GNU Coreutils

Peter Hill

What are coreutils?

- “Standard” set of (mostly) POSIX-compatible tools
  - Portable Operating System Interface
  - Family of standards for compatibility between UNIX-like OSes (Linux, OS X, BSD)
- Combination of tools for operating on/with files, shells and text
- Has extensions to the POSIX-standard
- 103 separate programs
  - Many are used all the time, some have fallen by the wayside

UNIX Philosophy

Philosophy

Peter H. Salus in A Quarter-Century of Unix (1994):

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

Pipes

Use pipes to connect the output from one program directly into the input of the next, and so build up complex pipelines out of the simple building blocks.

Summarise your top ten most used functions:

```
sed 's/|/\n/g' ~/.bash_history | awk '{CMD[$1]++;count++;} END \
{ for (a in CMD) print CMD[a] " " CMD[a]/count*100 "% " a;}' \
| grep -v "./" | column -c3 -s " " -t | sort -nr | nl | head -n10
```

Getting help

- `info coreutils`: coreutils manual
  - `?`: Key bindings
- `info <command>`: manual for `<command>`
- `man <command>`: manual for `<command>`
  - The `info` pages tend to be a better for learning about a command, whereas the `man` pages tend to be better for refreshing your memory

Common options

Flags

- `--help`: Print a usage message
  - Even terser than the manpage!
- `--version`: Print the version number
  - Useful for verifying you have the correct thing installed in scripts
- `--verbose`: Print exactly what operation is being performed (less common)
- `--`: Delimit the option list
  - Useful if you need to operate on files beginning with a dash
- `-`: Read from standard in/write to standard out

Exit status

- Majority of tools use an exit status of 0 for "success" and non-zero for failure

Output of entire files

`cat [OPTION] [FILE]...`

- Concatenate files together and print them to standard output
- Also useful to just dump contents of a file to screen

`tac [OPTION] [FILE]...`

- Concatenate files together, reversing them separately, and print them to standard output

Output parts of files

`head [OPTION]... [FILE]...`

- Print the first `n` (default: 10) lines of each FILE
- `--lines K`: Print first K lines
- `--lines -K`: Print all but the last K lines

`tail [OPTION]... [FILE]...`

- Print the last `n` (default: 10) lines of each FILE
- `--lines K`: Print last K lines
- `--lines +K`: Print all but the first K lines
- `--follow`: Keep trying to read characters from files
  - Useful for watching log file(s)

Summarising files

`wc [OPTION]... [FILE]...`

- Count the number of bytes, characters, words and lines in each FILE
- `--lines`: Print only the number of lines

`md5sum [OPTION]... [FILE]...`

- Computes the MD5 digest of each FILE
- Useful for verifying downloads haven't been corrupted

Operating on sorted files

`sort [OPTION]... [FILE]...`

- Sorts files alphabetically
- `--numeric-sort`: Sort numerically
- `--human-numeric-sort`: Sort numerically, but understand SI suffixes (M, G, etc.)
- `--key=POS1[,POS2]`: Sort between fields POS1 and POS2 (inclusive) on each line
  - A field is some text surrounded by whitespace
- `--debug`: Annotate the part of the line used to sort
- `--reverse`: Reverse the ordering

`uniq [OPTION]... [INPUT [OUTPUT]]`

- Discard all but the first of adjacent repeated lines
- `--count`: Print the number of times each line occurred along with the line

Operating on fields

- Coreutils has the following, but see `awk` for a more powerful tool

`cut [OPTION]... [FILE]...`

- Write to stdout selected parts of each line of each FILE

`paste [OPTION]... [FILE]...`

- Writes all the first lines of each FILE, followed by all the second lines, etc.

`join [OPTION]... FILE1 FILE2`

- Joins each pair of lines from FILE1 and FILE2 that have identical join fields

Operating on characters

`tr [OPTION]... SET1 [SET2]`

- Translate SET1 into SET2
- Input from stdin, output to stdout (i.e. you probably want this in a pipe)
- e.g. Convert output of `some-command` from lowercase to uppercase:
  - `some-command | tr a-z A-Z`
- `--delete`: delete SET1 from input
  - Useful to remove extra tabs, etc.

