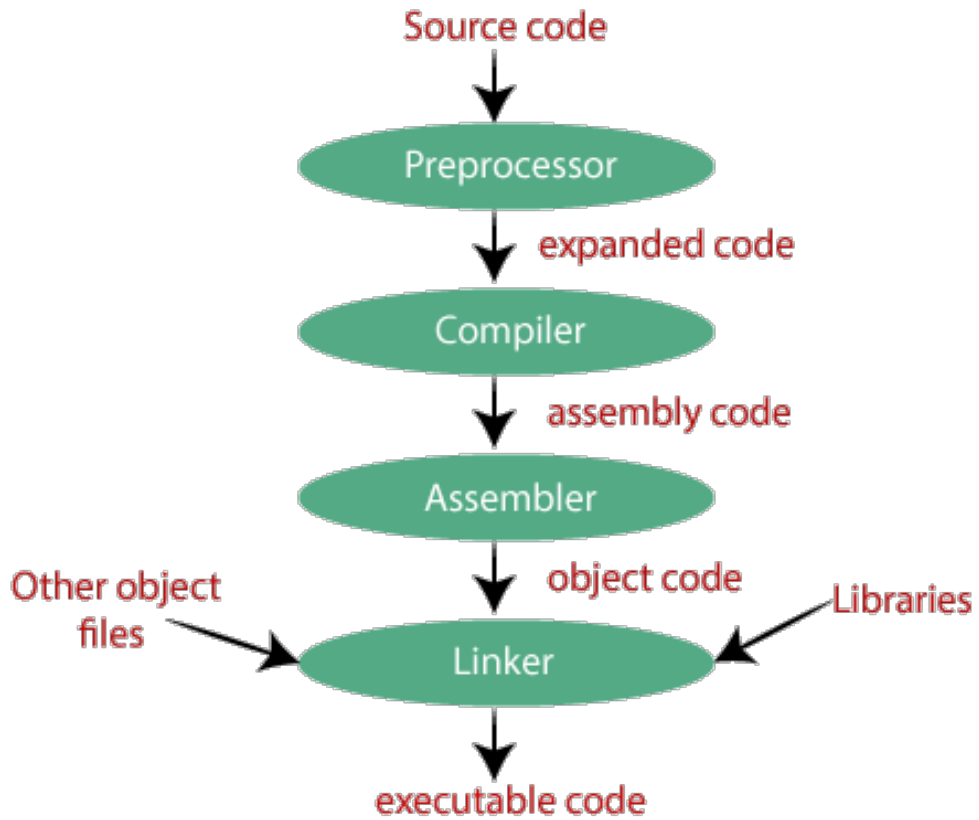
CS 136L Lecture 11 Module 8

Make

Make

Key idea: Compile small parts separately and link them.
(`-E`, `-S`, `-c`, `noflag`)



Source:
`https://www.javatpoint.com/compilation-process-in-c`

Make

What is the main principle behind `Make`? How does it make things more efficient?

Make Format

- Rules have the form:

```
target: prerequisite
    recipe
```

  - `target` should be the executable or .o file you want a rule for.
  - `prerequisites` should be a collection of dependencies. For the executable, usually just object files. For object file targets, a combination of `.c`, `.o` or `.h` files.
- Remember indentation is done by a single tab! Can't use spaces for indentation in Make files!

Example from Module

```
EXEC = myprogram
CC = clang
CFLAGS = -std=c99 -g -Wall -I/u2/cs136l/pub/
    common
LIB = /u2/cs136l/pub/common/*.o
SRC = $(wildcard *.c)
OBJECTS = $(SRC:.c=.o)
${EXEC}: ${OBJECTS}
    ${CC} ${CFLAGS} ${OBJECTS} ${LIB} -o ${EXEC}
vec.o: vec.c vec.h
    ${CC} ${CFLAGS} -c vec.c
linalg.o: linalg.c vec.h linalg.h
    ${CC} ${CFLAGS} -c linalg.c
main.o: main.c vec.h linalg.h /u2/cs136l/pub/
    common/cs136.h
    ${CC} ${CFLAGS} -c main.c
.PHONY: clean
clean:
    rm *.o ${EXEC}
```

Example from Module

```
EXEC = myprogram
CC = clang
CFLAGS = -std=c99 -g -Wall -I/u2/cs136l/pub/
    common -MMD
LIB = /u2/cs136l/pub/common/*.o
SRC = $(wildcard *.c)
OBJECTS = $(SRC:.c=.o)
DEPENDS = $(OBJECTS:.o=.d)
${EXEC}: ${OBJECTS}
    ${CC} ${CFLAGS} ${OBJECTS} ${LIB} -o ${
    EXEC}
# copy .d files providing .c dependencies
-include ${DEPENDS}
.PHONY: clean
clean: #Include .d below
    rm *.o *.d ${EXEC}
```