Makefile

RAVITEJA K
Applications Analyst, Cray inc

Makefile

Small program - Entire code in a single file

Complex program –

- Many individual files
- Many header files
- May require external linking
- Require coordination of different files to build

Makefile

Small program - Entire code in a single file

If we have single file:

main.c: The main program file

Compilation :

> cc -c main.c

Makefile

If we have simple applicaiton:

main.c: The main program file

library.c: Library .c file

library.h: Library header file

Compilation :

- cc -c main.c -o main.o
- cc -c library.c -o library.o
- cc library.o main.o -o main_exec

Makefile

For a simple application:

main.c: The main program file

library.c: Library .c file

library.h: Library header file

Compilation :

- cc -c main.c -o main.o
- cc -c library.c -o library.o
- cc library.o main.o -o main_exec

Makefile

What if you want to use flags? Like,

-O2 or –O3: For optimization

Compilation :

- cc -c –O2 main.c -o main.o
- cc -c –O2 library.c -o library.o
- cc library.o main.o -o main_exec

Makefile

What if you want to use other flags?

-Wall: To display warning

-g : To debug application

Compilation :

- cc -c –O2 –Wall –g main.c -o main.o
- cc -c –O2 –Wall –g library.c -o library.o
- cc –Wall –g library.o main.o -o main_exec

Makefile

What if you want to use other flags?

-Wall: To display warning

-g : To debug application

Compilation :

- cc -c –O2 –Wall –g main.c -o main.o
- cc -c –O2 –Wall –g library.c -o library.o
- cc –Wall –g library.o main.o -o main_exec

Makefile

Compilation :

- cc -c –O2 –Wall –g main.c -o main.o
- cc -c –O2 –Wall –g library.c -o library.o
- cc –Wall –g library.o main.o -o main_exec

```
CC = cc

CFLAGS = -g -Wall

OUTPUT = main_exec
```

Makefile

Syntax:

Target : dependency1 dependency2…
<TAB> [actions]

Example:

```
main_exec: library.o main.o
        cc -g -Wall library.o main.o -o main_exec
```

Makefile

```
main_exec: library.o main.o
    cc -g -Wall library.o main.o -o main_exec

library.o: library.c
    cc -g -Wall -c library.c -o library.o

main.o: main.c
    cc -g -Wall -c main.c -o main.o

clean:
    rm *.o main_exec
```

Makefile

```
main_exec: library.o main.o
    cc -g -Wall library.o main.o -o main_exec

library.o: library.c
    cc -g -Wall -c library.c -o library.o

main.o: main.c
    cc -g -Wall -c main.c -o main.o

clean:
    rm *.o main_exec
```

Makefile

```
CC = cc
CFLAGS = -g –Wall

main_exec: library.o main.o
    $(CC) $(CFLAGS) library.o main.o -o main_exec
library.o: library.c
    $(CC) $(CFLAGS) -c library.c -o library.o
main.o: main.c
    $(CC) $(CFLAGS) -c main.c -o main.o
clean:
    rm *.o main_exec
```

Makefile

```
CC = cc
CFLAGS = -g –Wall
EXEC= main_exec
OBJFILES= library.o main.o

$(EXEC) : $(OBJFILES)
	$(CC) $(CFLAGS) $(OBJFILES) -o main_exec
library.o: library.c
	$(CC) $(CFLAGS) -c library.c -o library.o
main.o: main.c
	$(CC) $(CFLAGS) -c main.c -o main.o
clean:
	rm *.o main_exec
```

Makefile

```
CC = cc
CFLAGS = -g –Wall
EXEC= main_exec
OBJFILES= library.o main.o

$(EXEC) : $(OBJFILES)
	$(CC) $(CFLAGS) $(OBJFILES) -o $(EXEC)


%.o: %.c
	# $<: dependency (%.c)
	# $@: target (%.o)
	$(CC) $(CFLAGS) -c $< -o $@
clean:
	rm *.o $(EXEC)
```

$< the name of the related file that caused the action

$@ is the name of the file to be made

Makefile uage

Naming:

makefile or Makefile are standard

Running make

- make
- make –f filename

  Eg : make –f makefile.test
- make target_name

  Eg: make gpu

Questions?