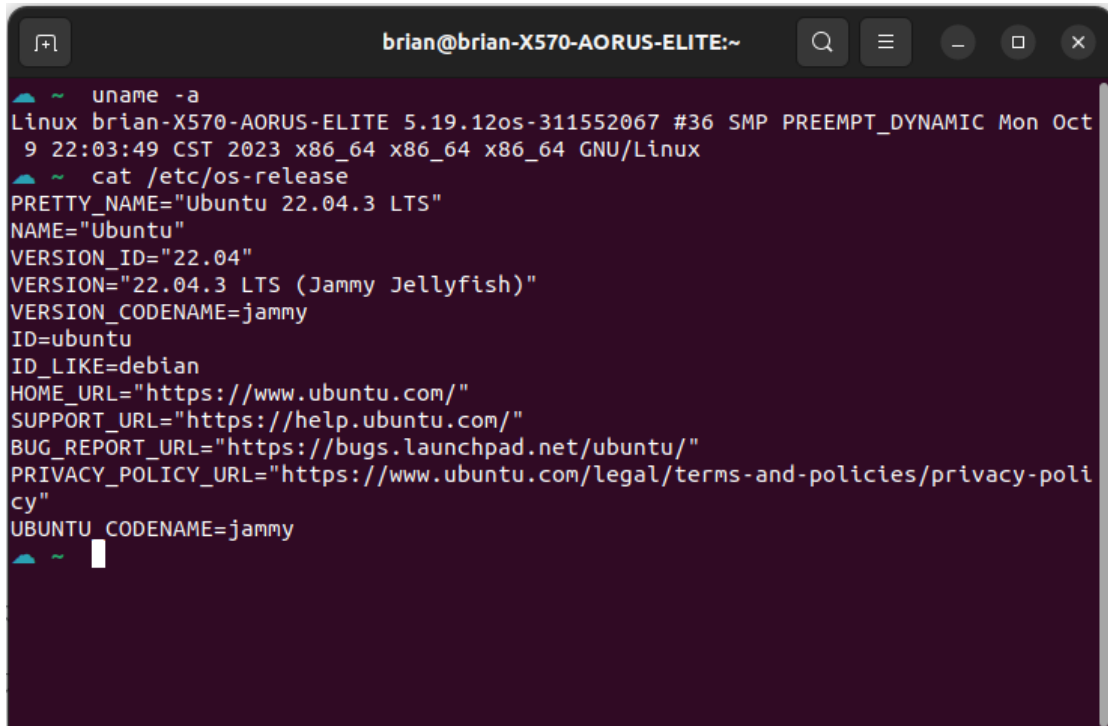


Assignment 1: Compiling Linux Kernel and Adding Custom System Calls

311552067 劉承熙

1. results of executing `uname -a` and `cat /etc/os-release` commands

A terminal window with a dark purple background and light blue text. The window title is 'brian@brian-X570-AORUS-ELITE:~'. The terminal shows the output of two commands: 'uname -a' and 'cat /etc/os-release'.

```
brian@brian-X570-AORUS-ELITE:~  
~ uname -a  
Linux brian-X570-AORUS-ELITE 5.19.12os-311552067 #36 SMP PREEMPT_DYNAMIC Mon Oct  
9 22:03:49 CST 2023 x86_64 x86_64 x86_64 GNU/Linux  
~ 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-poli  
cy"  
UBUNTU_CODENAME=jammy  
~
```

2. Which kernel sources did you modify? What do they do?

1. Add syscall function code and Makefile for each new syscalls
 - Makefile is to ensure that the c file is compiled and included in the kernel source code
2. Add new syscall folder in Makefile of Linux Kernel
 - This is to tell the compiler that the source files of new syscalls are in present in the new syscall directory
3. Add new syscall to syscall table_64
 - Define syscall number for syscall function and name
4. Add new syscall to syscall header file
 - Defines the prototype of the function of system call
5. Compile kernel code, install and update kernel

3. Each system call implemented (Source code)

Add syscall function code and Makefile for each new syscalls

```
// linux-5.19.12/hello/hello.c
#include <linux/kernel.h>
#include <linux/syscalls.h>
SYSCALL_DEFINE0(hello){
    printk("Hello world!\n");
    printk("311552067\n");
    return 0;
}
```

```
// linux-5.19.12/hello/Makefile
obj-y := hello.o
```

```
// linux-5.19.12/linux-5.19.12/revstr/revstr.c
#include <linux/kernel.h>
#include <linux/syscalls.h>
#include <linux/string.h>
#include <linux/uaccess.h>
SYSCALL_DEFINE2(revstr, int, len, char __user *, src){
    char str_in[100];
    char reverse[100];
    if( copy_from_user(str_in, src, len)){
        return -EFAULT;
    }
    for(int i=0; i<len; i++){
        reverse[i] = str_in[(len-1) - i];
    }
    str_in[len] = '\0';
    reverse[len] = '\0';
    printk("The origin string: %s\n", str_in);
    printk("The reversed string: %s\n", reverse);
    return 0;
}
```

```
// linux-5.19.12/revstr/Makefile
obj-y := revstr.o
```

