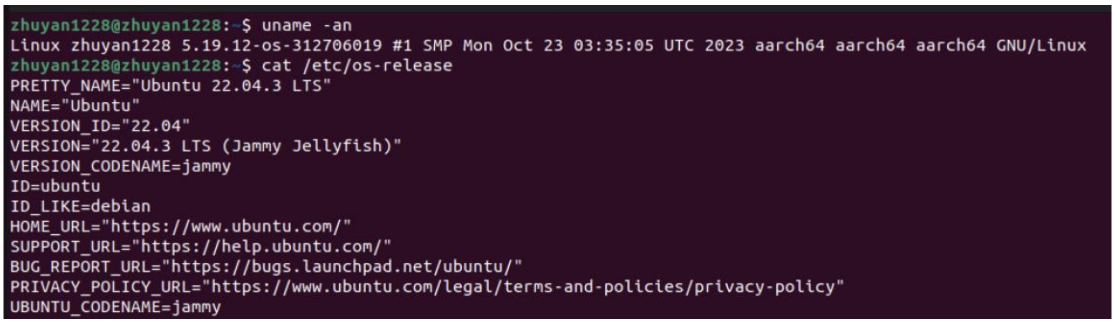
312706019 Assignment 1

For the kernel compilation part:

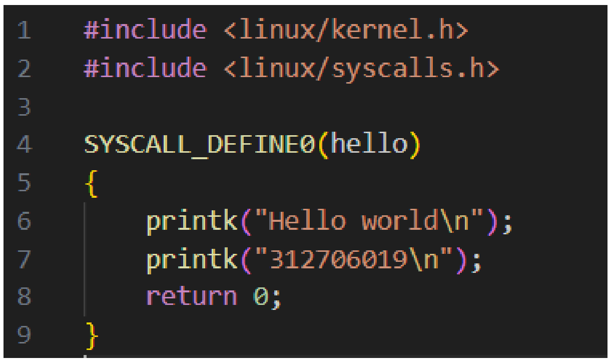
Paste the screenshot of the results of executing uname -a and cat /etc/os-release commands.



```
zhuyan1228@zhuyan1228:~$ uname -an
Linux zhuyan1228 5.19.12-os-312706019 #1 SMP Mon Oct 23 03:35:05 UTC 2023 aarch64 aarch64 aarch64 GNU/Linux
zhuyan1228@zhuyan1228:~$ 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-policy"
UBUNTU_CODENAME=jammy
```

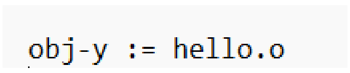
For the system call part:

1. sys_hello:
   a. create the folder hello to store system call files. (in linux-5.19.12) Then, create a file **hello.c** to implement our system call:



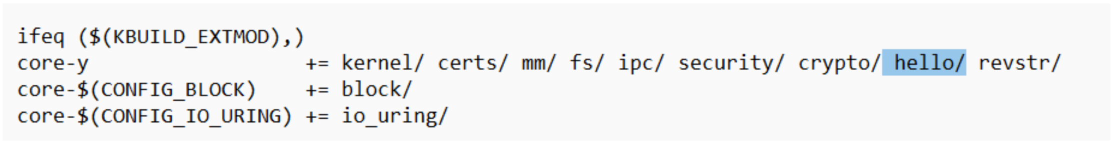
```c
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(hello)
{
    printk("Hello world\n");
    printk("312706019\n");
    return 0;
}
```

And create a **Makefile** for our system call, also in the same hello file:



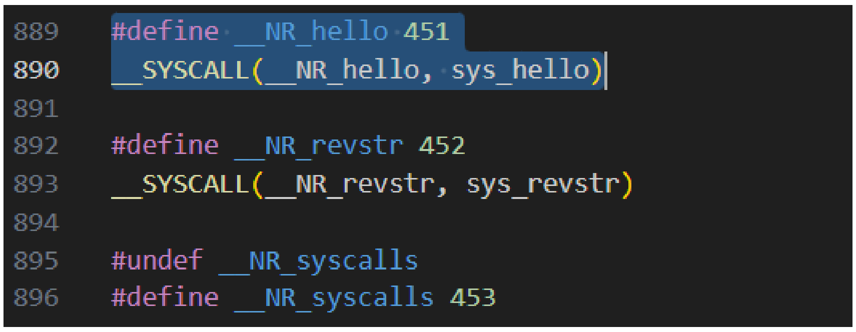
```
obj-y := hello.o
```

   b. Modify kernel's **Makefile** (in linux-5.19.12), add our system call module in it. Ensure our system call file can be found.



