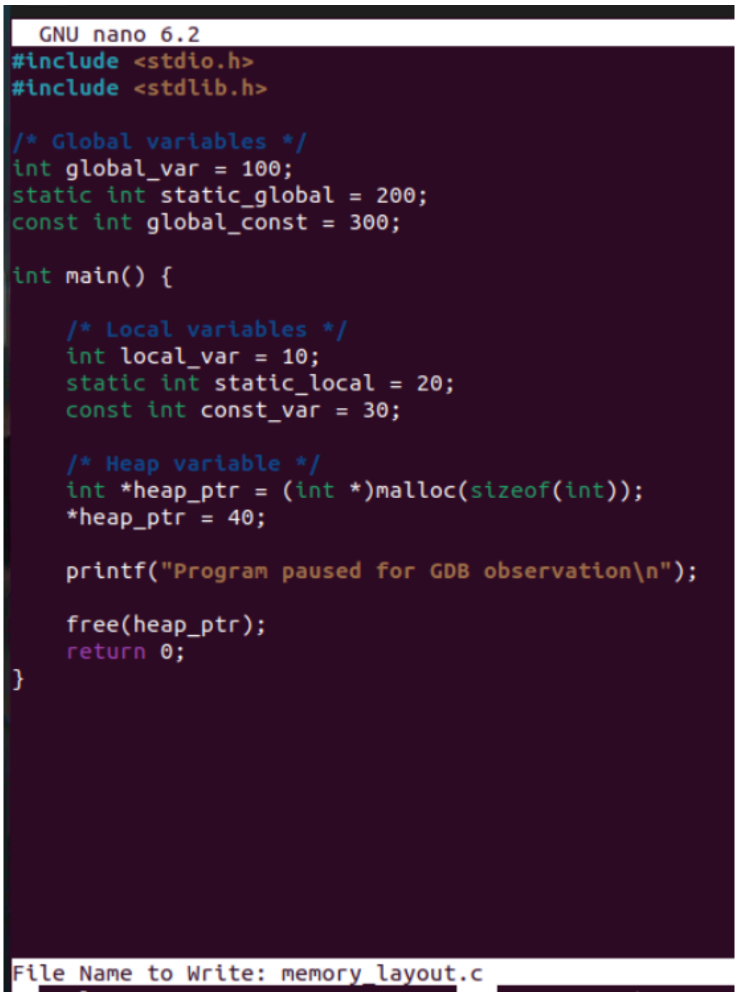
Task 1A: Observe Layout of Data in Memory using GDB

Objective:
To understand how different types of variables are laid out in memory by using GDB and observing stack, heap, data, and text segments.

1. C Program Used

Below is the C program used for this task. Insert a screenshot of the code editor showing this program.

Screenshot :



```
  GNU nano 6.2
#include <stdio.h>
#include <stdlib.h>

/* Global variables */
int global_var = 100;
static int static_global = 200;
const int global_const = 300;

int main() {

    /* Local variables */
    int local_var = 10;
    static int static_local = 20;
    const int const_var = 30;

    /* Heap variable */
    int *heap_ptr = (int *)malloc(sizeof(int));
    *heap_ptr = 40;

    printf("Program paused for GDB observation\n");

    free(heap_ptr);
    return 0;
}
```

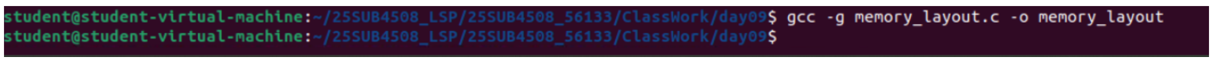
```
File Name to Write: memory_layout.c
```

2. Compilation with Debug Symbols

The program is compiled using gcc with the -g flag to include debug symbols.

Command used:
gcc -g memory_layout.c -o memory_layout

Screenshot :



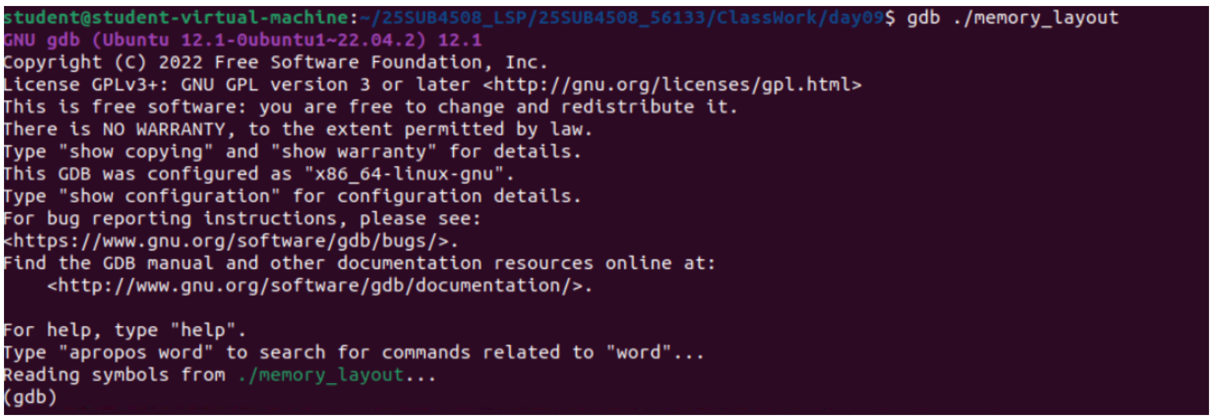
```
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day09$ gcc -g memory_layout.c -o memory_layout
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day09$
```

3. Starting GDB Session

GDB is started with the compiled executable.

