Operating Systems Project Report

Project Number (01 / 02 / 03):	02
Name:	蕭望緯
Student ID:	0811521
YouTube link (Format youtube.com/watch?v=[key]):	https://youtu.be/zGuZGQTrDb8
Date (YYYY-MM-DD):	2021-11-17
Names of the files uploaded to E3:	OS_Project02_0811521.pdf
Physical Machine Total RAM (Example: 8.0 GB):	16GB
Physical Machine CPU (Example: Intel i7-2600K):	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz

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

Individual questions

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

Ans:

Kernel space: kernel runs and provides services (e.g., system calls) in the kernel space.

User space: User processes (e.g., applications) run in the user space.

Differences: kernel space is responsible for the stability of the system and for process, memory management, file systems, device control and networking. Important jobs run in kernel space instead of user space.

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

Ans:

protection rings: The mechanism of rings protects data and system from fault or attack. Each ring represents a level of privilege which limits the access of resources in operating systems.

There are four rings. Ring0 is the most privileged level, which interacts most directly with the physical hardware such as the CPU and memory. Ring1 has less privilege than Ring0 but more privilege than Ring2.

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

Ans:

system call: A way that user programs request services from the kernel

Five types in total.

Process Control: process creation, termination, etc.

File Management: for example, creating, reading, or writing files.

Device Management: for example, request, release, write to devices.

Information Maintenance: handling information between user programs and kernel.

Communications: inter-process communication.

4. For the custom kernel built in project 01, where is the list of system calls? (Give the file name and path)

Ans**: /arch/x86/entry/syscalls/syscall_64.tbl**

5. What is the system call ID?

Ans: each system call has its unique ID, an integer, and the ID is used as an argument when calling a specific system call.

6. What do the reserved words "*asmlinkage*" and "*printk*" mean?

Ans:

asmlinkage: a #define for some gcc magic that tells the compiler that the function should not expect to find any of its arguments in registers, but only on the CPU's stack.

printk: prints to the kernel's log file.

7. How do you use **printk**? How do you read the messages printed by **printk**?

Ans:

printk(KERN_INFO "Message: %s ", arg);

KERN_INFO is the log level that specifies the importance of a message.

log messages are read through **dmesg.**

8. What is the **kernel ring buffer**? How do you read its contents?

Ans:

kernel ring buffer: records messages related to the operation of the kernel.

log messages are read through **dmesg.**

9. What is a function signature?

Ans:

function signature: defines input (parameters and their types) and output of functions; i.e., defines the prototype.

10. What does **SYSCALL_DEFINE[n]** mean? What is **n**?

Ans:

a macro, a wrapper of system call function.

"n" means the number of arguments of the system call.

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?

Ans:

SYSCALL_DEFINE0(*syscall_name*);

SYSCALL_DEFINE1(*syscall_name*, int);

SYSCALL_DEFINE2(*syscall_name*, int, int);

SYSCALL_DEFINE3(*syscall_name*, int, int, int);

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

Ans:

SYSCALL_DEFINE is a macro; it is equivalent to **asmlinkage long *function_name*(*parameters*)**

The return type is defined to be always **long**.

13. What is **#include <linux/kernel.h>**? What is **#include <linux/syscalls.h>**?
Ans:
<linux/kernel.h>: header file that contains macros for kernel functions.
<linux/syscalls.h>: header file that contains system calls' function prototypes.

Screenshot #1
content of **/usr/src/**

```
usertest0811521@usertest0811521:/usr/src$ ls
linux-4.19.148      linux-5.13.19      linux-headers-5.11.0-27-generic  linux-hwe-5.11-headers-5.11.0-27
linux-4.19.148.tar  linux-5.13.19.tar  linux-headers-5.11.0-38-generic  linux-hwe-5.11-headers-5.11.0-38
```