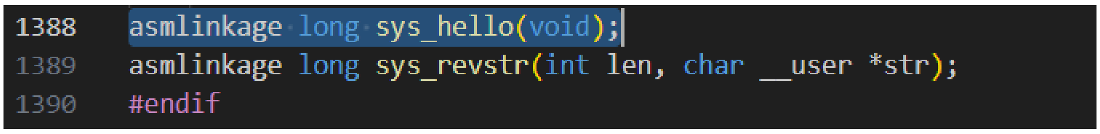
```
ifeq ($(KBUILD_EXTMOD),)
core-y                  += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ hello/ revstr/
core-$(CONFIG_BLOCK)    += block/
core-$(CONFIG_IO_URING) += io_uring/
```

c. Move to linux-5.19.12/include/uapi/asm-generic, modify the file of **unistd.h.** add system call into this table, define our sys_hello. and change __NR_syscalls number to the last one.



```
#define __NR_hello 451
__SYSCALL(__NR_hello, sys_hello)

#define __NR_revstr 452
__SYSCALL(__NR_revstr, sys_revstr)

#undef __NR_syscalls
#define __NR_syscalls 453
```

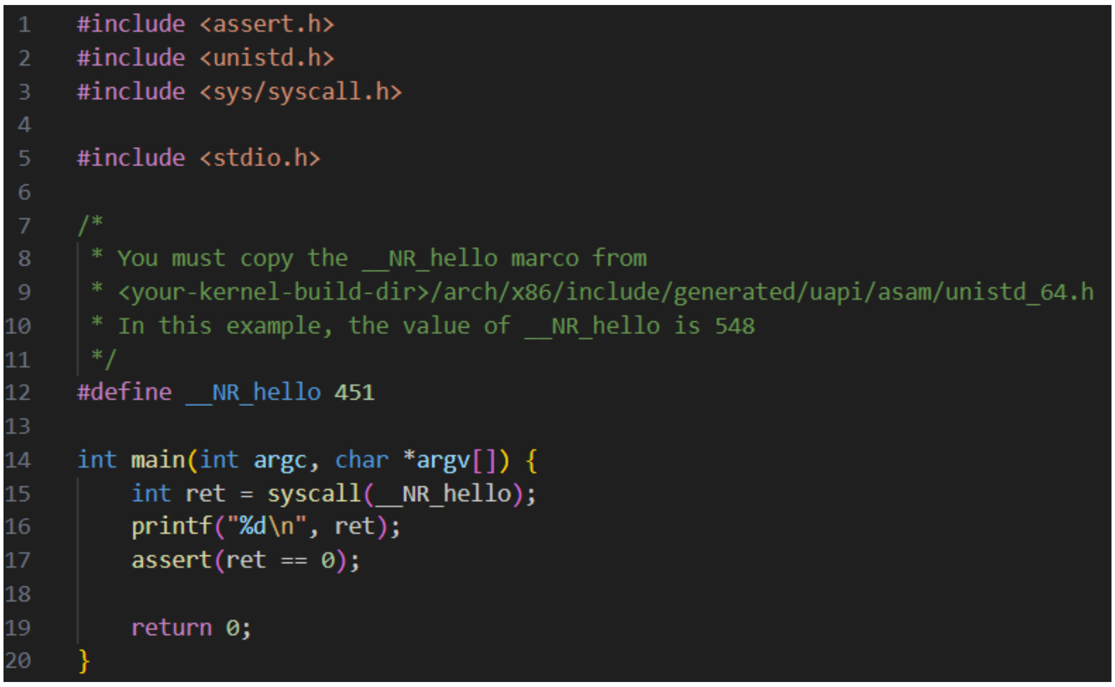
d. Move to linux-5.19.12/include/linux, modify the file of **syscalls.h**. Add our system call to kernel header file.



```
asmlinkage long sys_hello(void);
asmlinkage long sys_revstr(int len, char __user *str);
#endif
```

