Operating Systems Project Report

Project Number (01 / 02 / 03):	02
Name:	陳宥安
Student ID:	109550073
YouTube link (Format youtube.com/watch?v=[key]):	https://youtu.be/QCXMKKt54Ns
Date (YYYY-MM-DD):	2021/11/25
Names of the files uploaded to E3:	K4_numericalTest.c, K4_syscall_64.tbl, K4_syscalls.h K5_numericalTest.c, K5_syscall_64.tbl, K5_syscalls.h syscallsNumerical.c, OS_Project02_109550073.pdf
Physical Machine Total RAM (Example: 8.0 GB):	16.0 GB
Physical Machine CPU (Example: Intel i7-2600K):	Intel(R) Core(TM) i5-8250U CPU @1.60GHz

Checklist	
Yes/No	**Item**
Yes	**The report name follows the format "OS_ProjectXX_StudentID.pdf".**
Yes	**The report was uploaded to E3 before the deadline.**
Yes	**The YouTube video is public, and anyone with the link can watch it.**
Yes	**The audio of the video has a good volume.**
Yes	**The pictures in your report and video have a good quality.**
Yes	**All the questions and exercises were answered inside the report.**
Yes	**I understand that late submission is late submission, regardless of the time uploaded.**
Yes	**I understand that any cheating in my report / video / code will not be tolerated.**

1. Screenshots

Section 1: Kernels Download / Build.

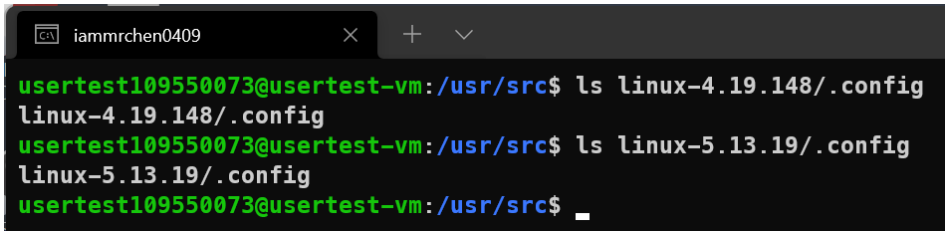
SCREENSHOT #1 Kernels

The screenshot shows two kernels (linux-4.19.148 / linux-5.13.19) are under the folder /usr/src.

```
usertest109550073@usertest-vm:/usr/src$ ls
linux-4.19.148      linux-headers-5.11.0-27-generic
linux-4.19.148.tar  linux-headers-5.11.0-40-generic
linux-5.13.19       linux-hwe-5.11-headers-5.11.0-27
linux-5.13.19.tar   linux-hwe-5.11-headers-5.11.0-40
usertest109550073@usertest-vm:/usr/src$
```

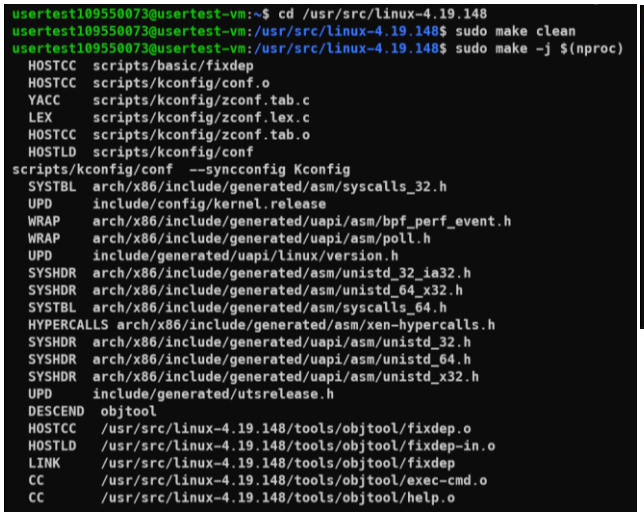
SCREENSHOT #2 .config files

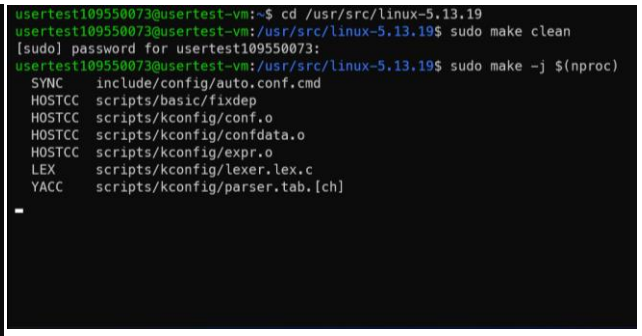
The screenshot implies two .config files inside two kernels' folder.



SCREENSHOT #3 Kernel running

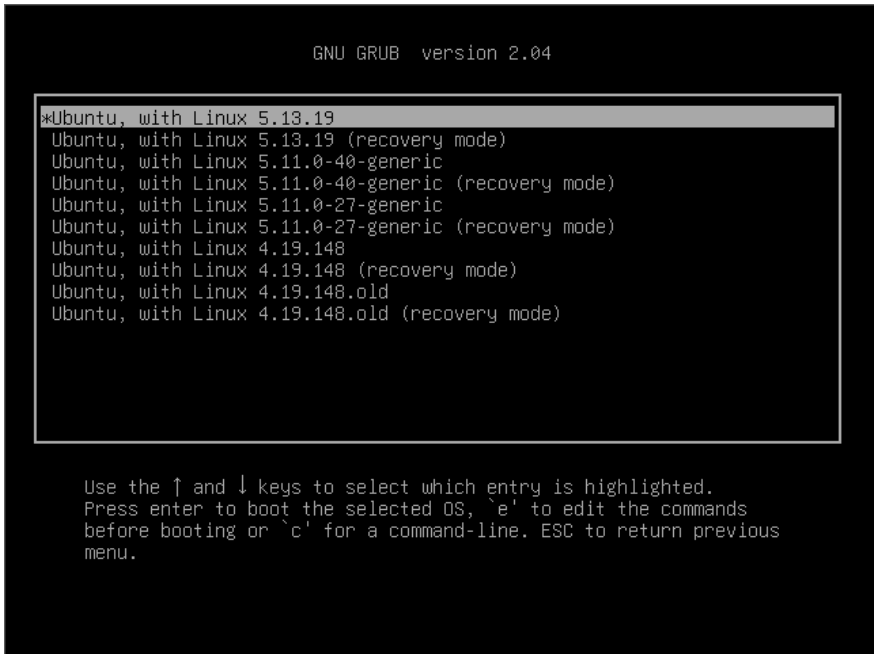
The screenshot represents the compiling process of two kernels.





