**Example 1:**

C program to find factorial of a number

$ vi factorial.c

# include <stdio.h>

int main()

{

int i, num, j;

printf ("Enter the number: ");

scanf ("%d", &num );

for (i=1; i<num; i++)

j=j\*i;

printf("The factorial of %d is %d\n", num, j);

}

$ cc factorial.c

$ ./a.out

Enter the number: 3

The factorial of 3 is 65532

Steps to debug the code:

1)Compile the C program with debugging option -g

Compile the C program with -g option. This allows the compiler to collect the debugging information.

**$ cc -g factorial.c**

The above command creates a.out file which will be used for debugging

2) Launch gdb

Launch the C debugger (gdb)

**$ gdb a.out**

3) Set up a break point inside C program

Syntax:

break line\_number

Other formats to set a breakpoint are:

* break [file\_name]:line\_number
* break [file\_name]:func\_name

Places break point in the C program, where you suspect errors. While executing the program, the debugger will stop at the break point, and gives you the prompt to debug.So before starting up the program, place the break point in the program.

Example:

break 10

Breakpoint 1 at 0x804846f: file factorial.c, line 10.

4)Execute the C program in gdb debugger

**run**

Starting program: /home/user/a.out

Once you executed the C program, it would execute until the first break point, and give you the prompt for debugging.

Breakpoint 1, main () at factorial.c:10

10 j=j\*i;

5)Printing the variable values inside gdb debugger

Syntax: print {variable}

Examples:

print i

print j

print num

(gdb) p i

$1 = 1

(gdb) p j

$2 = 32767

(gdb) p num

$3 = 3

(gdb)

As we can see, in the factorial.c, we have not initialized the variable j. So, it gets garbage value resulting in a big numbers as factorial values.So we will fix this issue by initializing variable j with 1, compile the C program and execute it again.

**Example 2:**

C program to find average of numbers by reading them from a file.

#include <stdio.h>

int sum=0, val , num =0; // sum should be 0 and num should

be 0.

double ave;

int main()

{

// sum should be the total of the num input values processed

while(scanf("%d\n",&val ) != EOF){

sum += val;

num++;

}

// sum should be the total of the num input values and there is no more input.

if (num > 0){

ave = sum/num; // ave should be the floating point mean of the num input //data values.

printf("Average is %f\n", ave);

}

}

Output:

% a.out

1

2

3

4

Average is 2.000000

This clearly shows that there is an error which is responsible for wrong output.

Steps to debug the code using gdb:

1. Make sure to compile source with the –g switch asserted. In our case, **cc -g average.c** .
2. **Breakpoint**: line in source code at which debugger will pause execution. At breakpoint, can examine values of relevant components of program state. **break** command sets a breakpoint; clear removes the breakpoint.
3. Diagnostic **printf(**)crude, but effective way of getting a snapshot of program state at a given point.
4. Once paused at a breakpoint, use **gdb** print, or display to show variable or expression values. **display** will automatically print values when execution halts at breakpoint.
5. From a breakpoint, may step or next to single step the program. **step** stops after next source line is executed. **next** is also similar, but executes functions without stopping.

(gdb) p ave $1 = 2 // Everything was fine until ave is computed. Integer division was the problem. It was supposed to be in float or double.

(gdb) p (double)sum/(double)num //So in this line we convert the datatype to double $2 = 2.5 //Now after conversion the correct value is obtained

(gdb) c Continuing.