Chapter 5

Development Tools

Development Tools

Read … :

BLP: Chapter 9

- make
- **Revision control**
- Writing manual pages
- Distributing software: patch, tar, rpm
- Development Environments

info make

Makefile

```
simple_calc: simple_parser.tab.o simple_lexer.o Expr.o
        $(CC) $(CFLAGS) –o $@ $^

simple_parser.tab.h simple_parser.tab.c: simple_parser.y
        bison –d $<

simple_lexer.o: simple_parser.tab.h Expr.h
simple_parser.tab.o: Expr.h
```


The Language “Make”

- **Rule-based** artefact production language
- Rules (normally) specify how to produce **targets** from their prerequisites
- Rules consist of a description of a **dependency relation** and and **action**
- A *make* run performs a bottom-up traversal of the dependency graph
- Actions are triggered unless a target is newer than all its prerequisites
- Actions are specified in a shell language (sh, bash)
- Performing the action of a rule should satisfy its dependency
- make and in particular gmake has a wealth of **built-in rules** and default definitions

Makefile Syntax

- **Dependencies:** *target* **:** *sources*
- **Rules:** ***shell*** commands ***after a TAB character (^I)***
- **Comments:** line comments starting with "#"
- **Variable** definitions (spaces around "=" allowed)
  - Variable references: `${VARNAME}` or `$(VARNAME)`
  - Variable references default to environment variables
  - Make variable do not enter command environments
  - (Local) shell variable references need escape: `$$SHELLVARNAME`

General Rules

- Suffix rules (BLP):

```
.c.o
        $(CC) $(CFLAGS) –c $<
```

"***the old-fashioned way of defining implicit rules***"

- **Pattern Rules:**

```
%.o : %.c
        $(CC) $(CFLAGS) $(CPPFLAGS) –c $< –o $@
%.tab.c %.tab.h: %.y
        bison –d $<
```

- **Static Pattern Rules:**

```
objects = foo.o bar.o
$(objects): %.o: %.c
        $(CC) –c $(SPECIALCFLAGS) $< –o $@
```


Building Software Packages From Source

- source-code only
  — look for main module, config.h; read; compile…
- README-based
  — **read** README, INSTALL; follow instructions
- Makefile-based
  - edit configuration part of Makefile
  - make all
  - make install (as root)
- configure-based (GNU auto-tools)
  - ./configure --help — notice enable* and with* options
  - ./configure --prefix=*/usr/local/packages/XYZ*
  - make — sometimes separately: make doc html ps
  - make install (as root)

Make Discipline

- The first target is a useful default, e.g., all
- Targets all, install, clean, distclean are defined as appropriate
- install is responsible for permissions! (→ umask)
- User-editable parts are in variable definitions at the top of the Makefile

  "-------- do not edit beyond this point ---------"
- Consider generating your Makefiles using the GNU auto-tools (autoconf, automake) for dealing with portability

  (configure.ac, configure.in, Makefile.in, …)