- Video: http://newslab.csie.ntu.edu.tw/course/OS2018/PJ1.html
- 作業上傳後如果有更新版本，檔名請加v#.
  - Example: OSPJ1_Team99v2.zip
- 報告內請放上學號姓名.
- 繳交的檔案(.c, .h)與報告放在同一個目錄下即可.
- 解析度問題
  - http://tern.logdown.com/posts/194790-virtualbox-ubuntu-1404-how-adjusting-screen-resolution
- Report可以用中文或英文.
- 使用system call計算數值時不用考慮負值.
- 每次重編譯kernel, 請從menuconfig之後的指令開始.
- sys_min(), 用來傳回兩數值中的最小值.

Operating Systems 2018 Project 1

Linux Kernel Building
System Call Implementation

Advisor: Prof. Tei-Wei Kuo

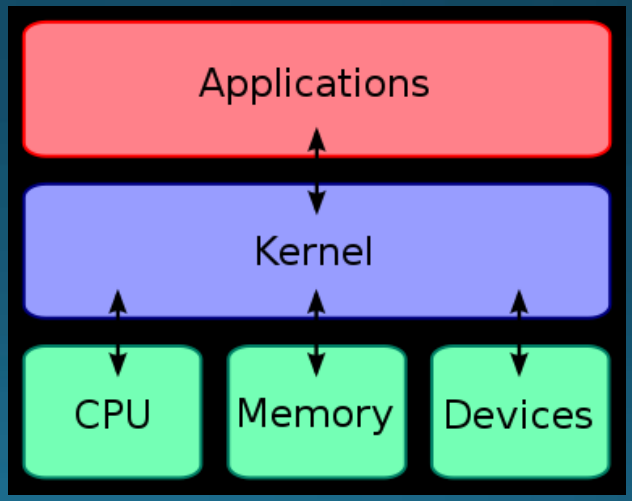
TAs: Han-Yi Lin, Yu-Chen Lin, Yi-Shen Chen, Yu-Chuan Chang

Outline

- Linux Kernel Building
- System Call Implementation
- Project Requirements
- Submission Rules

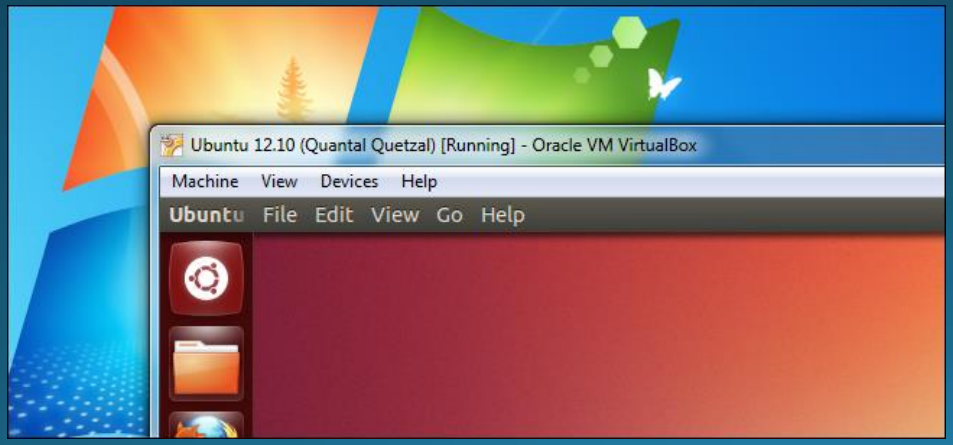
What is “Kernel”?

- The kernel[1] is a fundamental part of a modern computer‘s operating system.
- The kernel's primary functions are to
  - Manage the computer's hardware and resources
    - E.g., CPU, main memory, I/O devices, and so on.
  - Allow applications to run and use these resources



