

Software Development using GNU/Linux

Presentation for the Student ACM at Marquette

Michael J. Schultz

February 4, 2009

Overview

- Authoring, interpreting, compiling, debugging, project management, build testing, and documentation.
- Linux offers a wide range of support for many programming languages and development tools.
- This presentation can only scratch the surface of developing software using the Linux operating system.

Language Support

- Interpreted Languages
 - perl
 - python
 - php
 - ruby
 - lisp
 - scheme
 - ...
- Compiled Languages
 - C
 - C++
 - Objective-C
 - Ada
 - Java
 - C#

Language Tools

- Command line interpreters (python, php, perl, clisp)
- Compilers available include javac, mono, and gcc!

Makefiles

- Consist of variables, rules, targets, dependencies, and commands.

```
# Example Makefile for presentation
```

```
CONFIG = config.c
```

```
SOURCES = ${CONFIG} demo.c
```

```
CFLAGS = -m32 -Os
```

```
CC      := gcc ${CFLAGS}
```

```
target: ${SOURCES}
```

```
    $(CC) -o $@ $^
```

```
    @echo "    [CC] Complete"
```

Variables

- Two types of variables
 - 1 recursively expanded (=)
 - 2 simply expanded (:=)

```
CONFIG = config.c  
SOURCES = ${CONFIG} demo.c  
CFLAGS = -m32 -Os  
CC      := gcc ${CFLAGS}
```

Rule, targets, and dependencies

- Rules
 - A rule refers to the collection of a target, possible dependencies, and commands to execute.
- Targets
 - A target is what to build (`make target`)
 - Occur on the left hand side of a semi-color (`:`)
 - Can use a variable as a target (i.e. `%.o`)
- Dependencies
 - Dependencies are optional, and tell `make` what is needed to fulfill this target.
 - `${SOURCES}` is the example, expands to `config.c` `demo.c`—both of which are required to create target.

```
target: ${SOURCES}
```

Commands

- Commands run in the \$SHELL environment.
- Must be tabbed in one level.
- Command executed will be echoed (in expanded form) by default
 - Echoing can be suppressed by prefixing with @.
- Ignore errors values in return status by prefixing with -.

```
$(CC) -o $@ $^
@echo "    [CC] Complete"
```

Commands

- Commands can also take advantage of ‘automatic’ variables
 - `$$` replaced with name of target.
 - `$$<` replaced with first dependency.
 - `$$^` replaced with all dependencies (skipping duplicates).
 - `$$+` replaced with all dependencies (including duplicates).
 - `$$?` replaced with dependencies that are newer than target.

```
target: ${SOURCES}
    $(CC) -o $$ $^
    @echo "    [CC] Complete"
```

Example Run

```
# Example Makefile for presentation
CONFIG = config.c
SOURCES = ${CONFIG} demo.c
CFLAGS = -m32 -Os
CC      := gcc ${CFLAGS}

target: ${SOURCES}
    $(CC) -o $@ $^
    @echo "    [CC] Complete"
```

- Running 'make' will result in the following output:

```
gcc -m32 -Os -o target config.c demo.c
    [CC] Complete
```

Example Run

```
gcc -m32 -Os -o target config.c demo.c  
[CC] Complete
```

- `$(CC)` was replaced with `gcc -m32 -Os`
- `$(@)` was replaced with `target`
- `$(^)` was replaced with `config.c demo.c`
- The first command was echoed to the shell when executed, the second was not.
- The directory now contains an executable named `target`.
- If `make` is run a second time we get the message

```
make: 'target' is up to date.
```

Command Line Arguments

- By default `make` will build the first target in the Makefile.
- Specify different targets at the command line: `make target1 target2`, will build `target1` and `target2`.
- `make -n`: print what will happen but not perform the commands.
- `make -C <dir>`: change to `<dir>` before reading the Makefile.
- `make -j <njobs>`: tell `make` to spawn `<njobs>` for building.

Other Topics

- Ant (make for Java)
- Source Code Control (subversion, git, mercurial, etc.)
- Debugging (GNU debugger, -gddb)
- Autoconf and automake (./configure)

Thanks

Thank you for your attention. Questions are welcomed now or later.

- Make information found at <http://www.gnu.org/software/make/manual/make.html>
- Presentation available from <http://www.beyond-syntax.com/>
- Contact me at mjs@beyond-syntax.com