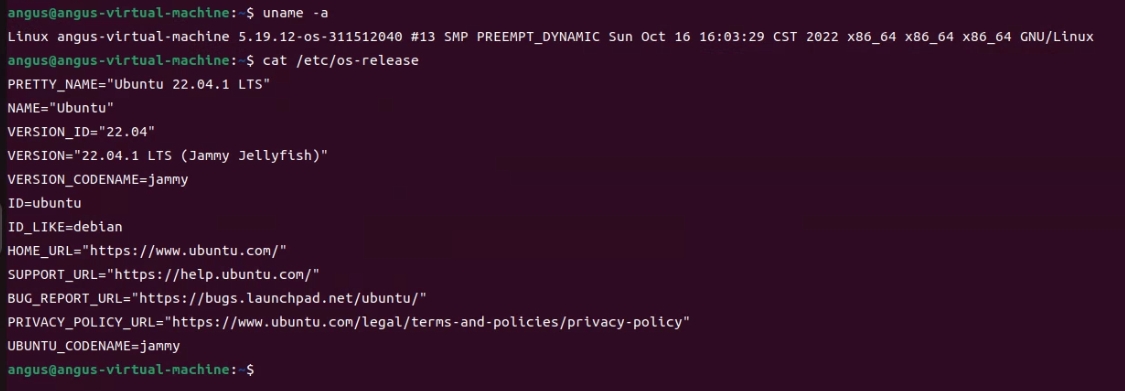
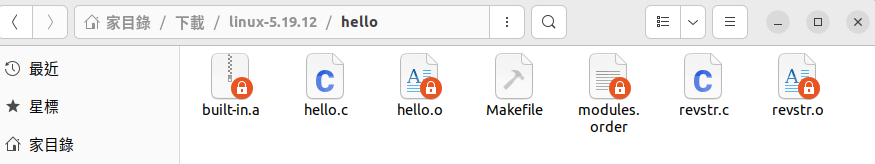
# kernel compilation and system call

* kernel compilation**:**

## system call:

### ****Define a new System Call, sys\_hello(),sys\_revstr()****

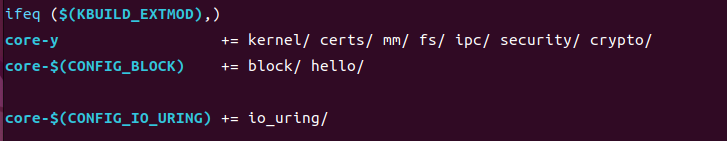
Switch into the compiled source directory by cd linux-5.19.12/. Create a new directory, mkdir hello and change into the directory by cd hello. Consequently, create new C files by vim hello.c and vim revstr.c in this directory in order to add the definition of our system call.



### Add the directory to the kernel’s Makefile

Create a new Makefile in the same folder by vim Makefile with the following line obj-y += hello.o and obj-y += revstr.o  This Makefile specifies the objects to be built and added to the source during the next kernel recompilation. We also need to male sure the parent Makefile points to this directory. So back to linux-5.19.12/ directory and edit the parent Makefile by vim Makefile.

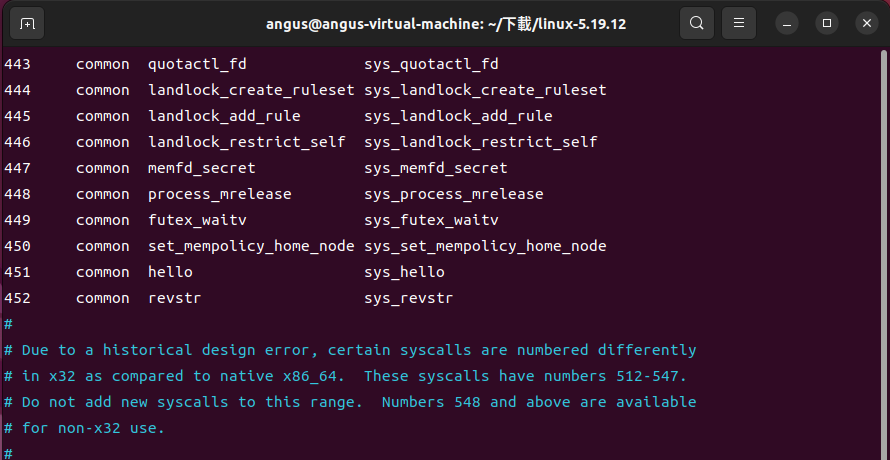
Find core-y and append the hello/ directory:



This amendment tell the compiler that the source files of our new system calls are in the hello/ directory.

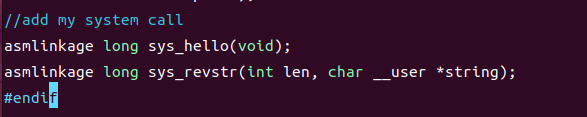
### Add the new system call into the System Call table

Edit the table by vim arch/x86/entry/syscalls/syscall\_64.tbl ,and add my system calls with number 451,452.



### Add the new System Call in the System Call header file

Change directory with cd include/linux/ Furthermore, vim syscalls.h and add asmlinkage long sys\_hello(void); and asmlinkage long sys\_revstr(int len, char \_\_user \*string); at the end of the file just before the #endif statement.



This defines the prototype of the function of our System Call. asmlinkage is a keyword used to indicate that all parameters of the function would be available on the stack.

### Recompile the kernel

Switch to the source directory linux-5.19.12/. Execute the following commands in sequence on the terminal:

sudo make -j &(nproc)

sudo make modules

sudo make modules\_install

sudo make install

For the system to now use the newly configured kernel, reboot.

### Sorce code

**Add the following code to hello.c file:**

# include <linux/kernel.h>

# include <linux/syscalls.h>

SYSCALL\_DEFINE0(hello)

{

printk("Hello world!\n");

printk("311512040\n");

return 0;

}

**Add the following code to revstr.c file:**

#include <linux/kernel.h>

#include <linux/syscalls.h>

#include <linux/linkage.h>

#include <linux/uaccess.h>

/\* syscall number 452 \*/

SYSCALL\_DEFINE2(revstr, int, len, char \_\_user \*, string)

{

char str[200]; // declare the size of character string

unsigned long strlen = len;

char temp;

copy\_from\_user(str, string, strlen);

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

for (int i = 0; i <(len/2); i++)

{

temp = str[i];

str[i] = str[strlen - i - 1];

str[strlen - i - 1] = temp;

}

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

return 0;

}

### System call printed(compile and run the code attached by the TA):

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述