Unix Command Line Tips and Tricks

Part2: Basic Unix commands

Michael F. Herbst
Girton College

Department of Chemistry
University of Cambridge

27 February 2013

Table of Contents

Table of Contents

Introduction

- Fairly vast and exhaustive language for patterns to match
- Distinct from filename metacharacters, but there are similarities
- Up to a certain point understood by `vim`, `sed`, `awk` and `grep`
- Since grep has two engines to parse RegExps we distinguish between `grep` and `egrep`
- The latter, invoked by `grep -E` or `egrep`, takes longer, but has more functionality. ("Extended regular expression")

Standards

These are implemented in all `vim`, `sed`, `awk`, `grep` and `egrep`.

- `.` Matches a single character
- `*` The previous expression is present zero, one or more times.
- `^` Matches the beginning of a line
- `$` Matches the end of a line
- `\` Escape the next character
- `[ ]` Match one character from the group, a leading `^` inverses the selection

e.g. `[abc]` matches exactly a,b or c

e.g. `[a-c]` matches exactly a,b or c

e.g. `[^a-e]` matches all but a to e

`awk` and `egrep` only

All of these are implemented in `awk` and `egrep` only.

- `+` The previous expression is present zero, one or more times. (same as *)
- `?` The previous expression is present once or more times.

`(` *regExp1* `|` *regExp2* `|` `...)` Combine regular expressions to a group. If either of the *regExp*s matches it is counted as a match.

POSIX character classes are also implemented in `awk` and `egrep` (see next slide)

POSIX character classes

All of these are implemented in `awk` and `egrep` only. The important ones are:

POSIX	ASCII	description
`[:alnum:]`	`[A-Za-z0-9]`	Alphanumeric characters
`[:alpha:]`	`[A-Za-z]`	Alphabetic characters
`[:blank:]`	`[ \t]`	Space and tab
`[:digit:]`	`[0-9]`	Digits
`[:lower:]`	`[a-z]`	Lowercase letters
`[:space:]`	`[ \t\r\n\v\f]`	Whitespace characters
`[:upper:]`	`[A-Z]`	Uppercase letters

Note: The brackets are part of the name, ie we need `[[:lower:]]` in order to match a lower case character.

`vim` only and `sed` only

`vim` only

`\<` `\>`	matches beginning or end of a word, respectively

`sed` only

`\( \)`	match this expression *and* save it for later reuse
`\n`	reuse the *n*th saved expression

Examples

`^house`	matches house at the beginning of a line
`fr(ien|ou|aul)d`	matches friend or froud or frauld
`Ha[unl]s`	matches Hans or Haus or Hals
`[^hbc]at`	matches any 3 letter word ending in at *but* hat, cat, bat
`"*word"*`	matches word in arbitrarily many (could be no) quotes
`"?word"?`	matches word in at least one quote either side
`^...$`	matches a line of 3 characters
`.*`	matches everything
`[[:space:]]*`	matches arbitrary number of space characters
`[^[:alnum:]_]`	matches arbitrary character but alphanumeric or underscore

Table of Contents

The `cd` command

`cd` *DIR*

- Shell overloaded
- Changes working directory to subdirectory *DIR*
- If *DIR* cannot be found in `$PWD` , searches all directories in `$CDPATH` as well

The `ls` command

`ls [`*Options*`] [`*Files*`]`

List the current directory or *Files* if given. Possible options:

`--color=auto` Enable colours (really helpful)

`-a` all – do not ignore . entries

`-A` almost all – all, but .. and .

`-h` print file sizes more human readable using K,M,G, ... (Factors of 1024)

`-l` use long listing format

The `ls` command

continued

`ls [`*Options*`] [`*Files*`]`

- `-r` reverse the listing order
- `-R` recursively parse directories
- `-s` print sizes of files
- `-S` sort by size
- `-t` sort by modification time, newest first
- `-v` natural ordering of version numbers
- `-1` list one item per line only (automatically done when output piped)

Searching files

find, whereis, locate, which(next to type as shell builtin)

Copying files

cp, mv

Creating/removing files and folders

touch, rm mkdir, rmdir

Changing UNIX rights

chmod, chown

Reading files

less more view

Downloading files

wget

Synchronisation and compression

tar, rsync, unison

Table of Contents

`cat` and friends

cat tac tee head tail

Simple tasks

cut, paste, join, sort, tr, uniq

grep

`grep [`*Options*`] `*Pattern*` [ `*File*` ...]`

Search for *Pattern* in current stream or in the *Files*. Returns 0 if a match is found, 1 otherwise.

- `-E` use extended RegExps (like `egrep`)
- `-e` *Pattern* Provide another pattern
- `-i` ignore case
- `-v` invert match – select non-matching lines
- `-w` word regexp – select lines where a word matches the pattern exactly
- `-x` line regexp – select lines where the whole line is matched exactly

grep
continued

`grep [`*Options*`] `*Pattern*` [ `*File*` ...]`

- `-c` count – only print the number of selected lines, but not the lines itself
- `-q` quiet – print nothing
- `-n` line number – prefix each line with its line number
- `-H` file name – print the file name for each match
- `-A` *Num* print *Num* lines before the actual match
- `-B` *Num* print *Num* lines after the actual match

sed

sed - Extended regular expressions! sed: `&` reuse the characters that matched previous substitution pattern

awk

awk

Table of Contents

Listing and killing processes

ps, kill, nice, renice, pwdx

Delayed or detached execution

at, nohup

Monitoring memory and load

free, top

Monitoring other users

uptime, w, who

Table of Contents

Calculations within the terminal

bc, expr

Reading help

man info

Output and Logging

echo printf logger

`date` and `time`

date, time

`xargs` building and executing a command

xargs

Other tools

mktemp, wc

Table of Contents

Examples

Examples

References

Most of the info in here came from these resources:

- `http://mywiki.wooledge.org`
- various program manuals
- `http://www.schatenseite.de/uploads/media/shell.pdf`