Portfolio 4

Sam Bowyer

2023-03-20

Linux Command Line

Introduction

It is very useful to be familiar with the Linux command line as it can be used to access remote systems and perform tasks that are not possible with a GUI, and can be used to automate tasks and speed up your workflow.

There also exist many command line tools such as `git` and `ssh` that are more commonly used with the command line than with a GUI (although there are GUIs for these tools as well).

Basic Commands

Moving Around the File System

The command line is a text interface to the operating system and as such any command you type is ran from within some point of the computer's file system.

For instance, the command `pwd` (print working directory) will print the current directory you are in:

```
pwd
```

```
## /home/dg22309/Documents/Compass/SC2/lec4
```

The command `ls` (list) will list the files in the current directory:

```
ls
```

```
## command-line-files
## example_directory_copy
## example_file
## intro-to-command-line
## numLines
## portfolio4.html
## portfolio4.md
## portfolio4.pdf
## portfolio4.rmd
```

You can add the `-l` flag to `ls` to get more information about the files:

```
ls -l
```

```
## total 1036
## drwxr-xr-x 6 dg22309 dg22309   4096 Oct 11 11:08 command-line-files
## drwxrwxr-x 3 dg22309 dg22309   4096 Mar 20 16:50 example_directory_copy
## -rw-rw-r-- 1 dg22309 dg22309     32 Mar 20 16:50 example_file
## drwxrwxr-x 2 dg22309 dg22309   4096 Feb 16 14:36 intro-to-command-line
## -rw-rw-r-- 1 dg22309 dg22309      2 Mar 20 16:50 numLines
```

```
## -rw-rw-r-- 1 dg22309 dg22309 815208 Mar 20 16:29 portfolio4.html
## -rw-rw-r-- 1 dg22309 dg22309   5700 Mar 20 16:49 portfolio4.md
## -rw-rw-r-- 1 dg22309 dg22309 202037 Mar 20 16:50 portfolio4.pdf
## -rw-rw-r-- 1 dg22309 dg22309   5700 Mar 20 16:51 portfolio4.rmd
```

The flag `-a` will list all files including hidden files and can be combined with `-l` as follows:

```
ls -la
```

```
## total 1048
## drwxrwxr-x  6 dg22309 dg22309   4096 Mar 20 16:50 .
## drwxrwxr-x 10 dg22309 dg22309   4096 Mar 16 14:09 ..
## drwxr-xr-x  6 dg22309 dg22309   4096 Oct 11 11:08 command-line-files
## drwxrwxr-x  3 dg22309 dg22309   4096 Mar 20 16:50 example_directory_copy
## -rw-rw-r--  1 dg22309 dg22309     32 Mar 20 16:50 example_file
## -rw-rw-r--  1 dg22309 dg22309      0 Mar 20 16:15 .hidden_file
## drwxrwxr-x  2 dg22309 dg22309   4096 Mar 20 16:15 .hidden_folder
## drwxrwxr-x  2 dg22309 dg22309   4096 Feb 16 14:36 intro-to-command-line
## -rw-rw-r--  1 dg22309 dg22309      2 Mar 20 16:50 numLines
## -rw-rw-r--  1 dg22309 dg22309 815208 Mar 20 16:29 portfolio4.html
## -rw-rw-r--  1 dg22309 dg22309   5700 Mar 20 16:49 portfolio4.md
## -rw-rw-r--  1 dg22309 dg22309 202037 Mar 20 16:50 portfolio4.pdf
## -rw-rw-r--  1 dg22309 dg22309   5700 Mar 20 16:51 portfolio4.rmd
```

