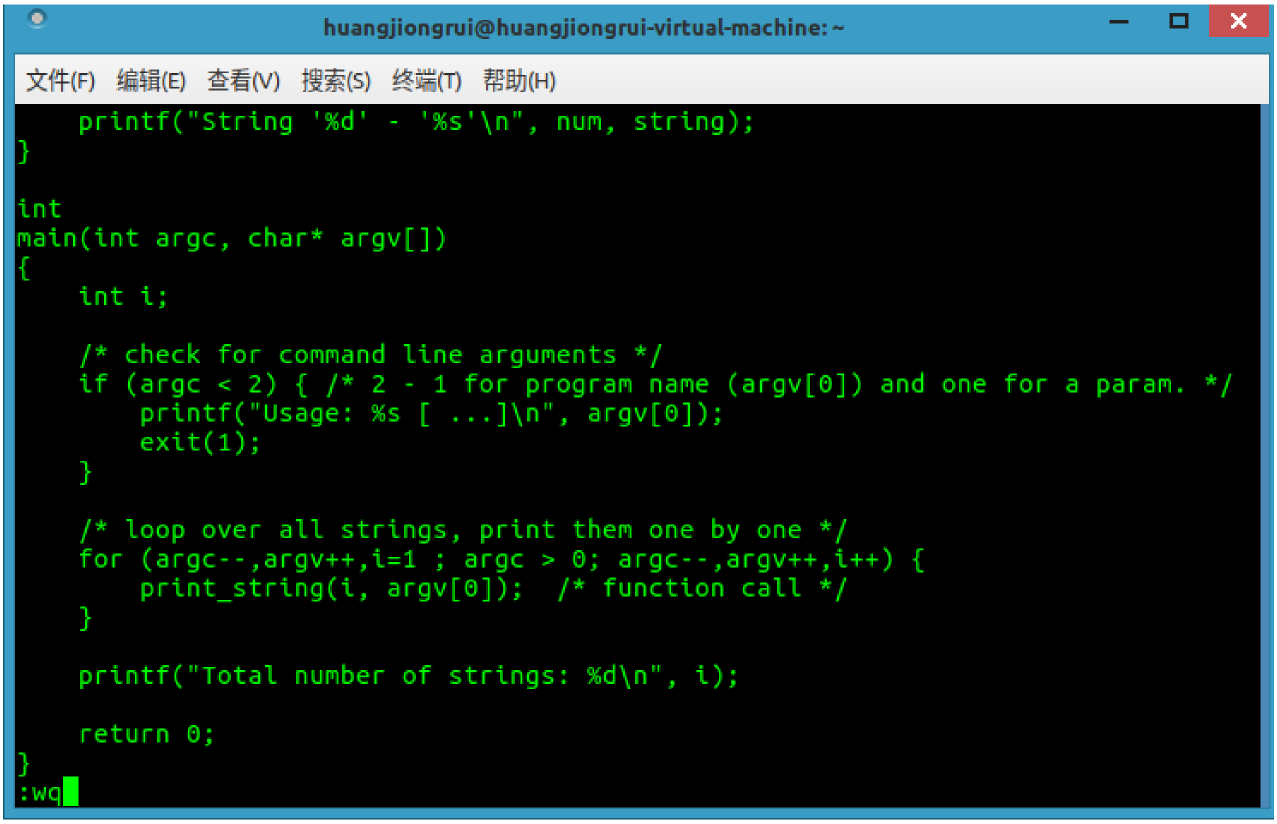
Lab 2.2 Running a Hello World Program in C using GCC

