Algorithm Engineering

Jens K. Mueller

jkm@informatik.uni-jena.de

Department of Mathematics and Computer Science
Friedrich Schiller University Jena

Monday 27th October, 2014

Building Software

Configure, Build, and Install

1. Configure
   Are all dependencies (like libraries, system calls) fulfilled?
   Is there a working compiler? What features to enable?
2. Build
   Controls the building process and inter source dependencies.
3. Install
   Copy files to the appropriate places.

In the Unix world the configuration is done via autotool's configure. This generates a Makefile that is used for building. The Makefile provides an install target.

Guidelines for Building Software

- Enable warnings
- Decent optimization level
- Release/Testing/Debug builds

Make

- Dependency tracking
- Based on file time-stamps
- Different (also non-standard) implementations (g)make (GNU Make), nmake (Microsoft), pmake (BSD), ...
- Reads file `Makefile` by default
- `$ make target`

Make language (excerpt)

- Macros

```
CFLAGS = -Wall
```

- Suffix rules

```
%.o: %.d
    $(DC) $(DFLAGS) -c $< -o $@
```

- Rules

```
targets: prerequesites
    commands
```

At least one target (comma separated), zero or more prerequisites (comma separated) and zero or more commands (newline separated).

Example Makefile

```
 1  DC = dmd
 2  DFLAGS = -property -w -wi -gc -Isrc
 3  LD = dmd
 4
 5  SOURCES = $(wildcard src/*.d)
 6  OBJECTS = $(patsubst %.d, %.o, $(SOURCES))
 7  BINARY = wordcloud
 8
 9  %.o: %.d
10      $(DC) $(DFLAGS) -c $< -of$@
11
12  .PHONY: all
13  all: build
```

Example Makefile (cont.)

```
14
15 .PHONY: build
16 build: $(BINARY)
17
18 $(BINARY): $(OBJECTS)
19     $(LD) $(LDFLAGS) $^ -of$@
20
21 .PHONY: clean
22 clean:
23     rm -f $(OBJECTS) $(BINARY)
```