SCREENSHOT #4 Grub

The screenshot shows the grub menu with both kernel linux-5.13.19 and linux-4.19.148.



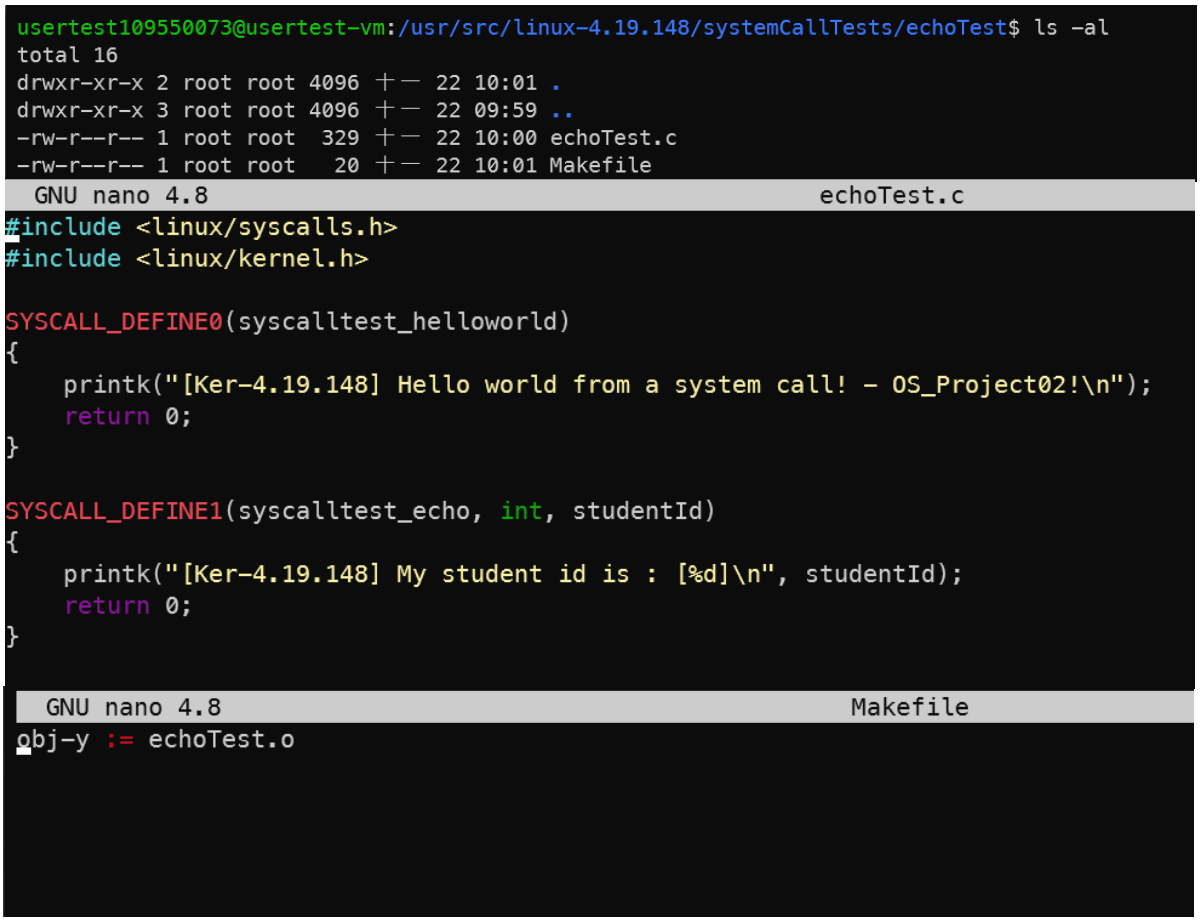
SCREENSHOT #5 Kernel Version

The screenshot represents the version of two kernels.

Section 2: System Calls.

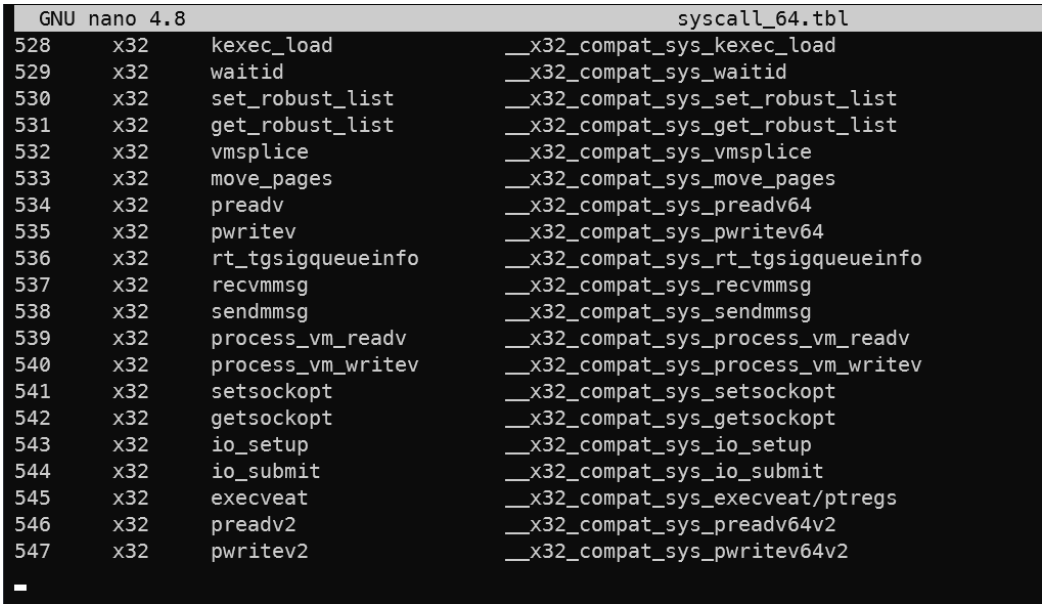
SCREENSHOT #6 echoTest-4.19.148

The screenshot indicates the contents inside linux-4.19.148 / systemCallTests / echoTest, which are echoTest.c and Makefile.