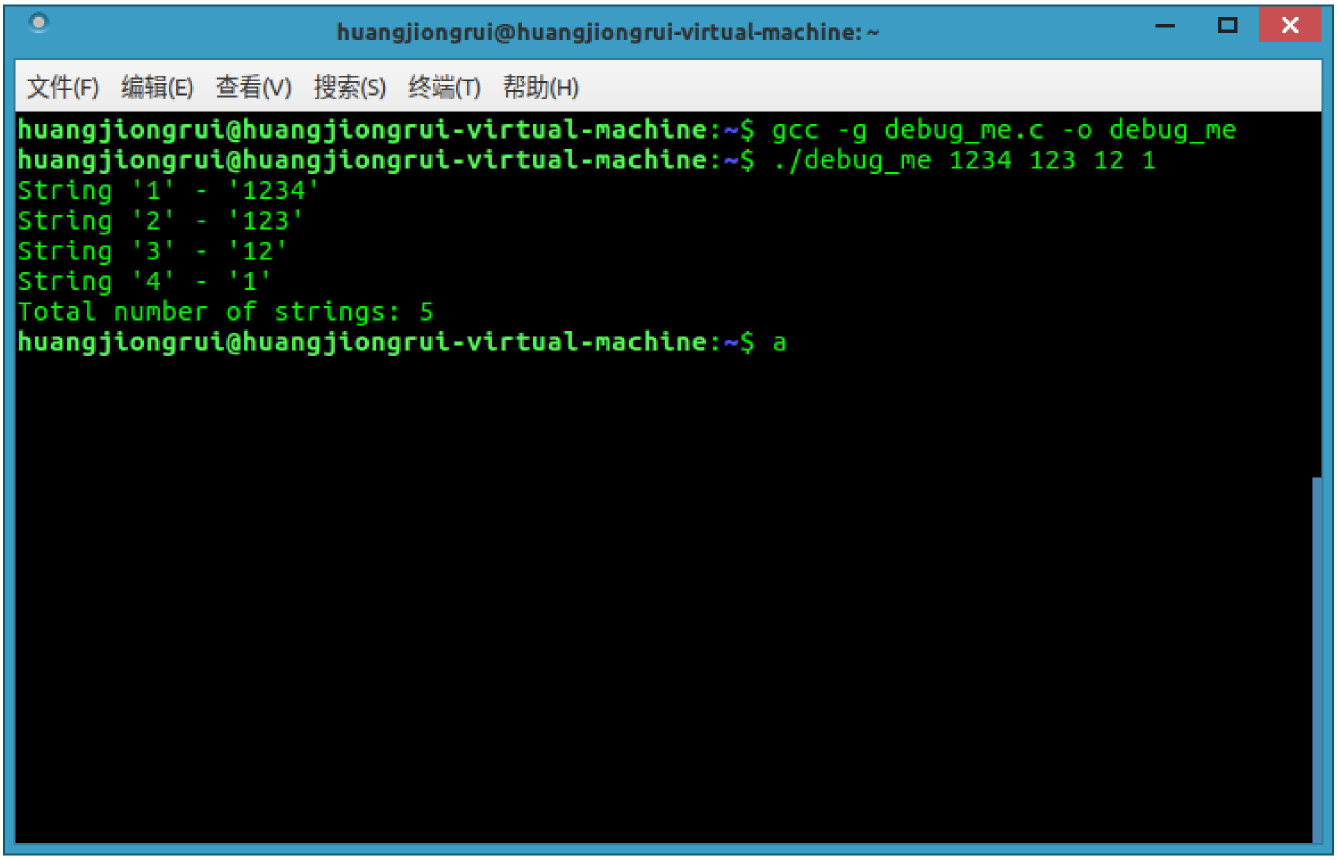
1. 用vi编辑需要gdb的点c源文件，保存后退出。



huangjiongrui@huangjiongrui-virtual-machine: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

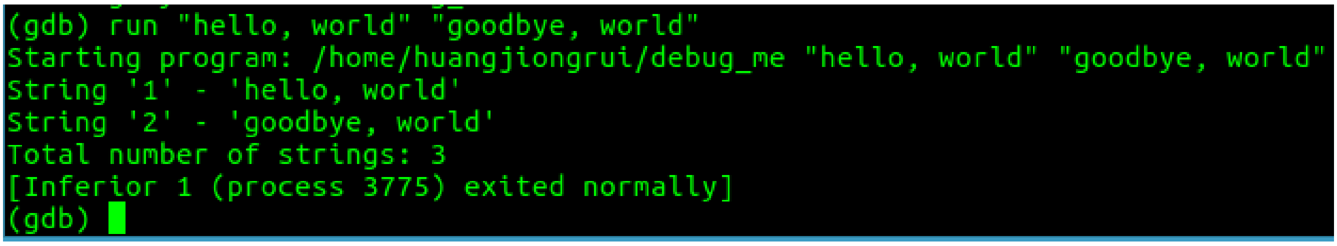
```
    printf("String '%d' - '%s'\n", num, string);
}

int
main(int argc, char* argv[])
{
    int i;

    /* check for command line arguments */
    if (argc < 2) { /* 2 - 1 for program name (argv[0]) and one for a param. */
        printf("Usage: %s [ ...]\n", argv[0]);
        exit(1);
    }

    /* loop over all strings, print them one by one */
    for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
        print_string(i, argv[0]);  /* function call */
    }

    printf("Total number of strings: %d\n", i);

    return 0;
}
:wq
```

