an empty 'file' that lets you write data to it yet that data is never stored (like a black hole). Say you have a program that requires to output to a file but you don't want/need this output stored. Then direct the program to '/dev/null'.

/dev/zero - this 'file' is just all zeros. Say you wish to create a big empty file, just use dd to copy contents from '/dev/zero'.

/dev/random - this is a file full of random numbers (they are generated on the fly).

/dev/mem - this is a file which contains an image of the main memory of your system.

/dev/hda - (or something similar) the 'file' that contains the raw blocks of the harddisk.