Module 1

Advanced Linux commands

Standard output

What do we mean by *standard output*?

- All the commands outputting text on your terminal do it by writing to the standard output
- Standard output can be written to a file (i.e., redirected) using the > operator
- Standard output can be appended to an existing file using the >> operator

Examples

- `ls saddam/* > gwb/weapons_mass_destruction.txt`
- `cat obiwan_kenobi.txt > starwars_biographies.txt`
  `cat han_solo.txt >> starwars_biographies.txt`
- `echo "README: No such file or directory" > README`

Standard input

What do we mean by *standard input*?

- A number of commands (when not provided with arguments) take the input from standard input
- `sort`
  (user enters from keyboard)
  `windows`
  `linux`
  `<Ctrl+d>`
  (printed on screen)
  `linux`
  `windows`

  `sort` takes its inputs from the standard input (i.e., what you type in your terminal) ended by entering a `<Ctrl+d>` sequence
- `sort < participants.txt`
  The standard input of `sort` is taken from the given file

Pipes

- Linux pipes (|) are useful for redirecting the standard output of a command to the standard input of another one

Examples

- `cat *.log | grep -i error | sort`
- `grep -ri error . | grep -v "ignored" | sort -u \`
  `> serious_error.txt`
- `cat /home/*/homework.txt | grep mark | more`

- This is the one of the most powerful features in Linux shells

tee command

`tee [-a] file`

Used to send standard output both to the screen and to a file

Examples

- `make | tee build.log`
  Runs the `make` command and stores its output to `build.log` file
- `make install | tee -a build.log`
  Runs the `make install` command and appends its output to `build.log` file

Standard error

- Error messages are usually output (assuming that the program is well written) to *standard error* instead of standard input
- Standard error is redirected using `2>` or `2>>`
- Both standard output and standard error are redirected using `&>`

Examples

- `cat f1 f2 nofile > newfile 2> errfile`
- `cat f1 f2 nofile &> wholefile`

yes command

Useful to send some default string to standard input

- `yes "some string" | <command>`
  Keeps sending the standard input of a `<command>` with `some string` (`y` by default)

Examples

```
yes | rm -r dir/
bank > yes no | credit_applicant
yes "" | make oldconfig
```

The later is useful for configuring Linux kernel, as it is equivalent to hitting `<Enter>` to accept all default settings

Special devices

- `/dev/null`
  Discards all the data written to it. Useful to trash out unwanted output (usually some unnecessary log information).
- `/dev/zero`
  Reading this file return zero. Useful to fill a file with all zeros.
- `/dev/random`
  Returns random bytes when read. Use your imagination where this could come handy (cryptography, maybe).
- `/dev/urandom`
  Returns pseudo random number when read.
- `/dev/full`
  Emulates a full device. Useful for testing if an application properly handles such a condition.

Everything in Linux is a file...

Everything in Linux that is not a file is a process

- Process – an instance of a running program
- There can be several instances of the same program running at the same time
- Every process contain some associated data: open files, allocated memory, stack, process id, parent, priority, state, etc.
- Thread – considered as *lightweight process* (shares the context with other threads within the same process)

Background tasks

- You can run a task in background by adding `&` at the end of the line
  `some_background_job --search --exhaustive &`
- `jobs`
  Displays the list of currently running jobs
- `fg [%<n>]`
  Puts last (or n*th*) job in foreground mode
- Moving the current task in background mode
  `<Ctrl+z>`
  `bg`
- `kill %<n>`
  Kills the n*th* job

Listing all processes

Effectively, the commands obtain information from `/proc/` virtual filesystem

- `ps -ux`
  Lists all the processes belonging to the current user
- `ps -aux`
  Lists all the processes running on the system
- `top`
  Shows the most important processes interactively

Module 1

Advanced Linux commands

Practical Demonstration