2. 使用gcc对c文件编译，之后运行，可以看到此程序输出了输入的参数。


huangjiongrui@huangjiongrui-virtual-machine: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

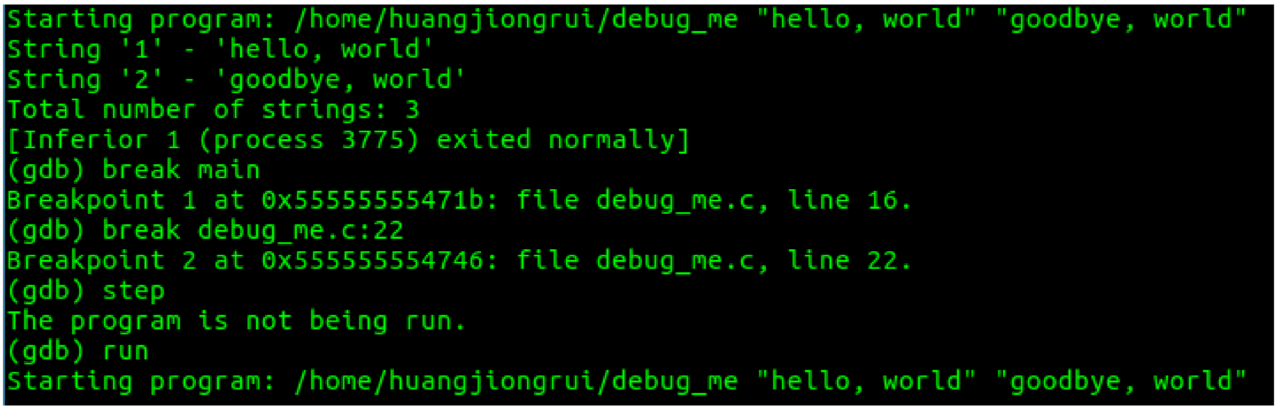
```
huangjiongrui@huangjiongrui-virtual-machine:~$ gcc -g debug_me.c -o debug_me
huangjiongrui@huangjiongrui-virtual-machine:~$ ./debug_me 1234 123 12 1
String '1' - '1234'
String '2' - '123'
String '3' - '12'
String '4' - '1'
Total number of strings: 5
huangjiongrui@huangjiongrui-virtual-machine:~$ a
```