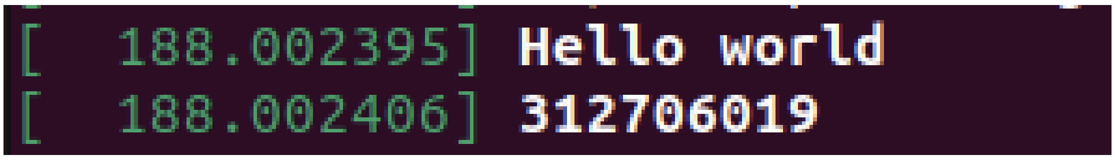
e. Then compile our kernel again and reboot.

f. Move to user program: create the file name **test.c**, compile it to invoke the system call sys_hello



```
#include <assert.h>
#include <unistd.h>
#include <sys/syscall.h>

#include <stdio.h>

/*
 * You must copy the __NR_hello marco from
 * <your-kernel-build-dir>/arch/x86/include/generated/uapi/asam/unistd_64.h
 * In this example, the value of __NR_hello is 548
 */
#define __NR_hello 451

int main(int argc, char *argv[]) {
    int ret = syscall(__NR_hello);
    printf("%d\n", ret);
    assert(ret == 0);

    return 0;
}
```

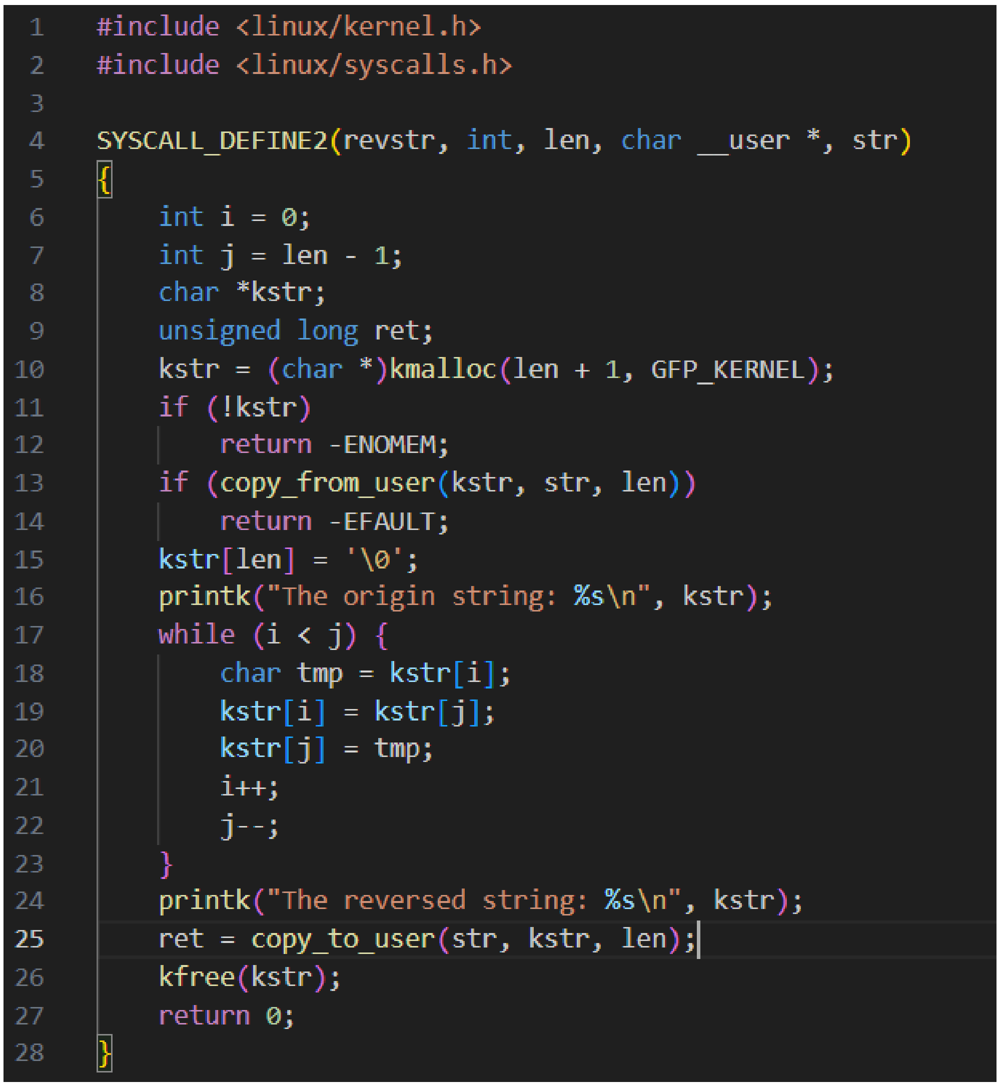
g. Use dmesg command, show the screenshot of the messages the system call printed:



```
[  188.002395] Hello world
[  188.002406] 312706019
```