Command used:
gdb ./memory_layout

Screenshot :



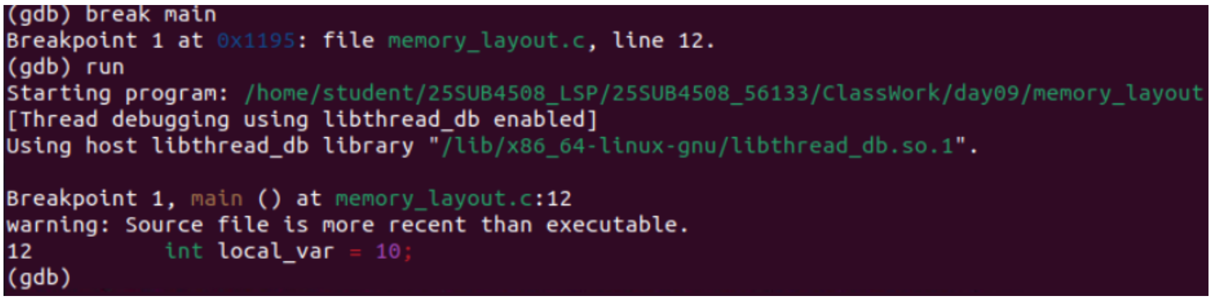
```
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day09$ gdb ./memory_layout
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you 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:
<https://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 ./memory_layout...
(gdb)
```

4. Breakpoint at main()

A breakpoint is set at the main function and program execution is started.

Commands used:
break main
run

Screenshot :



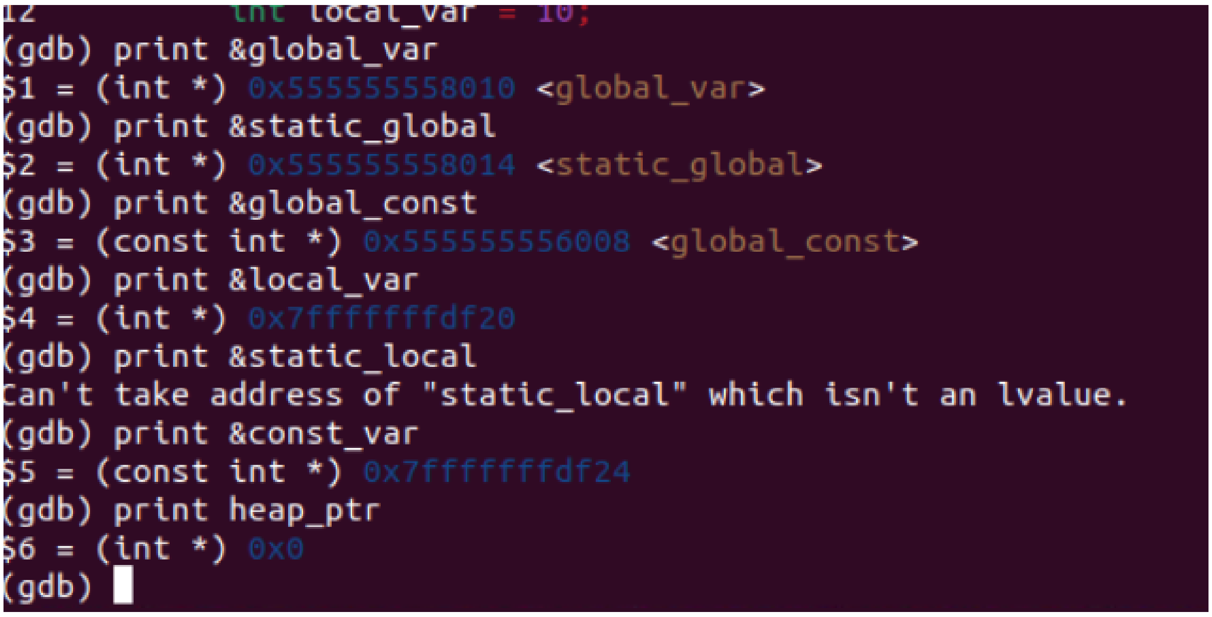
```
(gdb) break main
Breakpoint 1 at 0x1195: file memory_layout.c, line 12.
(gdb) run
Starting program: /home/student/25SUB4508_LSP/25SUB4508_56133/ClassWork/day09/memory_layout
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at memory_layout.c:12
warning: Source file is more recent than executable.
12          int local_var = 10;
(gdb)
```

5. Observing Variable Addresses

The addresses of different variables are printed using the 'print &variable' command.

Commands used:
print &global_var
print &static_global
print &local_var
print &static_local
print &const_var
print heap_ptr

Screenshot :



```
            int local_var = 10;
(gdb) print &global_var
$1 = (int *) 0x555555558010 <global_var>
(gdb) print &static_global
$2 = (int *) 0x555555558014 <static_global>
(gdb) print &global_const
$3 = (const int *) 0x555555556008 <global_const>
(gdb) print &local_var
$4 = (int *) 0x7fffffffdf20
(gdb) print &static_local
Can't take address of "static_local" which isn't an lvalue.
(gdb) print &const_var
$5 = (const int *) 0x7fffffffdf24
(gdb) print heap_ptr
$6 = (int *) 0x0
(gdb) 
```

6. Understanding Memory Segments

Based on the observed addresses:
- Stack: Local variables
- Heap: Dynamically allocated memory (malloc)
- Data Segment: Global and static variables
- Read-only Data: Constants