

Lab 2.2 Running a Hello World Program in C using GCC

3200105872 软工 2020 庄毅非

Overview

The lab helps familiarize you with writing a simple Hello World program using C, the GCC compiler [link](#), and Pico(a text editor, [link](#)). It uses Ubuntu VM created in Lab 2.1. Here is lab objective:

1. Learn to run a program in gcc.
2. Learn to debug a program in gdb.

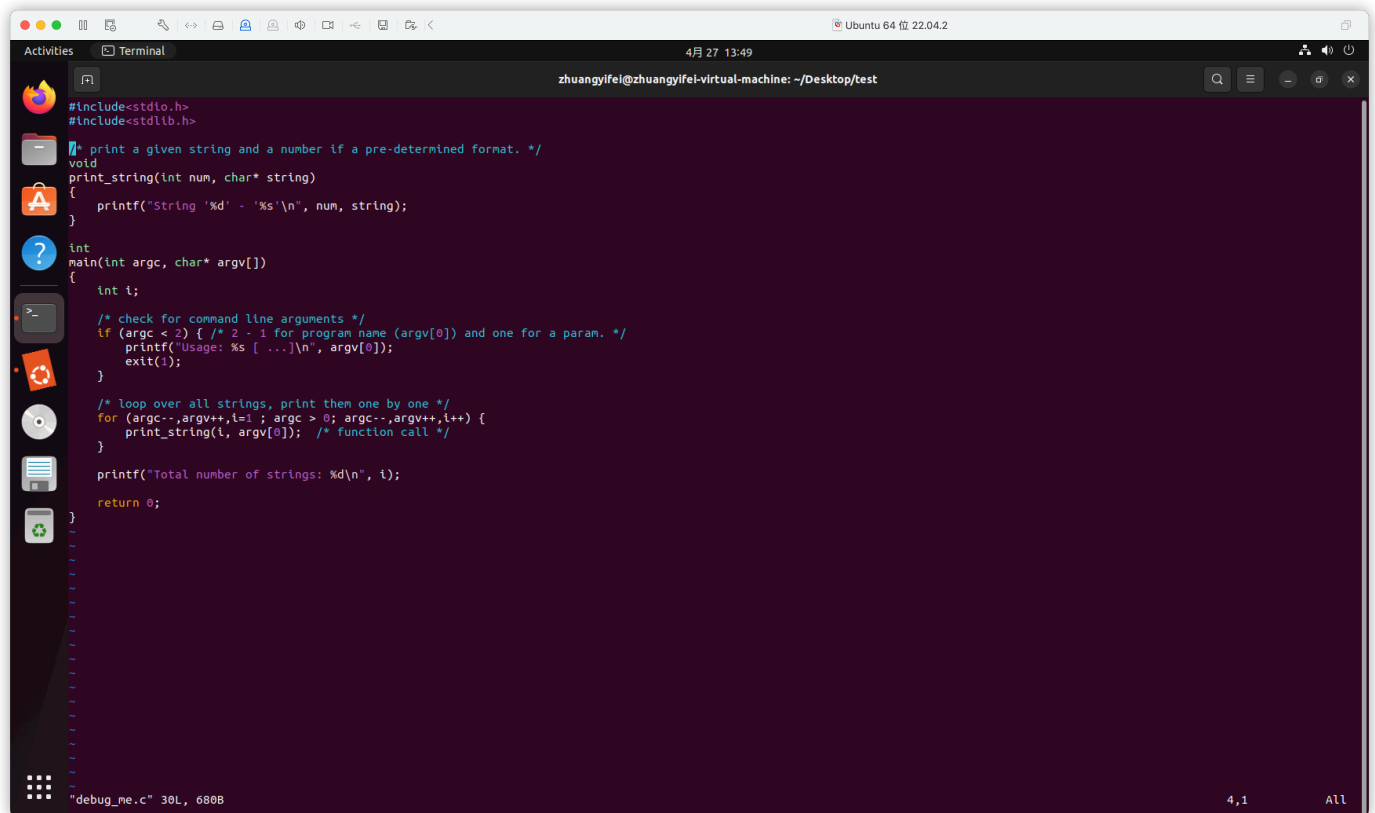
Open the Terminal in Ubuntu.

