Operating System Assignment 1:

Compiling Linux Kernel and Adding Custom System Calls

周睦鈞 311553060

For the kernel compilation:

1. Paste the screenshot of the results of executing `uname -a` and `cat /etc/os-release` commands as the example shows

```
michael@OperatingSystem: ~
michael@OperatingSystem: ~$ uname -a
Linux OperatingSystem 5.19.12-os-311553060 #1 SMP PREEMPT_DYNAMIC Tue Oct 24 23:51:43 CST 2023 x86_64 x86_64 x86_64 GNU/Linux
michael@OperatingSystem: ~$ 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:

1. Describe how you implemented the two system calls in detail. Which kernel sources did you modified? What do they do?

1-1. Change the working directory to the root directory of the recently unpacked source code

```
$ cd linux-5.19.12/
```

1-2. Create the directory `mycall` for system call and create two C files `hello.c` and `revstr.c`

```
michael@OperatingSystem: ~/linux-5.19.12/mycall
michael@OperatingSystem:~/linux-5.19.12/mycall$ ls *.c
hello.c  revstr.c
```

```
$ vim hello.c
```

```
michael@OperatingSystem: ~/linux-5.19.12/mycall
#include <linux/syscalls.h>
#include <linux/kernel.h>

SYSCALL_DEFINE0(hello) {
    printk("Hello, world!\n");
    printk("311553060\n");

    return 0;
}
```

```
$ vim revstr.c
```

```
michael@OperatingSystem: ~/linux-5.19.12
#include <linux/syscalls.h>
#include <linux/kernel.h>
#include <linux/uaccess.h>

SYSCALL_DEFINE2(revstr, int, length, char __user *, str) {
    char original_string[256];  // Assuming a maximum string length of 256.
    char reversed_string[256];
    int i;

    // Copy the user-space string to kernel space.
    if (copy_from_user(original_string, str, length)) {
        return -EFAULT; // Error handling
    }

    original_string[length] = '\0';


    // Reverse the string.
    for (i = 0; i < length; i++) {
        reversed_string[i] = original_string[length - i - 1];
    }

    // Null-terminate the reversed string.
    reversed_string[length] = '\0';

    printk("The origin string: %s\n", original_string);
    printk("The reversed string: %s\n", reversed_string);

    return 0;
}
```

1-3. Create a Makefile for my system call

Make sure our code will be compile into Kernel

```
michael@OperatingSystem: ~/linux-5.19.12/mycall
michael@OperatingSystem:~/linux-5.19.12/mycall$ cat Makefile
obj-y := hello.o revstr.o
```

1-4. Add home directory of my system to main Makefile of the kernel

Search for the `core-y` directive and append the directory for my system call, named `mycall/`, at the end, indicating to the compiler the location of our new system call.

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

1-5. Add a corresponding function prototype for my system call to the header file of system calls

```
vim include/linux/syscalls.h
```

Navigate to the bottom of it and write the following code just above `#endif`

The term `asmlinkage` signifies that our parameters are accessible on the stack.

```
asmlinkage long sys_hello(void);
asmlinkage long sys_revstr(int length, char __user *str);

#endif
```

1-6. Add my system call to the kernel's system call table

```
vim arch/x86/entry/syscalls/syscall_64.tbl
```

Add the following code at the end of this file. Use Tab for space

```
548     common  hello                   sys_hello
549     common  revstr                  sys_revstr
```

1-7. Compile the kernel's source code and install the kernel

```
sudo make -j4 && sudo make -j4 modules_install && sudo make -j4 install
```

1-8. Update the bootloader of the operating system with new kernel

```
sudo update-grub
```

1-9. Reboot the computer

```
reboot
```

2. For each system call you implemented:

2-1. Paste the **well-formatted** source code into the document

When implementing the system call function, `SYSCALL_DEFINE0` indicates that the system call function does not take any parameters from user space. `SYSCALL_DEFINE2` signifies that the system call function requires two parameters from user space

`hello.c`

```
michael@OperatingSystem: ~/linux-5.19.12/mycall
#include <linux/syscalls.h>
#include <linux/kernel.h>

SYSCALL_DEFINE0(hello) {
    printk("Hello, world!\n");
    printk("311553060\n");

    return 0;
}
```

`revstr.c`

Since the data from user space are not shared with the kernel space.

As a result, the function `copy_from_user` is used to copy data, such as strings, from user space to kernel space.

```
michael@OperatingSystem: ~/linux-5.19.12
#include <linux/syscalls.h>
#include <linux/kernel.h>
#include <linux/uaccess.h>

SYSCALL_DEFINE2(revstr, int, length, char __user *, str) {
    char original_string[256];  // Assuming a maximum string length of 256.
    char reversed_string[256];
    int i;

    // Copy the user-space string to kernel space.
    if (copy_from_user(original_string, str, length)) {
        return -EFAULT; // Error handling
    }

    original_string[length] = '\0';


    // Reverse the string.
    for (i = 0; i < length; i++) {
        reversed_string[i] = original_string[length - i - 1];
    }

    // Null-terminate the reversed string.
    reversed_string[length] = '\0';

    printk("The origin string: %s\n", original_string);
    printk("The reversed string: %s\n", reversed_string);

    return 0;
}
```

2-2. Paste the screenshot of the messages the system call printed

```
michael@OperatingSystem: ~/Desktop/HW1
michael@OperatingSystem:~/Desktop/HW1$ ./hello
michael@OperatingSystem:~/Desktop/HW1$ sudo dmesg
[  238.510084] Hello, world!
[  238.510088] 311553060
```

```
michael@OperatingSystem: ~/Desktop/HW1
michael@OperatingSystem:~/Desktop/HW1$ ./revstr
michael@OperatingSystem:~/Desktop/HW1$ sudo dmesg
[  267.245129] The origin string: hello
[  267.245134] The reversed string: olleh
[  267.245135] The origin string: 5Y573M C411
[  267.245136] The reversed string: 114C M375Y5
```