3. 在gdb下运行程序：

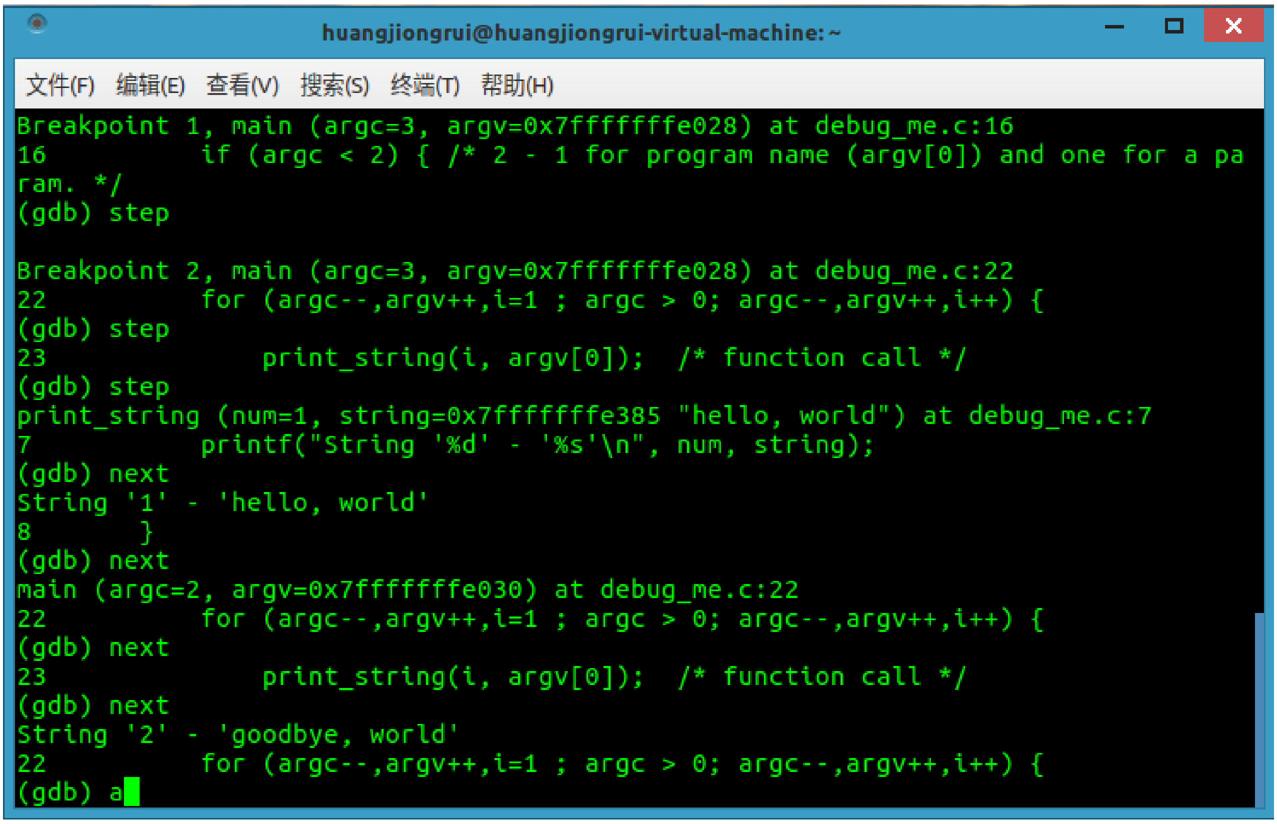


```
(gdb) run "hello, world" "goodbye, world"
Starting program: /home/huangjiongrui/debug_me "hello, world" "goodbye, world"
String '1' - 'hello, world'
String '2' - 'goodbye, world'
Total number of strings: 3
[Inferior 1 (process 3775) exited normally]
(gdb) 
```

4. 在main函数和for循环上设置断点，并使用next和step进行单步的调试。调试中发现step是步入，程序会执行包括自己定义地函数在内的代码并显示出来；next是下一步，对于自己定义的函数不会显示具体信息，而是直接返回程序执行的结果

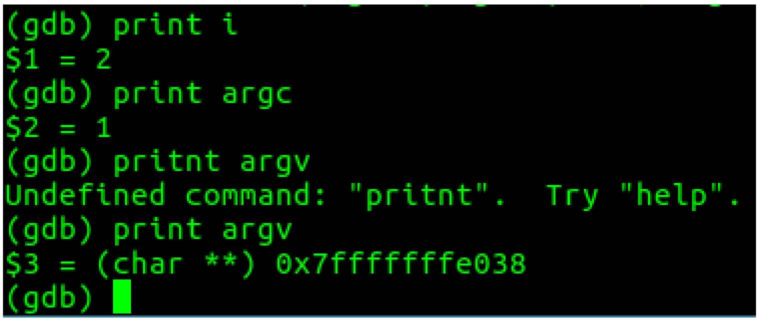


```
Starting program: /home/huangjiongrui/debug_me "hello, world" "goodbye, world"
String '1' - 'hello, world'
String '2' - 'goodbye, world'
Total number of strings: 3
[Inferior 1 (process 3775) exited normally]
(gdb) break main
Breakpoint 1 at 0x55555555471b: file debug_me.c, line 16.
(gdb) break debug_me.c:22
Breakpoint 2 at 0x555555554746: file debug_me.c, line 22.
(gdb) step
The program is not being run.
(gdb) run
Starting program: /home/huangjiongrui/debug_me "hello, world" "goodbye, world"
```



