Task 1D: Pointer Initialized with String Constant – GDB Memory Observation

Objective:
To define a pointer initialized with a string constant and use GDB to observe:
1. The address where the pointer variable itself is stored
2. The address stored inside the pointer (address of string literal)
3. The memory segments where the pointer and string literal reside

This task helps understand storage classes such as stack, read-only data section, and the behavior of string literals in C.

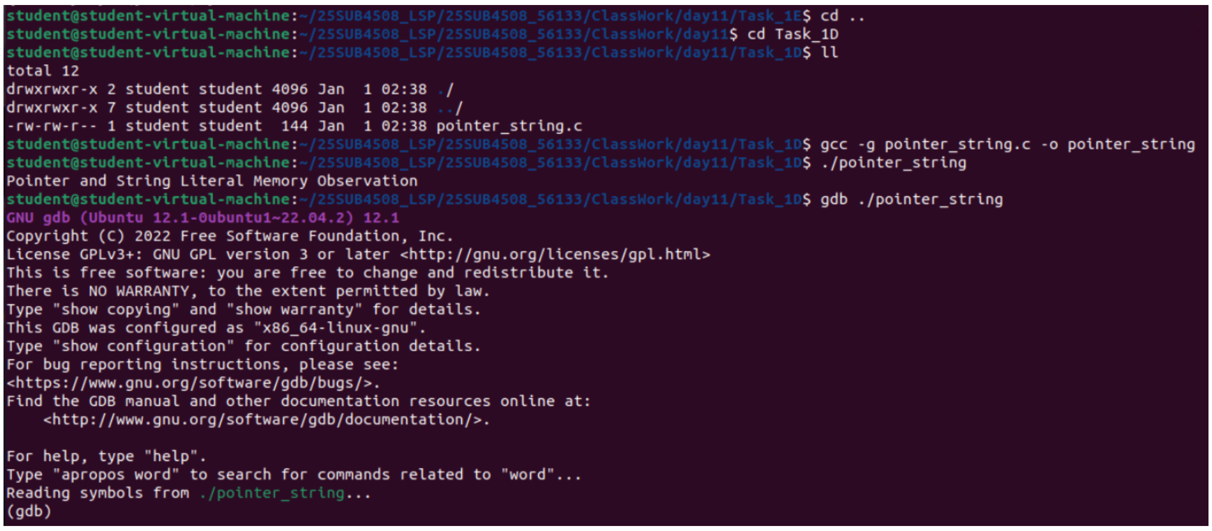
1. Program Code

```
#include <stdio.h>

 int main() {
        char *ptr = "HELLO_WORLD";

        printf("Pointer and String Literal Memory Observation\n");
        return 0;
 }
```

2. Compilation Instructions

Compile the program with debug symbols enabled:

```
 gcc -g pointer_string.c -o pointer_string
```

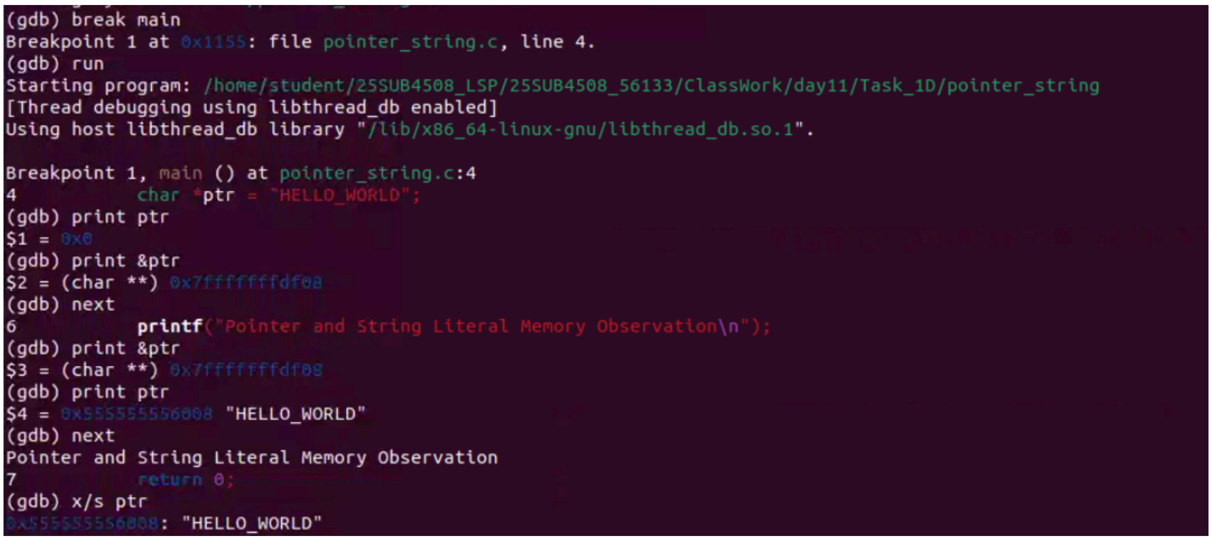


```
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1E$ cd ..
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11$ cd Task_1D
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1D$ ll
total 12
drwxrwxr-x 2 student student 4096 Jan  1 02:38 ./
drwxrwxr-x 7 student student 4096 Jan  1 02:38 ../
-rw-rw-r-- 1 student student  144 Jan  1 02:38 pointer_string.c
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1D$ gcc -g pointer_string.c -o pointer_string
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1D$ ./pointer_string
Pointer and String Literal Memory Observation
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1D$ gdb ./pointer_string
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 ./pointer_string...
(gdb)
```