Screenshot #2
.config for v4 and v5

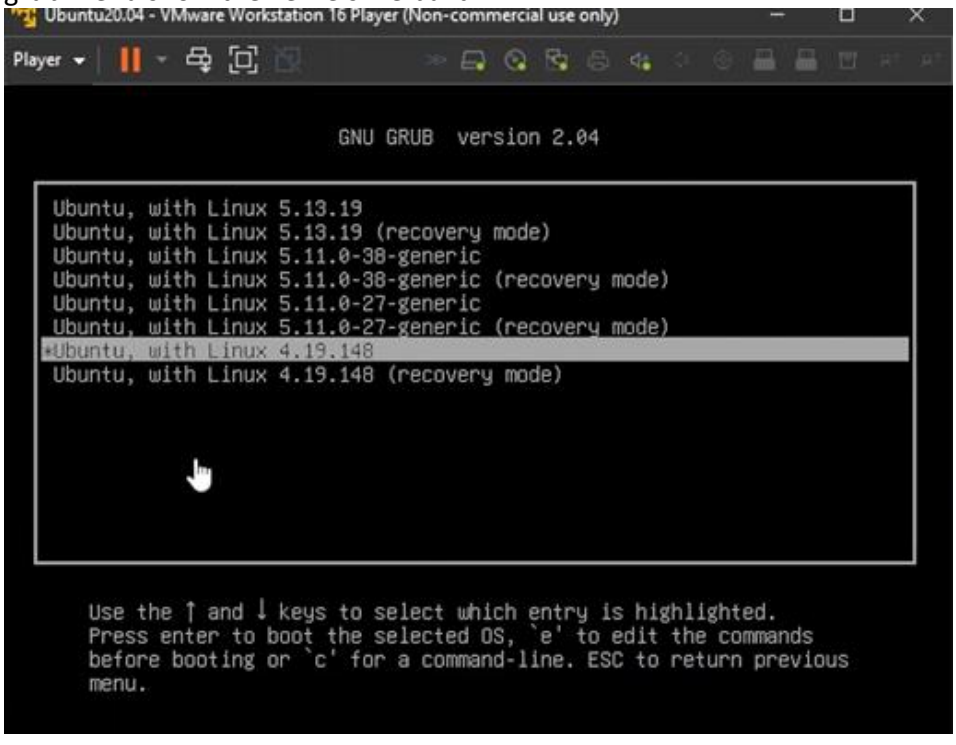
```
usertest0811521@usertest0811521:/usr/src$ ls linux-4.19.148/.config
linux-4.19.148/.config
usertest0811521@usertest0811521:/usr/src$ ls linux-5.13.19/.config
linux-5.13.19/.config
```

Screenshot #3
compile the kernel for v4 and v5

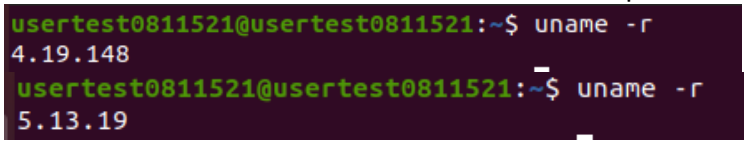
```
usertest0811521@usertest0811521: /usr/src/linux-5.13.19
usertest0811521@usertest0811521:/usr/src$ cd linux-5
bash: cd: linux-5: No such file or directory
usertest0811521@usertest0811521:/usr/src$ cd linux-5.13.19/
usertest0811521@usertest0811521:/usr/src/linux-5.13.19$ sudo make clean
[sudo] password for usertest0811521:
usertest0811521@usertest0811521:/usr/src/linux-5.13.19$ sudo make -j $(nproc)
  SYNC    include/config/auto.conf.cmd
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/menu.o
```

```
usertest0811521@usertest0811521:/usr/src/linux-4.19.148$ sudo nano .config
usertest0811521@usertest0811521:/usr/src/linux-4.19.148$ sudo make clean
usertest0811521@usertest0811521:/usr/src/linux-4.19.148$ sudo make -j $(nproc)
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  YACC    scripts/kconfig/zconf.tab.c
  LEX     scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf  --syncconfig Kconfig
.config:1545:warning: symbol value 'm' invalid for NF_TABLES_BRIDGE
.config:1631:warning: symbol value 'm' invalid for NET_DSA_TAG_BRCM
.config:1632:warning: symbol value 'm' invalid for NET_DSA_TAG_BRCM_PREPEND
.config:1636:warning: symbol value 'm' invalid for NET_DSA_TAG_DSA
```

Screenshot #4
grub menu show the kernels we built



Screenshot #5
both kernels can be run and their versions can be printed in the terminal



Screenshot #6 (linux-4.19.148)

echoTest.c is our source code of self-defined system calls.