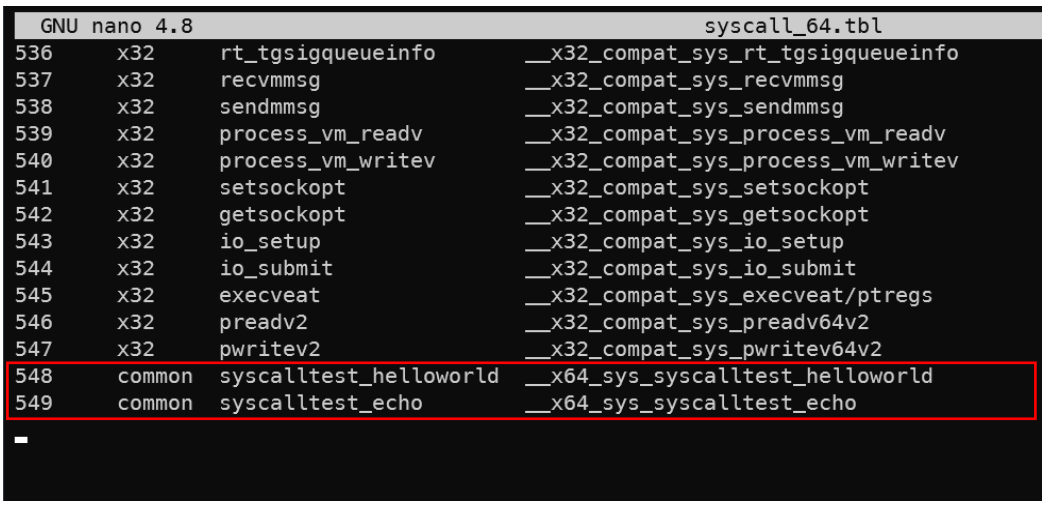
SCREENSHOT #7 syscall_64.tbl_original-4.19.148

The screenshot implies the original version of syscall_64.tbl.



SCREENSHOT #8 syscall_64.tbl_modified-4.19.148

The screenshot implies the modified version of syscall_64.tbl.



SCREENSHOT #9 syscalls.h-4.19.148

The screenshot shows the added line in syscalls.h.

```
 GNU nano 4.8                                                  syscalls.h
}

extern long do_sys_truncate(const char __user *pathname, loff_t length);

static inline long ksys_truncate(const char __user *pathname, loff_t length)
{
        return do_sys_truncate(pathname, length);
}

static inline unsigned int ksys_personality(unsigned int personality)
{
        unsigned int old = current->personality;

        if (personality != 0xffffffff)
                set_personality(personality);

        return old;
}
asmlinkage long syscalltest_helloworld(void);
asmlinkage long syscalltest_echo(int);

#endif
```

SCREENSHOT #10 echoTest-5.13.19

The screenshot indicates the contents inside linux-5.13.19 / systemCallTests / echoTest, which are echoTest.c and Makefile.

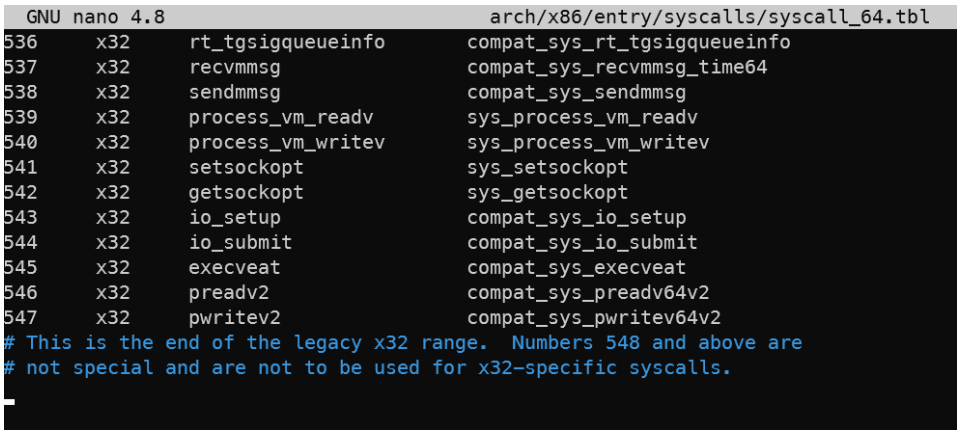
```
usertest109550073@usertest-vm:/usr/src/linux-5.13.19/systemCallTests/echoTest$ ls -al
total 16
drwxr-xr-x 2 root root 4096 十一 22 10:19 .
drwxr-xr-x 3 root root 4096 十一 22 10:17 ..
-rw-r--r-- 1 root root  325 十一 22 10:18 echoTest.c
-rw-r--r-- 1 root root   20 十一 22 10:19 Makefile
```

```
 GNU nano 4.8                                                  echoTest.c
#include <linux/syscalls.h>
#include <linux/kernel.h>

SYSCALL_DEFINE0(syscalltest_helloworld)
{
    printk("[Ker-5.13.19] Hello world from a system call! OS_Project02!\n");
    return 0;
}

SYSCALL_DEFINE1(syscalltest_echo, int, studentId)
{
    printk("[Ker-5.13.19] My student id is : [%d]\n", studentId);
    return 0;
}
```

```
 GNU nano 4.8                                                  Makefile
obj-y := echoTest.o
```

