CS 332/532 Systems Programming

Lecture 11

-Makefile-

Professor : Mahmut Unan – UAB CS

Agenda

- makefile

Make Utility

- *make* is a utility that is used to automatically detect which program need to be recompiled while working on a large number of source programs and will recompile only those programs that have been modified.
- The *make* utility uses a *Makefile* to describe the rules for determining the dependencies between the various programs and the compiler and compiler options to use for compiling the programs.
- In case of C programs, an executable is created from object files (*.o files) and object files are created from source files.
- Source files are often divided into header files (*.h files) and actual source files (*.c files).

- The simplest way to organize the code compilation
- Make figures out automatically which files it needs to update, based on which source files have changed.

Exercise 1

```c
#include <stdio.h>

int main (int argc, char *argv[]){

char charArr[20]="Hello CS330";
printf("%s",charArr);
return 0;

}
```

```
myExecutable: main.c
  gcc -o myExecutable main.c

```

```
(base) mahmutunan@MacBook-Pro exercise1 % ls
Makefile        main.c
(base) mahmutunan@MacBook-Pro exercise1 % make
gcc -o myExecutable main.c
(base) mahmutunan@MacBook-Pro exercise1 % ./myExecutable
Hello CS330%
(base) mahmutunan@MacBook-Pro exercise1 %
```

Exercise 2

```c
#include "someFunc.h"

int main (int argc, char *argv[]){

  printHellofunct();
  return 0;

}
```

- main.c

```c
#include <stdio.h>
#include "someFunc.h"

void printHellofunct(){

  printf("Hello CS330");

}
```

- someFunc.c

```c
void printHellofunct();

```

- someFunc.h

Exercise 2

- we can compile multiple file using command line

```
(base) mahmutunan@MacBook-Pro exercise2 % ls
main.c          someFunc.c      someFunc.h
(base) mahmutunan@MacBook-Pro exercise2 % gcc -o exercise2 main.c someFunc.c -I
(base) mahmutunan@MacBook-Pro exercise2 % ./exercise2
Hello CS330%
(base) mahmutunan@MacBook-Pro exercise2 %
```

- The -I. is included so that gcc will look in the current directory (.) for the include file someFunc.h

Exercise 2

- We can use the make file to automate the compilation process

Makefile

```
1  exercise2withMake: main.c someFunc.c
2    gcc -o exercise2withMake main.c someFunc.c -I
3
```

```
(base) mahmutunan@MacBook-Pro exercise2 % ls
Makefile        exercise2       main.c          someFunc.c      someFunc.h
(base) mahmutunan@MacBook-Pro exercise2 % make
gcc -o exercise2withMake main.c someFunc.c -I
(base) mahmutunan@MacBook-Pro exercise2 % ./exercise2withMake
Hello CS330%
(base) mahmutunan@MacBook-Pro exercise2 %
```

Exercise 2

- Let's modify the make file a little bit

Makefile

```
1  CC=gcc
2  CFLAGS=-I
3  DEPS=someFunc.h
4  OBJ=main.o someFunc.o
5
6  %.o: %.c $(DEPS)
7    $(CC) -c -o $@ $< $(CFLAGS)
8
9  exercise2withMake: $(OBJ)
10   $(CC) -o $@ $^ $(CFLAGS)
```

```
(base) mahmutunan@MacBook-Pro exercise2 % ls
Makefile        main.c          someFunc.c      someFunc.h
(base) mahmutunan@MacBook-Pro exercise2 % make
gcc -c -o main.o main.c -I
gcc -c -o someFunc.o someFunc.c -I
gcc -o exercise2withMake main.o someFunc.o -I
(base) mahmutunan@MacBook-Pro exercise2 % ls
Makefile                main.o                  someFunc.o
exercise2withMake       someFunc.c
main.c                  someFunc.h
(base) mahmutunan@MacBook-Pro exercise2 % ./exercise2withMake
Hello CS330%
(base) mahmutunan@MacBook-Pro exercise2 %
```

Exercise - Lab04

- To illustrate the use of make, let us consider adding a new function to measure the time taken by the insertion sort program that we wrote in Lab 2.
- Instead of adding this method to the same file as the insertion sort, let us create a new file and create a header file that has the method prototype.