2. sys_revstr:

   a. create the folder revstr to store system call files. (in linux-5.19.12)
      Then, create a file **revstr.c** to implement our system call



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

SYSCALL_DEFINE2(revstr, int, len, char __user *, str)
{
    int i = 0;
    int j = len - 1;
    char *kstr;
    unsigned long ret;
    kstr = (char *)kmalloc(len + 1, GFP_KERNEL);
    if (!kstr)
        return -ENOMEM;
    if (copy_from_user(kstr, str, len))
        return -EFAULT;
    kstr[len] = '\0';
    printk("The origin string: %s\n", kstr);
    while (i < j) {
        char tmp = kstr[i];
        kstr[i] = kstr[j];
        kstr[j] = tmp;
        i++;
        j--;
    }
    printk("The reversed string: %s\n", kstr);
    ret = copy_to_user(str, kstr, len);
    kfree(kstr);
    return 0;
}
```

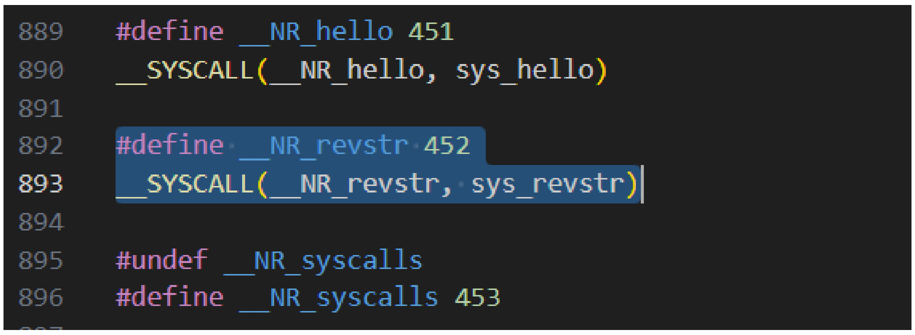
      And create a **Makefile** for our system call, also in the same revstr file:

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

   b. Modify kernel's **Makefile** (in linux-5.19.12), add our system call
      module in it. Ensure our system call file can be found.

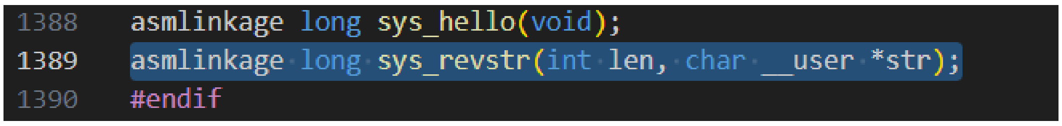
```
ifeq ($(KBUILD_EXTMOD),)
core-y                  += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ hello/ revstr/
core-$(CONFIG_BLOCK)    += block/
core-$(CONFIG_IO_URING) += io_uring/
```

c. Move to linux-5.19.12/include/uapi/asm-generic, modify the file of **unistd.h.** add system call into this table, define our sys_revstr. and change __NR_syscalls number to the last one.