The general format for commands is:

```
command [options] [arguments]
```

where `command` is the name of the command, `options` are flags that change the behaviour of the command, and `arguments` are the arguments to the command. You can get details on the available options for a command by using the `man` command, e.g. `man ls` will give you the manual page for the `ls` command:

```
man ls
```

```
## LS(1)                            User Commands                           LS(1)
##
## NAME
##        ls - list directory contents
##
## SYNOPSIS
##        ls [OPTION]... [FILE]...
##
## DESCRIPTION
##        List  information  about  the FILEs (the current directory by default).
##        Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
##        fied.
##
##        Mandatory  arguments  to  long  options are mandatory for short options
##        too.
##
##        -a, --all
##               do not ignore entries starting with .
##
##        -A, --almost-all
##               do not list implied . and ..
##
##        --author
##               with -l, print the author of each file
```

```
##
##        -b, --escape
##               print C-style escapes for nongraphic characters
##
##        --block-size=SIZE
##               with  -l,  scale  sizes  by  SIZE  when  printing  them;   e.g.,
##               '--block-size=M'; see SIZE format below
##
##        -B, --ignore-backups
##               do not list implied entries ending with ~
##
##        -c     with -lt: sort by, and show, ctime (time of last modification of
##               file status information); with -l: show ctime and sort by  name;
##               otherwise: sort by ctime, newest first
##
##        -C     list entries by columns
##
##        --color[=WHEN]
##               colorize  the output; WHEN can be 'always' (default if omitted),
##               'auto', or 'never'; more info below
##
##        -d, --directory
##               list directories themselves, not their contents
##
##        -D, --dired
##               generate output designed for Emacs' dired mode
##
##        -f     do not sort, enable -aU, disable -ls --color
##
##        -F, --classify
##               append indicator (one of */=>@|) to entries
##
##        --file-type
##               likewise, except do not append '*'
##
##        --format=WORD
##               across -x, commas -m, horizontal -x, long -l, single-column  -1,
##               verbose -l, vertical -C
##
##        --full-time
##               like -l --time-style=full-iso
##
##        -g     like -l, but do not list owner
##
##        --group-directories-first
##               group directories before files;
##
##               can   be  augmented  with  a  --sort  option,  but  any  use  of
##               --sort=none (-U) disables grouping
##
##        -G, --no-group
##               in a long listing, don't print group names
##
##        -h, --human-readable
```

```
##               with -l and -s, print sizes like 1K 234M 2G etc.
##
##        --si   likewise, but use powers of 1000 not 1024
##
##        -H, --dereference-command-line
##               follow symbolic links listed on the command line
##
##        --dereference-command-line-symlink-to-dir
##               follow each command line symbolic link
##
##               that points to a directory
##
##        --hide=PATTERN
##               do not list implied entries matching shell  PATTERN  (overridden
##               by -a or -A)
##
##        --hyperlink[=WHEN]
##               hyperlink file names; WHEN can be 'always' (default if omitted),
##               'auto', or 'never'
##
##        --indicator-style=WORD
##               append indicator with style WORD to entry names: none (default),
##               slash (-p), file-type (--file-type), classify (-F)
##
##        -i, --inode
##               print the index number of each file
##
##        -I, --ignore=PATTERN
##               do not list implied entries matching shell PATTERN
##
##        -k, --kibibytes
##               default  to  1024-byte  blocks for disk usage; used only with -s
##               and per directory totals
##
##        -l     use a long listing format
##
##        -L, --dereference
##               when showing file information for a symbolic link, show informa-
##               tion  for  the file the link references rather than for the link
##               itself
##
##        -m     fill width with a comma separated list of entries
##
##        -n, --numeric-uid-gid
##               like -l, but list numeric user and group IDs
##
##        -N, --literal
##               print entry names without quoting
##
##        -o     like -l, but do not list group information
##
##        -p, --indicator-style=slash
##               append / indicator to directories
##
```