SCREENSHOT #11 syscall_64.tbl_original-5.13.19

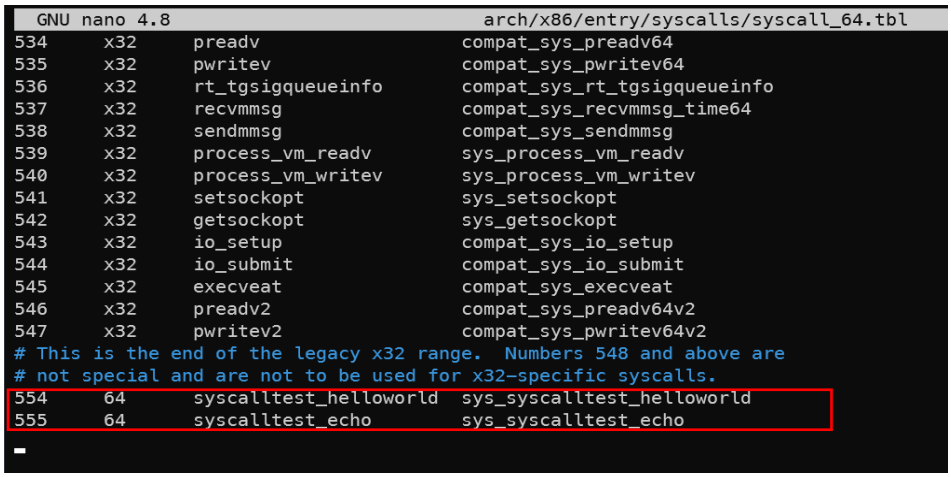
The screenshot implies the original version of syscall_64.tbl.



```
  GNU nano 4.8                                arch/x86/entry/syscalls/syscall_64.tbl
536     x32     rt_tgsigqueueinfo       compat_sys_rt_tgsigqueueinfo
537     x32     recvmmsg                compat_sys_recvmmsg_time64
538     x32     sendmmsg                compat_sys_sendmmsg
539     x32     process_vm_readv        sys_process_vm_readv
540     x32     process_vm_writev       sys_process_vm_writev
541     x32     setsockopt              sys_setsockopt
542     x32     getsockopt              sys_getsockopt
543     x32     io_setup                compat_sys_io_setup
544     x32     io_submit               compat_sys_io_submit
545     x32     execveat                compat_sys_execveat
546     x32     preadv2                 compat_sys_preadv64v2
547     x32     pwritev2                compat_sys_pwritev64v2
# This is the end of the legacy x32 range.  Numbers 548 and above are
# not special and are not to be used for x32-specific syscalls.
```

SCREENSHOT #12 syscall_64.tbl_modified-5.13.19

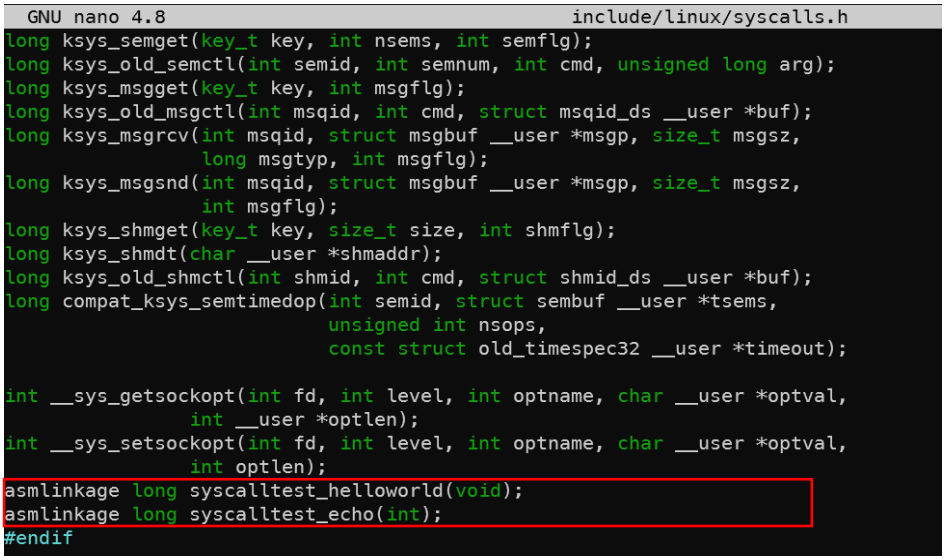
The screenshot implies the modified version of syscall_64.tbl.



```
  GNU nano 4.8                                arch/x86/entry/syscalls/syscall_64.tbl
534     x32     preadv                  compat_sys_preadv64
535     x32     pwritev                 compat_sys_pwritev64
536     x32     rt_tgsigqueueinfo       compat_sys_rt_tgsigqueueinfo
537     x32     recvmmsg                compat_sys_recvmmsg_time64
538     x32     sendmmsg                compat_sys_sendmmsg
539     x32     process_vm_readv        sys_process_vm_readv
540     x32     process_vm_writev       sys_process_vm_writev
541     x32     setsockopt              sys_setsockopt
542     x32     getsockopt              sys_getsockopt
543     x32     io_setup                compat_sys_io_setup
544     x32     io_submit               compat_sys_io_submit
545     x32     execveat                compat_sys_execveat
546     x32     preadv2                 compat_sys_preadv64v2
547     x32     pwritev2                compat_sys_pwritev64v2
# This is the end of the legacy x32 range.  Numbers 548 and above are
# not special and are not to be used for x32-specific syscalls.
554     64      syscalltest_helloworld  sys_syscalltest_helloworld
555     64      syscalltest_echo        sys_syscalltest_echo
```

SCREENSHOT #13 syscalls.h-5.13.19

The screenshot shows the added line in syscalls.h.



```
  GNU nano 4.8                                        include/linux/syscalls.h
long ksys_semget(key_t key, int nsems, int semflg);
long ksys_old_semctl(int semid, int semnum, int cmd, unsigned long arg);
long ksys_msgget(key_t key, int msgflg);
long ksys_old_msgctl(int msqid, int cmd, struct msqid_ds __user *buf);
long ksys_msgrcv(int msqid, struct msgbuf __user *msgp, size_t msgsz,
                 long msgtyp, int msgflg);
long ksys_msgsnd(int msqid, struct msgbuf __user *msgp, size_t msgsz,
                 int msgflg);
long ksys_shmget(key_t key, size_t size, int shmflg);
long ksys_shmdt(char __user *shmaddr);
long ksys_old_shmctl(int shmid, int cmd, struct shmid_ds __user *buf);
long compat_ksys_semtimedop(int semid, struct sembuf __user *tsems,
                            unsigned int nsops,
                            const struct old_timespec32 __user *timeout);

int __sys_getsockopt(int fd, int level, int optname, char __user *optval,
                int __user *optlen);
int __sys_setsockopt(int fd, int level, int optname, char __user *optval,
                int optlen);
asmlinkage long syscalltest_helloworld(void);
asmlinkage long syscalltest_echo(int);
#endif
```

