Shell Handout

Zilogic Systems

1. Processes

- Process is a program in execution.
- A process is always created by another process.
- When process X creates a new process Y, X is called the parent of Y.
- Each process is given a unique number called the process ID. The processes can be referred to using the process ID.
- `init` is the first program to be executed, and hence the first process. It has process ID of 1.
- The first user process, is usually the shell.

2. Listing Processes

- The `ps` command is used to list processes.

```
❶   ❷             ❸   ❹
  PID TTY          TIME CMD
 6423 tty1     00:00:00 bash
 6457 tty1     00:00:00 ps
```

❶ The process ID of the task.
❷ The terminal associated with the task.
❸ The total CPU time used by the process
❹ Command used to start the process.

- Without arguments, the `ps` command lists processes started by the user in the current terminal.
- The command to list all processes in the system the `-A` option can be used as shown below.

```
$ ps -A
```

- There options for filtering processes for filtering processes based on user, command, and process ID.
  - `ps -U <username-list>`
  - `ps -C <command-list>`
  - `ps -p <pid-list>`
- The parent child relation ship of processes can be displayed as tree, using the `pstree` command.

Try Out

- List all processes belonging to `root`.
- List all processes corresponding to the program `getty`.
- List the process tree of the system.

3. Resource Utilisation

- The `top` command can be used to watch the resource utilisation of processes.

- The `top` command is a full screen application. `top` displays, one screen full of processes and their information. Some of important fields are listed below
  - `PR` - priority of the process
  - `VIRT` - the amount of virtual memory used by the task
  - `RES` - the amount of physical memory used by the task
  - `SHR` - the amount of shared memory used by the task
  - `S` - the state of the process
  - `%CPU` - the task's share of the elapsed CPU time since the last screen update
  - `%MEM` - the task's currently used share of available physical memory.
- Key strokes within `top`.
  - `M` key stroke - Sort on memory usage
  - `P` key stroke - Sort on CPU usage
  - `q` key stroke - quit

Try Out

- Find out the process that occupies the greatest amount of memory in the system.
- Find out the process that uses the CPU the most.

4. Job Control

- Shell does not provide prompt till the last entered command is completed. Further commands cannot be entered till the previous command completes execution.
- If this behaviour is not desired, the program can be run in the background: `cmd &`
- Each command entered in the shell is given a job no. When a command is executed in the background the shell prints the shell prints the job no. and the process id.

```
$ sleep 10 &
[1] 11274
```

- Sometime it is required to background, a command that has already been started. This can be achieved by first stopping the job using `Ctrl-Z`, and then running `bg <job-no>`.
- Stopping a job freezes its execution, no instructions in the program are executed, till the process is continued, for example by the `bg` command.

```
$ sleep 10
^Z
[1]+ Stopped
$ bg %1
[1]+ sleep 10 &
```

- Usually job numbers are prefixed by a percentage character.
- We could end up with lots of stopped jobs, or jobs running in the background. In case we would like to know what jobs are stopped or running in the background, the `jobs` command can be used.
- A stopped job or a background-ed job can be brought to foreground using `fg <job-no>`.

5. Terminator!

- The job running in the foreground, can be immediately terminated using the `Ctrl-C` key stroke.

- Processes running in the background or in other terminals can be terminated using, the `kill` command. The command can be invoked as `kill <pid>` or `kill <job-no>`.
- Processes can also be killed by specifying the command name they were started with. But since there could be more than one processes started with the same command name, this could potentially kill more than one process. The command to terminate processes by command name is `killall` and the general syntax is `killall <command-name>`

Try Out

1. Suspend and resume process
   - Start `top`.
   - Press Ctrl-Z to suspend `top`
   - Resume `top` in the foreground.
2. Suspend and background process
   - Make a copy of `/usr/bin/inkscape` to your home directory.
   - Press Ctrl-Z during the copy, to suspend the copy.
   - Resume the command in the background.
3. Run in background
   - Make a copy of `/usr/bin/inkscape` in the background.
4. Killing processes
   - Start multiple copies in the background.
   - Kill a copy process with the kill command.
   - Check the file size to verify that the copy command terminated.

6. I/O Redirection

- Programs usually produce some output. By default the output goes to the terminal.
- The output can be saved to file by using the output redirection operator `>`. The following example, stores the output of `ls` to `myfile`.

```
$ ls > myfile
```

- Note that, when the output is redirected to a file, the original contents of the file will be lost.
- If instead the output should be appended to the existing contents of the file the `>>` redirection operator can be used.
- Apart from regular output, a command can print errors.
- To redirect errors prefix the output redirection operator with `2` as `2>` and `2>>`.

Try Out

- Store the output of `pstree` to a file.
- Perform a file copy using the `cat` command and redirection operators.
- Switch to the home directory.
- Join the 3 files `elements1.txt`, `elements2,txt`, `elements3.txt` into a single file using a single `cat` command and the redirection operator.
- Join the 3 files using multiple `cat` commands and the redirection with append operator.