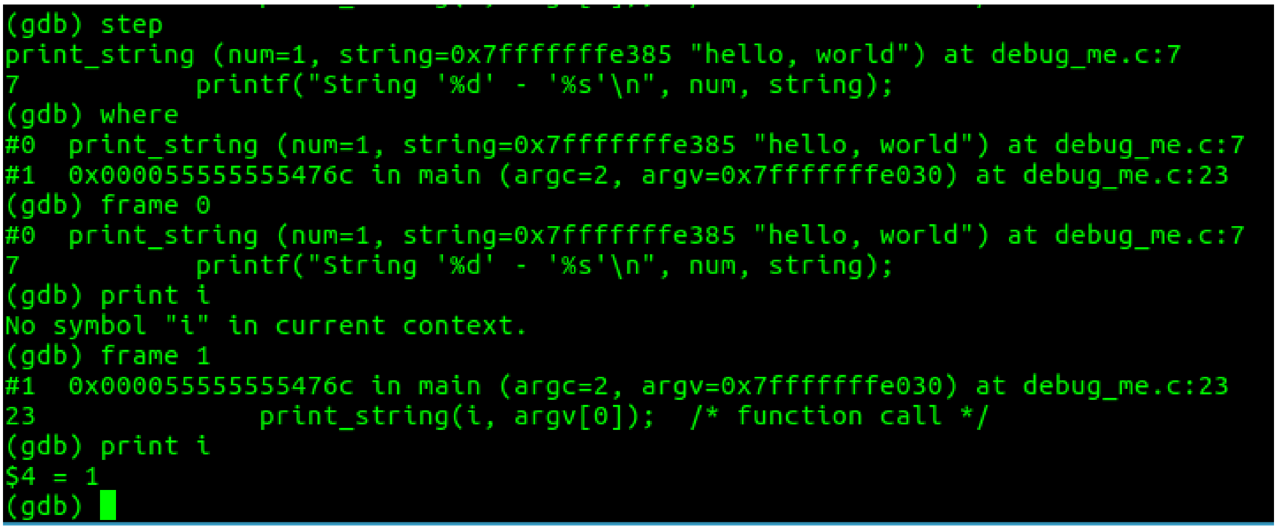
```
Breakpoint 1, main (argc=3, argv=0x7fffffffe028) at debug_me.c:16
16          if (argc < 2) { /* 2 - 1 for program name (argv[0]) and one for a pa
ram. */
(gdb) step

Breakpoint 2, main (argc=3, argv=0x7fffffffe028) at debug_me.c:22
22          for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) step
23              print_string(i, argv[0]);  /* function call */
(gdb) step
print_string (num=1, string=0x7fffffffe385 "hello, world") at debug_me.c:7
7           printf("String '%d' - '%s'\n", num, string);
(gdb) next
String '1' - 'hello, world'
8       }
(gdb) next
main (argc=2, argv=0x7fffffffe030) at debug_me.c:22
22          for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) next
23              print_string(i, argv[0]);  /* function call */
(gdb) next
String '2' - 'goodbye, world'
22          for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) a
```

5. 此时是第二次进入for循环了，使用print查看各个变量，可以看到几个变量都发生了改变



```
(gdb) print i
$1 = 2
(gdb) print argc
$2 = 1
(gdb) pritnt argv
Undefined command: "pritnt".  Try "help".
(gdb) print argv
$3 = (char **) 0x7fffffffe038
(gdb) 
```

6. 重新运行程序，step到print_string函数内的语句。使用where语句，可以看到当前执行的函数名和调用该函数的函数名。frame语句进入特定的函数空间内进行调试，可以观察到print_string函数中因为没有i变量，所以在此调试print i会报错； 而进入main函数进行调试，输入print i则会显示当前的i值。



```
(gdb) step
print_string (num=1, string=0x7fffffffe385 "hello, world") at debug_me.c:7
7           printf("String '%d' - '%s'\n", num, string);
(gdb) where
#0  print_string (num=1, string=0x7fffffffe385 "hello, world") at debug_me.c:7
#1  0x000055555555476c in main (argc=2, argv=0x7fffffffe030) at debug_me.c:23
(gdb) frame 0
#0  print_string (num=1, string=0x7fffffffe385 "hello, world") at debug_me.c:7
7           printf("String '%d' - '%s'\n", num, string);
(gdb) print i
No symbol "i" in current context.
(gdb) frame 1
#1  0x000055555555476c in main (argc=2, argv=0x7fffffffe030) at debug_me.c:23
23              print_string(i, argv[0]);  /* function call */
(gdb) print i
$4 = 1
(gdb) 
```