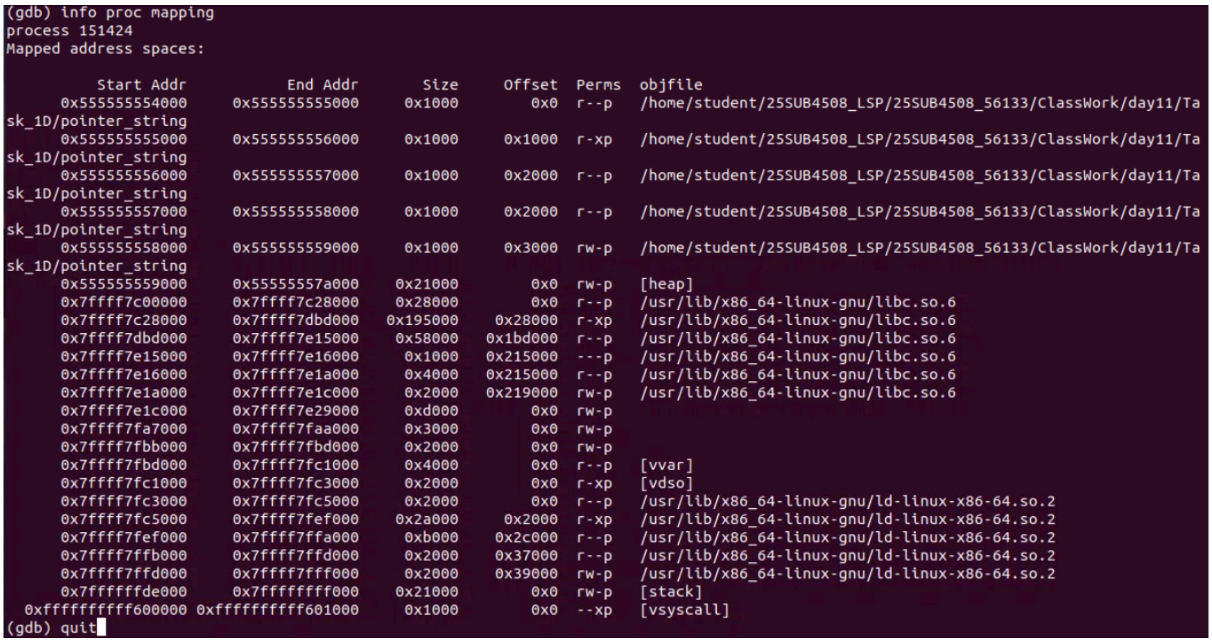
3. GDB Commands Used

break main
 run
 print ptr
 print &ptr
 x/s ptr
 info proc mappings

4. GDB Output



```
(gdb) break main
Breakpoint 1 at 0x1155: file pointer_string.c, line 4.
(gdb) run
Starting program: /home/student/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1D/pointer_string
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at pointer_string.c:4
4           char *ptr = "HELLO_WORLD";
(gdb) print ptr
$1 = 0x0
(gdb) print &ptr
$2 = (char **) 0x7fffffffdf08
(gdb) next
6           printf("Pointer and String Literal Memory Observation\n");
(gdb) print &ptr
$3 = (char **) 0x7fffffffdf08
(gdb) print ptr
$4 = 0x555555556008 "HELLO_WORLD"
(gdb) next
Pointer and String Literal Memory Observation
7           return 0;
(gdb) x/s ptr
0x555555556008: "HELLO_WORLD"
```



```
(gdb) info proc mapping
process 151424
Mapped address spaces:

          Start Addr           End Addr       Size     Offset  Perms  objfile
      0x555555554000     0x555555555000     0x1000        0x0  r--p   /home/student/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Ta
sk_1D/pointer_string
      0x555555555000     0x555555556000     0x1000     0x1000  r-xp   /home/student/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Ta
sk_1D/pointer_string
      0x555555556000     0x555555557000     0x1000     0x2000  r--p   /home/student/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Ta
sk_1D/pointer_string
      0x555555557000     0x555555558000     0x1000     0x2000  r--p   /home/student/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Ta
sk_1D/pointer_string
      0x555555558000     0x555555559000     0x1000     0x3000  rw-p   /home/student/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Ta
sk_1D/pointer_string
      0x555555559000     0x55555557a000    0x21000        0x0  rw-p   [heap]
      0x7ffff7c00000     0x7ffff7c28000    0x28000        0x0  r--p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0x7ffff7c28000     0x7ffff7dbd000   0x195000    0x28000  r-xp   /usr/lib/x86_64-linux-gnu/libc.so.6