Create a **Makefile** that includes the definition of **echoTest.o** so that when we recompile the kernel later, **echoTest.o** will be built.

```
root@usertest0811521:/usr/src/linux-4.19.148/systemCallTests/echoTest# ls -al
total 16
drwxr-xr-x 2 root root 4096 ┼─  8 11:02 .
drwxr-xr-x 3 root root 4096 ┼─  8 10:55 ..
-rw-r--r-- 1 root root  339 ┼─  8 11:00 echoTest.c
-rw-r--r-- 1 root root   20 ┼─  8 11:02 Makefile
```

the content of **echoTest.c**

```
root@usertest0811521: /usr/src/linux-4.19.148/systemCallT...
  GNU nano 4.8                    echoTest.c
#include <linux/syscalls.h>
#include <linux/kernel.h>

SYSCALL_DEFINE0(syscalltest_helloworld)
{
    printk("[Ker-4.19.148] Hello world from a system call! - OS_Project02!\n");
    return 0;
}
SYSCALL_DEFINE1(syscalltest_echo, int, studentId)
{
    printk("[Ker-4.19.148] My student id is : [%d]\n", studentId);
    return 0;
}
```

the content of **Makefile**

```
root@usertest0811521: /usr/src/linux-4.19.148/systemCallT...
  GNU nano 4.8                    Makefile
obj-y := echoTest.o
```

Screenshot #7 (linux-4.19.148)
the original system call table

```
root@usertest0811521: /usr/src/linux-4.19.148
  GNU nano 4.8        arch/x86/entry/syscalls/syscall_64.tbl        Modified
528     x32     kexec_load              __x32_compat_sys_kexec_load
529     x32     waitid                  __x32_compat_sys_waitid
530     x32     set_robust_list         __x32_compat_sys_set_robust_list
531     x32     get_robust_list         __x32_compat_sys_get_robust_list
532     x32     vmsplice                __x32_compat_sys_vmsplice
533     x32     move_pages              __x32_compat_sys_move_pages
534     x32     preadv                  __x32_compat_sys_preadv64
535     x32     pwritev                 __x32_compat_sys_pwritev64
536     x32     rt_tgsigqueueinfo       __x32_compat_sys_rt_tgsigqueueinfo
537     x32     recvmmsg                __x32_compat_sys_recvmmsg
538     x32     sendmmsg                __x32_compat_sys_sendmmsg
539     x32     process_vm_readv        __x32_compat_sys_process_vm_readv
540     x32     process_vm_writev       __x32_compat_sys_process_vm_writev
541     x32     setsockopt              __x32_compat_sys_setsockopt
542     x32     getsockopt              __x32_compat_sys_getsockopt
543     x32     io_setup                __x32_compat_sys_io_setup
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
```

Screenshot #8 (linux-4.19.148)
add our system calls:

index 548 is for ***syscalltest_helloworld***

index 549 is for ***syscalltest_echo***

```
543     x32     io_setup                __x32_compat_sys_io_setup
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
```

Screenshot #9 (linux-4.19.148)

define functions' prototype in **syscalls.h** and add the flag *asmlinkage* so that the kernel will know the parameters of functions are on the stack.

```
root@usertest0811521: /usr/src/linux-4.19.148
  GNU nano 4.8                include/linux/syscalls.h                Modified
}

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

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell
```

