**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>**?

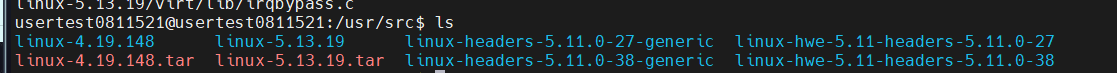
Ans:

**<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/**



Screenshot #2

**.config** for v4 and v5

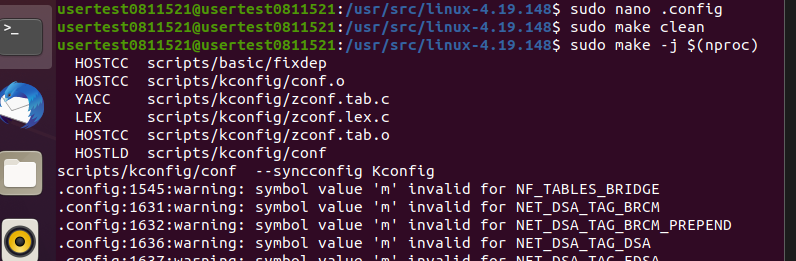
Text

Description automatically generated

Screenshot #3

compile the kernel for v4 and v5

Text

Description automatically generated

Screenshot #4

grub menu show the kernels we built

Text

Description automatically generated

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.

A picture containing calendar

Description automatically generated

the content of **echoTest.c**

Text

Description automatically generated

the content of **Makefile**

Text

Description automatically generated

Screenshot #7 (linux-4.19.148)

the original system call table

A picture containing text

Description automatically generated

Screenshot #8 (linux-4.19.148)

add our system calls:

index 548 is for ***syscalltest\_helloworld***

index 549 is for ***syscalltest\_echo***

Text

Description automatically generated

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.

Text

Description automatically generated

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.

Text

Description automatically generated with medium confidence Graphical user interface

Description automatically generated with low confidenceText

Description automatically generated

Screenshot #11 (linux-5.13.19)

the original system call table for linux-5.13.19

Text

Description automatically generated

Screenshot #12 (linux-5.13.19)

add our system calls:

index 554 is for ***syscalltest\_helloworld***

index 555 is for ***syscalltest\_echo***

Graphical user interface, text

Description automatically generated

Screenshot #13

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

Text

Description automatically generated

Screenshot #14

Rebuild both versions of kernel

Text

Description automatically generatedText

Description automatically generated

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

Text

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

Screenshot #17

numericalTest.c for v4

Text

Description automatically generated

Screenshot #18

numericalTest.c for v5

Text

Description automatically generated

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



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***

A computer screen capture

Description automatically generated with medium confidence

Screenshot #21 (linux-4.19.148)

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

Text

Description automatically generated

Screenshot #22 (linux-5.13.19)

add path to the **Makefile**



Screenshot #23 (linux-5.13.19)

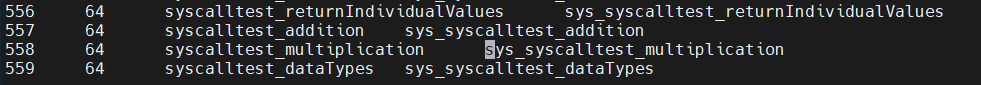
add our system calls in **syscall\_64.tbl**:

index 556 is for ***syscalltest\_*** ***returnIndividualValues***

index 557 is for ***syscalltest\_addition***

index 558 is for ***syscalltest\_multiplication***

index 559 is for ***syscalltest\_dataTypes***



Screenshot #24 (linux-5.13.19)

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

Text

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

Screenshot #26 (linux-4.19.148)

log messages of **syscallsNumericals.c**

A screenshot of a computer

Description automatically generated with medium confidence

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

Text

Description automatically generated

Screenshot #28 (linux-5.13.19)

log messages of **syscallsNumericals.c**

Text

Description automatically generated