```
##        -q, --hide-control-chars
##               print ? instead of nongraphic characters
##
##        --show-control-chars
##               show nongraphic characters as-is (the default, unless program is
##               'ls' and output is a terminal)
##
##        -Q, --quote-name
##               enclose entry names in double quotes
##
##        --quoting-style=WORD
##               use  quoting style WORD for entry names: literal, locale, shell,
##               shell-always,  shell-escape,  shell-escape-always,   c,   escape
##               (overrides QUOTING_STYLE environment variable)
##
##        -r, --reverse
##               reverse order while sorting
##
##        -R, --recursive
##               list subdirectories recursively
##
##        -s, --size
##               print the allocated size of each file, in blocks
##
##        -S     sort by file size, largest first
##
##        --sort=WORD
##               sort  by  WORD instead of name: none (-U), size (-S), time (-t),
##               version (-v), extension (-X)
##
##        --time=WORD
##               with -l, show time as WORD instead of default modification time:
##               atime  or  access  or  use  (-u); ctime or status (-c); also use
##               specified time as sort key if --sort=time (newest first)
##
##        --time-style=TIME_STYLE
##               time/date format with -l; see TIME_STYLE below
##
##        -t     sort by modification time, newest first
##
##        -T, --tabsize=COLS
##               assume tab stops at each COLS instead of 8
##
##        -u     with -lt: sort by, and show, access time; with -l:  show  access
##               time  and  sort  by name; otherwise: sort by access time, newest
##               first
##
##        -U     do not sort; list entries in directory order
##
##        -v     natural sort of (version) numbers within text
##
##        -w, --width=COLS
##               set output width to COLS.  0 means no limit
##
```

```
##        -x     list entries by lines instead of by columns
##
##        -X     sort alphabetically by entry extension
##
##        -Z, --context
##               print any security context of each file
##
##        -1     list one file per line.  Avoid '\n' with -q or -b
##
##        --help display this help and exit
##
##        --version
##               output version information and exit
##
##        The SIZE argument is an integer and  optional  unit  (example:  10K  is
##        10*1024).   Units  are  K,M,G,T,P,E,Z,Y  (powers  of 1024) or KB,MB,...
##        (powers of 1000).
##
##        The TIME_STYLE argument can be  full-iso,  long-iso,  iso,  locale,  or
##        +FORMAT.   FORMAT  is  interpreted  like in date(1).  If FORMAT is FOR-
##        MAT1<newline>FORMAT2, then FORMAT1 applies to non-recent files and FOR-
##        MAT2  to  recent files.  TIME_STYLE prefixed with 'posix-' takes effect
##        only outside the POSIX locale.  Also the TIME_STYLE  environment  vari-
##        able sets the default style to use.
##
##        Using  color  to distinguish file types is disabled both by default and
##        with --color=never.  With --color=auto, ls emits color codes only  when
##        standard  output is connected to a terminal.  The LS_COLORS environment
##        variable can change the settings.  Use the dircolors command to set it.
##
##    Exit status:
##        0      if OK,
##
##        1      if minor problems (e.g., cannot access subdirectory),
##
##        2      if serious trouble (e.g., cannot access command-line argument).
##
## AUTHOR
##        Written by Richard M. Stallman and David MacKenzie.
##
## REPORTING BUGS
##        GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
##        Report ls translation bugs to <https://translationproject.org/team/>
##
## COPYRIGHT
##        Copyright © 2018 Free Software Foundation, Inc.   License  GPLv3+:  GNU
##        GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
##        This  is  free  software:  you  are free to change and redistribute it.
##        There is NO WARRANTY, to the extent permitted by law.
##
## SEE ALSO
##        Full documentation at: <https://www.gnu.org/software/coreutils/ls>
##        or available locally via: info '(coreutils) ls invocation'
##
```

```
## GNU coreutils 8.30                September 2019                        LS(1)
```

