

Software Engineering

Assignments-3 on gdb



Name – Anuska Nath
**Department - Information
Technology**
Roll No - 002311001003
Section – A1

The program code is as follows:

```
#include <stdio.h>
#include <stdlib.h>
enum states {
    START,
    LOOP,
    END,
} state;

enum events {
    START_LOOPING,
    PRINT_HELLO,
    STOP_LOOPING,
};

void step_state(enum events event) {
    switch(state) {
        case START:
            switch(event) {
                case START_LOOPING:
                    state = LOOP;
                    break;
                default:
                    exit(1);
                    break;
            }
            break;
        case LOOP:
            switch(event) {
                case PRINT_HELLO:
                    printf("Hello World!\n");
                    break;
                case STOP_LOOPING:
                    state = END;
                    break;
                default:
                    exit(1);
                    break;
            }
    }
}
```

```
        break;
    case END:
        exit(1);
        break;
    }
}
int main(void) {
    int cntr = 0 ;
    events events_arr[] = { START_LOOPING, PRINT_HELLO, PRINT_HELLO, PRINT_HELLO,
    PRINT_HELLO,
    PRINT_HELLO,PRINT_HELLO,PRINT_HELLO,STOP_LOOPING,PRINT_HELLO,PRINT_HELLO,STOP
    _LOOPING};
    while(events_arr[cntr] != STOP_LOOPING)
    {
        step_state(events_arr[cntr]);
        cntr++;
    }

    if (cntr == 10) {
        printf("PASS");
    } else {
        printf("FAIL %d", cntr);
    }

    return 0;
}
```

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

a. Put a breakpoint in line 49

Ans> break e.c:49

```
(gdb) break e.c:49
Breakpoint 1 at 0x4006a1: file e.c, line 49.
(gdb)
```

b. Try next command

Ans> next - it will show program is not being run but if the program runs then output will be

```
(gdb) run
Starting program: /home/usr/student/ug/yr23/be2303/SE/assign3/prog

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

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

Ans> run, step

```
(gdb) run
Starting program: /home/usr/student/ug/yr23/be2303/SE/assign3/prog

Breakpoint 1, main () at e.c:49
49          step_state(events_arr[ctr]);
(gdb) step
step_state (event=START_LOOPING) at e.c:15
15          switch(state) {
(gdb)
```

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:15
main () at e.c:50
50          cntr++;
```

The code is as follows :

```
/******
```

Welcome to GDB Online.

GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby,
C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS
Code, Compile, Run and Debug online from anywhere in world.

```
*****/
```

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

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

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

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

```
enum states {
```

```
    START,
```

```
    LOOP,
```

```
    END,
```

```
} state;
```

```
enum events {
```

```
    START_LOOPING,
```

```
    USERID_MATCHED,
```

```
    SHOW_DETAIL,
```

```
    STOP_LOOPING,
```

```
};
```

```
int valid_id(int id)
```

```

{
    if ( id > 0 && id < 20)
        return 1;
    else
        return 0;
}

int valid_pw(int id, int password)
{
    if (password == id*1000)
        return 1;
    return 0;
}
int show(int id)
{
    return id*100000;
}
void step_state(enum events event) {
    int cntr= 0;
    while(1) {
        int id , password;
        cntr++;
        switch(state) {
        case START:
            switch(event) {
            case START_LOOPING:
                {
                    state = LOOP;
                    if (cntr > 10) {
                        printf("Session expired!");
                        event = STOP_LOOPING;
                        state = END;
                    } else {
                        printf("Hello Please Provide User Id and Password to see your details!\n");
                        printf("User Id: ");
                        scanf("%d", &id);
                        if (valid_id(id)) {
                            event = USERID_MATCHED ;
                        } else {
                            printf("Incorrect User Id!!");
                            event = START_LOOPING ;
                            state = START ;
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    }

    break;
}
case STOP_LOOPING:
{
    printf("Invaoid state\n");
    break;
}
default:
    exit(1);
    break;
}
break;
case LOOP:
    switch(event) {
    case USERID_MATCHED:
        printf("Password: ");
        scanf("%d", &password);
        if (valid_pw(id,password)) {
            event = SHOW_DETAIL ;
        } else {
            printf("Incorrect password!!\n");
            event = STOP_LOOPING ;
            state = START ;
        }
        break;
    case SHOW_DETAIL:
        {
            char c = 'p';
            printf("User Id : %d, Password: %d , Amount : %d\n", id,password,show(id));
            state = START ;
            event = START_LOOPING;

        }
        break;
    default:
        exit(1);
        break;
    }
    break;
case END:

