kernel compilation and system call

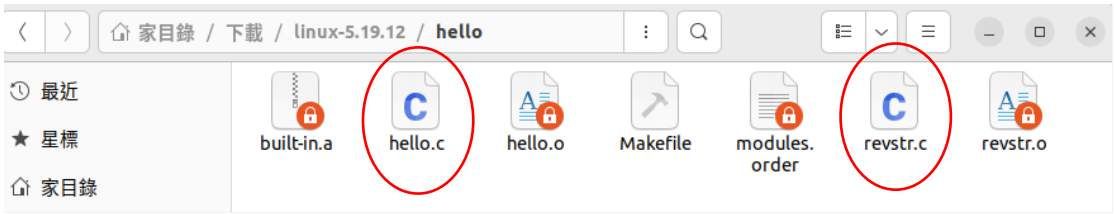
- **kernel compilation:**

```
angus@angus-virtual-machine:~$ uname -a
Linux angus-virtual-machine 5.19.12-os-311512040 #13 SMP PREEMPT_DYNAMIC Sun Oct 16 16:03:29 CST 2022 x86_64 x86_64 x86_64 GNU/Linux
angus@angus-virtual-machine:~$ 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-policy"
UBUNTU_CODENAME=jammy
```

- **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:

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

core-$(CONFIG_IO_URING) += io_uring/
```

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.

```
angus@angus-virtual-machine: ~/下载/linux-5.19.12
443     common  quotactl_fd             sys_quotactl_fd
444     common  landlock_create_ruleset sys_landlock_create_ruleset
445     common  landlock_add_rule       sys_landlock_add_rule
446     common  landlock_restrict_self  sys_landlock_restrict_self
447     common  memfd_secret            sys_memfd_secret
448     common  process_mrelease        sys_process_mrelease
449     common  futex_waitv             sys_futex_waitv
450     common  set_mempolicy_home_node sys_set_mempolicy_home_node
451     common  hello                   sys_hello
452     common  revstr                  sys_revstr
```

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.

```
//add my system call
asmlinkage long sys_hello(void);
asmlinkage long sys_revstr(int len, char __user *string);
#endif
```

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):

```
[15247.483018] Hello world!
[15247.483022] 311512040
```

```
[  550.172890] The origin string: hello
[  550.172893] The reversed string: olleh
[  550.172893] The origin string: 5Y573M C411
[  550.172894] The reversed string: 114C M375Y5
```