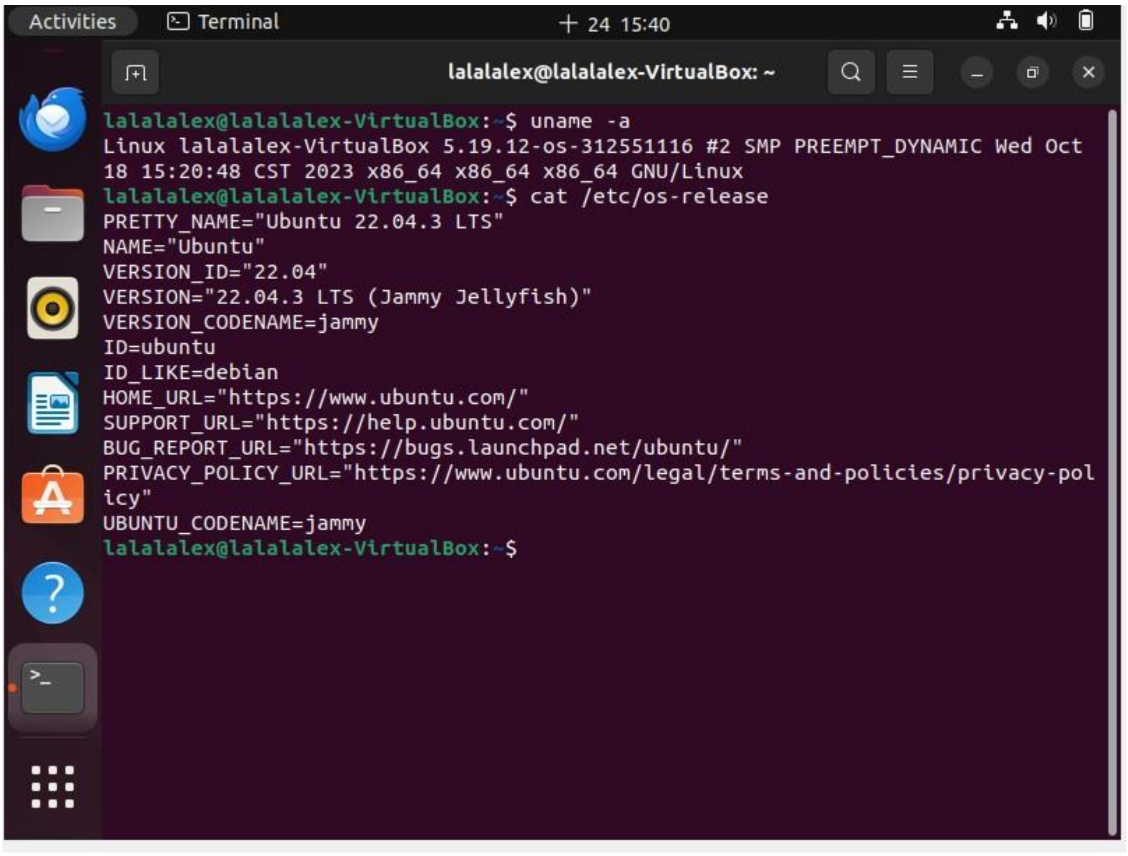
Operating System – Assignment 1

Compiling Linux Kernel and Adding Custom System Calls

I. Kernel Compilation

i. Paste the screenshot of the results of executing uname -a and cat /etc/os-release commands as the example sho

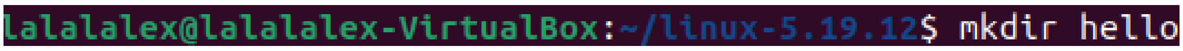


```
lalalalex@lalalalex-VirtualBox:~$ uname -a
Linux lalalalex-VirtualBox 5.19.12-os-312551116 #2 SMP PREEMPT_DYNAMIC Wed Oct
18 15:20:48 CST 2023 x86_64 x86_64 x86_64 GNU/Linux
lalalalex@lalalalex-VirtualBox:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.3 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-pol
icy"
UBUNTU_CODENAME=jammy
lalalalex@lalalalex-VirtualBox:~$
```

II. System Call

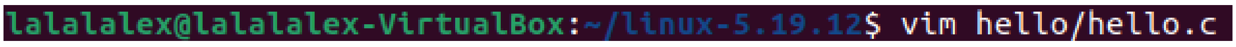
i. Describe how you implemented the two system calls in detail

(1) First, create the directory for each system call, e.g. ~/hello/, ~/revstr/



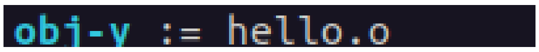
```
lalalalex@lalalalex-VirtualBox:~/linux-5.19.12$ mkdir hello
```

(2) Second, Create the c file for the system call and write the system call



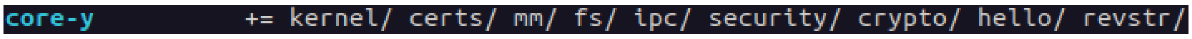
```
lalalalex@lalalalex-VirtualBox:~/linux-5.19.12$ vim hello/hello.c
```

(3) Add a Makefile in the directory, it will let kernel know there is a system call file to compile and get the out(.o) file.



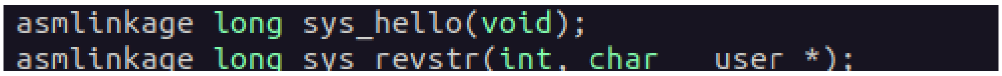
```
obj-y := hello.o
```

(4) Edit the Make file in the ~/linux-5.19.12. It will let kernel recursive and compile all the file in the directory. And with previous step, it will help us generate hello.o and revstr.o.