```

```

        exit(1);
        break;
    }
}
}
int main(void) {
    step_state(START_LOOPING);

    return 0;
}

```

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 show, run

```

(gdb) break show
Breakpoint 1 at 0x40067b: file f.c, line 43.
(gdb) run
Starting program: /home/usr/student/ug/yr23/be2303/SE/assign4/prog
Hello Please Provide  User Id and Password to see your details!
User Id: 15
Password: 15000

Breakpoint 1, show (id=15) at f.c:43
43         return id*100000;
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.2.x86_64

```


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

Ans> run, info stack, backtrace

```
(gdb) run
Starting program: /home/usr/student/ug/yr23/be2303/SE/assign4/prog
Hello Please Provide User Id and Password to see your details!
User Id: 15
Password: 15000

Breakpoint 1, show (id=15) at f.c:43
43         return id*1000000;
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.2.x86_64
(gdb) info stack
#0  show (id=15) at f.c:43
#1  0x0000000000400805 in step_state (event=SHOW_DETAIL) at f.c:101
#2  0x000000000040085a in main () at f.c:119
(gdb) backtrace
#0  show (id=15) at f.c:43
#1  0x0000000000400805 in step_state (event=SHOW_DETAIL) at f.c:101
#2  0x000000000040085a in main () at f.c:119
(gdb) █
```

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

Ans> first run the program , then use the step command and use the watch event command to see the number of hardware watchpoints.

run, step, watch event, step, watch event, step, watch event, step , watch event.

```

(gdb) run
Starting program: /home/usr/student/ug/yr23/be2303/SE/assign4/pro
Hello Please Provide User Id and Password to see your details!
User Id: 15
Password: 15000

Breakpoint 1, show (id=15) at f.c:43
43         return id*100000;
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.2.x86_64
(gdb) step
44     }
(gdb) watch event
No symbol "event" in current context.
(gdb) step
User Id : 15, Password: 15000 , Amount : 1500000
step_state (event=SHOW_DETAIL) at f.c:102
102         state = START ;
(gdb) watch event
Hardware watchpoint 2: event
(gdb) step
103         event = START_LOOPING;
(gdb) watch event
Hardware watchpoint 3: event
(gdb) step
Hardware watchpoint 2: event

Old value = SHOW_DETAIL
New value = START_LOOPING
Hardware watchpoint 3: event

Old value = SHOW_DETAIL
New value = START_LOOPING
step_state (event=START_LOOPING) at f.c:106
106         break;
(gdb) watch event
Hardware watchpoint 4: event
(gdb) █

```

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]

Ans> Just change the STOP_LOOPING to START_LOOPING in the USERID SECTION, then it will not go to infinite looping.

Before:

```
case LOOP:
    switch(event) {
    case USERID_MATCHED:
        printf("Password: ");
        scanf("%d", &password);
        if (valid_pw(id,password)) {
            event = SHOW_DETAIL ;
        } else {
            printf("Incorrect password!!\n");
            event = STOP_LOOPING ;
            state = START ;
        }
    }
```

After :

```
case LOOP:
    switch(event) {
    case USERID_MATCHED:
        printf("Password: ");
        scanf("%d", &password);
        if (valid_pw(id,password)) {
            event = SHOW_DETAIL ;
        } else {
            printf("Incorrect password!!\n");
            event = START_LOOPING ;
            state = START ;
        }
    }
```

```
(gdb) run
Starting program: /home/usr/student/ug/yr23/be2303/SE/assign4/pro
Hello Please Provide  User Id and Password to see your details!
User Id: 15
Password: 8932
Incorrect password!!
Hello Please Provide  User Id and Password to see your details!
User Id: 
```

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

Ans> x <function_name or variable_name>

example - x show

```
(gdb) x show  
0x400674 <show>:      0xe5894855  
(gdb) █
```