GNU Tool Chain

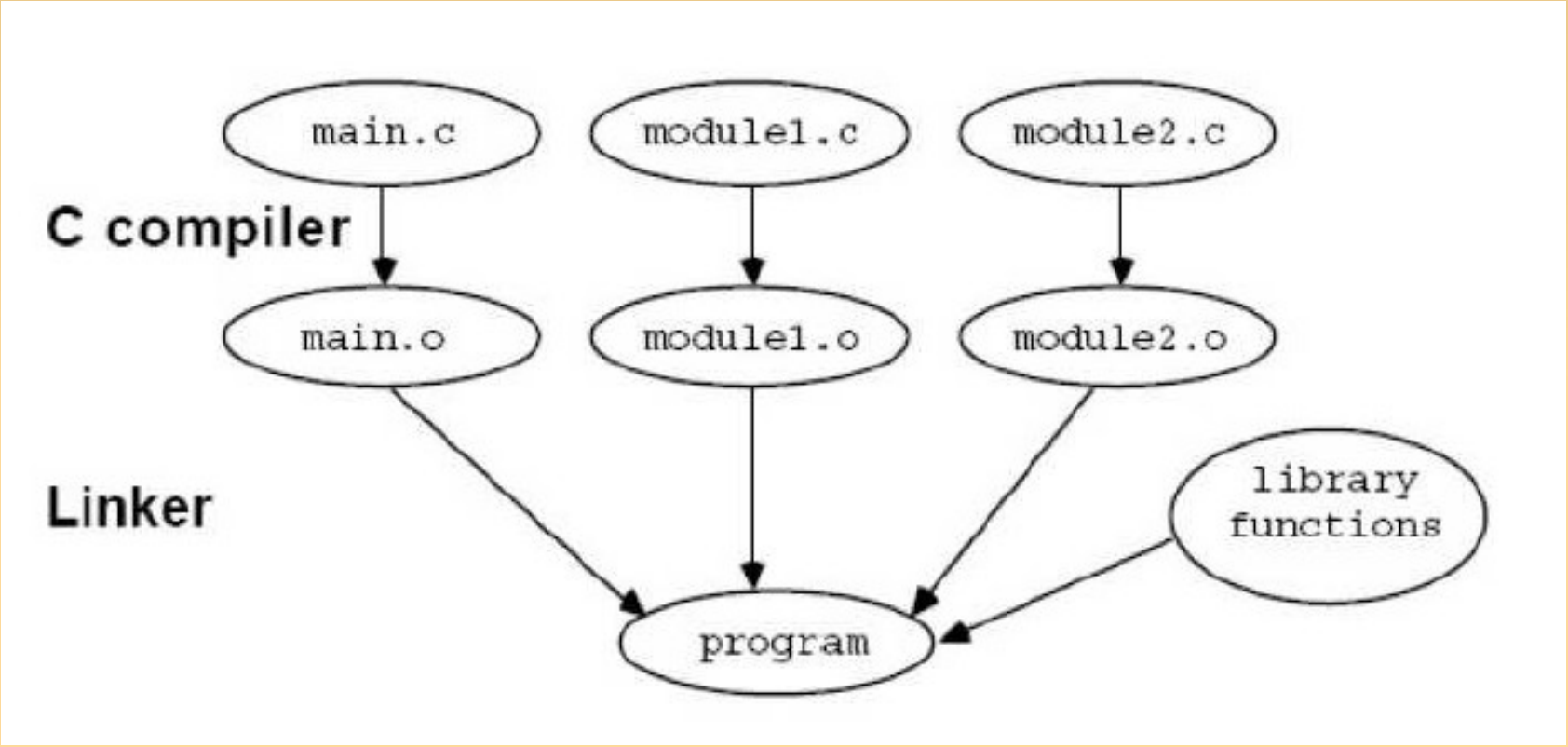
What is GNU

- GNU Not Unix
- Free software organization

GNU Tool Chain

- This is an outcome of the GNU Project.
- Tool Chain includes
  - GNU Make
  - GNU Compiler Collection (GCC)
  - GNU Binutils
  - GNU Debugger
  - GNU build system

GNU Compiler Collection (GCC)



- Advantages:
  - Pretty up-to-date and reliable
  - Available on variety of platforms
  - Free and open source
  - Can compile C, C++
  - It's both compiler and linker
  - Eg: gcc main.c module1.c module2.c –o program

options

- syntax:

  gcc *options files*

- Most common options to gcc are

  -c, -o, -g, -Wall, -I, -L, -l

GCC : Simple Example

```
/* main.c */

#include<stdio.h>

int main()
{
        printf("Hello world\n") ;
        return 0 ;
}
```