Directory listing

`ls [OPTION]... [FILE]...`

- List directory contents
- `--almost-all`: Show files beginning with `.`, except `.` and `..`
- `--human-readable`: Print human readable sizes (in powers of 1024)

- `--format=long`: Print as list, showing file information
- `--sort=size`: Sort by file size
- `--sort=time`: Sort by modification time
- `--reverse`: Reverse sort
- `--color`: Use colours to distinguish file types
- `--classify`: Append character to indicate file types

Common `ls` aliases

```
alias ls='ls -hF --color'   # add colors for filetype recognition
alias la='ls -Alh'          # show hidden files
alias lk='ls -lSrh'         # sort by size, biggest last
alias lt='ls -ltrh'         # sort by date, most recent last
alias lr='ls -lhR'          # recursive ls
```

`dircolors [OPTION]... [FILE]`

- Output a sequence of shell commands to set up the terminal for colour output from `ls`
- Run like: `eval "$(dircolors)"`

Basic operations

`cp`, `mv`, `rm`

- Copy, move/rename, delete files
- `cp/mv [OPTION]... SOURCE... DEST`: DEST may be a directory
- `rm [OPTION]... [FILE]...`
- Share most of the same options:
  - `--force`: Don't prompt before overwriting/deleting existing files
  - `--interactive`: Prompt before overwriting existing files
  - `--no-clobber`: Don't overwrite existing files (not `rm`)
  - `--recursive`: Copy/delete directories recursively (not `mv`)
- For complicated copies/backups, look at `rsync`
- Shell `{}` syntax is useful here:
  - `cp file.txt{,.bak}`: Same as `cp file.txt file.txt.bak`

Special file types (directories)

`mkdir`, `rmdir [OPTION]... NAME...`

- Make/remove directories

- `--parents`: Make/remove parent directories
  - `mkdir -p a/b/c` will make `./a`, `./a/b` and `./a/b/c` if they don't already exist
  - `rmdir -p a/b/c` will remove `./a`, `./a/b` and `./a/b/c` if they are all empty

Special file types (links)

`ln [OPTION]... [-T] TARGET LINKNAME`

- Make a link from TARGET to LINKNAME
- `--force`: Replace existing destination files without asking
- `--symbolic`: Make symbolic links – generally the kind you want
  - Hard links (the default) are another name for the same physical file on disk. They have to be on the same filesystem. A file is only deleted once all hard links are deleted
  - Symbolic links are just links to a filename. If you delete the original, the symbolic link is "dangling", not pointing to anything. Symbolic links can be across filesystems

`readlink [OPTION]... [FILE]...`

- Print the value of the given symbolic links

Changing file attributes

`chmod [OPTION]... {MODE | --reference=REF_FILE} FILE...`

- Change the access permissions of the named files
- `--recursive`: Recursively change permissions of directories and their contents
- Permissions on Linux:
  - Read/Write/eXecute for
  - User who owns the file/Group who owns the file/Other users
- MODE should look like USERS OPERATION PERMISSIONS:
  - `chmod u+x ./my_script`: Give User eXecute permission
  - `chmod go-w important_file`: Remove write permission for everyone but user
  - `chmod go= secret_file`: Remove all permissions for everyone else

`touch [OPTION]... [FILE]...`

- Change the access/modification times of FILEs
- Default is change the times to "now"
- `--date=TIME`: Use TIME instead, e.g.:
  - `--date="2017-01-02 13:00"`
  - `--date="yesterday"`

    - `--date="2004-02-27 14:19:13.489392193 +0530"`
- Date formats are *complicated* because human times are way, way, *way* more complicated than you think

Disk usage

`df [OPTION]... [FILE]...`

- Report the amount of disk space used and available on file systems
- `--human-readable`: Print with sensible suffixes (powers of 1024)
- `--portability`: Among other things, print each file system on a single line
- `--print-type`: Print each file system's type
    - Useful to see network disks, tmpfs, etc.

`du [OPTION]... [FILE]...`

- Report the amount of disk space used by the specified files and for each subdirectory
- `--total`: Print a grand total
- `--max-depth=DEPTH`: Show the total for each directory that is at most DEPTH levels down from the root of the hierarchy. The root is at level 0, so `du --max-depth=0` is equivalent to `du -s`
- `--human-readable`: Print with sensible suffixes (powers of 1024)
- `--summarise`: Only display the total

Printing text

`echo [OPTION]... [STRING]...`

- Write each STRING to stdout
- Useful for basic debugging of scripts

`printf FORMAT [ARGUMENT]...`

- Print formatted text
- FORMAT is mandatory, and is mostly the same as the C `printf()` function
- Useful for more complicated printing

Conditions

`false`, `true`

- `false`: Do nothing, unsuccessfully
- `true`: Do nothing, successfully
- Useful as placeholders in scripts

`test EXPRESSION` or `[ EXPRESSION ]`

- Return a status of 0 (true) or 1 (false) depending on EXPRESSION
- Lots of different tests:
  - File type (is it a normal file, a directory, a symlink?)
  - Access permission (can I read/write/execute this file?)
  - File characteristics (does it exist? Is it newer than another file?)
  - String tests (is the length of this string zero? Is it the same as another string?)
  - Numeric tests (is this number larger/smaller/equal to another?)

