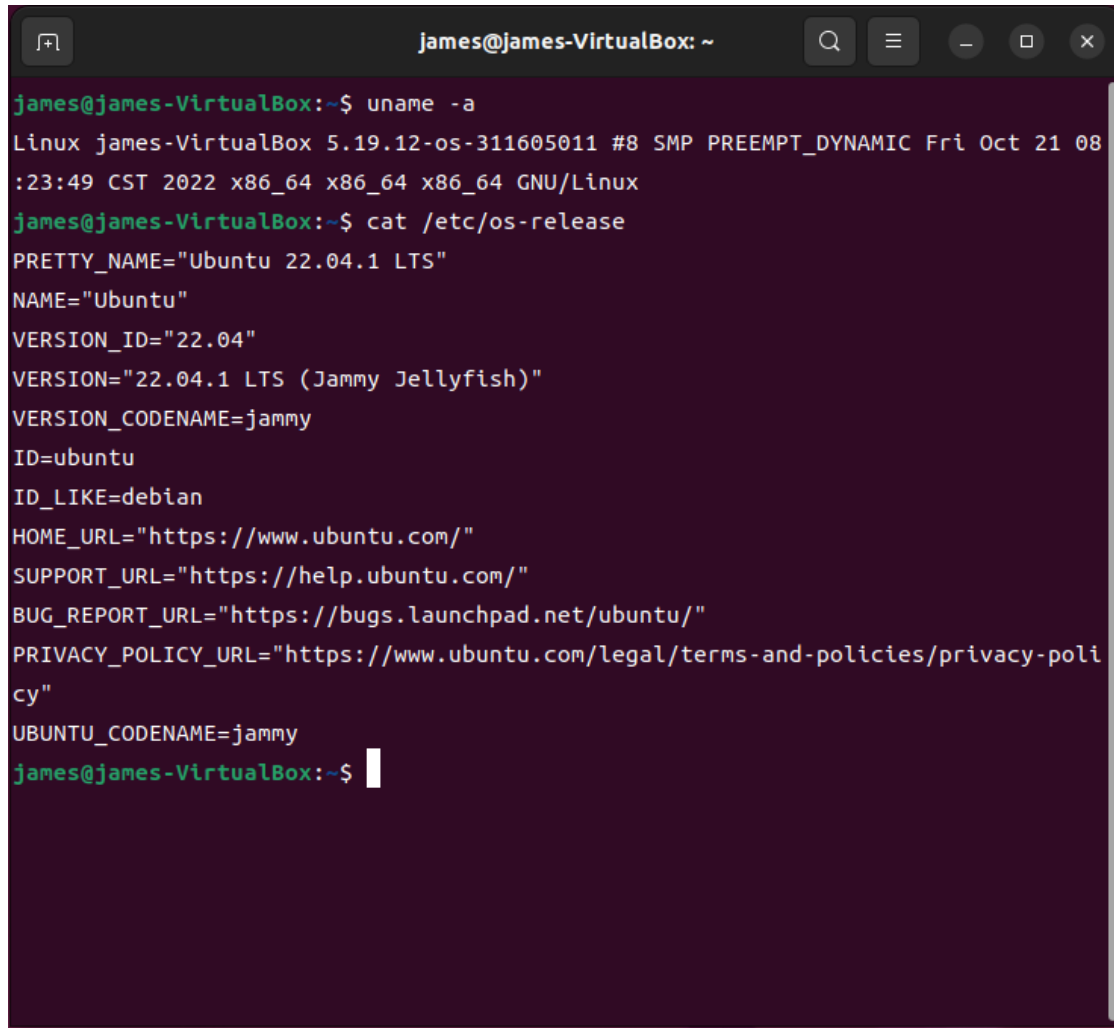


OS Assignment1

311605011 黃品振

一、 Kernel compilation

A terminal window titled 'james@james-VirtualBox: ~' with standard window controls. It shows the output of 'uname -a' and 'cat /etc/os-release'.

```
james@james-VirtualBox:~$ uname -a
Linux james-VirtualBox 5.19.12-os-311605011 #8 SMP PREEMPT_DYNAMIC Fri Oct 21 08
:23:49 CST 2022 x86_64 x86_64 x86_64 GNU/Linux
james@james-VirtualBox:~$ 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
james@james-VirtualBox:~$
```

二、 System calls

First system call:

第一個 system call 比較單純只需要將一個字串列印出來，步驟如下:

1. 將程式碼寫好，這裡有個需要注意的地方是不能用 `printf()`，要用 `printk()` 去印字串，因為 `printk()` 是從 kernel 端印出，還有因為第一個 system call 不需要傳入任何東西，所以是 `DEFINE0`。
2. 進入 `linux-5.19.12/kernel`，將我們寫好的.c 檔(hello.c)放到裡面，然後修

改同一個資料夾內的 Makefile，在 obj-y 後面將我們想編譯的檔案名從.c 變成.o 加入，如下所示(hello.o):

```
obj-y      = fork.o exec_domain.o panic.o \
             cpu.o exit.o softirq.o resource.o \
             sysctl.o capability.o ptrace.o user.o \
             signal.o sys.o umh.o workqueue.o pid.o task_work.o \
             extable.o params.o platform-feature.o \
             kthread.o sys_ni.o nsproxy.o \
             notifier.o ksysfs.o cred.o reboot.o \
             async.o range.o smpboot.o ucount.o regset.o hello.o
```

3. 接著進入 linux-5.19.12/ arch/ x86/ entry/ syscalls 修改 syscall_64.tbl，將我們的新增的 system call "sys_hello()" 到 system call 的 table，如下所示：

```
548      common  hello                      sys_hello
```

4. 新增新的 system call "sys_hello()" 到 system call 的標頭檔，如下所示：
`asmlinkage long sys_hello(void);`
5. 最後重新 compile kernel，system call 就加進去了
6. 成果圖：

```
[16168.106849] Hello world
[16168.106853] 311605011
```

Second system call:

第二個 system call 的方法跟第一個大同小異，比較需要注意的有兩個，第一個是程式的寫法，第二個是標頭檔那邊要將 void 改掉，詳細步驟如下：

1. 將程式碼寫好，因為操作時會從傳入兩個引數，所以需要將 DEFINE0 改成 DEFINE2，還有一個地方需要注意，因為傳入的引數是從 user 端傳入，所以需要 call 一個 copy_from_user() 的函示將其轉換到 kernel 端。
2. 進入 linux-5.19.12/kernel，將我們寫好的.c 檔(reverse.c)放到裡面，然後修改同一個資料夾內的 Makefile，在 obj-y 後面將我們想編譯的檔案名從.c 變成.o 加入，如下所示(reverse.o):

```
obj-y      = fork.o exec_domain.o panic.o \
             cpu.o exit.o softirq.o resource.o \
             sysctl.o capability.o ptrace.o user.o \
             signal.o sys.o umh.o workqueue.o pid.o task_work.o \
             extable.o params.o platform-feature.o \
             kthread.o sys_ni.o nsproxy.o \
             notifier.o ksysfs.o cred.o reboot.o \
             async.o range.o smpboot.o ucount.o regset.o hello.o reverse.o
```

3. 接著進入 linux-5.19.12/ arch/ x86/ entry/ syscalls 修改 syscall_64.tbl，將我們的新增的 system call "sys_revstr()" 到 system call 的 table，如下所示：

```
549      common reverse      sys_revstr
```

4. 新增新的 system call "sys_revstr()" 到 system call 的標頭檔，這裡有個需要注意的地方，因為需要傳入兩個引數，所以要改成如下所示：

```
asm linkage long sys_revstr(char __user* str, int len);
```

5. 最後重新 compile kernel，system call 就加進去了
6. 成果圖：

```
[16771.142715] The origin string: hello
[16771.142739] The reversed string: olleh
[16771.142740] The origin string: 5Y573M C411
[16771.142740] The reversed string: 114C M375Y5
```

三、 System calls code:

- First system call:

```
#include <linux/kernel.h>
#include <linux/syscalls.h>
SYSCALL_DEFINE0(hello)
{
    printk("Hello world\n");
    printk("311605011");
    return 0;
}
```

- Second system call:

(*我這裡把跟助教的範本有點不同，助教寫的 test code 第一個放的是 string 的長度，第二個才是 string，我的版本第一個放的是 string，第二個放的是儲存字串的陣列長度，也就是字串長度+1*)

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

```
SYSCALL_DEFINE2(revstr, char __user *, src, int, len)
{
    char buf[256];
    unsigned long lenleft = len;
```

```

unsigned long klen = len - 1;
char temp;
unsigned long chunklen = sizeof(buf);
int i;
while(lenleft > 0)
{
    if(lenleft < chunklen) chunklen = lenleft;

    if(copy_from_user(buf, src, chunklen))
    {
        return -EFAULT;
    }
    lenleft -= chunklen;
}

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

for (i=0; i<(klen/2); i++)
{
    temp = buf[i];
    buf[i] = buf[klen - i - 1];
    buf[klen - i - 1] = temp;
}
printk("The reversed string: %s", buf);

return 0;
}

```