Quick Guide to UNIX

By default, the CSIF system use a command interpreter (called a *shell*) called *tcsh*. The shell commands described below work in this shell. They may, or may not, work in others.

File Systems in UNIX

- File hierarchy — directories are arranged in a hierarchy with root (/) at the top. The position of any file within the hierarchy is described by its pathname.
  - Absolute pathname — a pathname which describes the file location starting from the root
  - Relative pathname — a pathname which describes the location of a file or directory relative to the current directory
- File types
  - Files — an ordinary file is a file on the system that contains data, text, or program instructions.
  - Directories — directories store both special and ordinary files and are equivalent to folders.
  - Hard links — directory entry that references the inode of a file. If more than one hard link exists for a file, if the original is deleted, the others can still access the file; it just means that the original reference to the file is deleted.
  - Soft links (also called symbolic links) — a pointer that contains information about the path to another file. If the original file is deleted then the soft link can no longer access the file.
  - Hidden files — files that start with '`.`' and are usually used to store configuration information
- File permissions — read ('`r`'), write ('`w`'), and execute ('`r`'), for example `drwxrwxrwx`
  - The first character refers to the file type ('`d`' for directory, '`-`' for regular file, "`l`' for soft link)
  - Characters 2–4 are permissions for the owner of the file
  - Characters 5–7 are permissions for groups that this file belongs
  - Characters 8–10 are permissions for everyone else
- Useful directory related commands
  - `ls` — provides a list of all the files under the named directory (by default it uses the current directory)
  - `mkdir` — make a new empty directory

    ```
    mkdir dir_name
    ```
  - `rmdir` — remove a new empty directory

    ```
    rmdir dir_name
    ```
  - `cd` — change directories
    - to go to any directory: `cd pathname/directory`
    - to go to your home directory: `cd ~` or `cd`
    - to move one level up - `cd ..`
    - to go to the root directory - `cd /`
  - `pwd` — present working directory
  - `mv` — move (or rename) files or directories

    ```
    mv old_file new_file
    ```
  - `cp` — copy files from one directory to another

    ```
    cp source_file destination_file
    ```
  - `rm` — remove files from directories (***be very careful while using variations of this command***)

    ```
    rm [option] filename(s)
    ```
  - `find` — to find files and directories based on name, date or other parameters
- Useful file related commands
  - `cat` — display file contents

    ```
    cat filename
    ```
  - `chmod` — changing file permissions
    - allowing others to read and execute file: `chmod o+rx filename`
    - allowing user to write into file: `chmod u=w filename`
    - revoking group execute permission: `chmod g-x filename`

    * using absolute permissions: `0` = no permission, `1` = `--x`, `2` = `-w-`, `3` = `-wx`, `4` = `r--`, `5` = `r-x`, `6` = `rw-`, `7` = `rwx`. For example, `chmod 755 filename` sets the permissions to `rwxr-xr-x`
  - `diff` — finds the difference between 2 files

```
diff file1 file2
```

  - `head` — display the first few lines of a file

```
head filename
```

  - `tail` — displays the last lines of the file
  - `touch` — update access and modification date of a file.
  - `grep` — searches for files that have a particular pattern, without a file name it reads the standard input (or from a pipeline)

```
grep [options] pattern filename(s)
```

  - `sort` — arranges lines of text in the file alphabetically or numerically

```
sort [options] filename
```

Processes in UNIX

- Shell — command line interface similar to DOS or the command window in Windows
  - loads when you open a terminal window
  - prompt followed by a cursor (`$`, `%`, >)
  - commands are usually words with blank spaces in between; first word is the UNIX command and the others are considered options and arguments.
- Processes — instance of a running program with a 1–5 digit ID called the PID
  - foreground — by default every process that has been started by the user and takes an input from the user and outputs the result.
  - background — a process that isn't connected to the keyboard and waits when it has to interact with the user.
- Redirection — redirects between standard input and standard output and files
  - uses >, <, >>, and `|`
  - > — redirects output to a file; creates if not existing; overwrites if exists
  - >> — same as above, but if the file exists, it appends
  - < — redirects standard input, that is, causes the process to take input from a file; you can combine these, so

```
x < y.txt > z.txt
```