SCREENSHOT #14 Rebuild

The screenshot represents that the both kernels are successfully rebuilt.

```
usertest109550073@usertest-vm:/usr/src/linux-4.19.148$ sudo make install
[sudo] password for usertest109550073:
sh ./arch/x86/boot/install.sh 4.19.148 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 4.19.148 /boot/vmlinuz-4.19.148
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.19.148 /boot/vmlinuz-4.19.148
update-initramfs: Generating /boot/initrd.img-4.19.148
I: The initramfs will attempt to resume from /dev/sda5
I: (UUID=0b74727d-1ce8-4bc7-9465-b44e169c9b79)
I: Set the RESUME variable to override this.
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 4.19.148 /boot/vmlinuz-4.19.148
run-parts: executing /etc/kernel/postinst.d/update-notifier 4.19.148 /boot/vmlinuz-4.19.148
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 4.19.148 /boot/vmlinuz-4.19.148
I: /boot/initrd.img.old is now a symlink to initrd.img-5.13.19
I: /boot/initrd.img is now a symlink to initrd.img-4.19.148
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 4.19.148 /boot/vmlinuz-4.19.148
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.13.19
Found initrd image: /boot/initrd.img-5.13.19
Found linux image: /boot/vmlinuz-5.13.19.old
Found initrd image: /boot/initrd.img-5.13.19
Found linux image: /boot/vmlinuz-5.11.0-40-generic
Found initrd image: /boot/initrd.img-5.11.0-40-generic
Found linux image: /boot/vmlinuz-5.11.0-27-generic
Found initrd image: /boot/initrd.img-5.11.0-27-generic
Found linux image: /boot/vmlinuz-4.19.148
Found initrd image: /boot/initrd.img-4.19.148
Found linux image: /boot/vmlinuz-4.19.148.old
Found initrd image: /boot/initrd.img-4.19.148
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
usertest109550073@usertest-vm:/usr/src/linux-4.19.148$
```

```
usertest109550073@usertest-vm:/usr/src/linux-5.13.19$ sudo make install
arch/x86/Makefile:148: CONFIG_X86_X32 enabled but no binutils support
sh ./arch/x86/boot/install.sh 5.13.19 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 5.13.19 /boot/vmlinuz-5.13.19
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.13.19 /boot/vmlinuz-5.13.19
update-initramfs: Generating /boot/initrd.img-5.13.19
I: The initramfs will attempt to resume from /dev/sda5
I: (UUID=0b74727d-1ce8-4bc7-9465-b44e169c9b79)
I: Set the RESUME variable to override this.
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.13.19 /boot/vmlinuz-5.13.19
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.13.19 /boot/vmlinuz-5.13.19
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.13.19 /boot/vmlinuz-5.13.19
I: /boot/initrd.img.old is now a symlink to initrd.img-4.19.148
I: /boot/initrd.img is now a symlink to initrd.img-5.13.19
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.13.19 /boot/vmlinuz-5.13.19
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.13.19
Found initrd image: /boot/initrd.img-5.13.19
Found linux image: /boot/vmlinuz-5.13.19.old
Found initrd image: /boot/initrd.img-5.13.19
Found linux image: /boot/vmlinuz-5.11.0-40-generic
Found initrd image: /boot/initrd.img-5.11.0-40-generic
Found linux image: /boot/vmlinuz-5.11.0-27-generic
Found initrd image: /boot/initrd.img-5.11.0-27-generic
Found linux image: /boot/vmlinuz-4.19.148
Found initrd image: /boot/initrd.img-4.19.148
Found linux image: /boot/vmlinuz-4.19.148.old
Found initrd image: /boot/initrd.img-4.19.148
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
usertest109550073@usertest-vm:/usr/src/linux-5.13.19$
```

SCREENSHOT #15 Outputs-4.19.148

The screenshot shows the outputs of terminal and dmesg when running the program using the kernel linux-4.19.148.

```
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$ uname -r
4.19.148
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$ ls -a
.  ..  a.out  syscallsHelloEco.c
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$ ./a.out
studentId = [109550073]

=== Kernel 4.19.148 ===
helloworld : 0
echo : 0

=== Kernel 5.13.19 ===
helloworld : -1
echo : -1
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$ dmesg
[  644.490285] [Ker-4.19.148] Hello world from a system call! - OS_Project02!
[  644.490288] [Ker-4.19.148] My student id is : [109550073]
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$
```

SCREENSHOT #16 Outputs-5.13.19

The screenshot shows the outputs of terminal and dmesg when running the program using the kernel linux-5.13.19.

```
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$ uname -r
5.13.19
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$ ls -a
.  ..  a.out  syscallsHelloEco.c
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$ ./a.out
studentId = [109550073]

=== Kernel 4.19.148 ===
helloworld : -1
echo : -1

=== Kernel 5.13.19 ===
helloworld : 0
echo : 0
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$ dmesg
[  209.517143] [Ker-5.13.19] Hello world from a system call! OS_Project02!
[  209.517161] [Ker-5.13.19] My student id is : [109550073]
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/echo$
```

SCREENSHOT #17 numericalTest.c-4.19.148

The screenshot indicates the completed numericalTest.c in kernel linux-4.19.148.

```
 GNU nano 4.8                                                    numericalTest.c
#include <linux/syscalls.h>
#include <linux/kernel.h>
//STUDENT ID: 109550073
int returnValue(int studentId, int a, int b){
        printk("[%d][Ker-4.19.148] syscalltest_returnIndividualValues : [%d][%d]\n", studentId, a, b);
        return 0;
}
int minimum(int studentId, int a, int b, int c){
        int min = a;
        if(b < min)     min = b;
        if(c < min)     min = c;
        printk("[%d][Ker-4.19.148] syscalltest_minimum : [%d][%d][%d] - [%d]\n", studentId, a, b, c, min);
        return min;
}
int maximum(int studentId, int a, int b, int c){
        int max = a;
        if(b > max)     max = b;
        if(c > max)     max = c;
        printk("[%d][Ker-4.19.148] syscalltest_maximum : [%d][%d][%d] - [%d]\n", studentId, a, b, c, max);
        return max;
}
```

```
int displayDatatypes(int studentId) {
       printk("[%d][Ker-4.19.148] size of unsigned int : [%d] bytes\n", studentId, sizeof(unsigned int));
       printk("[%d][Ker-4.19.148] size of signed int : [%d] bytes\n", studentId, sizeof(signed int));
       printk("[%d][Ker-4.19.148] size of unsigned long : [%d] bytes\n", studentId, sizeof(unsigned long));
       printk("[%d][Ker-4.19.148] size of signed long : [%d] bytes\n", studentId, sizeof(signed long));
       printk("[%d][Ker-4.19.148] size of unsigned long long : [%d] bytes\n", studentId, sizeof(unsigned long long));
       printk("[%d][Ker-4.19.148] size of signed long long : [%d] bytes\n", studentId, sizeof(signed long long));
       printk("[%d][Ker-4.19.148] size of double : [%d] bytes\n", studentId, sizeof(double));
       printk("[%d][Ker-4.19.148] size of char : [%d] bytes\n", studentId, sizeof(char));
       return 0;
}
SYSCALL_DEFINE3(syscalltest_returnIndividualValues, int, studentId, int, a, int, b){
        return returnValue(studentId, a, b);
}

SYSCALL_DEFINE4(syscalltest_minimum, int, studentId, int, a, int, b, int, c){
        return minimum(studentId, a, b, c);
}

SYSCALL_DEFINE4(syscalltest_maximum, int, studentId, int, a, int, b, int, c){
        return maximum(studentId, a, b, c);
}

SYSCALL_DEFINE1(syscalltest_dataTypes, int, studentId){
        return displayDatatypes(studentId);
}
```

