GDB - DEBUGGING EXAMPLE1

Let us write a program to generate a core dump.

```
#include <iostream>
using namespace std;

int divint(int, int);
int main()
{
   int x = 5, y = 2;
   cout << divint(x, y);

   x =3; y = 0;
   cout << divint(x, y);

   return 0;
}

int divint(int a, int b)
{
   return a / b;
}
```

To enable debugging, the program must be compiled with the -g option.

```
$g++ -g crash.cc -o crash
```

NOTE: We are using g++ compiler because we have used C++ source code.

Now, when you run this program on your linux machine, it will produce the following result:

```
Floating point exception (core dumped)
```

You will find a *core* file in your current directory.

Now to debug the problem, start gdb debugger at the command prompt:

```
$gdb crash
# Gdb prints summary information and then the (gdb) prompt

(gdb) r
Program received signal SIGFPE, Arithmetic exception.
0x08048681 in divint(int, int) (a=3, b=0) at crash.cc:21
21        return a / b;

# 'r' runs the program inside the debugger
# In this case the program crashed and gdb prints out some
# relevant information.  In particular, it crashed trying
# to execute line 21 of crash.cc.  The function parameters
# 'a' and 'b' had values 3 and 0 respectively.

(gdb) l
# l is short for 'list'.  Useful for seeing the context of
# the crash, lists code lines near around 21 of crash.cc

(gdb) where
#0  0x08048681 in divint(int, int) (a=3, b=0) at crash.cc:21
#1  0x08048654 in main () at crash.cc:13
# Equivalent to 'bt' or backtrace.  Produces what is known
# as a 'stack trace'.  Read this as follows:  The crash occurred
# in the function divint at line 21 of crash.cc.  This, in turn,
# was called from the function main at line 13 of crash.cc
```

```
(gdb) up
# Move from the default level '0' of the stack trace up one level
# to level 1.

(gdb) list
# list now lists the code lines near line 13 of crash.cc

(gdb) p x
# print the value of the local (to main) variable x
```

In this example, it is fairly obvious that the crash occurs because of the attempt to divide an integer by 0.

To debug a program 'crash' that has crashed and produced a core file named 'core', type the following at the command line:

```
gdb crash core
```

As this is mostly equivalent to starting gdb and typing the 'r' command, all of the commands above could now be used to debug the file.