4. Listing variables and examining their values.

Code :

```c
#include <stdio.h>


void func(int a, int b, char* c) {
    int arr[10] = {1, 2, 3, 4, 5};
    printf("%d %d %s\n", a, b, c);
}

int main(int argc, char** argv) {
    int a = 5;
    func(10, 20, "test");
    return 0;
}

```

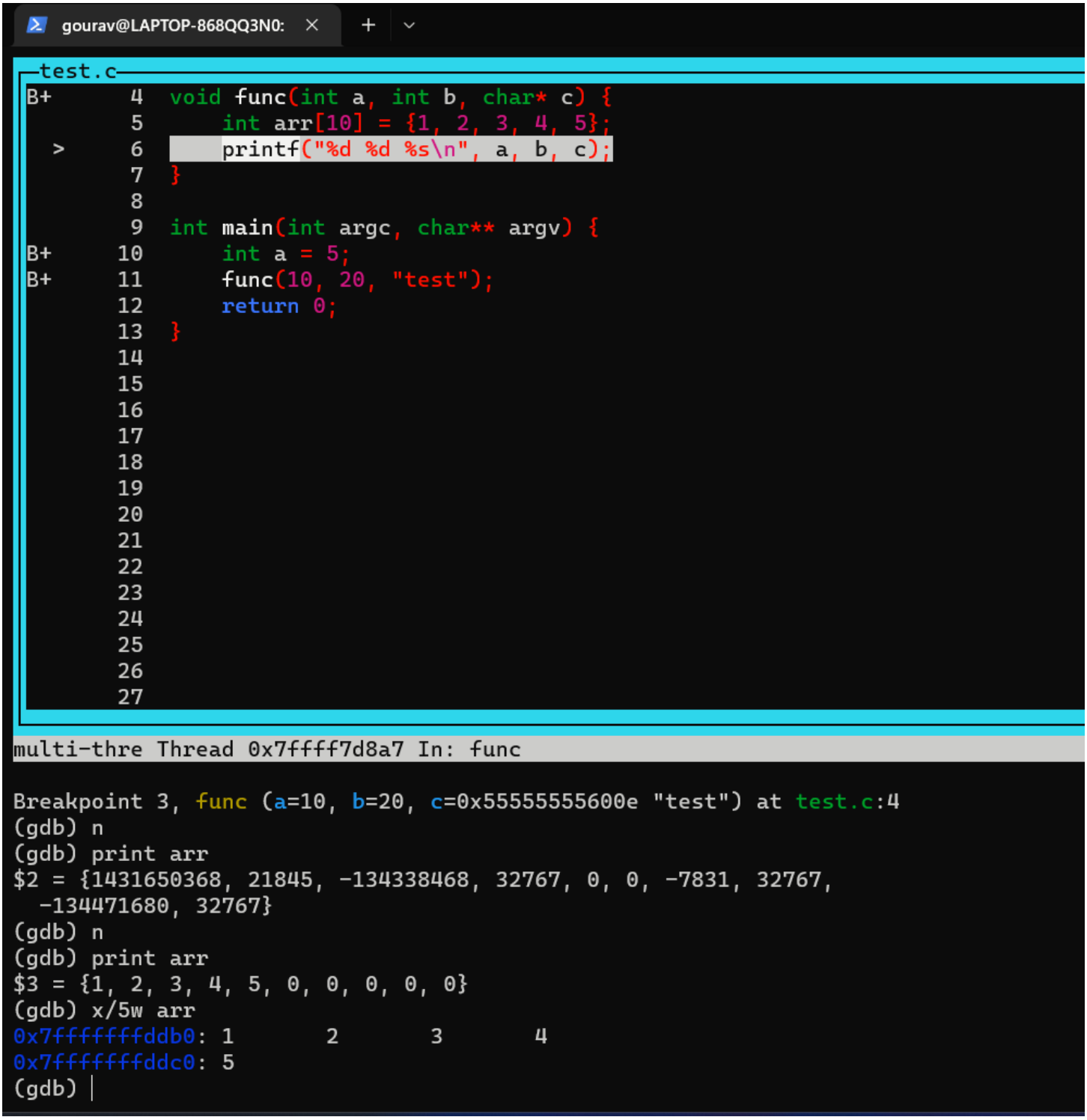
```
gourav@LAPTOP-868QQ3N0:

test.c
        4  void func(int a, int b, char* c) {
        5      int arr[10] = {1, 2, 3, 4, 5};
        6      printf("%d %d %s\n", a, b, c);
        7  }
        8
        9  int main(int argc, char** argv) {
B+     10      int a = 5;
B+>    11      func(10, 20, "test");
       12      return 0;
       13  }

    0x555555555169 <func>           endbr64
    0x55555555516d <func+4>         push   %rbp
    0x55555555516e <func+5>         mov    %rsp,%rbp
    0x555555555171 <func+8>         sub    $0x40,%rsp
    0x555555555175 <func+12>        mov    %edi,-0x34(%rbp)
    0x555555555178 <func+15>        mov    %esi,-0x38(%rbp)
    0x55555555517b <func+18>        mov    %rdx,-0x40(%rbp)
    0x55555555517f <func+22>        mov    %fs:0x28,%rax
    0x555555555188 <func+31>        mov    %rax,-0x8(%rbp)
    0x55555555518c <func+35>        xor    %eax,%eax
    0x55555555518e <func+37>        movq   $0x0,-0x30(%rbp)
    0x555555555196 <func+45>        movq   $0x0,-0x28(%rbp)

multi-thre Thread 0x7ffff7d8a7 In: main                L11   PC: 0x55555555522a
(gdb) print a
$1 = 21845
(gdb) break 11
Breakpoint 2 at 0x55555555522a: file test.c, line 11.
(gdb) continue
Continuing.

Breakpoint 2, main (argc=1, argv=0x7fffffffdf28) at test.c:11
(gdb) print a
$2 = 5
(gdb) info locals
a = 5
(gdb)
```