7. Pipes and Filters

- Some programs get input from the terminal. For example, the figlet commands reads input from the terminal and prints it in a large font.

```
$ figlet
abc
      _
  __ _| |__   ___
 / _` | '_ \ / __|
| (_| | |_) | (__
 \__,_|_.__/ \___|
```

- When no more input, needs to be given the user can press `Ctrl-D` to indicated end of input.
- This way of getting input from the user, is called reading from standard input.
- Pipes are mechanism provided by the operating system, to send output of one command as the input another command. The general syntax is `cmd1 | cmd2`. The output of `cmd1` is provided as input to `cmd2`.

```
$ echo "Hello" | figlet
 _          _ _
| |__   ___| | | ___
| '_ \ / _ \ | |/ _ \
| | | |  __/ | | (_) |
|_| |_|\___|_|_|\___/
```

- One of the design philosophies of Unix was the KISS philosophy - Keep It Small and Simple. Most Unix programs were simple tools and Unix provided frameworks like pipes and scripting to combine these simple programs to do complex work
- Filters are programs that read data from standard input and write to standard output.

8. Heads or Tails

- Display portion of a file: `head`, `tail`
- `head` - reads input and outputs only the first n lines.
- `tail` - reads input and outputs only the last n lines.
- The `-n` option specifies the no. of lines.

```
$ cat /var/log/Xorg.0.log | tail -n 2
(II) AIGLX: Resuming AIGLX clients after VT switch
(II) Configured Mouse: ps2EnableDataReporting: succeeded
$ cat myfile
line 1
line 2
line 3
line 4
$ cat myfile | head -n 3 | tail -n 1
line 3
```

- `tail` is useful for viewing log files.
- `head` can be used in combination with `tail` to extract lines from the middle of a file.

Try Out

- Switch to the home directory.
- `cities.txt` contains the 10 largest cities in the world (by population).
- Display the 5th largest city in the word using `head` and `tail`.

9. Summarising and Sorting

- Summarising with `wc`
  - `wc` program prints the no. of characters, words and lines in its input.
  - By default prints all the three.
  - Only character `-c`, only words `-w` and only lines `-l`.
- Selecting fields in a line with `cut`
  - Field are specified using the option `-f <field1>-<field2>`, fields can be separated by commands, or ranges can be specified.
  - The default delimiter is Tab. The delimiter can be changed using the option `-d <delim-char>`
  - Useful for processing text based database files like `/etc/passwd`.
  - To extract all users and their user ids.

```
cat /etc/passwd | cut -f 1,3 -d ":"
```

- Sorting: `sort`
  - Sorts input based on a key field.
  - Uses entire line as key, by default.
  - If a particular field is to be treated as the key, the `-k <field no.>` option can be used.
  - By default fields are considered to be separated by white spaces. A different separator can be specified using the `-t` option.
  - The sorting order can be reversed using the `-r` option.
  - Sort output of `ps` by command name.

```
ps -A --no-headers | sort -k 4
```

  - By default, sort performs as string sort, use `-g` option to specify numeric sort.
  - For an example data of 1, 2, 3, 10, 11.
  - String Sort yields: 1, 10, 11, 2, 3
  - Numeric Sort yields: 1, 2, 3, 10, 11
  - In string sort, the character at which the strings do not match, is used to determine if the string is lesser or greater than the other string. And a missing character is lesser than any other character.
  - Sort the files in the directory on the order of size

```
ls -l | sort -k 5 -g
```

Try Out

- Sort all elements in `elements.txt` on the basis of abbreviation.
- Display only the element names in `elements.txt`.

10. Find and Replace

- Selecting lines that match an pattern: `grep <exp>`
- Get all the words from `/usr/share/dict/words` that contain the word `bug`.

```
cat /usr/share/dict/words | grep bug
```

- Searching and replacing text: `tr`
  - translates input from one set of characters to another set of characters
  - squeeze repetitions `-s <char>`
  - delete characters `-d <char>`
- Filtering repetitions: `uniq`
  - filters out consecutive repetitive lines in the input.

11. Examples

- List of users in the system
- List files in a directory with filename and size alone
- Find the PID given the command name
- Using the dictionary, find number of words in the English language starting with in each alphabet
- Which letter in the English language is the most common starting letter for words?

12. Further Reading

- Introduction to Linux: Processes: Processes inside out - http://www.tldp.org/LDP/intro-linux/html/sect_04_01.html
- Bash Reference Manual: Bash Features: Job Control - http://www.gnu.org/software/bash/manual/html_node/Job-Control.html#Job-Control
- FreeBSD Handbook: Unix Bascis: Processes and Daemons - http://www.freebsd.org/doc/handbook/basics-processes.html
- Debian Reference: GNU/Linux tutorials: Unix-like text processing - http://www.debian.org/doc/manuals/reference/ch-tutorial.en.html