Screenshot #10 (linux-5.13.19)

echoTest.c is our source code of system calls.
Create a **Makefile** that includes the definition of **echoTest.o** so that when we recompile the kernel later, **echoTest.o** will be built.

```
root@usertest0811521:/usr/src/linux-5.13.19/systemCallTests/echoTest# ls -al
total 16
drwxr-xr-x 2 root root 4096 ┼─  8 15:03 .
drwxr-xr-x 3 root root 4096 ┼─  8 14:57 ..
-rw-r--r-- 1 root root  325 ┼─  8 15:03 echoTest.c
-rw-r--r-- 1 root root   20 ┼─  8 15:02 Makefile
```

```
root@usertest0811521: /usr/src/linux-5.13.19/systemCallTe...
  GNU nano 4.8                    Makefile
obj-y := echoTest.o
```

```
root@usertest0811521: /usr/src/linux-5.13.19/systemCallTe...
  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;
}
```

Screenshot #11 (linux-5.13.19)

the original system call table for linux-5.13.19

```
root@usertest0811521: /usr/src/linux-5.13.19
  GNU nano 4.8        arch/x86/entry/syscalls/syscall_64.tbl        Modified
533     x32     move_pages              compat_sys_move_pages
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.
```

Screenshot #12 (linux-5.13.19)

add our system calls:

index 554 is for ***syscalltest_helloworld***

index 555 is for ***syscalltest_echo***

```
root@usertest0811521: /usr/src/linux-5.13.19
  GNU nano 4.8        arch/x86/entry/syscalls/syscall_64.tbl        Modified
533     x32     move_pages              compat_sys_move_pages
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

define functions' prototype in **syscalls.h** and add the flag *asmlinkage*

```
root@usertest0811521: /usr/src/linux-5.13.19
  GNU nano 4.8              include/linux/syscalls.h              Modified
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 both versions of kernel

```
usertest0811521@usertest0811521:/usr/src/linux-4.19.148$ sudo make install
[sudo] password for usertest0811521:
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/vml
inuz-4.19.148
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.19.148 /boot/vmli
nuz-4.19.148
update-initramfs: Generating /boot/initrd.img-4.19.148
I: The initramfs will attempt to resume from /dev/sda5
I: (UUID=a2ac42cc-9639-4133-99ee-1ec08c5f9e78)
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/vmli
nuz-4.19.148
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 4.19.148 /bo
ot/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/vmlin
uz-4.19.148
```

```
usertest0811521@usertest0811521:/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/vmli
nuz-5.13.19
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.13.19 /boot/vmlin
uz-5.13.19
update-initramfs: Generating /boot/initrd.img-5.13.19
I: The initramfs will attempt to resume from /dev/sda5
I: (UUID=a2ac42cc-9639-4133-99ee-1ec08c5f9e78)
I: Set the RESUME variable to override this.
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.13.19 /boot/v
mlinuz-5.13.19
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.13.19 /boot/vmlin
uz-5.13.19
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.13.19 /boo
t/vmlinuz-5.13.19
I: /boot/initrd.img.old is now a symlink to initrd.img-4.19.148
```

Screenshot #15

system calls for v4 return 0 and system calls for v5 return -1 (not executed) when we are in version 4

demsg print the log messages

```
usertest0811521@usertest0811521:~/Desktop/E3/Section 2.1$ uname -r
4.19.148
usertest0811521@usertest0811521:~/Desktop/E3/Section 2.1$ ./syscallsHelloEco
studentId = [811521]

=== Kernerl 4.19.148 ===
helloworld : 0
echo : 0

=== Kernerl 5.13.19 ===
helloworld : -1
echo : -1
usertest0811521@usertest0811521:~/Desktop/E3/Section 2.1$ dmesg
[  223.402271] [Ker-4.19.148] Hello world from a system call! - OS_Project02!
[  223.402278] [Ker-4.19.148] My student id is : [811521]
```