- **`info locals`:** The **`info locals`** command is used to display the values of local variables in the current scope. When you are stopped at a breakpoint or when your program is running in the debugger, we can use **`info locals`** to see the values of all local variables in the current function.
- In this example, **`info locals`** was used to display the value of the local variable `a`. Here **`info locals`** gives the output **`a=5`**.

- **`info locals`** will only show the values of local variables in the current scope. If we have nested scopes or if we are using function calls, we may need to use the `up` and `down` commands to change the current scope and display the values of local variables in other scopes.

- Note that the **`info locals`** command does not display the information about the function arguments

5. Printing content of an array or contiguous memory



- **`x/5w arr`:** In GDB, the x command is used to examine memory. The **`/5w`** in the **`x/5w arr`** command is a format specifier that tells GDB how to format the output.

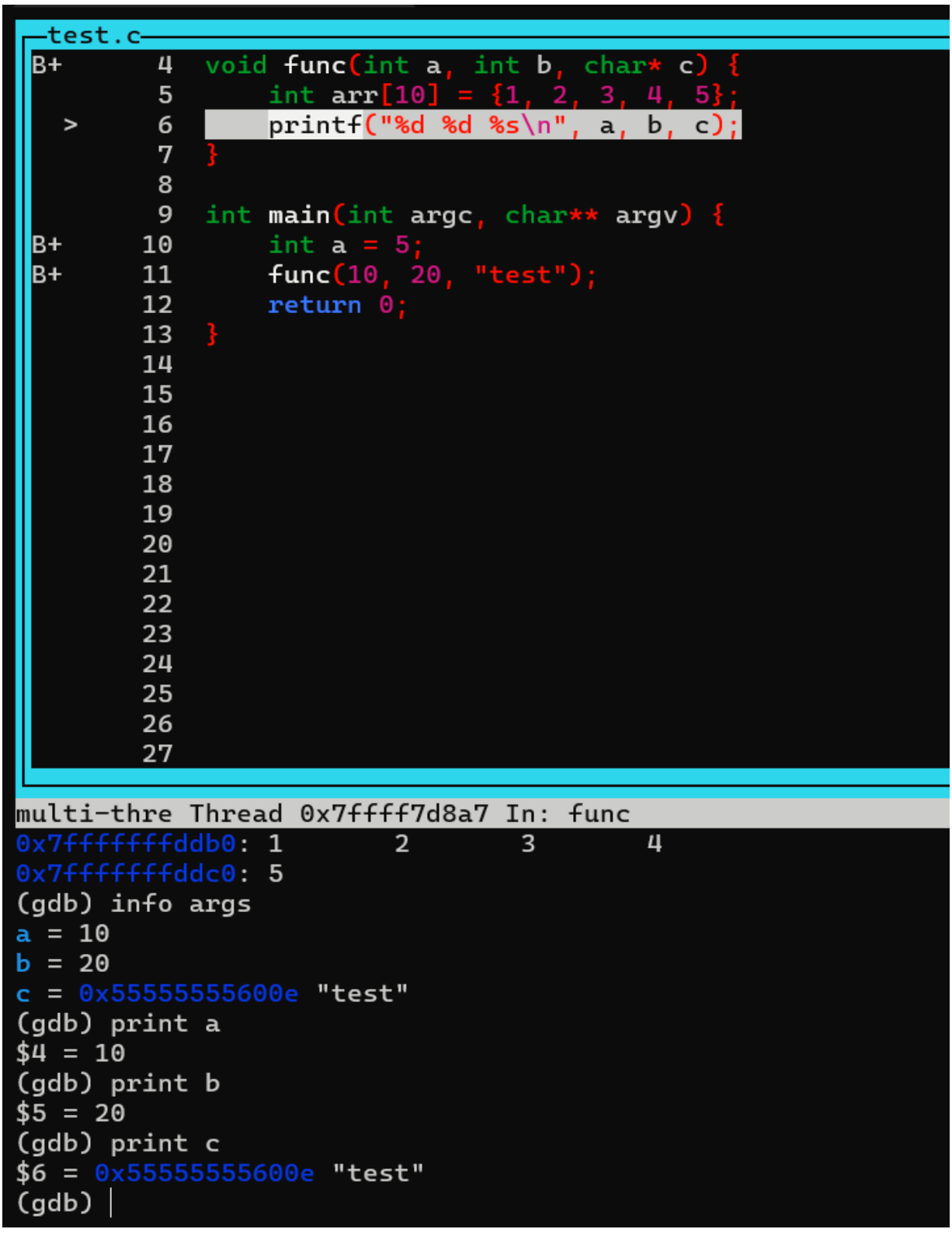
- The **`/5w`** format specifier is used to display 5 memory words in hexadecimal format. A word is typically 4 bytes on most systems, so this command will display the values of 20 bytes of memory.

- The **`arr`** argument specifies the memory address to start displaying from, which is name of array. In this example, **`x/5w arr`** was used to display the values of the first 5 elements of the **`arr`** array. The output shows that the values of the elements are stored in consecutive memory locations, starting from the address :**`0x7fffffffddb0.`**

- If we gives the command **`print arr`** then it will gives all the array elements including 0.

- But if we want only particular elements to print then we have to use **`x/nw arr`** were n is number of element we want to print **.**

6. Printing function arguments



- **`info args:`** the **`info args`** command is used to display the arguments of the current function. When we are stopped at a breakpoint or when our program is running in the debugger.

- we can use **`info args`** to see the values of all arguments passed to the current function.

- **`info args`** will only show the values of arguments in the current function. If we are calling other functions, we need to set breakpoints in those functions and use **`info args`** to see the values of arguments passed to those functions.

- Here in our program if we give the command **`info args`** `it gives the values of arguments` **`a=10, b=20 and c=0x5555555600e “test”.`**