

Part 1: Kernel Compilation

In part 1, we need to change the kernel version to “-os-id”. The current Linux kernel .config was copied as our kernel, then the CONFIG_LOCAL_VERSION was modified as requested. Also, the KEYS were changed and Linux headers need to be reinstalled again. After kernel made and restart, the result should be shown as below Figure 1.

1. After unzip the downloaded kernel to corresponding directory, change authorizes to prevent authorized problem, where my username is don.

```
chown -R don:don linux-5.19.12
```

2. Make config file, copy current Linux release(uname -r) config file to our kernel build directory

```
cd linux-5.19.12/  
make mrproper  
cp -v /boot/config-$(uname -r) .config
```

3. Make config file

```
make menuconfig
```

4. Change version, CONFIG_LOCALVERSION to change suffix, CONFIG_SYSTEM_TRUSTED_KEYS a PEM-encoded file for additional certificates, CONFIG_SYSTEM_REVOCATION_KEYS include X.509 certificates

```
vim .config  
/\<CONFIG_LOCALVERSION\>  
# change "" to "-os-id"  
CONFIG_LOCALVERSION = "-os-id"  
CONFIG_SYSTEM_TRUSTED_KEYS=""  
CONFIG_SYSTEM_REVOCATION_KEYS=""
```

5. Check version before execute, CONFIG_LOCALVERSION_AUTO=y means add suffix

```
sudo apt install --reinstall linux-headers-$(uname -r)  
make oldconfig && make prepare  
make CONFIG_LOCALVERSION_AUTO=y kernelrelease
```

6. Install kernel and modules

```
make -j 4  
sudo make modules_install  
sudo make install
```

7. Install grub properly to prevent boot issue

```
sudo grub-install /dev/sda  
sudo update-grub
```

8. Restart and check version

```
uname -a  
cat /etc/os-release
```

```

don@don-VirtualBox:~/Don$ uname -a
Linux don-VirtualBox 5.19.12-os-310551003 #11 SMP PREEMPT_DYNAMIC Tue Oct 18 22:
25:12 CST 2022 x86_64 x86_64 x86_64 GNU/Linux
don@don-VirtualBox:~/Don$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.1 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.1 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-poli
cy"
UBUNTU_CODENAME=jammy

```

Figure 1 kernel compilation part – screenshot of the results

Part 2: System Call

In this part, we would like to add 2 new system calls to our kernel.

Task 1: add print “Hello, world!” and student ID “310551003” system call to kernel.

1. Restart computer and go to original Linux kernel and we can modify our own kernel.
2. As provided sample C code, the new system call “__NR_hello” was defined to 548. Looking into “linux-5.19.12/arch/x86/entry/syscalls/syscall_64.tbl” which is Linux System Call Table contains all system call number and their corresponding system call function name. That is, the new system call is also need to be added on this table so that the system call number 548 will match to corresponding system call function.

```

vim ~/Don/kernelbuild/linux-5.19.12/arch/x86/entry/syscalls/syscall_64.tbl
548    common hello                sys_hello

```

3. Add system call function “asmlinkage long sys_hello(void);” at system call header file “linux-5.19.12/include/linux/syscalls.h”

```

vim include/linux/syscalls.h
asmlinkage long sys_hello(void);

```

4. Add system call function code, in corresponding kernel file. In my case, a new hello folder was created and hello.c code was created and the content was shown as below:
Where DEFINE0 mean 0 parameter will be requested to provide, and printk is the print function used in Linux kernel, we add “Hello, world!” and my student ID “310551003” accordingly, so that, when sys_hello was called, it will print these 2 strings accordingly.

```

cd ~/Don/kernelbuild/linux-5.19.12/
mkdir hello
vim hello/hello.c

```

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

SYSCALL_DEFINE0(hello)
{
    printk("Hello, world!\n");
    printk("310551003\n");
    return 0;
}
```

5. Add a Makefile for hello.c script inside hello folder and input "obj-y :=hello.c" for compile hello.o to kernel when execute Make command.

```
vim hello/Makefile
obj-y := hello.o
```

6. Add "hello/" to "core-y" row from the kernel Makefile, thus, it will contain the new create hello folder when Linux kernel Make.

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

7. Install kernel and restart Linux to check on our kernel.