Screenshot #16

system calls for v5 return 0 and system calls for v4 return -1 (not executed) when we are in version 5

demsg print the log messages

```
root@usertest0811521:/home/usertest0811521/Desktop/E3/Section 2.1# uname -r
5.13.19
root@usertest0811521:/home/usertest0811521/Desktop/E3/Section 2.1# ./syscallsHelloEco
studentId = [811521]

=== Kernerl 4.19.148 ===
helloworld : -1
echo : -1

=== Kernerl 5.13.19 ===
helloworld : 0
echo : 0
root@usertest0811521:/home/usertest0811521/Desktop/E3/Section 2.1# dmesg
[ 1016.641291] [Ker-5.13.19] Hello world from a system call! OS_Project02!
[ 1016.641307] [Ker-5.13.19] My student id is : [811521]
```

Screenshot #17

numericalTest.c for v4

syscallsNumerical.c 4 | numericalTest.c ...\Kernel 5.13.19 3 | numericalTest.c ...\Kernel 4.19.148 3

Section 2.2 > Kernel 4.19.148 > numericalTest.c > SYSCALL_DEFINE1(syscalltest_dataTypes, int, studentId)

```c
#include <linux/syscalls.h>
#include <linux/kernel.h>

//STUDENT ID: 0811521 (With leading 0)

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 maximum(int studentId, int a, int b, int c){ ...

int displayDatatypes(int studentId) { ...


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);
}

```

Install the recommended C extension to
debugging, linting, and more.

Screenshot #18

numericalTest.c for v5

```
syscallsNumerical.c 4   numericalTest.c ...\Kernel 5.13.19 3   numericalTest.c ...\Kernel 4.19.148 3
Section 2.2 > Kernel 5.13.19 > numericalTest.c > SYSCALL_DEFINE1(syscalltest_dataTypes, int, studentId)
1   #include <linux/syscalls.h>
2   #include <linux/kernel.h>
3
4   //STUDENT ID: 0811521 (With leading 0)
5
6 > int returnValue(int studentId, int a, int b){ ···
10
11 > int addition(int studentId, int a, int b){ ···
15
16 > int multiplication(int studentId, int a, int b){ ···
20
21 > int displayDatatypes(int studentId) { ···
45
46  SYSCALL_DEFINE3(syscalltest_returnIndividualValues, int, studentId, int, a, int, b){
47      return returnValue(studentId, a, b);
48  }
49
50  SYSCALL_DEFINE3(syscalltest_addition, int, studentId, int, a, int, b){
51      return addition(studentId, a, b);
52  }
53
54  SYSCALL_DEFINE3(syscalltest_multiplication, int, studentId, int, a, int, b){
55      return multiplication(studentId, a, b);
56  }
57
58  SYSCALL_DEFINE1(syscalltest_dataTypes, int, studentId){
59      return displayDatatypes(studentId);
60  }
61
```

Screenshot #19 (linux-4.19.148)

add path to the **Makefile** so that later when we rebuild the kernel, the files in the **numericalTest/** would be included

```
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ systemCallTests/echoTest/ systemCallTests/numericalTest/
```

Screenshot #20 (linux-4.19.148)

add our system calls in **syscall_64.tbl**:

index 550 is for ***syscalltest_ returnIndividualValues***

index 551 is for ***syscalltest_minimum***

index 552 is for ***syscalltest_maximum***

index 553 is for ***syscalltest_dataTypes***

```
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 (linux-4.19.148)

define functions' prototype in **syscalls.h**

```
asmlinkage long syscalltest_returnIndividualValues(int, int, int);
asmlinkage long syscalltest_minimum(int, int, int, int);
asmlinkage long syscalltest_maximum(int, int, int, int);
asmlinkage long syscalltest_dataTypes(int);
```

Screenshot #22 (linux-5.13.19)

add path to the **Makefile**

```
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ systemCallTests/echoTest/ systemCallTests/numericalTest/
```

Screenshot #23 (linux-5.13.19)

add our system calls in **syscall_64.tbl**:

index 556 is for ***syscalltest_ returnIndividualValues***

index 557 is for ***syscalltest_addition***

index 558 is for ***syscalltest_multiplication***

index 559 is for ***syscalltest_dataTypes***