```
#define __NR_hello 451
__SYSCALL(__NR_hello, sys_hello)

#define __NR_revstr 452
__SYSCALL(__NR_revstr, sys_revstr)

#undef __NR_syscalls
#define __NR_syscalls 453
```

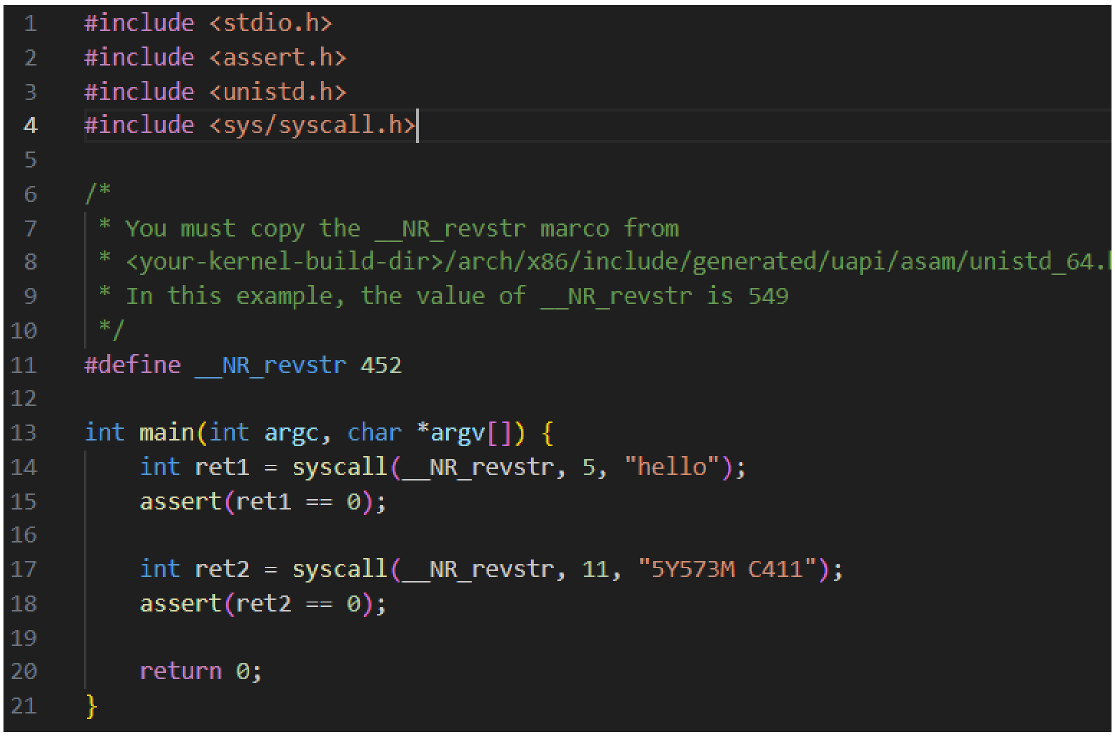
d. Move to linux-5.19.12/include/linux, modify the file of **syscalls.h**. Add our system call to kernel header file.



```
asmlinkage long sys_hello(void);
asmlinkage long sys_revstr(int len, char __user *str);
#endif
```

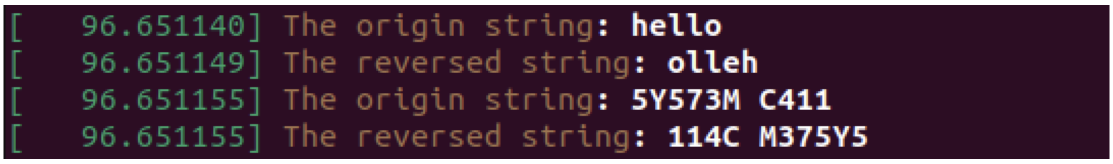
e. Then compile our kernel again and reboot.

f. Move to user program: create the file name **test_rev.c**, compile it to invoke the system call sys_revstr:



```
#include <stdio.h>
#include <assert.h>
#include <unistd.h>
#include <sys/syscall.h>

/*
 * You must copy the __NR_revstr marco from
 * <your-kernel-build-dir>/arch/x86/include/generated/uapi/asam/unistd_64.
 * In this example, the value of __NR_revstr is 549
 */
#define __NR_revstr 452

int main(int argc, char *argv[]) {
    int ret1 = syscall(__NR_revstr, 5, "hello");
    assert(ret1 == 0);

    int ret2 = syscall(__NR_revstr, 11, "5Y573M C411");
    assert(ret2 == 0);

    return 0;
}
```

g. Use dmesg command, show the screenshot of the messages the system call printed:
the screenshot of the messages the system call printed:



```
[   96.651140] The origin string: hello
[   96.651149] The reversed string: olleh
[   96.651155] The origin string: 5Y573M C411
[   96.651155] The reversed string: 114C M375Y5
```