SCREENSHOT #18 numericalTest.c-5.13.19

The screenshot indicates the completed numericalTest.c in kernel linux-5.13.19.

```
  GNU nano 4.8                          ../../../linux-5.13.19/systemCallTests/numericalTest/numericalTest.c
#include <linux/syscalls.h>
#include <linux/kernel.h>
//studentId:109550073
int returnValue(int studentId, int a, int b){
        printk("[%d][Ker-5.13.19] syscalltest_returnIndividualValues : [%d][%d]\n", studentId, a, b);
        return 0;
}

int addition(int studentId, int a, int b, int c){
        printk("[%d][Ker-5.13.19] syscalltest_addition : [%d][%d][%d]\n", studentId, a, b, a+b);
        return a+b;
}

int multiplication(int studentId, int a, int b, int c){
        printk("[%d][Ker-5.13.19] syscalltest_multiplication : [%d][%d][%d]\n", studentId, a, b, a*b);
        return a*b;
}

int displayDatatypes(int studentId) {
        printk("[%d][Ker-5.13.19] size of unsigned int : [%d] bytes\n", studentId, sizeof(unsigned int));
        printk("[%d][Ker-5.13.19] size of signed int : [%d] bytes\n", studentId, sizeof(signed int));
        printk("[%d][Ker-5.13.19] size of unsigned long : [%d] bytes\n", studentId, sizeof(unsigned long));
        printk("[%d][Ker-5.13.19] size of signed long : [%d] bytes\n", studentId, sizeof(signed long));
        printk("[%d][Ker-5.13.19] size of unsigned long long : [%d] bytes\n", studentId, sizeof(unsigned long long));
        printk("[%d][Ker-5.13.19] size of signed long long : [%d] bytes\n", studentId, sizeof(signed long long));
        printk("[%d][Ker-5.13.19] size of double : [%d] bytes\n", studentId, sizeof(double));
        printk("[%d][Ker-5.13.19] size of char : [%d] bytes\n", studentId, sizeof(char));
        return 0;
}
SYSCALL_DEFINE3(syscalltest_returnIndividualValues, int, studentId, int, a, int, b){
        return returnValue(studentId, a, b);
}

SYSCALL_DEFINE4(syscalltest_addition, int, studentId, int, a, int, b, int, c){
        return addition(studentId, a, b, c);
}

SYSCALL_DEFINE4(syscalltest_multiplication, int, studentId, int, a, int, b, int, c){
        return multiplication(studentId, a, b, c);
}

SYSCALL_DEFINE1(syscalltest_dataTypes, int, studentId){
        return displayDatatypes(studentId);
}
```

SCREENSHOT #19 Makefile-4.19.148

The screenshot implies the modification of systemCallTests/numericalTest/ Makefile and Makefile for kernel linux-4.19.148.

```
obj-y := numericalTest.o
```

```
  GNU nano 4.8                                                    Makefile
HOST_LIBELF_LIBS = $(shell pkg-config libelf --libs 2>/dev/null || echo -lelf)

ifdef CONFIG_STACK_VALIDATION
  has_libelf := $(call try-run,\
                echo "int main() {}" | $(HOSTCC) -xc -o /dev/null $(HOST_LIBELF_LIBS) -,1,0)
  ifeq ($(has_libelf),1)
    objtool_target := tools/objtool FORCE
  else
    SKIP_STACK_VALIDATION := 1
    export SKIP_STACK_VALIDATION
  endif
endif

PHONY += prepare0

ifeq ($(KBUILD_EXTMOD),)
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ systemCallTests/echoTest/ systemCallTests/numericalTest/

vmlinux-dirs    := $(patsubst %/,%,$(filter %/, $(init-y) $(init-m) \
                     $(core-y) $(core-m) $(drivers-y) $(drivers-m) \
                     $(net-y) $(net-m) $(libs-y) $(libs-m) $(virt-y)))
```

SCREENSHOT #20 syscall_64.tbl-4.19.148

The screenshot shows the modification of syscall_64.tbl for kernel linux-4.19.148

```
544     x32     io_submit               __x32_compat_sys_io_submit
545     x32     execveat                __x32_compat_sys_execveat/ptregs
546     x32     preadv2                 __x32_compat_sys_preadv64v2
547     x32     pwritev2                __x32_compat_sys_pwritev64v2
548     common  syscalltest_helloworld  __x64_sys_syscalltest_helloworld
549     common  syscalltest_echo        __x64_sys_syscalltest_echo
550     common  syscalltest_returnIndividualValues      __x64_sys_syscalltest_returnIndividualValues
551     common  syscalltest_minimum     __x64_sys_syscalltest_minimum
552     common  syscalltest_maximum     __x64_sys_syscalltest_maximum
553     common  syscalltest_dataTypes   __x64_sys_syscalltest_dataTypes
```