1. 安装必要套件

为了完成实验, 我们首先需要安装 gcc 和 gdb, 命令是 `sudo apt install build-essential`

2. Create 'debug_me.c':

使用 vim, 将题目中的 c 程序放到 `debug_me.c` 文件中



```
#include<stdio.h>
#include<stdlib.h>

/* print a given string and a number if a pre-determined format. */
void
print_string(int num, char* string)
{
    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;
}
```

3. Invoke gdb to debug 'debug_me.c':

使用 `gcc -g debug_me.c -o debug_me` 进行编译, 使用 `gdb` 运行。

```
zhuangyifei@zhuangyifei-virtual-machine: ~/Desktop/test$ vim debug_me.c
zhuangyifei@zhuangyifei-virtual-machine: ~/Desktop/test$ gcc -g debug_me.c -o debug_me
zhuangyifei@zhuangyifei-virtual-machine: ~/Desktop/test$ gdb debug_me
GNU gdb (Ubuntu 12.1-0ubuntu1-22.04) 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 debug_me...
(gdb)
```

4. Run the program inside gdb:

运行 `run "hello, world" "goodbye, world"`, 查看输出。

```
Reading symbols from debug_me...
(gdb) run "hello, world" "goodbye, world"
Starting program: /home/zhuangyifei/Desktop/test/debug_me "hello, world" "goodbye, world"
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
String '1' - 'hello, world'
String '2' - 'goodbye, world'
Total number of strings: 3
Inferior 1 (process 4268) exited normally
```

5. Set breakpoints. You can set breakpoints using two methods:

使用两种方式设置断点。

```
(gdb) break debug_me.c:9
Breakpoint 1 at 0x55555555199: file debug_me.c, line 9.
(gdb) break main
Breakpoint 2 at 0x555555551af: file debug_me.c, line 17.
(gdb)
```

6. Step a command at a time. After setting breakpoints, you can run the program step by step. There are also two methods:

使用两种 `next` 和 `step` 两种方式继续程序运行, 区别主要在于后者如果是函数调用的话会进入到子函数里面, 前者则不会。

```

(gdb) run "hello, world" "goodbye, world"
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/zhuangyifei/Desktop/test/debug_me "hello, world" "goodbye, world"
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 2, main (argc=3, argv=0x7fffffffdf98) at debug_me.c:17
17     if (argc < 2) { /* 2 - 1 for program name (argv[0]) and one for a param. */
(gdb) next
23     for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb)
24         print_string(i, argv[0]); /* function call */
(gdb) step
print_string (num=1, string=0x7fffffff31a "hello, world") at debug_me.c:8
8         printf("String '%d' - '%s'\n", num, string);
(gdb)
__printf (format=0x555555556004 "String '%d' - '%s'\n") at ./stdio-common/printf.c:28
28     ./stdio-common/printf.c: No such file or directory.

```

7. Print variables

使用 `print` 观察变量的值。

```
zhuangyifei@zhuangyifei-virtual-machine: ~/Desktop/test
debug_me.c
7 {
8     printf("String '%d' - '%s'\n", num, string);
9 }
10
11 int
12 main(int argc, char* argv[])
13 {
14     int i;
15
16     /* check for command line arguments */
17     if (argc < 2) { /* 2 - 1 for program name (argv[0]) and one for a param. */
18         printf("Usage: %s [ ...]\n", argv[0]);
19         exit(1);
20     }
21
22     /* loop over all strings, print them one by one */
23     for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
24         print_string(i, argv[0]); /* function call */
25     }
26
27     printf("Total number of strings: %d\n", i);
28
29     return 0;
30 }
31
32
33
34
35
36

multi-thre Thread 0x7ffff7fa97 In: main L24 PC: 0x5555555551ef
(gdb) Quit
(gdb) n
main (argc=2, argv=0x7ffffffffffdfa0) at debug_me.c:23
(gdb) print i
$2 = 1
(gdb) n
(gdb) print i
$3 = 2
(gdb) █
```

8. Examine the function call stack.

使用 where 观察函数调用链。

```
value returned is $4 = 30
(gdb) where
#0  print_string (num=2, string=0x7fffffff327 "goodbye, world") at debug_me.c:9
#1  0x0000555555555203 in main (argc=1, argv=0x7fffffffdfa8) at debug_me.c:24
(gdb) frame 0
#0  print_string (num=2, string=0x7fffffff327 "goodbye, world") at debug_me.c:9
(gdb) print i
No symbol "i" in current context.
(gdb) frame 1
#1  0x0000555555555203 in main (argc=1, argv=0x7fffffffdfa8) at debug_me.c:24
(gdb) print i
$5 = 2
(gdb)
```

可以看到 frame 0 实际上是没有 i 的, 原因是没有定义, 只有主函数 main 定义了 i。