```
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 (linux-5.13.19)

define functions' prototype in **syscalls.h**

```
asmlinkage long syscalltest_returnIndividualValues(int, int, int);
asmlinkage long syscalltest_addition(int, int, int);
asmlinkage long syscalltest_multiplication(int, int, int);
asmlinkage long syscalltest_dataTypes(int);
```

Screenshot #25 (linux-4.19.148)

syscallsNumericals.c execution result

system calls of v4 return correct value

system calls of v5 do not respond and give outputs of -1

```
root@usertest0811521:/home/usertest0811521/Desktop/E3/Section 2.2# uname -r
4.19.148
root@usertest0811521:/home/usertest0811521/Desktop/E3/Section 2.2# ./syscallsNumerical
a = [15]
b = [43]
c = [30]
studentId = [811521]

=== 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
substraction : -1
dataTypes : -1
```

Screenshot #26 (linux-4.19.148)

log messages of **syscallsNumericals.c**

```
root@usertest0811521:/home/usertest0811521/Desktop/E3/Section 2.2# dmesg
[  239.141220] [Ker-4.19.148] Hello world from a system call! - OS_Project02!
[  239.141222] [Ker-4.19.148] My student id is : [811521]
[  239.141233] [811521][Ker-4.19.148] syscalltest_returnIndividualValues : [15][43]
[  239.141240] [811521][Ker-4.19.148] syscalltest_minimum : [15][43][30] - [15]
[  239.141241] [811521][Ker-4.19.148] syscalltest_maximum : [15][43][30] - [43]
[  239.141242] [811521][Ker-4.19.148] size of unsigned int : [4] bytes
[  239.141242] [811521][Ker-4.19.148] size of signed int : [4] bytes
[  239.141243] [811521][Ker-4.19.148] size of unsigned long : [8] bytes
[  239.141243] [811521][Ker-4.19.148] size of signed long : [8] bytes
[  239.141243] [811521][Ker-4.19.148] size of unsigned long long : [8] bytes
[  239.141243] [811521][Ker-4.19.148] size of signed long long : [8] bytes
[  239.141244] [811521][Ker-4.19.148] size of double : [8] bytes
[  239.141244] [811521][Ker-4.19.148] size of char : [1] bytes
```

Screenshot #27 (linux-5.13.19)

syscallsNumericals.c execution result

system calls of v4 do not respond and give outputs of -1

system calls of v5 return correct value

```
root@usertest0811521:/home/usertest0811521/Desktop/E3/Section 2.2# uname -r
5.13.19
root@usertest0811521:/home/usertest0811521/Desktop/E3/Section 2.2# ./syscallsNumerical
a = [15]
b = [43]
c = [30]
studentId = [811521]

=== 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
substraction : 645
dataTypes : 0
```

Screenshot #28 (linux-5.13.19)

log messages of **syscallsNumericals.c**

```
root@usertest0811521:/home/usertest0811521/Desktop/E3/Section 2.2# dmesg
[  129.082670] [Ker-5.13.19] Hello world from a system call! OS_Project02!
[  129.082683] [Ker-5.13.19] My student id is : [811521]
[  129.082694] [811521][Ker-5.13.19] syscalltest_returnIndividualValues : [15][43]
[  129.082698] [811521][Ker-5.13.19] syscalltest_addition : [15][43][58]
[  129.082699] [811521][Ker-5.13.19] syscalltest_multiplication : [15][43][645]
[  129.082700] [811521][Ker-5.13.19] size of unsigned int : [4] bytes
[  129.082713] [811521][Ker-5.13.19] size of signed int : [4] bytes
[  129.082714] [811521][Ker-5.13.19] size of unsigned long : [8] bytes
[  129.082714] [811521][Ker-5.13.19] size of signed long : [8] bytes
[  129.082715] [811521][Ker-5.13.19] size of unsigned long long : [8] bytes
[  129.082715] [811521][Ker-5.13.19] size of signed long long : [8] bytes
[  129.082715] [811521][Ker-5.13.19] size of double : [8] bytes
[  129.082716] [811521][Ker-5.13.19] size of char : [1] bytes
```