    tells the command `x` to read input from the file `y.txt` and write its output to `z.txt`
  - `|` — pipeline, output of command on the left is used as input to the command on the right
- File naming conventions — files can be named anything including special characters, but in order for the shell to be able to distinguish the special characters as being a part of a file name and not as a special character, each special character needs to be escaped by a backslash (\) or the whole file name must be in quotes. For example, '`?`' is a special character (see the next item). So to give the file name `x?  y:z` (there's a blank after the `?` in the file name)' to a command, that file name would be written as either `x\?\ y\:z` or `'x?  y:z'`
- Wildcards — used for pattern matching
  - `*` — matches any string of 0 or more characters
  - `?` — matches a single character
  - `[...]` — matches any among the list provided between `[]`
    * `ls [0-4,a-d,f].doc` — lists any of the files `0.doc`, `1.doc`, `2.doc`, `3.doc`, `4.doc`, `a.doc`, `b.doc`, `c.doc`, `d.doc`, and `f.doc` that exist
    * `ls G?[^a-c]*x*.doc` — lists all files that start with `G`, has any character in the second position (the `?`), has a third character that is not `a`, `b`, or `c` (the `[^a-c]`; the `^` means "that is not any of"), has an `x` as the fourth or later character (the `*x*`), and ends in `.doc`.
- Variables — can be set to user defined values; called *shell* or *environment* variables
  - Local environment variables — these values can only be used in the current shell; to set, say `set variable_name = value` and to erase, say `unset variable_name`
  - Global environment variables — these values can be used in child shells too (created by executing shell scripts); to set, say `setenv variable_name value` and to erase, say `unsetenv variable_name`

  - To use the value of a variable `variable_name` in a command, say `$variable_name`
- File name completion — by pressing TAB, this automatially fills in the rest of the file name if it is unique (and beeps if not)
- History — remembers the last few commands typed (to save time)
  - `$history` variable that says how many commands are remembered, so
    ```
    set history=50
    ```
    causes the shell to remember the last 50 commands
  - The command `history` will produce a list of commands numbered as they were entered; that is, the first command is 1, and so on
  - `!` — command completion from history; `!r` will look for previous commands starting with `r` and execute the most recent one; `!`*n* will execute the command numbered *n*, if it is in the history
- Aliases — way of creating your own commands that are built up from other commands and options. For example,
  ```
  alias myLS "ls l"
  ```
  allows you to give the command `myLS`, which executes `ls -l`
- Process related commands
  - `ps` — lists running processes, with options to show processes from all users, extended information, and so forth; see the manual page for details
  - `kill` — used to stop a background process after getting its PID
  - control-C (hold down the control and C keys simultaneously) — used to stop the foreground process
    * `kill PID` — terminate the process with process identification number PID
    * `kill -9 PID` — if a process will not stop after the above, use this; it stops the process immediately (with a few rare exceptions), but the process cannot save any work (so be very careful while using this command!)
  - `top` — used to list processes sorted by various criteria

Information Gathering Commands

- `apropos` — searches the manual pages for a keyword or regular expression; call it as
  ```
  apropos [options] keywords
  ```
  and it displays manual pages that match any keyword Some commonly used options are:
  - `-a` — display items that match all keywords
  - `-e` — display items that match the keyword(s) exactly
- `man` — the system's manual page viewer; gives information on how to use a program or command; call it as
  ```
  man [options] command
  ```
  Some commonly used options are:
  - `-k` — same as the `apropos` command
  - `-a` — show all manual pages that match the command (usually, only one of the pages is displayed)

More Information

See `http://www.tutorialspoint.com/unix/` and `http://www.computerhope.com/unix.htm`

Credit

This was written for ECS 30, Programming and Problem Solving, in Fall 2015 by Pallavi Kudigrama, and modified slightly by Matt Bishop.