# LAB TASK:423132

**GNU (GNU's Not Unix)** is a free and open-source operating system.

## Important GNU Tools

| Tool | Purpose |
|---|---|
| **GCC** | GNU Compiler Collection (C, C++, etc.) |
| **GDB** | GNU Debugger (debug C/C++ programs) |
| **Make** | Automate build processes |
| **Bash** | GNU shell (command-line interface) |
| **Coreutils** | Basic Unix-like commands (ls, cp, rm, etc.) |

**GNU/Linux** (commonly called Linux) is the combination of the **Linux kernel** and **GNU software**.

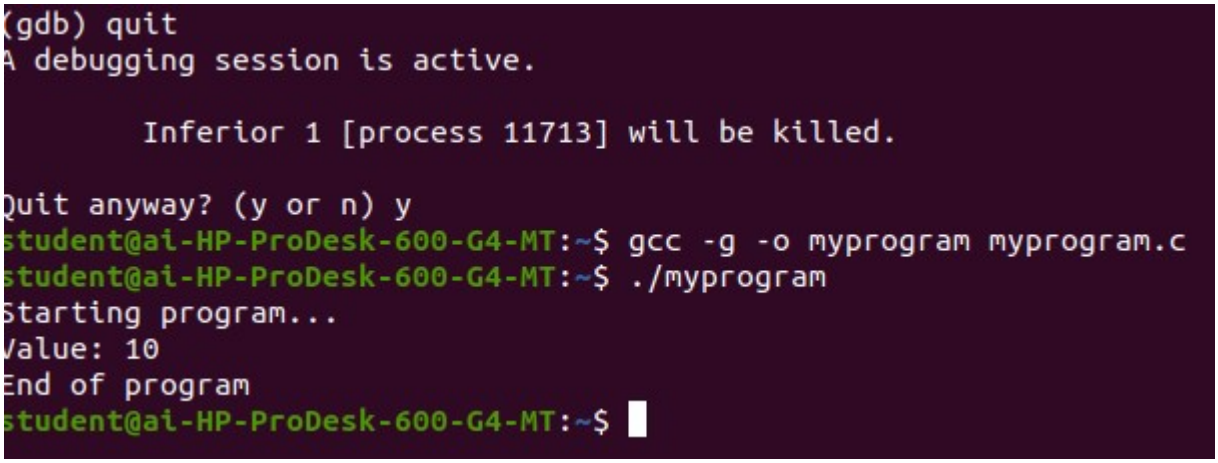
**Core dumping** refers to the process of saving a **snapshot of a program's memory (RAM) and execution state** when it crashes due to a fatal error.
A **crash** occurs when a program unexpectedly stops running due to a **fatal error.**
**Debugging is the process of finding and fixing errors (bugs) in a program. It helps ensure that the program runs correctly and efficiently.**
**,.....you can effectively utilize GDB to debug your C programs, making the process of identifying and fixing bugs more manageable.**

```
student@ai-HP-ProDesk-600-G4-MT:~$ gcc -g -o myprogram myprogram.c
student@ai-HP-ProDesk-600-G4-MT:~$ ./myprogram
Starting program...
Segmentation fault (core dumped)
student@ai-HP-ProDesk-600-G4-MT:~$ gdb ./myprogram
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.2) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
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:
<http://www.gnu.org/software/gdb/bugs/>.
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 ./myprogram...
(gdb) run
Starting program: /home/student/myprogram
Starting program...

Program received signal SIGSEGV, Segmentation fault.
0x000055555555515d in buggy_function () at myprogram.c:5
5           *ptr = 10;        // ❌ Causes segmentation fault
(gdb) bt
#0  0x000055555555515d in buggy_function () at myprogram.c:5
#1  0x0000555555555184 in main () at myprogram.c:10
```

# LAB TASK:423132




**1. Compile Your Program with Debugging Information**
**2. Starting GDB**
**3. Running Your Program Inside GDB**

**IS ALL COVERED ABOVE.BELOW TO BE COVERED.**

**4. Setting Breakpoints**
**5. Stepping Through Code**
**6. Inspecting and Modifying Variables**
**7. Using Watchpoints**
**8. Backtracing**
**9. Handling Core Dumps**

*****A breakpoint stops your program at a specific line so you can inspect what's happening.**
*****Step-by-step execution helps you analyze how your program behaves.**

# LAB TASK:423132

```
student@ai-HP-ProDesk-600-G4-MT:~$ gdb ./myprogram
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.2) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
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:
<http://www.gnu.org/software/gdb/bugs/>.
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 ./myprogram...
(gdb) break 10
Breakpoint 1 at 0x11d8: file myprogram.c, line 10.
(gdb) run
Starting program: /home/student/myprogram
Starting program...

Breakpoint 1, buggy_function () at myprogram.c:10
10          *ptr = 10;  // ✅ Safe operation
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00005555555551d8 in buggy_function at myprogram.c:10
        breakpoint already hit 1 time
(gdb) delete 1
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb)
```

**5)*next** # Go to next line
***step** # Step into function
***continue** # Run until next breakpoint

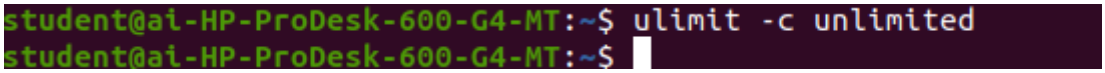
6)**print x**
MODIFYING.
**set var x = 20**
**print x # Now x = 20**

7) A watchpoint stops execution when a variable changes.
If x changes, GDB pauses execution!
**watch x**
**info watchpoints** # Show active watchpoints
**delete 1** # Remove watchpoint #1

8)Find Where the Error Happened
---to see the function call history.
**bt**

# LAB TASK:423132

**9)Enable Core Dumps**




## Summary of Commands

| Action | Command |
|---|---|
| **Set breakpoint** | `break 10` |
| **Step over a line** | `next` |
| **Step into a function** | `step` |
| **Continue execution** | `continue` |
| **Print variable** | `print x` |
| **Modify variable** | `set var x = 20` |
| **Set watchpoint** | `watch x` |
| **View backtrace** | `bt` |
| **Enable core dumps** | `ulimit -c unlimited` |
| **Analyze core dump** | `gdb ./myprogram core.1234` |