SCREENSHOT #21 syscalls.h-4.19.148

The screenshot shows the modification of syscalls.h for kernel linux-4.19.148.

```
  GNU nano 4.8                                                include/linux/syscalls.h
extern long do_sys_truncate(const char __user *pathname, loff_t length);

static inline long ksys_truncate(const char __user *pathname, loff_t length)
{
        return do_sys_truncate(pathname, length);
}

static inline unsigned int ksys_personality(unsigned int personality)
{
        unsigned int old = current->personality;

        if (personality != 0xffffffff)
                set_personality(personality);

        return old;
}
asmlinkage long syscalltest_helloworld(void);
asmlinkage long syscalltest_echo(int);
asmlinkage long syscalltest_returnIndividualValues(int);
asmlinkage long syscalltest_minimum(int);
asmlinkage long syscalltest_maximum(int);
asmlinkage long syscalltest_dataTypes(int);
#endif
```

SCREENSHOT #22 Makefile-5.13.19

The screenshot implies the modification of systemCallTests/numericalTest/ Makefile and Makefile for kernel linux-5.13.19

```
obj-y := numericalTest.o
```

```
  GNU nano 4.8                                                    Makefile
ifdef CONFIG_DEBUG_INFO_BTF
  ifeq ($(has_libelf),1)
    resolve_btfids_target := tools/bpf/resolve_btfids FORCE
  else
    ERROR_RESOLVE_BTFIDS := 1
  endif
endif # CONFIG_DEBUG_INFO_BTF
endif # CONFIG_BPF

PHONY += prepare0

export extmod_prefix = $(if $(KBUILD_EXTMOD),$(KBUILD_EXTMOD)/)
export MODORDER := $(extmod_prefix)modules.order
export MODULES_NSDEPS := $(extmod_prefix)modules.nsdeps

ifeq ($(KBUILD_EXTMOD),)
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ systemCallTests/echoTest/ systemCallTests/numericalTest/

vmlinux-dirs    := $(patsubst %/,%,$(filter %/, \
                     $(core-y) $(core-m) $(drivers-y) $(drivers-m) \
                     $(libs-y) $(libs-m)))
```

SCREENSHOT #23 syscall_64.tbl-5.13.19

The screenshot shows the modification of syscall_64.tbl for kernel linux-5.13.19

```
544     x32     io_submit               compat_sys_io_submit
545     x32     execveat                compat_sys_execveat
546     x32     preadv2                 compat_sys_preadv64v2
547     x32     pwritev2                compat_sys_pwritev64v2
# This is the end of the legacy x32 range.  Numbers 548 and above are
# not special and are not to be used for x32-specific syscalls.
554     64      syscalltest_helloworld  sys_syscalltest_helloworld
555     64      syscalltest_echo        sys_syscalltest_echo
556     64      syscalltest_returnIndividualValues      sys_syscalltest_returnIndividualValues
557     64      syscalltest_addition    sys_syscalltest_addition
558     64      syscalltest_multiplication      sys_syscalltest_multiplication
559     64      syscalltest_dataTypes   sys_syscalltest_dataTypes
```

SCREENSHOT #24 syscalls.h-5.13.19

The screenshot shows the modification of syscalls.h for kernel linux-5.13.19.

```
  GNU nano 4.8                                                    include/linux/syscalls.h
long ksys_msgrcv(int msqid, struct msgbuf __user *msgp, size_t msgsz,
                 long msgtyp, int msgflg);
long ksys_msgsnd(int msqid, struct msgbuf __user *msgp, size_t msgsz,
                 int msgflg);
long ksys_shmget(key_t key, size_t size, int shmflg);
long ksys_shmdt(char __user *shmaddr);
long ksys_old_shmctl(int shmid, int cmd, struct shmid_ds __user *buf);
long compat_ksys_semtimedop(int semid, struct sembuf __user *tsems,
                            unsigned int nsops,
                            const struct old_timespec32 __user *timeout);

int __sys_getsockopt(int fd, int level, int optname, char __user *optval,
                int __user *optlen);
int __sys_setsockopt(int fd, int level, int optname, char __user *optval,
                int optlen);

asmlinkage long syscalltest_helloworld(void);
asmlinkage long syscalltest_echo(int);
asmlinkage long syscalltest_returnIndividualValues(int);
asmlinkage long syscalltest_addition(int);
asmlinkage long syscalltest_multiplication(int);
asmlinkage long syscalltest_dataTypes(int);
#endif
```

SCREENSHOT #25 Output_terminal-4.19.148

The screenshot indicates the output of the program in terminal under kernel 4.19.148.

```
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/numericalRuns$ uname -r
4.19.148
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/numericalRuns$ ls -a
.  ..  a.out  syscallsNumerical.c
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/numericalRuns$ ./a.out
a = [15]
b = [43]
c = [30]
studentId = [109550073]

=== Kernel 4.19.148 ===
helloworld : 0
echo : 0
returnIndividualValues : 0
minimum : 15
maximum : 43
dataTypes : 0

=== Kernel 5.13.19 ===
helloworld : -1
echo : -1
returnIndividualValues : -1
addition : -1
multiplication : -1
dataTypes : -1
```

SCREENSHOT #26 Output_log-4.19.148

The screenshot indicates the output of the program in kernel ring buffer under kernel 4.19.148

```
=== Kernel 5.13.19 ===
helloworld : -1
echo : -1
returnIndividualValues : -1
addition : -1
multiplication : -1
dataTypes : -1
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/numericalRuns$ dmesg
[11191.019061] [Ker-4.19.148] Hello world from a system call! - OS_Project02!
[11191.019067] [Ker-4.19.148] My student id is : [109550073]
[11191.019072] [109550073][Ker-4.19.148] syscalltest_returnIndividualValues : [15][43]
[11191.019077] [109550073][Ker-4.19.148] syscalltest_minimum : [15][43][30] - [15]
[11191.019082] [109550073][Ker-4.19.148] syscalltest_maximum : [15][43][30] - [43]
[11191.019086] [109550073][Ker-4.19.148] size of unsigned int : [4] bytes
[11191.019087] [109550073][Ker-4.19.148] size of signed int : [4] bytes
[11191.019088] [109550073][Ker-4.19.148] size of unsigned long : [8] bytes
[11191.019089] [109550073][Ker-4.19.148] size of signed long : [8] bytes
[11191.019090] [109550073][Ker-4.19.148] size of unsigned long long : [8] bytes
[11191.019091] [109550073][Ker-4.19.148] size of signed long long : [8] bytes
[11191.019092] [109550073][Ker-4.19.148] size of double : [8] bytes
[11191.019092] [109550073][Ker-4.19.148] size of char : [1] bytes
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/numericalRuns$
```

