Ayudantía de Makefile

Ayudante: Luis Loyola Vidal
email: luis.loyola@usach.cl
Usach 2016.1

Makefile

hellomake.c	hellofunc.c	hellomake.h

#include <hellomake.h>
 	
int main() {
  // call a function in another file
  myPrintHelloMake();
 	
  return(0);
}
	#include <stdio.h>
	#include <hellomake.h>
	
	void
	myPrintHelloMake(void) {
	
	  printf("Hello makefiles!\n");
	
	  return;
	}
		/*
		example include file
		*/
		
		void
		myPrintHelloMake(void);

Makefile

- gcc -o hellomake hellomake.c hellofunc.c -I.
- Compila los 2 archivos .c y hace un ejecutable llamado hellomake.
- -I. Hace que gcc busque el “include hellomake.h” en el directorio actual (.)
- Problema: hay que compilar todos los archivos siempre.

Makefile1

- Regla: archivos que dependen de la regla
- \tcommando
- Ejecutamos el makefile escribiendo
  $ make
- Sin argumentos se ejecuta la primera regla.
- Problema: sigue compilando ambos archivos.

```
hellomake: hellomake.c hellofunc.c
    gcc -o hellomake hellomake.c hellofunc.c -I.
```

Makefile2

- Definimos constantes.
- CC para el compilador
- CFLAGS para los flags del compilador

```
CC=gcc
CFLAGS=-I.

hellomake: hellomake.o hellofunc.o
	$(CC) -o hellomake hellomake.o hellofunc.o -I.
```

- Al definir en la regla los archivos .o gcc sabe que debe compilarlos por separado y luego crear un executable con estos.

Makefile3

- La macro DEPS tiene los .h que requieren los .c para funcionar.
- Agregamos una regla para todos los archivos que terminen en .o (%.0:)que dice que se debe compilar el .c (%.c)

```
CC=gcc
CFLAGS=-I.
DEPS = hellomake.h

%.o: %.c $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

hellomake: hellomake.o hellofunc.o
	gcc -o hellomake hellomake.o hellofunc.o -I.
```

Makefile3

- -c indica que se debe generar el archivo objeto
- -o $@ indica que ese archivo objeto debe ser colocado en lo que está a la izquierda de los dos puntos. (en %.o).

```
CC=gcc
CFLAGS=-I.
DEPS = hellomake.h

%.o: %.c $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

hellomake: hellomake.o hellofunc.o
	gcc -o hellomake hellomake.o hellofunc.o -I.
```

Makefile3

- $< es el primer elemento en la lista de dependencias.
- $@ a la izquierda de “:”
- $^ a la derecha de “:”

```
CC=gcc
CFLAGS=-I.
DEPS = hellomake.h

%.o: %.c $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

hellomake: hellomake.o hellofunc.o
	gcc -o hellomake hellomake.o hellofunc.o -I.
```

Makefile4

- Todos los includes van en DEPS.
- Todos los archivo objeto van en OBJ.
- Todos los OBJ ejecutarán la última regla que dice: compile (gcc) con salida en hellomake (-o $@)(-o hellomake), los archivos que están a la derecha ($^), o sea, $(OBJ), o sea, hellomake.o y hellofunc.o con los flags definidos en CFLAGS.

```
CC=gcc
CFLAGS=-I.
DEPS = hellomake.h
OBJ = hellomake.o hellofunc.o

%.o: %.c $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

hellomake: $(OBJ)
	gcc -o $@ $^ $(CFLAGS)
```

Makefile5

- Para mantener el orden es recomendable tener los .h, los .c y los .o en carpetas distintas.
- El siguiente make define una carpeta para libs.
- Los objetos quedan en una carpeta obj dentro de source.

```
IDIR =../include
CC=gcc
CFLAGS=-I$(IDIR)

ODIR=obj
LDIR =../lib

LIBS=-lm

_DEPS = hellomake.h
DEPS = $(patsubst %,$(IDIR)/%,$(_DEPS))

_OBJ = hellomake.o hellofunc.o
OBJ = $(patsubst %,$(ODIR)/%,$(_OBJ))


$(ODIR)/%.o: %.c $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

hellomake: $(OBJ)
	gcc -o $@ $^ $(CFLAGS) $(LIBS)

.PHONY: clean

clean:
	rm -f $(ODIR)/*.o *~ core $(INCDIR)/*~
```

Makefile5

- Se define además una regla que limpia los archivos fuente y objeto si se hace:
  $make clean
- .PHONY impide que make trate el comando clean como si fuera un archivo llamado clean.

```
IDIR =../include
CC=gcc
CFLAGS=-I$(IDIR)

ODIR=obj
LDIR =../lib

LIBS=-lm

_DEPS = hellomake.h
DEPS = $(patsubst %,$(IDIR)/%,$(_DEPS))

_OBJ = hellomake.o hellofunc.o
OBJ = $(patsubst %,$(ODIR)/%,$(_OBJ))


$(ODIR)/%.o: %.c $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

hellomake: $(OBJ)
	gcc -o $@ $^ $(CFLAGS) $(LIBS)

.PHONY: clean

clean:
	rm -f $(ODIR)/*.o *~ core $(INCDIR)/*~
```

Bibliografía

- http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/
- http://www.gnu.org/software/make/manual/make.html