

Makefile

Ansuman Banerjee
Arijit Bishnu
Debapriyo Majumdar

Data and File Structures Lab
M.Tech. Computer Science 1st Year, Semester I
Indian Statistical Institute Kolkata

August 26, 2014

Compilation stages

1. Compiler stage:

C file → lower level assembly language code

2. Assembler stage:

Assembly language code → Object code (machine directly understands) [.o files]

3. Linker stage:

Object codes → Linked to produce an executable file

May also link with some built in functions

The Codebase

main.c

add.c

mean.c

add.h

mean.h

- Download:
<http://www.isical.ac.in/~pdslab/ProgramsForMakefile.tar.gz>

- Trivial compilation:

```
$ gcc main.c add.c mean.c -o prog
```

- Create object files, and compile

```
$ gcc -c add.c -o add.o  
$ gcc -c mean.c -o mean.o  
$ gcc -c main.c -o main.o  
$ gcc main.o mean.o add.o -o prog
```

Dependencies

```
$ gcc -c add.c -o add.o  
$ gcc -c mean.c -o mean.o  
$ gcc -c main.c -o main.o  
$ gcc main.o mean.o add.o -o prog
```

prog

add.o

mean.o

main.o

add.c

mean.c

main.c

add.h

mean.h

The mean function

Arithmetic mean

```
#include "mean.h"

int mean(int n1, int n2)
{
    return (n1+n2)/2;
}
```

Geometric mean

```
#include "mean.h"
#include <math.h>

int mean(int n1, int n2)
{
    return sqrt(n1*n2);
}
```

prog

add.o

mean.o

main.o

add.c

mean.c

main.c

add.h

mean.h

Which object and executable files should change?

Recompiling

```
# Recompiling only mean.o and the prog
$ gcc -c mean.c -o mean.o
$ gcc main.o mean.o add.o -o prog
```

- A Makefile
 - Define dependencies and compilation instructions beforehand
 - When parts of a large codebase is changed, the make command recompiles selectively and builds the program

A Simple Makefile

```
# comment
```

The format

```
target: dependency1 dependency2 ...
```

```
    <tab> build instruction / command
```

```
prog: main.o mean.o add.o
```

Makefile

```
    gcc -Wall main.o mean.o add.o -o prog
```

```
main.o: main.c add.h mean.h
```

```
    gcc -c -Wall main.c -o main.o
```

```
mean.o: mean.c mean.h
```

```
    gcc -c -Wall mean.c -o mean.o
```

```
add.o: add.c add.h
```

```
    gcc -c -Wall add.c -o add.o
```

A Simple Makefile

```
$ make
```

```
$ make prog
```

Specifying the Target

Command

```
prog: main.o mean.o add.o
```

```
    gcc -Wall main.o mean.o add.o -o prog -lm
```

```
main.o: main.c add.h mean.h
```

```
    gcc -c -Wall main.c -o main.o
```

```
mean.o: mean.c mean.h
```

```
    gcc -c -Wall mean.c -o mean.o
```

```
add.o: add.c add.h
```

```
    gcc -c -Wall add.c -o add.o
```

Makefile

Defining more targets

Makefile

```
default: prog
```

```
all: prog
```

If there were more executable files, prog1 prog2 ... or the object files

```
prog: main.o mean.o add.o
```

```
    gcc -Wall main.o mean.o add.o -o prog -lm
```

```
main.o: main.c add.h mean.h
```

```
    gcc -c -Wall main.c -o main.o
```

```
mean.o: mean.c mean.h
```

```
    gcc -c -Wall mean.c -o mean.o
```

```
add.o: add.c add.h
```

```
    gcc -c -Wall add.c -o add.o
```

Variables

```
default: prog
```

```
all: prog
```

```
prog: main.o mean.o add.o
```

Objects

```
gcc -Wall main.o mean.o add.o -o prog -lm
```

```
main.o: main.c add.h mean.h
```

```
gcc -c -Wall main.c -o main.o
```

```
mean.o: mean.c mean.h
```

```
gcc -c -Wall mean.c -o mean.o
```

```
add.o: add.c add.h
```

```
gcc -c -Wall add.c -o add.o
```

Command

Flags

Makefile

Variables

Makefile

```
CC=gcc
CFLAGS=-c -Wall
LDFLAGS=-lm
OBJECTS=main.o add.o mean.o
default: prog
all: prog
prog: $(OBJECTS)
    $(CC) $(OBJECTS) -o prog $(LDFLAGS)
main.o: main.c add.h mean.h
    $(CC) $(CFLAGS) main.c -o main.o
mean.o: mean.c mean.h
    $(CC) $(CFLAGS) mean.c -o mean.o
add.o: add.c add.h
    $(CC) $(CFLAGS) add.c -o add.o
clean:
    rm -f $(OBJECTS)
```

Experiments for yourself

- Make clean and then run make
- Change the file add.h and run make
- Change the file main.c and run make