Standard process I/O

In Unix, a process normally has at least 3 I/O channels (or *file descriptors*) associated with it:
0 — `stdin` — standard input
1 — `stdout` — standard output
2 — `stderr` — standard error

Most programs print their output to `stdout` and error messages to `stderr` (screen–oriented programs like `vi`, `pine` or `less` and X-Windows programs are an exception). Many programs that normally operate on files would operate on `stdin` when no file argument is given, for example `grep`, `sort`, `wc`. Many programs would allow users to specify "–" instead of a file name to mean reading from `stdin` or writing to `stdout`.

By default, all 3 point to your current terminal, but any of them can be redirected.

Redirecting stdout

`command` `>` *`file`* redirects the output (but not the error messages) of the *`command`* to *`file`*.

Redirecting stdin

`command` `<` *`file`* takes the input of the of the *`command`* from *`file`*.

Pipes

`command1` `|` *`command2`* takes the output of *`command1`* and gives is an input to *`command2`*.

It's possible to do many redirections at the same time:
`command1` `<` *`infile`* `|` *`command2`* `|` *`command3`* `|` *`command4`* `>` *`outfile`*

ShellsThe program that reads the command line, parses it, does I/O redirection, calls appropriate commands with appropriate options, *etc* is called *command shell*. There are different flavors of shells:

`sh` — Bourne **sh**ell — very basic one
`csh` — **C sh**ell
`ksh` — **K**orn **sh**ell — `sh`-compatible shell
`bash` — **B**ourne **a**gain **sh**ell — another `sh`-compatible shell, incorporates features from both `ksh` and `csh`
`tcsh` — an extension of `csh`

To switch between shells temporarily, just run a new shell. To make it permanent, run `chsh` command.

Shell command line substitutions (expansions)

...‘*`command`*‘... — substitute the output of a *`command`*
`*`, `?`, `[ ]` — substitute all matching filenames
`{}` — substitute all choices
`\` — escape next char
`$`*`var`* — substitute value of variable *`var`*
`"..."` — only variable and command expansion
`'...'` — pass everything as-is