

SP-lab2.2

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.

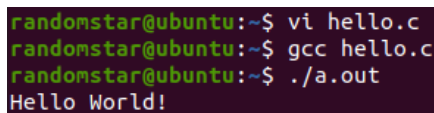
实验过程

- 编写一个hello world的C程序(这一部分实验标题里有但是实验正文里却没有，但是还是简单尝试一下)
 - 打开vi编辑器，编写hello.c文件中的内容

A screenshot of a terminal window titled 'randomstar@ubuntu: ~'. The terminal shows the contents of a C program being edited in the vi editor. The code is as follows:

```
#include<stdio.h>
int main()
{
    printf("Hello World!\n");
    return 0;
}
```

- 保存并退出，之后在命令行中输入命令 `gcc hello.c` 编译该C文件，输入 `./a.out` 即可执行编译生成的文件

A screenshot of a terminal window showing the compilation and execution of the C program. The commands and output are as follows:

```
randomstar@ubuntu:~$ vi hello.c
randomstar@ubuntu:~$ gcc hello.c
randomstar@ubuntu:~$ ./a.out
Hello World!
```

- 接下来是实验步骤中安排的内容，首先打开vi编辑器，编写如下内容，文件命名位debug_me.c，保存并退出，另外记得在复制实验网站的代码时别忘记添加两个头文件 `#include<stdio.h>` 和 `#include<stdlib.h>`

```

#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;
}

```

- 在命令行中执行如下两条命令 `gcc -g debug_me.c -o debug_me` 和 `gdb debug_me`

```

randomstar@ubuntu:~$ gcc -g debug_me.c -o debug_me
randomstar@ubuntu:~$ gdb debug_me
GNU gdb (Ubuntu 8.3-0ubuntu1) 8.3
Copyright (C) 2019 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:
<http://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)

```

- 在gdb中输入 `run "hello, world" "goodbye, world"`

```

(gdb) run "hello,world" "goodbye,world"
Starting program: /home/randomstar/debug_me "hello,world" "goodbye,world"
String '1' - 'hello,world'
String '2' - 'goodbye,world'
Total number of strings: 3
[Inferior 1 (process 3081) exited normally]

```

- 设置断点，使用 `break main` 等命令来设置断点

```

(gdb) break main
Breakpoint 1 at 0x55555555199: file debug_me.c, line 11.
(gdb) break debug_me.c:22
Breakpoint 2 at 0x555555551e9: file debug_me.c, line 22.

```

- 断点运行：使用 `next` 和 `step` 命令来调试程序，二者的主要区别是
 - `next` 遇到函数时将其当作一个语句来执行

- step遇到函数时会进入函数并且一步步执行
- 调试的结果如下

```
(gdb) run
Starting program: /home/randomstar/debug_me "hello,world" "goodbye,world"

Breakpoint 1, main (argc=21845, argv=0x555555555230 <__libc_csu_init>)
    at debug_me.c:11
11      {
(gdb) step
15      if (argc < 2) { /* 2 - 1 for program name (argv[0]) and one for a p
aram. */
(gdb) step
21      for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) next

Breakpoint 2, main (argc=2, argv=0x7fffffffe0f0) at debug_me.c:22
22      print_string(i, argv[0]); /* function call */
(gdb) next
String '1' - 'hello,world'
21      for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) next

Breakpoint 2, main (argc=1, argv=0x7fffffffe0f8) at debug_me.c:22
22      print_string(i, argv[0]); /* function call */
(gdb) next
String '2' - 'goodbye,world'
21      for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
```

- 在第二次循环时，我们可以用print的方法来查看中间变量的值，这里可以用 `print i` 来查看for循环中i的值

```
Breakpoint 2, main (argc=2, argv=0x7fffffffe0f0) at debug_me.c:22
22      print_string(i, argv[0]); /* function call */
(gdb) next
String '1' - 'hello,world'
21      for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) next

Breakpoint 2, main (argc=1, argv=0x7fffffffe0f8) at debug_me.c:22
22      print_string(i, argv[0]); /* function call */
(gdb) next
String '2' - 'goodbye,world'
21      for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) print i
$1 = 2
(gdb) print i
$2 = 2
(gdb) next
25      printf("Total number of strings: %d\n", i);
(gdb) next
Total number of strings: 3
27      return 0;
(gdb) print i
$3 = 3
```

- 可以用where指令来查看函数栈

```

Breakpoint 1, main (argc=21845, argv=0x555555555230 <__libc_csu_init>)
  at debug_me.c:11
11      {
(gdb) step
15      if (argc < 2) { /* 2 - 1 for program name (argv[0]) and one for a p
aram. */
(gdb) step
21      for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) step

Breakpoint 2, main (argc=2, argv=0x7fffffffef0) at debug_me.c:22
22      print_string(i, argv[0]); /* function call */
(gdb) step
print_string (num=21845, string=0x0) at debug_me.c:6
6      {
(gdb) where
#0  print_string (num=21845, string=0x0) at debug_me.c:6
#1  0x0000555555551fd in main (argc=2, argv=0x7fffffffef0) at debug_me.c:22
(gdb) step
7      printf("String '%d' - '%s'\n", num, string);
(gdb) step
__CDROM__ (format=0x555555556004 "String '%d' - '%s'\n") at printf.c:28
28      printf.c: No such file or directory.
(gdb) where
#0  __printf (format=0x555555556004 "String '%d' - '%s'\n") at printf.c:28
#1  0x000055555555196 in print_string (num=1,
    string=0x7fffffffef0 "hello,world") at debug_me.c:7
#2  0x0000555555551fd in main (argc=2, argv=0x7fffffffef0) at debug_me.c:22

```

- frame指令的使用，可以用frame指令进行切换，不同的frame中打印的变量i的值不同，原因是子函数中变量i没有被定义，而主函数中i=1

```

(gdb) step
21      for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) print i
$1 = 0
(gdb) frame 0
#0  main (argc=3, argv=0x7fffffffef0e8) at debug_me.c:21
21      for (argc--,argv++,i=1 ; argc > 0; argc--,argv++,i++) {
(gdb) step

Breakpoint 2, main (argc=2, argv=0x7fffffffef0) at debug_me.c:22
22      print_string(i, argv[0]); /* function call */
(gdb) step
print_string (num=21845, string=0x0) at debug_me.c:6
6      {
(gdb) step
7      printf("String '%d' - '%s'\n", num, string);
(gdb) step
__printf (format=0x555555556004 "String '%d' - '%s'\n") at printf.c:28
28      printf.c: No such file or directory.
(gdb) step
32      in printf.c
(gdb) print i
No symbol "i" in current context.
(gdb) frame 1
#1  0x000055555555196 in print_string (num=1,
    string=0x7fffffffef0 "hello,world") at debug_me.c:7

```

- 本次实验到此结束，完成了要求的所有内容