Redirection

`tee [OPTION]... [FILE]...`

- Copy stdin to stdout and also FILEs
- `--append`: Append to the files rather than overwriting them
- Very useful for creating logs for a command whilst still seeing the results on screen
  - `some_command | tee command.log`

File name manipulation

`basename OPTION... NAME...`, `dirname OPTION... NAME...`

- `basename`: Remove all the leading directory components
- `dirname`: Print all the leading directory components
- Useful for cleaning up paths got from other programs

```
$ dirname /usr/bin/sort
/usr/bin
$ basename /usr/bin/sort
sort
```

`realpath [OPTION]... FILE...`

- Expand all symbolic links and resolves references to "/./", "/../" and extra "/" characters

`mktemp [OPTION]... [TEMPLATE]`

- Safely create a temporary file or directory based on TEMPLATE, and print its name. TEMPLATE must include at least three consecutive "X"s in the last component (default "tmp.XXXXXXXXXX")
- `--directory`: Create a directory rather than a file
- `--tmpdir[=DIR]`: Treat TEMPLATE relative to the directory DIR (defaults to "/tmp")

Working context

`pwd`

- Print the current (working) directory

User information

`id [OPTION]... [USER]`

- Print information about USER (default: you)
- Useful to find out what groups you belong to, and the numeric user/group IDs

`groups [USER]`

- Print the names of the groups USER is in

`who`

- Print information about currently logged in users
- Useful to see if a system is busy

System context

`date [OPTION]... [+FORMAT]`

- Get the current time and date in FORMAT

- Useful in scripts to make uniquely named files
- `--date=DATESTR`: Get the time and date from DATESTR
  - Can be a string like "1 month ago" or "+1 year"
  - Time specifiers: %[HIklMNpPrRsSTXzZ]
  - Date specifiers: %[aAbBcCdDeFgGhjmuUVwWxyY]
- Can be used to convert between different periods of time
- Can be used to convert to "seconds since UNIX epoch", 1970-01-01 00:00:00 UTC, useful for sorting dated data

`uname [OPTION]...`

- Print information about the machine and OS
- `--all`: Print all the available information
- Very useful when you diagnosing computer problems
- Gives info on: hardware platform, machine name, processor type, OS, kernel
- When you need the exact name/version of the OS, a portable method that works almost everywhere is `cat /etc/*release`

`uptime`

- Print the current time, the system's uptime, the number of logged-in users and the current load average
- Useful to quickly gauge how busy a machine is

Modified command invocation

`nice [OPTION]... [COMMAND [ARG]...]`

- Run COMMAND with modified niceness
- The niceness affects how favourably the command is scheduled in the system
- Values range from -20 (highest priority) to 19 (lowest priority)
- With no options, runs COMMAND with nice of 10
- You generally need sudo to set a negative nice value

`nohup COMMAND [ARG]...`

- Run given COMMAND with hangup signals ignored, so that the command can continue running in the background after you log out
- You still need to background the process by ending the line with "&"
  - `nohup long_running_command > file.log &`

Delaying

`sleep NUMBER[smhd]...`

- Pause for an amount of time
- `[smhd]`: seconds, minutes, hours, days

Numeric operations

`seq [OPTION]... [[FIRST] [INCREMENT]] LAST`

- Print the numbers from FIRST to LAST by INCREMENT
- `--separator=STRING`: Set the separator (default: newline)
- `--equal-width`: Print all numbers with same width by padding with leading zeros
- Useful for generating lots of sequential names!

Findutils - utilities for finding

`find [-H] [-L] [-P] [-D DEBUGOPTIONS] [-OLEVEL] [STARTING-POINT...] [EXPRESSION]`

- Search the directory tree from STARTING-POINT for EXPRESSION
- Around 100 expressions… Most useful is the basic:
  - `find . -type f -name "*.[ch]xx"`
  - "Starting here, find normal files that end in either 'cxx' or 'hxx' "
- Multiple expressions can be chained
  - Expressions are evaluated left-to-right with implied "and"
- The expressions can be tests or actions
  - `-name`, `-ctime`, `-ls`, `-delete`, `-exec`

`xargs [OPTION...] [COMMAND [INITIAL-ARGUMENTS]]`

- Build and execute command lines from stdin
- Useful in conjunction with `find`
- Reads from stdin and passes as arguments to COMMAND
- `find . -type f -name "*.[ch]xx" | xargs grep "variable"`

`locate [OPTION...] PATTERN...`

- Find files by name

- Reads a local database of file names
- Very handy for tracking down libraries or headers
- `--basename`: Match the pattern only against the last component of the filename

Acknowledgements

- I relied heavily on the GNU Coreutils documentation to write this
- Copyright (C) 1994-2014 Free Software Foundation, Inc.