### Shell Command Line Substitutions

**Expansions**
- `$(command)` — Substitutes value of variable `command`.
- `@command` — Substitutes all characters in `command`.
- `{command}` — Substitutes the output of `command`

### Piping
- `command | command` — Takes the output of `command` and gives it as input to `command`.

### Redirecting Standard Input
- `command < file` — Takes the input from file and gives it as input to `command`.

### Redirecting Standard Output
- `command > file` — Prints the output to file.
- `command > file &` — Prints the output to file and runs `command` in the background.

### Standard Process I/O
- `0` — Standard input
- `1` — Standard output
- `2` — Standard error

### Additional Notes
- To switch between shells temporarily, just run a new shell. To make it permanent, run `source` command.
- To graph an expansion of each subshell, use `set -p` (set all). For example, `set -o debugging`.
- There are different flavors of shells: called `command shells`. There are different flavors of shells: `bash` — Bourne again shell, another shell compatible shell, incorporates `ksh` — Korn shell — a compatible shell.
- `csh` — C shell
- `sh` — Bourne shell — very basic one
- `Sh` — The program that reads the command line: `pass` does I/O be redirected. By default, all 3 point to your current terminal, but any of them can be redirected:
- `&` — An asterisk to mean reading from stdin or writing to stdout.
- `>file` — Same as `>`
- `=file` — Same as `=`
- `>` — Standard output
- `>` — Standard error

In Unix, a process normally has at least 3 I/O channels (or file descriptors):