Software Engineering Lab

Assignment – 3

Swapnadeep Mishra
Roll-002211001115, Group- A3

1) Consider the program in Assign3.It is a simple state machine.

a. Put a breakpoint in line 49

Ans: break 49

```
[be22115@localhost ~]$ gcc -g e.c -o e
[be22115@localhost ~]$ gdb e
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-94.el7
Copyright (C) 2013 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/usr/student/ug/yr22/be22115/e...done.
(gdb) break 49
Breakpoint 1 at 0x4006a1: file e.c, line 49.
```

b) Try next command

```
The program is not being run.
(gdb) r
Starting program: /home/usr/student/ug/yr22/be22115/e

Breakpoint 1, main () at e.c:49
49              step_state(events_arr[cntr]);
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.2.x86_64
(gdb) next
50              cntr++;
```

c) How will you get inside the function without using breakpoint?

Ans: step

```
Breakpoint 1, main () at e.c:49
49              step_state(events_arr[cntr]);
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.2.x86_64
(gdb) step
step_state (event=START_LOOPING) at e.c:15
15          switch(state) {
(gdb) step
17              switch(event) {
```

d) How will you come out the of the function without using next and continue?
Ans: finish

```
(gdb) finish
Run till exit from #0  step_state (event=START_LOOPING) at e.c:17
main () at e.c:50
50              cntr++;
```

2) Consider the program in Assign4 .It is also a simple state machine.If you provide user id and password properly account details will be displayed. The basic rule is user id should be positive and less than 20 .password is userid *b1000 .The loop will terminate after 10 iteration. It works fine if you provide valid user id and password.It works fine for invalid userid. But it goes to infiniteloop for invalid password.Run the program .It goes into infinite loop.you need to kill the program by [ctrl^c]

a) Set a suitable breakpoint in gdb in the routine show.give valid input and run :
Ans: break 43

```
(gdb) break 43
Breakpoint 1 at 0x4006bb: file f.c, line 43.
(gdb) run
Starting program: /home/usr/student/ug/yr22/be22115/f

Breakpoint 1, step_state (event=START_LOOPING) at f.c:44
44              switch(event) {
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.2.x86_64
(gdb) c
Continuing.
Hello Please Provide  User Id and Password to see your details!
User Id: 12
Password: 12000
User Id : 12, Password: 12000 , Amount : 1200000

Breakpoint 1, step_state (event=START_LOOPING) at f.c:44
44              switch(event) {
```

b) How you can see the call stack of the routine.

Ans: bt

```
(gdb) bt
#0  step_state (event=START_LOOPING) at f.c:44
#1  0x000000000040085a in main () at f.c:111
```

c) Which commands will help you to see each value change of variable “event”?

Ans: watch event

```
(gdb) watch event
Hardware watchpoint 2: event
(gdb) n
47                  state = LOOP;
(gdb) n
48                  if (cntr > 10) {
(gdb) n
53                      printf("Hello Please Provide  User Id and Password to see your details!\n");
(gdb) n
Hello Please Provide  User Id and Password to see your details!
```

d) Correct the program so that it doesn’t go to infinite loop for wrong password. Rather main iteration restarts . [follow the value change path of event for wrong password]

```
        printf("Incorrect password!!\n");
        event = STOP_LOOPING ;
        state = START ;
```

```
}else{
        printf("Incorrect password!!\n");
        event = STOP_LOOPING ;
        state = START ;
}
```

Explore the commands found for 5c to see/use content of a pointer

```
Breakpoint 1, step_state (event=USERID_MATCHED) at f.c:94
94                          printf("Incorrect password!!\n");
Missing separate debuginfos, use: debuginfo-install glibc-2.17-15
7.el7_3.2.x86_64
(gdb) watch event
Hardware watchpoint 2: event
(gdb) n
Incorrect password!!
95                          event = START_LOOPING ;
(gdb) n
Hardware watchpoint 2: event

Old value = USERID_MATCHED
New value = START_LOOPING
step_state (event=START_LOOPING) at f.c:96
96                          state =  START;
```