# **EXPERIMENT:2**

**Aim:** Program maintenance using make utility.

**Tools / Apparatus:** Linux OS, make, cc Procedure:

**1. Explain the use of make utility & structure of Makefile.**

# **The make Utility and Makefiles:**

You can use the make utility and makefiles to help automate building of an application with Sun Workshop.

## The Makefile

A file called makefile tells the make utility in a structured manner which source and object files depend on other files. It also defines the commands required to compile and link the files.

1. **The make Utility**

To start the make utility, type the following at a command line:

The make utility looks for a file named makefile or Makefile in the current directory and takes its instructions from that file.

The make utility:

1. Reads makefile to determine all the target files it must process, the files they depend on, and the commands needed to build them

1. Finds the date and time each file was last changed

1. Rebuilds any target file that is older than any of the files it depends on, using the commands from makefile for that target

To make writing a makefile easier, the make utility has default rules that it uses depending on the suffix of a target file. Recognizing the .f suffix, make uses the f77 compiler, passing as arguments any flags specified by the FFLAGS macro, the -c flag, and the name of the source file to be compiled.

**2. Create 2 separate C Program and compile both of them using Makefile.**

**File1: dmx.c**

#include<stdio.h>

int main()

{

printf("This is first c code file.\n");

return 0;

}

**File2: dmx2.c**

#include<stdio.h>

int main()

{

printf("This is second file.\n");

return 0;

}

**Compile using Makefile:**

Makefile:

all: dmx.o dmx2.o dmx.out dmx2.out run

dmx.o: dmx.c

cc dmx.c -c

dmx2.o: dmx2.c

cc dmx2.c -c

dmx.out: dmx.o

cc dmx.o -o dmx.out

dmx2.out: dmx2.o

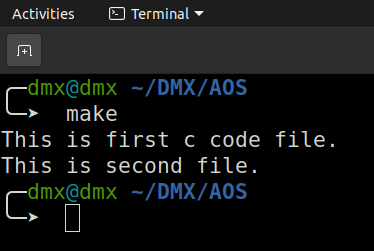
cc dmx2.o -o dmx2.out

run:

@./dmx.out

@./dmx2.out

**Output:**



**3. Write a C program that is spread over multiple files (Dependency) & Execute that program using Makefile.**

**File1: Main.c**

#include<stdio.h>

#include"Connect.h"

int main(){

printf("Main.c is executed.\n");

DMX();

DMX2();

printf("Main.c DMX.c DMX2.c are executed.\n");

return 0;

}

**File2: DMX.c**

#include<stdio.h>

#include"Connect.h"

void DMX()

{

printf("HELLO DMX.\n");

}

**File3: DMX2.c**

#include<stdio.h>

#include"Connect.h"

void DMX2()

{

printf("SECOND FILE RUNNING FOR DMX.\n");

}

**File4: Connect.h**

void DMX();

void DMX2();

**Compile using Makefile:**

Makefilemodule:

all: Main.o DMX.o DMX2.o out run clean

Main.o: Main.c Connect.h

cc Main.c -c

DMX.o: DMX.c Connect.h

cc DMX.c -c

DMX2.o: DMX2.c Connect.h

cc DMX2.c -c

out:

cc Main.o DMX.o DMX2.o -o Final.out

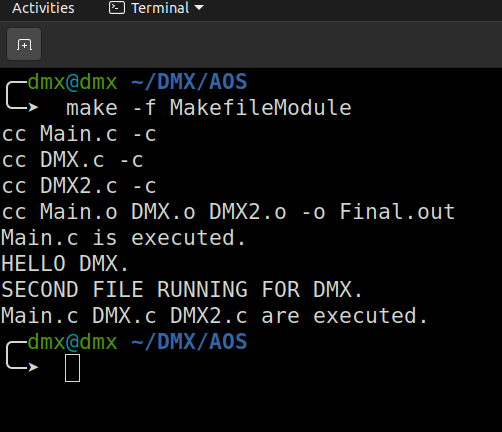
run:

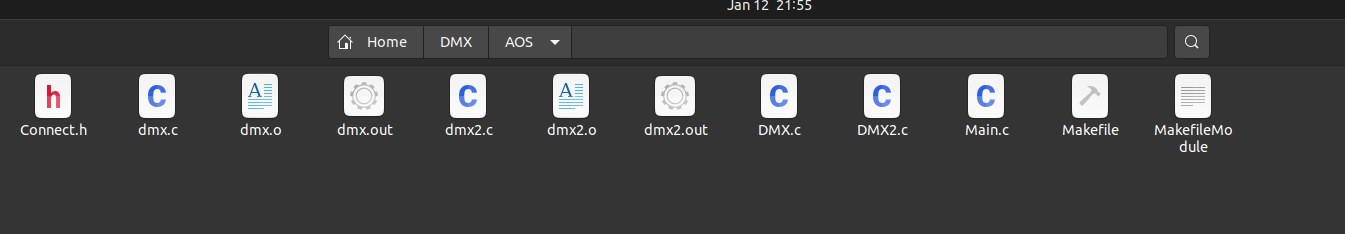
@./Final.out

clean:

@rm Main.o DMX.o DMX2.o Final.out

**Output:**





1. **Use following Makefile for program maintenance. To use make utility, use make command.**

**File1. Makefile**

test.out: test.0

cc test.o -o test.out

test.o: test.c test.h

cc test.c -c

clean:

rm -f \*.o \*.\*~ \*~

run:

./test

**File2. test.c**

#include<stdio.h>

int main()

{

printf(“Hello World.\n In test.c\n”);

return 0;

}

**File3. Makefile2**

clean: run

@rm -f \*.o\* \*.\*~

run: test.out

@./test.out

test.out: test.o

cc test.o -o test.out

test.o: test.c

cc test.c -c

**Output: -**