```
sudo apt install --reinstall linux-headers-$(uname -r)
make oldconfig && make prepare
make CONFIG_LOCALVERSION_AUTO=y kernelrelease
make -j 4
sudo make modules_install
sudo make install
sudo grub-install /dev/sda
sudo update-grub
```

8. Use the provide sample code as below to test the system call

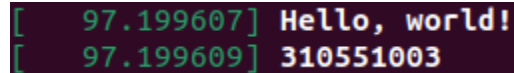
```
#include <assert.h>
#include <unistd.h>
#include <sys/syscall.h>

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

int main(int argc, char *argv[]) {
    int ret = syscall(__NR_hello);
    assert(ret == 0);

    return 0; }
```

9. The process of system call
Execute sample code call syscall → kernel looking into Linux System Call Table and find the system call function name → use system call function accordingly
10. Use sudo dmesg and check whether the result “Hello, world!” and “310551003” was shown as below Figure 2.



```
[ 97.199607] Hello, world!
[ 97.199609] 310551003
```

Figure 2 Result screenshot of system call printed messages

Task 2: provide length of string and string, utilize system call to output string and its reverse string.

1. Restart computer and go to original Linux kernel and we can modify our own kernel.
2. As provided sample C code, the new system call “__NR_revstr” was defined to 548. Looking into “linux-5.19.12/arch/x86/entry/syscalls/syscall_64.tbl” which is Linux System Call Table contains all system call number and their corresponding system call function name. That is, the new system call is also need to be added on this table so that the system call number 548 will match to corresponding system call function.

```
vim ~/Don/kernelbuild/linux-5.19.12/arch/x86/entry/syscalls/syscall_64.tbl
549    common revstr                sys_revstr.c
```

3. Add system call function “asmlinkage long sys_revstr(int str_len, char __user *input_str);” at system call header file “linux-5.19.12/include/linux/syscalls.h”

```
vim include/linux/syscalls.h
asmlinkage long sys_revstr(int str_len, char __user *input_str);
```

4. Add system call function code, in corresponding kernel file. In my case, a new revstr folder was created and revstr.c code was created and the content was shown as below:
Where DEFINE2 mean 2 parameters will be requested to input, char buf is char array which I used to store the data from user space by using copy_from_user function and return -EFAULT if error, create a for loop for store the char array in origin sequence and reverse sequence. Add a null character '\0' to the end of char array to imply the end of string while printing.

```
cd ~/Don/kernelbuild/linux-5.19.12/
mkdir revstr
vim revstr/revstr.c
```

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

SYSCALL_DEFINE2(revstr, int, len, char __user *, str)
{
    char buf[256], nor[256], rev[256];
    if(copy_from_user(buf, str, len)) return -EFAULT;
    for(int i=len-1, j=0; i>=0 ; i--, j++){
```

```

        rev[j] = buf[i];
        nor[j] = buf[j];
        printk("rev[j] nor[j]\n");
    }
    rev[len] = '\0';
    nor[len] = '\0';
    printk("The origin string: %s\n", nor);
    printk("The reversed string: %s\n", rev);
    return 0;
}

```

5. Add a Makefile for hello.c script inside hello folder and input "obj-y := revstr.c" for compile revstr.o to kernel when execute Make command.

```

vim revstr/Makefile
obj-y := revstr.o

```

6. Add "revstr/" to "core-y" row from the kernel Makefile, thus, it will contain the new create hello folder when Linux kernel Make.

```

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

```

7. Install kernel and restart Linux to check on our kernel.

```

sudo apt install --reinstall linux-headers-$(uname -r)
make oldconfig && make prepare
make CONFIG_LOCALVERSION_AUTO=y kernelrelease
make -j 4
sudo make modules_install
sudo make install
sudo grub-install /dev/sda
sudo update-grub

```

8. Use the provide sample code as below to test the system call

```

#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.h
 * In this example, the value of __NR_revstr is 549
 */
#define __NR_revstr 549

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);
}

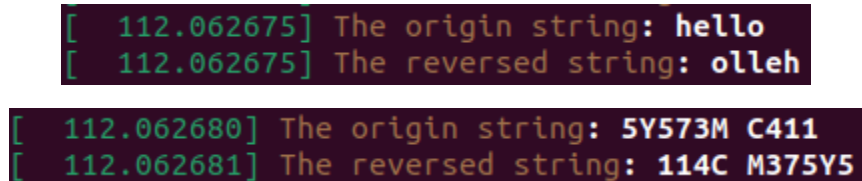
```

```
return 0;  
}
```

9. The process of system call

Execute sample code call syscall → kernel looking into Linux System Call Table and find the system call function name → use system call function accordingly

10. Use sudo dmesg and check whether the result origin and reversed string was shown as below Figure 3.



```
[ 112.062675] The origin string: hello  
[ 112.062675] The reversed string: olleh  
  
[ 112.062680] The origin string: 5Y573M C411  
[ 112.062681] The reversed string: 114C M375Y5
```

Figure 3 result of origin and reversed string

This homework is to familiar how to create a kernel, add and utilize system call from user space accordingly, understand the system call and user space relationship and implement it.