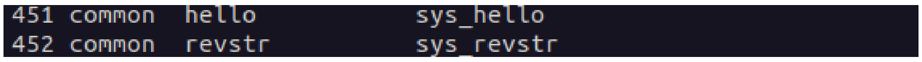
```
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ hello/ revstr/
```

(5) Next, we need to edit the ~/include/linux/syscalls.h. asmlinkage means that the system call will get parameters from stack, not from register. If we use C to write system call and need stack to get parameters, we will use it.



```
asmlinkage long sys_hello(void);
asmlinkage long sys revstr(int, char   user *);
```

(6) Go into ~/arch/x86/entry/syscalls/syscall_64.tbl, and add our system call in it, the first is the file name and second is the system call name. It will let kernel find our system call file.

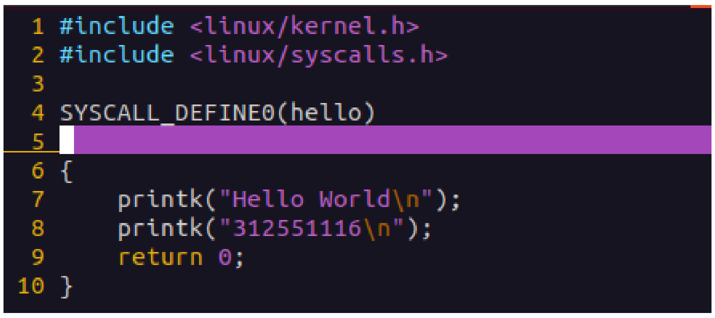


```
451 common  hello           sys_hello
452 common  revstr          sys_revstr
```

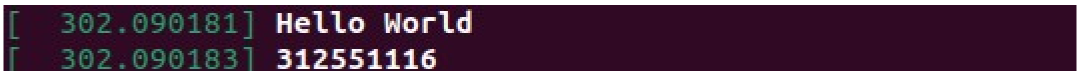
(7) Final, compile the kernel and test it.

ii. For each system call you implemented

(1) Hello World

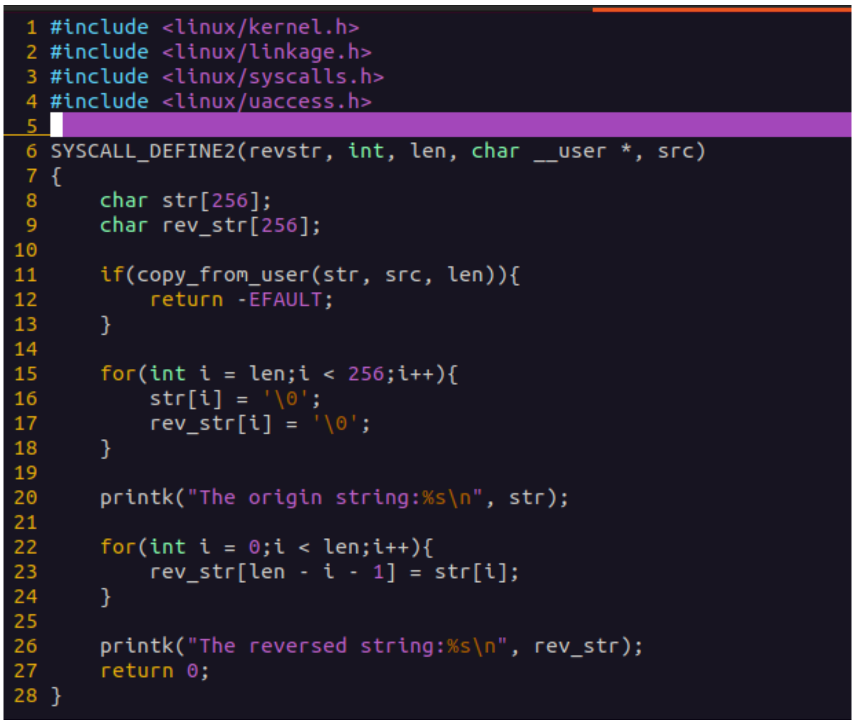


```c
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(hello)

{
    printk("Hello World\n");
    printk("312551116\n");
    return 0;
}
```

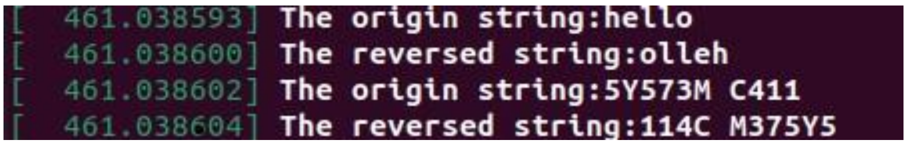


```
[  302.090181] Hello World
[  302.090183] 312551116
```

(2) Revstr



```c
#include <linux/kernel.h>
#include <linux/linkage.h>
#include <linux/syscalls.h>
#include <linux/uaccess.h>

SYSCALL_DEFINE2(revstr, int, len, char __user *, src)
{
    char str[256];
    char rev_str[256];

    if(copy_from_user(str, src, len)){
        return -EFAULT;
    }

    for(int i = len;i < 256;i++){
        str[i] = '\0';
        rev_str[i] = '\0';
    }

    printk("The origin string:%s\n", str);

    for(int i = 0;i < len;i++){
        rev_str[len - i - 1] = str[i];
    }

    printk("The reversed string:%s\n", rev_str);
    return 0;
}
```



```
[  461.038593] The origin string:hello
[  461.038600] The reversed string:olleh
[  461.038602] The origin string:5Y573M C411
[  461.038604] The reversed string:114C M375Y5
```

- The number after SYSCALL_DEFINE means the number of the parameters, e.g. SYSCALL_DEFINE2 means there are two parameters in revstr. And the first is the system call name.
- copy_from_user(to, from, len) is used to copy the data from user space.