Add new syscall folder in Makefile of Linux Kernel

```
// linux-5.19.12/Makefile
core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/

//add new syscalls folder(hello/ revstr/)

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

Add new syscall to syscall table_64

```
// linux-5.19.12/arch/x86/entry/syscalls/syscall_64.tbl
548 64 hello          sys_hello
549 64 revstr         sys_revstr
```

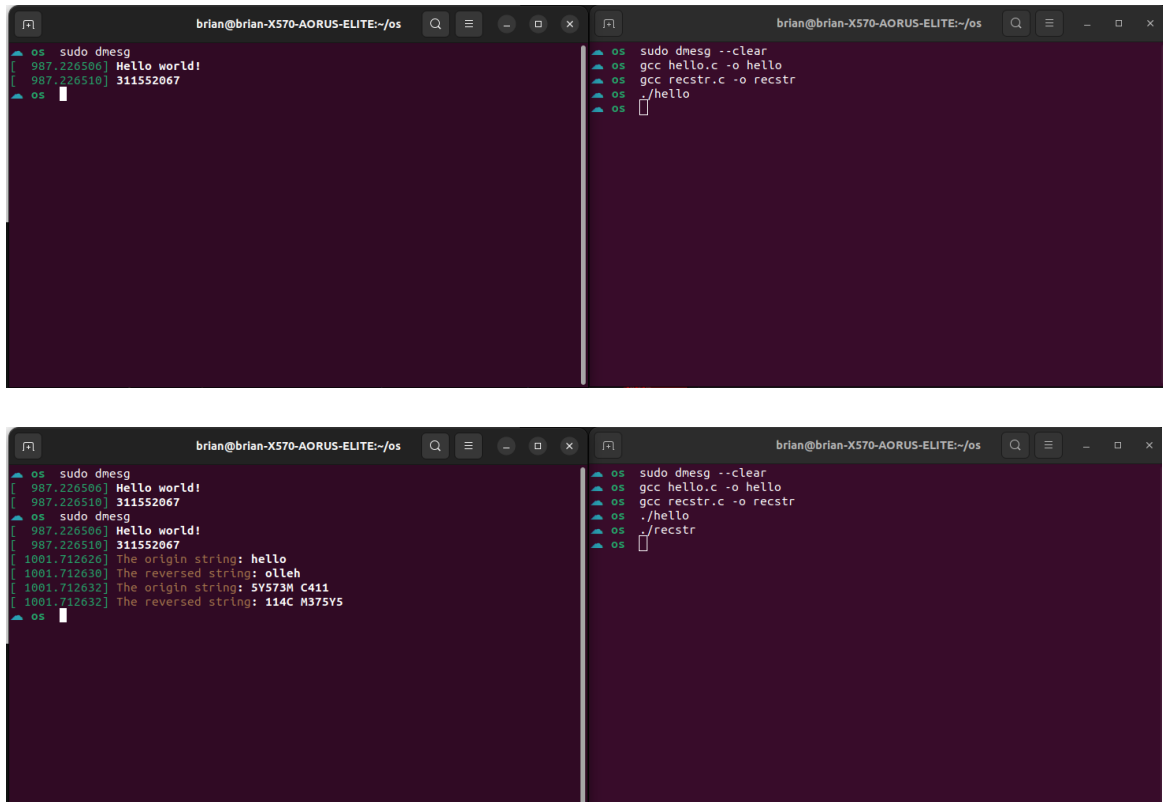
Add new syscall to syscall header file

```
// linux-5.19.12/include/linux/syscalls.h
asmlinkage long sys_hello(void);
asmlinkage long sys_revstr(int len, char __user *src);
```

compile

```
sudo make
sudo make modules_install
sudo make install
```

4. Result



The image displays two terminal windows side-by-side, both running on a system identified as 'brian@brian-X570-AORUS-ELITE:~/os'.

Top Left Terminal:

```
os sudo dmesg
[ 987.226506] Hello world!
[ 987.226510] 311552067
os
```

Top Right Terminal:

```
os sudo dmesg --clear
os gcc hello.c -o hello
os gcc recstr.c -o recstr
os ./hello
os
```

Bottom Left Terminal:

```
os sudo dmesg
[ 987.226506] Hello world!
[ 987.226510] 311552067
os sudo dmesg
[ 987.226506] Hello world!
[ 987.226510] 311552067
[ 1001.712626] The origin string: hello
[ 1001.712630] The reversed string: olleh
[ 1001.712632] The origin string: 5Y573M C411
[ 1001.712632] The reversed string: 114C M375Y5
os
```

Bottom Right Terminal:

```
os sudo dmesg --clear
os gcc hello.c -o hello
os gcc recstr.c -o recstr
os ./hello
os ./recstr
os
```