title: Programming Fundamentals
subtitle: Reference

Summaries and Resources

BCE (0-1_BCE.md)

- An operating system is a suite of programs which make the computer work.
- UNIX is an very popular operating system, used on Macs and bunch of others.
- UNIX has a kernel, file system, and programs.
- Linux is an open source Unix-like system
- A virtual machine is like using somebody else's computer on your very own laptop
- The Berkeley Common Environment (BCE) is a standardized VM using Linux
- BCE comes with Python, R, and a bunch of other cool stuff to do your science.
- You can share files between your laptop and your BCE.
- BCE will take some time getting use to, especially the keyboard if you're using a Mac.

Introducing the Shell (1-0_shell.md)

- A shell is a program whose primary purpose is to read commands and run other programs.
- The shell's main advantages are its high action-to-keystroke ratio,
  its support for automating repetitive tasks,
  and that it can be used to access networked machines.
- The shell's main disadvantages are its primarily textual nature
  and how cryptic its commands and operation can be.

Files and Directories (1-1_filedir.md)

- The file system is responsible for managing information on the disk.
- Information is stored in files, which are stored in directories (folders).
- Directories can also store other directories, which forms a directory tree.
- `cd path` changes the current working directory.
- `ls path` prints a listing of a specific file or directory;
  `ls` on its own lists the current working directory.
- pwd prints the user's current working directory.
- `whoami` shows the user's current identity.
- / on its own is the root directory of the whole filesystem.
- A relative path specifies a location starting from the current location.
- An absolute path specifies a location from the root of the filesystem.
- Directory names in a path are separated with '/' on Unix, but '\' on Windows.
- '..' means "the directory above the current one";
  '.' on its own means "the current directory".
- Most files' names are `something.extension`.
  The extension isn't required,
  and doesn't guarantee anything,
  but is normally used to indicate the type of data in the file.

- Most commands take options (flags) which begin with a '-'.
- ~ stands for the user's home directory. Use it at the beginning of a path, like `~/path/to/file`
- If you type enough letters of your command or argument, then you can press tab to have it automatically completed.
  Double tab displays all the available options.
- Up Arrow displays last command in the command line.
- Copy a file/directory in the GUI and paste them into the command line to give the file/directory's full path.

Creating Things (1-2-create.md)

- `cp old new` copies a file.
- `mkdir path` creates a new directory.
- `mv old new` moves (renames) a file or directory.
- `rm path` removes (deletes) a file.
- `rmdir path` removes (deletes) an empty directory.
- `touch path` creates an empty file if it doesn't already exist.
- Unix documentation uses '^A' to mean "control-A".
- The shell does not have a trash bin: once something is deleted, it's really gone.
- Nano is a very simple text editor --- please use something else for real work.
- * is a wildcard. It matches zero or more characters
- Naming/structuring your files and directories in a systematic way is important.

Pipes and Filters (1-3_pipe.me)

- `cat` displays the contents of its inputs.
- `head` displays the first few lines of its input.
- `tail` displays the last few lines of its input.
- `sort` sorts its inputs.
- `wc` counts lines, words, and characters in its inputs.
- `command > file` redirects a command's output to a file.
- `first | second` is a pipeline: the output of the first command is used as the input to the second.
- The best way to use the shell is to use pipes to combine simple single-purpose programs (filters).

Loops (1_4-loop.md)

- `history` displays recent commands, and `!number` to repeat a command by number.
- A `for` loop repeats commands once for every thing in a list.
- Every `for` loop needs a variable to refer to the current "thing".
- Use `$name` to expand a variable (i.e., get its value).
- Do not use spaces, quotes, or wildcard characters such as '*' or '?' in filenames, as it complicates variable expansion.
- Give files consistent names that are easy to match with wildcard patterns to make it easy to select them for looping.
- Use the up-arrow key to scroll up through previous commands to edit and repeat them.
- Use `history` to display recent commands, and `!number` to repeat a command by number.

Shell Scripts (1-5-script.md)

- Save commands in files (usually called shell scripts) for re-use.
- `bash filename` runs the commands saved in a file.
- `$*` refers to all of a shell script's command-line parameters.
- `$1`, `$2`, etc., refer to specified command-line parameters.
- `$@` refer to all command-line parameters. Especially helpful for wildcards.
- Place variables in quotes if the values might have spaces in them.
- Letting users decide what files to process is more flexible and more consistent with built-in Unix commands.

Finding Things: (Not covered, but helpful)

- `find` finds files with specific properties that match patterns.
- `grep` selects lines in files that match patterns.
- `man command` displays the manual page for a given command.
- `*` matches zero or more characters in a filename, so `*.txt` matches all files ending in `.txt`.
- `?` matches any single character in a filename, so `?.txt` matches `a.txt` but not `any.txt`.
- `$(command)` inserts a command's output in place.
- `man command` displays the manual page for a given command.
- Find the whole lesson [here (http://software-carpentry.org/v5/novice/shell/06-find.html)](http://software-carpentry.org/v5/novice/shell/06-find.html)

Python and Beyond (1-6_python.md)

- An interpreter is a program that reads and executes code.
- `which [program]` gives you the version of a program, and, by extension, whether you have it installed.
- Run Python in interactive mode in bash by typing python.
- Run Python in normal mode in bash by typing python `[scrip.py]`
- `quit()` gets you back into bash.
- A module is a python script that has helpful functions n such.
- A package is a collection of python modules.
- `pip install [package]` is the easiest way to install new packages
- BCE comes with most of the packages you need already installed.
- try sudo in the beginning of a bash command when you face permissions problems.
- Use an IDE like PyCharm to develop your Python code.

Getting Help (2-0_help.md)

- Don't learn specific programming languages; learn *how to program*
- Most of your programming will be spent debugging, looking things up on the internet, and testing.
- Google errors!

Glossary

absolute path
: FIXME

argument
: FIXME

command shell
: FIXME

command-line interface
: FIXME

comment
: FIXME

current working directory
: FIXME

file system
: FIXME

filename extension
: FIXME

filter
: FIXME

flag
: FIXME

graphical user interface
: FIXME

home directory
: FIXME

loop
: FIXME

loop body
: FIXME

orthogonal
: FIXME

parent
: FIXME

pipe
: FIXME

process
: FIXME

prompt
: FIXME

quoting
: FIXME

read-evaluate-print loop
: FIXME

redirect
: FIXME

regular expressions
: FIXME

relative path
: FIXME

root directory
: FIXME

shell script
: FIXME

standard input
: FIXME

standard output
: FIXME

sub-directories
: FIXME

tab completion
: FIXME

variable
: FIXME

wildcard
: FIXME