      0x7ffff7dbd000     0x7ffff7e15000    0x58000   0x1bd000  r--p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0x7ffff7e15000     0x7ffff7e16000     0x1000   0x215000  ---p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0x7ffff7e16000     0x7ffff7e1a000     0x4000   0x215000  r--p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0x7ffff7e1a000     0x7ffff7e1c000     0x2000   0x219000  rw-p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0x7ffff7e1c000     0x7ffff7e29000     0xd000        0x0  rw-p
      0x7ffff7fa7000     0x7ffff7faa000     0x3000        0x0  rw-p
      0x7ffff7fbb000     0x7ffff7fbd000     0x2000        0x0  rw-p
      0x7ffff7fbd000     0x7ffff7fc1000     0x4000        0x0  r--p   [vvar]
      0x7ffff7fc1000     0x7ffff7fc3000     0x2000        0x0  r-xp   [vdso]
      0x7ffff7fc3000     0x7ffff7fc5000     0x2000        0x0  r--p   /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
      0x7ffff7fc5000     0x7ffff7fef000    0x2a000     0x2000  r-xp   /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
      0x7ffff7fef000     0x7ffff7ffa000     0xb000    0x2c000  r--p   /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
      0x7ffff7ffb000     0x7ffff7ffd000     0x2000    0x37000  r--p   /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
      0x7ffff7ffd000     0x7ffff7fff000     0x2000    0x39000  rw-p   /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
      0x7ffffffde000     0x7ffffffff000    0x21000        0x0  rw-p   [stack]
  0xffffffffff600000 0xffffffffff601000     0x1000        0x0  --xp   [vsyscall]
(gdb) quit
```

5. Observations & Explanation

1. The pointer variable 'ptr' is a local variable and is stored in the stack segment.
   This is confirmed by its address being close to the stack range (0x7ffffff...).

2. The value stored inside the pointer points to a string literal.
   String literals are stored in the read-only data section (.rodata).

3. The pointer and the string literal reside in different memory segments.
   This separation prevents modification of string literals, which can lead to segmentation faults if attempted.

6. Memory Segment Summary

Pointer Variable (ptr): Stack Segment
String Literal ("HELLO_WORLD"): Read-Only Data Segment (.rodata)

7. Conclusion

This experiment demonstrates that a pointer initialized with a string constant stores the pointer itself in stack memory while the actual string literal resides in the read-only data section. Understanding this distinction is crucial for safe memory handling in C programming.