GCC : Simple Example

```
/* main.c */

#include<stdio.h>

int main()
{
        printf("Hello world\n") ;
        return 0 ;
}
```

```c
#include<stdio.h>                    /* sub.c */
int sub(int a, int b)
{
        printf("In the subtraction funtcion\n") ;
        return (a-b) ;
}
```

```c
#include <stdio.h>                   /* add.c */

int add(int a, int b)
{
        printf("In the addition function\n") ;
        return(a+b) ;
}
```

```c
// include.h

int add(int, int) ;
int sub(int, int) ;
```

```c
#include<stdio.h>                              /* main.c */
#include "include.h"
int main()
{
        int a = 5, b =4 ;
        printf("Addition of these two numbers is %d\n", add(a,b)) ;
        printf("Subtraction of these two numbers is %d\n",sub(a,b)) ;
        return 0 ;
}
```

GCC – generation of a binary file

Method - I

```
gcc main.c add.c sub.c  -o main
```

GCC – generation of a binary file

Method II

- gcc -c main.c
- gcc -c add.c
- gcc -c sub.c
- gcc main.o add.o sub.o -o main

GCC – generation of a binary file

Method II

- gcc -c main.c
- gcc -c add.c
- gcc -c sub.c
- gcc main.o add.o sub.o -o main

GNU Make

- **Make** is a utility for automatically building executable programs from source code.
- This is one of the dependency-tracking build utility.
- Make will look into the current directory for a file by the name Makefile or makefile

GNU Make

- **Make** is a utility for automatically building executable programs from source code.
- This is one of the dependency-tracking build utility.
- Make will look into the current directory for a file by the name Makefile or makefile

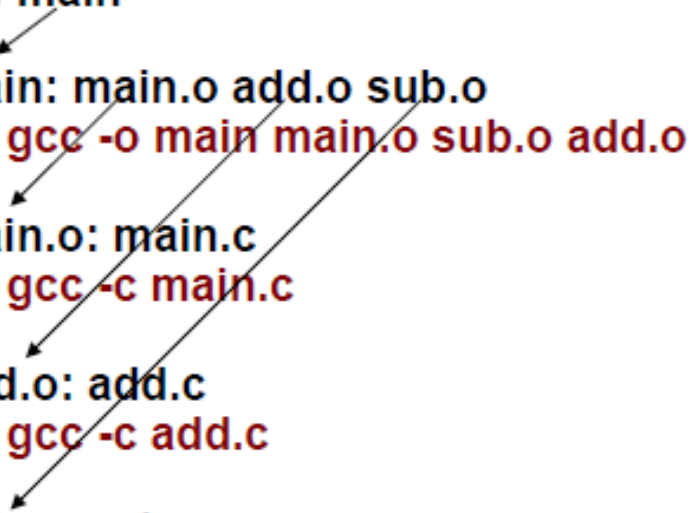
Makefile

- consists of a series of variable definitions and dependency rules
- Dependency rules

```
# simple make file
main: main.o add.o sub.o
        gcc -o main main.o sub.o add.o

main.o: main.c
        gcc -c main.c

add.o: add.c
        gcc -c add.c

sub.o: sub.c
        gcc -c sub.c

clean:
        rm –rf *.o
```

Make file Cont..

```
# simple make file

all: main

main: main.o add.o sub.o
    gcc -o main main.o sub.o add.o

main.o: main.c
    gcc -c main.c

add.o: add.c
    gcc -c add.c

sub.o: sub.c
    gcc -c sub.c

clean:
    rm –rf *.o
```



Make file Cont..

- Variable Definitions
  - Variables are not pre-declared, you just set them with '='
  - Predefined: CC, CFLAGS(-I, -g), LDFLAGS (-I, -L)
  - Eg:CC = gcc

  CFLAGS = -g -I/usr/abc/xyz/include

```
# A part of make file with variables
CC = gcc
main: main.o add.o sub.o
        $(CC) -o main main.o sub.o add.o
```

- Options:
  - -k ignore errors
  - -f <filename>
  - -n to print out what it would have done without actually doing it.

GNU Debugger

- Standard debugger for the GNU Softwares
- Compile the code with –g flag
- Steps in using the gdb
  - Starting the debugger – gdb *program*
  - Running the debugger – run, step, next, finish, return, jump *address*
  - Breakpoints – break, cont
  - Examining the stack – backtrace
  - Examining the source files – list
  - Examining data – print, set variable = expression