insertionsort.c

gettime.c | insertionsort.c

```c
#include<stdio.h>
#include<stdlib.h>

/* main method */
int main(int args, char** argv){
  int N, i;
  printf("Please enter number of elements in array: ");
  scanf("%d", &N);

  float arr[N];

  for (i=0; i<N; i++){
    printf("Please enter element %d of array: ", (i+1));
    scanf("%f", &arr[i]);
  }

  printf("Given array is: ");
  printf("[");
  for (i=0; i < N-1; i++){
    printf("%f, ", arr[i]);
  }
  printf("%f]\n", arr[N-1]);

  float temp;
  int currLoc;
  for (i=1; i < N; i++){
    currLoc = i;
    while (currLoc > 0 && arr[currLoc-1] > arr[currLoc]){
      temp = arr[currLoc];
      arr[currLoc] = arr[currLoc-1];
      arr[currLoc-1] = temp;
      currLoc--;
    }
  }

  printf("Sorted array is: ");
  printf("[");
  for (i=0; i < N-1; i++){
    printf("%f, ", arr[i]);
  }
  printf("%f]\n", arr[N-1]);

  return 0;
}
```

gettime.h

gettime.h

```
1  #ifndef _GETTIME_H_
2  #include <stdio.h>
3  #include <sys/time.h>
4  double gettime(void);
5  #endif
```

gettime.c

gettime.c

```c
#include "gettime.h"

double gettime(void) {
    struct timeval tval;
    gettimeofday(&tval, NULL);
    return((double)tval.tv_sec + (double)tval.tv_usec/1000000.0);
}

```

- Note that we can compile the file `gettime.c` separately and link the object file with any other program that uses the ***gettime*** function.
- To use the gettime function in the insertion sort program, we have to include the file `gettime.h` and invoke the ***gettime*** function before and after the call to ***insertionsort*** function

- Here are the steps involved in incrementally compiling and linking these two different files:

```
(base) mahmutunan@MacBook-Pro Desktop % cd lecture11
(base) mahmutunan@MacBook-Pro lecture11 % ls
gettime.c       gettime.h       insertionsort.c
(base) mahmutunan@MacBook-Pro lecture11 % gcc -c gettime.c
(base) mahmutunan@MacBook-Pro lecture11 % gcc -c insertionsort.c
(base) mahmutunan@MacBook-Pro lecture11 % gcc -o insertionsort insertionsort.o gettime.o
(base) mahmutunan@MacBook-Pro lecture11 % ./insertionsort
Please enter number of elements in array: 7
Please enter element 1 of array: 21
Please enter element 2 of array: 22
Please enter element 3 of array: 44
Please enter element 4 of array: 55
Please enter element 5 of array: 32
Please enter element 6 of array: 56
Please enter element 7 of array: 2
Given array is: [21.000000, 22.000000, 44.000000, 55.000000, 32.000000, 56.000000, 2.0000
00]
Sorted array is: [2.000000, 21.000000, 22.000000, 32.000000, 44.000000, 55.000000, 56.000
000]
(base) mahmutunan@MacBook-Pro lecture11 % _
```

`gcc -c` compiles source files without linking

```
$ gcc -c myfile.c
```

This compilation generates *myfile.o* object file.

- Also note that we don't have to recompile `gettime.c` if we are only making changes to the file `insertionsort.c`.
- These dependencies is what we can describe in a make file and let the make utility determine which files what been updated and recompile those files.

makefile

Makefile

```
# Sample Makefile to compile C programs
CC = gcc
CFLAGS = -Wall -g #replace -g with -O when not debugging
DEPS     = gettime.h Makefile
OBJS     = gettime.o insertionsort.o
EXECS    = insertionsort

all:     $(EXECS)

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

insertionsort: $(OBJS)
         $(CC) $(CFLAGS) -o $@ $^

clean:
         /bin/rm -i *.o $(EXECS)
```

- If the Makefile is saved as Makefile or makefile, you can invoke make utility by typing make.
- If you use a different file name other than Makefile or makefile then you have to specify the makefile using the -f option to make. If you type make, you should see the following output:

```
(base) mahmutunan@MacBook-Pro lecture11 % make
gcc -Wall -g  -c -o gettime.o gettime.c
gcc -Wall -g  -c -o insertionsort.o insertionsort.c
gcc -Wall -g  -o insertionsort gettime.o insertionsort.o
```

- If you change gettime.h then you should see all files recompiled and the following output:

```
(base) mahmutunan@MacBook-Pro lecture11 % touch gettime.h
(base) mahmutunan@MacBook-Pro lecture11 % make
gcc -Wall -g  -c -o gettime.o gettime.c
gcc -Wall -g  -c -o insertionsort.o insertionsort.c
gcc -Wall -g  -o insertionsort gettime.o insertionsort.o
```

If you change gettime.c then you should see the following output:

```
(base) mahmutunan@MacBook-Pro lecture11 % touch gettime.c
(base) mahmutunan@MacBook-Pro lecture11 % make
gcc -Wall -g  -c -o gettime.o gettime.c
gcc -Wall -g  -o insertionsort gettime.o insertionsort.o
```

- However, if you only change insertionsort.c you will see the following output:

```
(base) mahmutunan@MacBook-Pro lecture11 % make
gcc -Wall -g  -c -o insertionsort.o insertionsort.c
gcc -Wall -g  -o insertionsort gettime.o insertionsort.o
```

- If you have not modified any files, if you execute make, you will see that following output:

```
(base) mahmutunan@MacBook-Pro lecture11 % make
make: Nothing to be done for `all'.
```