One of the most useful commands is `cd` (change directory) which allows you to change the current directory:

```
cd command-line-files
```

There are some reserved shortcuts for directories:

- `cd ..` - change directory to parent directory
- `cd ~` - change directory to home directory
- `cd -` - change directory to previous directory

```
cd ~
```

Also useful to know is that the `.` character is a shortcut for the current directory, i.e. `./example_directory` is equivalent to `example_directory`.

File Manipulation

Some other common commands are:

- `mkdir` - make directory
- `rmdir` - remove directory
- `rm` - remove file
- `rm -r` - remove directory
- `rm -rf` - remove directory and all files in it
- `cp` - copy file
- `cp -r` - copy directory
- `cp -rf` - copy directory and all files in it
- `mv` - move file
- `mv -r` - move directory
- `mv -rf` - move directory and all files in it

For instance the following set of commands will create a directory `example_directory` and then copy it to a new directory `example_directory_copy` before removing the original directory:

```
mkdir example_directory
cp -r example_directory example_directory_copy
rm -rf example_directory
```

Reading and Writing Files

We can create a file using the `touch` command:

```
touch example_file
```

And then write to it using the `echo` command:

```
echo "Hello World" > example_file
```

We can then read the contents of the file using the `cat` command:

```
cat example_file
```

```
## Hello World
```

We can also append to the file using the `>>` operator:

```
echo "Hello, again, World" >> example_file
```

For longer files we might want to use the `head` and `tail` commands to print the first and last 10 lines of the file respectively. In this example we'll look into a file containing the text of `King Lear` by William Shakespeare:

```
head command-line-files/some_plays/king-lear.txt
tail command-line-files/some_plays/king-lear.txt
```

```
## King Lear
## by William Shakespeare
## Edited by Barbara A. Mowat and Paul Werstine
##   with Michael Poston and Rebecca Niles
## Folger Shakespeare Library
## https://shakespeare.folger.edu/shakespeares-works/king-lear/
## Created on Jul 31, 2015, from FDT version 0.9.2
##
## Characters in the Play
## ======================
## KENT
## I have a journey, sir, shortly to go;
## My master calls me. I must not say no.
##
## EDGAR
## The weight of this sad time we must obey,
## Speak what we feel, not what we ought to say.
## The oldest hath borne most; we that are young
## Shall never see so much nor live so long.
## [They exit with a dead march.]
```

An interactive file viewer (therefore one we can't effectively print out in this document) is also available using the `less` command, e.g. with:

```
less command-line-files/some_plays/king-lear.txt
```

Searching Files

One of the most useful commands is `grep` which allows you to search for a pattern in a file:

```
grep "land" command-line-files/some_plays/king-lear.txt
```

```
## Legitimate Edgar, I must have your land.
## Let me, if not by birth, have lands by wit.
## land comes to. He will not believe a Fool.
##     To give away thy land,
## Not in this land shall he remain uncaught,
## May have due note of him. And of my land,
##  When slanders do not live in tongues,
##  Child Rowland to the dark tower came.
## paper.] The army of France is landed.--Seek out
## I told him of the army that was landed;
## France spreads his banners in our noiseless land,
## It touches us as France invades our land,
```

Note that lines containing the words `slander` and `Rowland` are also returned as they contain the pattern `land`. To get only lines that contain the exact pattern we can use the `-w` flag:

```
grep -w "land" command-line-files/some_plays/king-lear.txt
```

```
## Legitimate Edgar, I must have your land.
## land comes to. He will not believe a Fool.
##     To give away thy land,
## Not in this land shall he remain uncaught,
```

```
## May have due note of him. And of my land,
## France spreads his banners in our noiseless land,
## It touches us as France invades our land,
```

We can also search in multiple files at once using the `-r` flag:

```
grep -rw "Quoth" command-line-files/some_plays
```

```
## command-line-files/some_plays/troilus-and-cressida.txt:PANDARUS  Quoth she, 'Here's but two and fifty
## command-line-files/some_plays/hamlet.txt:    Quoth she, before you tumbled me,
## command-line-files/some_plays/a-comedy-of-errors.txt:LUCIANA Quoth who?
## command-line-files/some_plays/a-comedy-of-errors.txt:DROMIO OF EPHESUS   Quoth my master:
```

Combining Commands

One of the most powerful features of the command line is the ability to combine commands together to perform more complex tasks. We can do this using the following operators:

- `;` - separate commands
- `&&` - execute second command if first command succeeds
- `||` - execute second command if first command fails
- `|` - pipe output of first command to input of second command
- `>` - redirect output of command to file
- `>>` - append output of command to file
- `<` - redirect input of command from file
- `<<` - redirect input of command from here document
- `<<<` - redirect input of command from here string

For example, we can run a `grep` command and then pipe the output to a `wc` command to count the number of lines and pipe that to a file called `numLines`:

```
grep -rw "yea" command-line-files/some_plays | wc -l > numLines
cat numLines
```

```
## 36
```