Shell Reference

Basic Commands

- `cat` displays the contents of its inputs.
- `cd path` changes the current working directory.
- `cp old new` copies a file.
- `find` finds files with specific properties that match patterns.
- `grep` selects lines in files that match patterns.
- `head` displays the first few lines of its input.
- `ls path` prints a listing of a specific file or directory; `ls` on its own lists the current working directory.
- `man command` displays the manual page for a given command.
- `mkdir path` creates a new directory.
- `mv old new` moves (renames) a file or directory.
- `pwd` prints the user's current working directory.
- `rm path` removes (deletes) a file.
- `rmdir path` removes (deletes) an empty directory.
- `sort` sorts its inputs.
- `tail` displays the last few lines of its input.
- `touch path` creates an empty file if it doesn't already exist.
- `wc` counts lines, words, and characters in its inputs.
- `whoami` shows the user's current identity.

Paths

- `/path/from/root` is an absolute path.
- `/` on its own refers to the root of the filesystem.
- `path/without/leading/slash` is a relative path.
- `.` refers to the current directory, `..` to its parent.
- `*` 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`.

Combining Commands

- `command > file` redirects a command's output to a file.
- `first | second` connects the output of the first command to the input of the second.
- A `for` loop repeats commands once for every thing in a list:

```
for variable in name_1 name_2 name_3
do
    ...commands refering to $variable...
done
```

- Use `$name` to expand a variable (i.e., get its value).
- `history` displays recent commands, and `!number` to repeat a command by number.
- `bash filename` runs commands saved in `filename`.
- `$*` refers to all of a shell script's command-line parameters.
- `$1`, `$2`, etc., refer to specified command-line parameters.
- `$(command)` inserts a command's output in place.