SCREENSHOT #27 Output_terminal-5.13.19

The screenshot indicates the output of the program in terminal under kernel 5.13.19

```
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/numericalRuns$ uname -r
5.13.19
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/numericalRuns$ ls -a
.  ..  a.out  syscallsNumerical.c
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/numericalRuns$ ./a.out
a = [15]
b = [43]
c = [30]
studentId = [109550073]

=== Kernel 4.19.148 ===
helloworld : -1
echo : -1
returnIndividualValues : -1
minimum : -1
maximum : -1
dataTypes : -1

=== Kernel 5.13.19 ===
helloworld : 0
echo : 0
returnIndividualValues : 0
addition : 58
multiplication : 645
dataTypes : 0
```

SCREENSHOT #28 Output_log-5.13.19

The screenshot indicates the output of the program in kernel ring buffer under kernel 5.13.19

```
=== Kernel 5.13.19 ===
helloworld : 0
echo : 0
returnIndividualValues : 0
addition : 58
multiplication : 645
dataTypes : 0
usertest109550073@usertest-vm:~/Desktop/SystemCallsRunTests/numericalRuns$ dmesg
[ 4325.800666] [Ker-5.13.19] Hello world from a system call! OS_Project02!
[ 4325.800671] [Ker-5.13.19] My student id is : [109550073]
[ 4325.800674] [109550073][Ker-5.13.19] syscalltest_returnIndividualValues : [15][43]
[ 4325.800677] [109550073][Ker-5.13.19] syscalltest_addition : [15][43][58]
[ 4325.800680] [109550073][Ker-5.13.19] syscalltest_multiplication : [15][43][645]
[ 4325.800682] [109550073][Ker-5.13.19] size of unsigned int : [4] bytes
[ 4325.800683] [109550073][Ker-5.13.19] size of signed int : [4] bytes
[ 4325.800684] [109550073][Ker-5.13.19] size of unsigned long : [8] bytes
[ 4325.800685] [109550073][Ker-5.13.19] size of signed long : [8] bytes
[ 4325.800686] [109550073][Ker-5.13.19] size of unsigned long long : [8] bytes
[ 4325.800686] [109550073][Ker-5.13.19] size of signed long long : [8] bytes
[ 4325.800687] [109550073][Ker-5.13.19] size of double : [8] bytes
[ 4325.800688] [109550073][Ker-5.13.19] size of char : [1] bytes
```

2. Questions

2.1. What is Kernel space? What is user space? What are the differences between them?

Kernel space is reserved in virtual memory for executing system kernel, kernel extensions, and most devices' drivers.

User space is where the application software and some drivers run.

The separation aims to protect memory and hardware with different permission, while kernel space has higher authority and user space has lower authority.

2.2. What are protection rings? How many are them? What is Ring 0? What is Ring 1?

Protection rings is created to provide computer security with different level of privileges. There are 4 levels ranging from 0 to 3.

Ring 0 is usually called kernel space with highest privilege and interact with physical hardware.

Ring1 is typically where other operating system components that are not in Ring0 execute.

2.3. What is a system call? How many types are they in total? What are the differences between all the types?

System call is method that the application program requires services from the kernel of the operating system. Basically, it provides an interface between a process and operating system.

There are 5 different kinds of system calls: process control, file management, device management, information maintenance and communication. Each type of system call is in charge of different prospective.

Process control deal with processes like process creation and process termination.

File management is responsible for file manipulation, such as creating and reading files.

Device management is in charge of device management such of reading from buffers and writing into buffer.

Information maintenance handles information update and maintenance

between user space and kernel space.

Communication is responsible for interprocess communication.

2.4. For the custom kernel built in project 01, where is the list of system calls? (Give the file name and path)
Under folder /usr/src/linux-4.4.101/arch/x86/entry/syscalls, we have list of system calls in syscall_32.tbl for 32 bits system and syscall_64.tbl for 64 bits system.

2.5. What is the system call ID?
System call ID is a unique integer that is assigned to system call. Since it is unique, application program can call it by ID when it needs the specific system call.

2.6. What do the reserved words “asmlinkage” and “printk” mean?
“asmlinkage” is a gcc tag that can be assumed to be “#define”. It tells the compiler that the function should look on stack instead of registers.

“printk” is a C function that is used in Linux kernel for printing messages to the kernel log.

2.7. How do you use printk? How do you read the messages printed by printk?
I use printk just like how I use printf since it is an output method too.

For getting the messages from printk, we can use command “dmesg” for getting kernel log.

2.8. What is the kernel ring buffer? How do you read its contents?
Kernel ring buffer is a space that records related messages about the operation of the kernel.

We can read its contents by using “dmesg” command or go to the /var/log/dmesg.

2.9. What is a function signature?
Function signature is a part of function declaration but not totally the same one. It is the information about a function that participates in overload

resolution.

2.10. What does SYSCALL_DEFINE[n] mean? What is n?

The SYSCALL_DEFINE[n] is like the definition and entry point of system call, which perform the steps required for the system call. It contains the system call name followed by the type and name of parameters as arguments.

The n in SYSCALL_DEFINE[n] is the number of arguments to the system call.

2.11. For a system call wrapper (SYSCALL_DEFINE), how does its function signature look like when it has 0 inputs as parameters? 1 integer number as input? 2 integer numbers as inputs? 3 integer numbers as inputs?

The function signature of system call wrapper consists of system call's name and the parameters. So, the difference among n inputs of parameters in function signature is the number, order and type of parameters.

2.12. Why the function signature of a SYSCALL_DEFINE wrapper doesn't change depending on the type of element returned?

Because in C, the function signature does not include the return type of value. Therefore, it does not change depending on the type of element returned.

2.13. What is #include <linux/kernel.h>? What is #include <linux/syscalls.h>?

Linux/kernel.h and linux/syscalls.h are both the linux built-in headers.

3. References

(1) Kernel Space v.s. User Space. 在作業系統中, virtual memory 常被分為 kernel… | by Carl | Medium

(2) Protection Ring - GeeksforGeeks

(3) Protection Rings | SpringerLink

(4) 03. System Call (系統呼叫) - HackMD

(5) What are the concepts of "kernel ring buffer", "user level", "log level"? - Unix & Linux Stack Exchange

(6) Adding a New System Call — The Linux Kernel documentation

(7) intro(2) - Linux manual page (man7.org)