

**Experiment 1****Date:21.09.2023****Advanced use of gcc****Aim:**

Advanced use of gcc : Important Options -o, -c, -D, -l, -I, -g, -O, -save-temps, -pg . Write a C program to add two numbers. Read the input from Standard Input and write output to Standard output. Compile and generate output using gcc command and its important options.

**Program**

```
//sum of two numbers
#include<stdio.h>
void main(){
    int sum,a,b;
    printf("Enter first number : ");
    printf("Enter second number : ");
    scanf("%d",&a);
    scanf("%d",&b);
    sum=a+b;
    printf("sum is= %d ",sum);
}
```

**GCC**

GCC is a Linux- Foundation which is usually operated via the command line. It often comes distributed freely with a Linux installation, so if you are running UNIX or a Linux variant you will probably have it on your system. You can invoke GCC on a source code file simply by typing:-

**gcc filename**

The default executable output of GCC is "a.out", which can be run by typing "./a.out". It is also possible to specify a name for the executable file at the command line by using the syntax "-o outputfile", as shown in the following example: -

**gcc filename -o outputfile**

Again, you can run your program with "./outputfile". (The ./ is there to ensure to run the program for the current working directory.)

Note: If you need to use functions from the math library (generally functions from math.h" such as sin or sqrt), then you need to explicitly ask it to link with library with the "-l" flag and the library "m":

**gcc filename -o outputfile -lm****Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc sum.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out
Enter first numbers : 7
```

Enter second numbers : 5  
sum is= 12

## **Important Options in GCC**

### **Option: -o**

To write and build output to output file.

#### **Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc sum.c -o sum_out
```

Here, GCC compiles the sum.c file and generates an executable named sum\_out.

### **Option: -c**

To compile source files to object files without linking.

#### **Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -c sum.c
```

This will generate an object file sum.o that can be linked separately.

### **Option: -D**

To define a preprocessor macro.

#### **Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -D debug=1 sum.c
```

This defines the macro 'DEBUG' with the value 1, which can be used in the source code.

### **Option: -I**

To include a directory of header files.

#### **Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -o sum.c sum_out.c -lm
```

Here, the -lm option links the math library (libm) with the sum.c.

### **Option: -I**

To look in a directory for library files.

#### **Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -o sum.c sum_out.c -I./ads_lab
```

This tells GCC to look for header files in the ads\_lab directory.

**Option: -g**

To debug the program using GDB.

**Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -g sum.c -o sum_out
```

This compiles sum.c with debug information, enabling you to debug the resulting executable.

**Option: -O**

To optimize for code size and execution time.

**Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -O3 -o my_pgm sum.c
```

This compiles sum.c with a high level of optimization.

**Option: -pg**

To enable code profiling.

**Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -pg -o my_pgm source.c
```

This compiles source.c with profiling support, allowing you to use profilers like gprof.

**Option: -save-temps**

To save temporary files generated during program execution.

**Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -save-temps -o my_pgm source.c
```

This will generate intermediate files, like sum.i (pre-processed source) and sum.s (assembly code), in addition to the final executable.

**Experiment 2****Date:21.09.2023****Familiarisation with GDB****Aim**

Familiarisation with gdb: Important Commands - break, run, next, print, display, help. Write a C program 'mul.c' to multiply two numbers. Read the input from Standard Input and write output to Standard output. Compile and generate sum.out which is then debug with gdb and commands.

**Program**

```
//product of two numbers
#include<stdio.h>
void
main(){int
multi,a,b;
printf("Enter first number : ");
printf("Enter second number : ");
scanf("%d",&a);
scanf("%d",&b);
multi=a*b;
printf("product= %d ",multi);
}
```

**Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -g mul.c -o mul_out
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gdb mul_out
GNU gdb (Ubuntu 12.0.90-0ubuntu1) 12.0.90 Copyright (C) 2022 Free Software Foundation,
Inc. License GPLv3+: GNU GPL version 3 or later This is free software: you are free to
change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Type
"show copying" and "show warranty" for details. This GDB was configured as "x86_64-
linux-gnu". Type "show configuration" for configuration details. For bug reporting
instructions, please see: . Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from sum1...
(gdb) run
```

Starting program: /home/mits/Desktop/s1mca/sum1 [Thread debugging using libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

Enter first numbers : 10

Enter second numbers : 3

Product : 30 [Inferior 1 (process 23588) exited normally]  
(gdb) quit

## **Important Commands in GDB**

### **Command: break**

Sets a breakpoint on a particular line.

#### **Output**

(gdb) break mul.c:5

### **Command: run**

Executes the program from start to end.

#### **Output**

(gdb) run

### **Command: next**

Executes the next line of code without diving into functions.

#### **Output**

(gdb) next

### **Command: print**

Displays the value of a variable.

#### **Output**

(gdb) print a

(gdb) a 10

### **Command: display**

Displays the current values of the specified variable after every step.

#### **Output**

(gdb) display a

(gdb)10

**Experiment 3****Date:29.09.2023****Familiarisation with gprof****Aim**

Write a program for finding the sum of two numbers using function. Then profile the executable with gprof.

**Program**

```
//sum of two numbers
#include<stdio.h>
int sum(int a, int b)
{
    int sum=a+b;
    return sum;
}
void
main(){ int
sum,a,b;
printf("Enter first number : ");
printf("Enter second number : ");
scanf("%d",&a);
scanf("%d",&b);
printf("sum is: %d ",sum(a,b));
}
```

**Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc sum.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc ./a.out sum.c
Enter first number:4
Enter first number:6
Sum : 10
```

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -o sum.out -pg sum.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./sum.out
Enter first number:4
Enter first number:6
Sum : 10
```

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gprof ./sum.out gmon.out >
pgm3.txt
```

**Experiment 4****Date: 29.09.2023****Different types of functions****Aim**

4. Write a program for finding the sum of two numbers using different types of functions.

**Algorithm:****main()**

1. Start
2. Declare n1,n2,choice
3. Display choices.
4. Read option choice.
  - a. if choice==1 call sum1().
  - b. if choice==2 input n1 and n2 and call sum2().
  - c. if choice==3 print sum3().
  - d. if choice==3 input n1 and n2 and print sum4().
5. Repeat steps 3 while choice>0&&choice<4
6. Stop.

**sum1()**

1. Start
2. Declare a and b.
3. Read a and b.
4. Print a+b.
5. Exit.

**sum2(int n1, int n2)**

1. Start
2. Print a+b.
3. Exit.

**sum3()**

1. Start
2. Declare a and b.
3. Read a and b.
4. Return a+b.
5. Exit.

**sum4()**

1. Start
2. Return a+b
3. Exit.

**Program**

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

```
void sum1()
```

```
{ int a,b;  
printf("Enter first & second numbers : ");  
scanf("%d %d",&a,&b);  
printf("Sum is: %d",a+b);  
}
```

```
void sum2(int a, int b)
```

```
{ printf("Sum is: %d",a+b);  
}
```

```
int sum3()
```

```
{ int a,b;  
printf("Enter first & second numbers : ");  
scanf("%d %d",&a,&b);  
return a+b;  
}
```

```
int sum4(int a, int b)
```

```
{ return a+b;  
}
```

```
void main()
```

```
{ int  
choice,a,b;  
do  
{  
printf("1. Function without return type and arguments\n2. Function without return type and  
with arguments\n3. Function with return type and without arguments\n4. Function with return  
type and arguments\n5. Exit\nEnter your choice: ");  
scanf("%d", &ch);  
switch(ch){  
case 1: sum1();  
break;  
case 2: printf("Enter first & second numbers : ");  
scanf("%d %d",&a,&b);  
sum2(a,b);  
break;
```



```
case 3: printf("Sum is: %d",sum3());
        break;
case 4: printf("Enter first & second numbers : ");
        scanf("%d %d",&a,&b);
        printf("Sum is: %d",sum4(a,b));
        break;
    }
}while(choice>0&&choice<4);
}
```

### **Output**

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc PGM1.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out PGM1.c
```

```
1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 1
Enter first & second numbers : 9 1
Sum : 10
```

```
1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 2
Enter first & second numbers : 6 3
Sum is: 9
```

```
1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 3
Enter first & second numbers : 2 5
Sum is: 7
```

1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit

Enter your choice: 4

Enter first & second numbers : 4 6

Sum is: 10

1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit

Enter your choice: 5

mits@mits-Lenovo-S510:~/Desktop/s1mca\$