OS
Assignment-4

Akshala Bhatnagar
2018012

- **Paste your code corresponding to `syscall_handler`**

```
ENTRY(syscall_handler)

push %eax
push %ecx
push %edx
push %fs

mov $216, %dx
mov %dx, %fs

push %ecx
push %eax

call syscall_handler_k

add $8, %esp

pop %fs
pop %edx
pop %ecx
add $4, %esp

iret
ENDPROC(syscall_handler)
```

- **Paste your code corresponding to `u_syscall`**

```
syscall_u:
mov 4(%esp), %eax
mov 8(%esp), %ecx
int $15
```

ret

- **Paste your code corresponding to `register_syscall`**

```
struct idt_desc current_idt;
struct idt_desc original_idt;
static void register_syscall(void)
{
   imp_copy_idt(&current_idt);
   struct idt_entry *index_15 = current_idt.base + 15;
   struct idt_entry *index_128 = current_idt.base + 128;
   memcpy(index_15, index_128, sizeof(struct idt_entry));
   unsigned long syscall_addr = (unsigned long)syscall_handler;
   unsigned long lower_16 = (unsigned short) (syscall_addr);
   unsigned long higher_16 = (unsigned short) (syscall_addr >> 16);
   index_15->lower16 = lower_16;
   index_15->higher16 = higher_16;
   imp_load_idt(&current_idt, &original_idt);
}
```

- **Paste your code corresponding to unregister_syscall**

```
static void unregister_syscall(void)
{
   if((current_idt.base != original_idt.base) && (current_idt != NULL){
         imp_load_idt(&original_idt, &current_idt);
         imp_free_desc(&current_idt);
   }
}
```

- **The output of user-program**

```
bhatnagar@bhatnagar-VirtualBox:~/Downloads/syscall-master/user$ ./syscall
syscall done!
```

- **How do you know the location of the original IDT in `unregister_syscall`?**

  I have maintained a global variable called original_idt. This contains the original IDT. I have also maintained a global variable called current_idt. This contains the current IDT. Each time when register_syscall is called, in the end the current_idt is loaded into the original_idt. Both current_idt and original_idt are passed to imp_load_idt by reference. This is how I get to know the location of the original IDT.

- **How do you know the location of the current IDT in `unregister_syscall`?**
  I have maintained a global variable called current_idt. This contains the current IDT. When the register_syscall function is called, current_idt is passed by reference into imp_copy_idt. Then the required changes are made in this IDT, that is contents of index 128 are copied to contents of index 15. The lower 16 and higher 16 bits of index 15 are set equal to the lower 16 and higher 16 bits of the address of the syscall_handler. As the global variable, current_idt was passed by reference, the changes are reflected in it and this can be directly used in unregister_syscall.

- **If somebody calls `unregister_syscall` twice, without calling `register_syscall` in between.**

  - **Do you load the original IDT twice?**
    No, I do not load the original IDT twice. This is because in the if condition, I am checking if the current_idt.base is not equal to the original_idt.base and that the current_idt is not null. If I have called unregister_syscall once, then it means that the current_idt has been freed, so current_idt would now be NULL. Hence we would not enter into the if condition and the original IDT is not loaded twice.

  - **Do you free the current IDT twice?**
    No, I do not free the current IDT twice. This is because in the if condition, I am checking if the current_idt.base is not equal to the original_idt.base and that the current_idt is not null. If I have called unregister_syscall once, then it means that the current_idt has been freed, so current_idt would now be NULL. Hence we would not enter into the if condition and the current IDT is not freed twice.