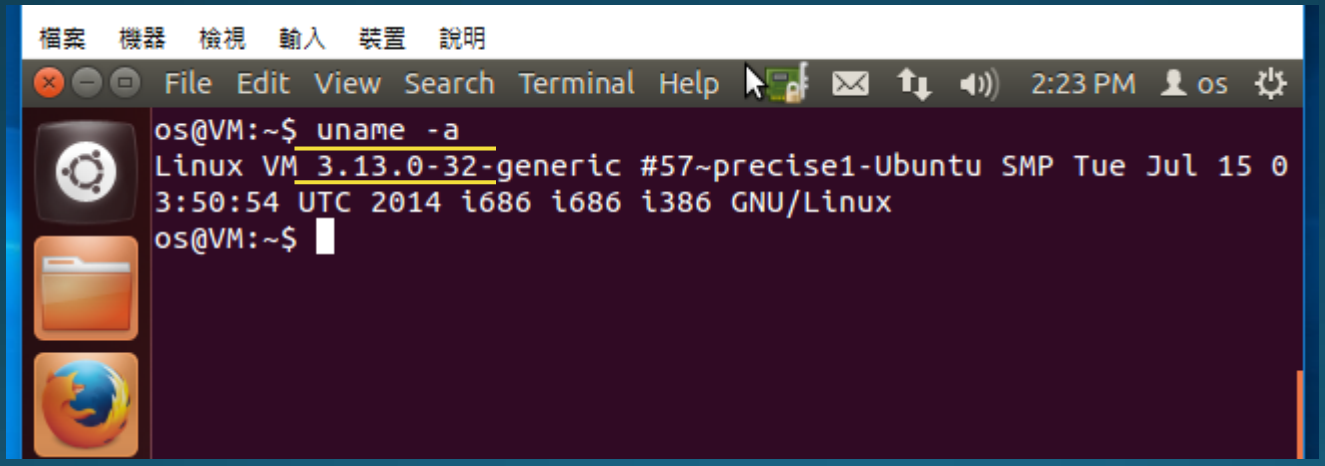
Environment Setup

- Oracle VM VirtualBox[2]
  - Download link: https://www.virtualbox.org/wiki/Downloads
- Ubuntu 12.04.5 32bits LTS[3]
  - Download link: http://tw.archive.ubuntu.com/ubuntu-cd/12.04.5/ubuntu-12.04.5-desktop-i386.iso
- Install the Ubuntu 12.04.5 on the VirtualBox



Build Linux Kernel (1/4)

- After the installation, please login Ubuntu and open a terminal to start building your Linux kernel[4]



Build Linux Kernel (2/4)

$ sudo apt-get install vim fakeroot build-essential kernel-package libncurses5 libncurses5-dev

$ cd /tmp

$ wget https://www.kernel.org/pub/linux/kernel/v2.6/longterm/v2.6.32/linux-2.6.32.60.tar.xz

$ sudo tar xvf linux-2.6.32.60.tar.xz -C /usr/src

$ cd /usr/src/linux-2.6.32.60

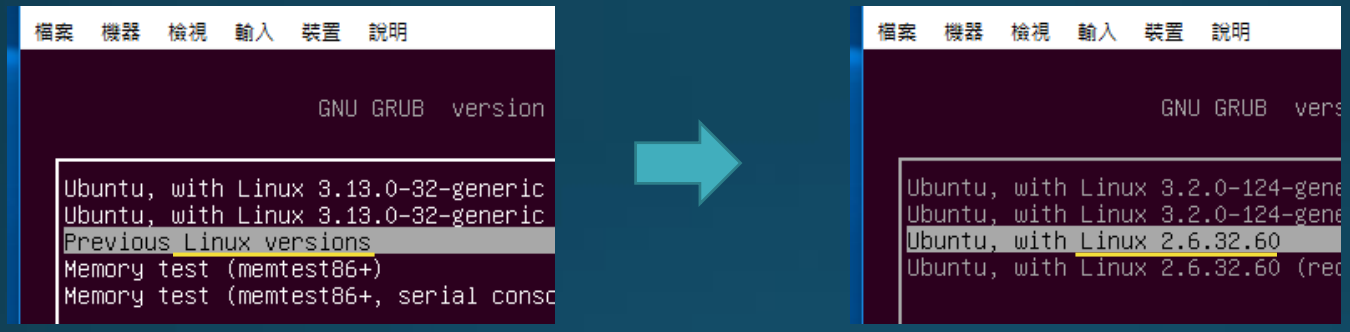
$ make mrproper

Build Linux Kernel (3/4)

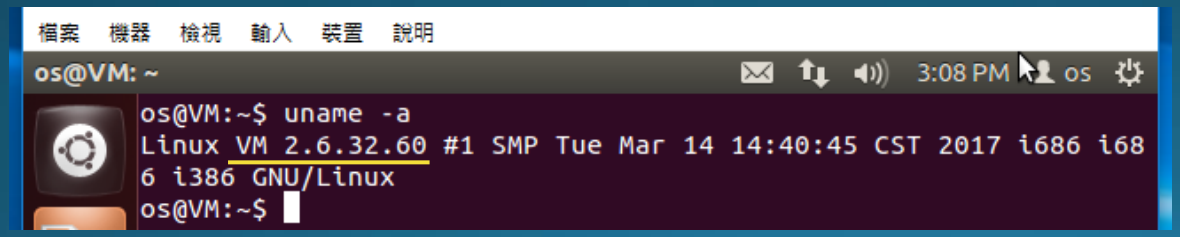
- $ sudo make menuconfig
  - Exit and save
- $ sudo make bzImage
  - You can use make –j# (# is the number of your physical cores) to create multiple threads to speed up the kernel building
- $ sudo make modules
- $ sudo make modules_install
- $ sudo make install
- $ sudo vim /etc/default/grub
  - Add “#” to comment the following 2 lines
    - #GRUB_HIDDEN_TIMEOUT=0
    - #GRUB_HIDDEN_TIMEOUT_QUIET=true
- $ sudo update-grub2
- $ sudo shutdown -r now

Build Linux Kernel (4/4)

- Now, you can select the version 2.6.32.60 kernel in the GNU grub to boot your Ubuntu.



- Then, you can use terminal and type “uname –a” to check the kernel version.



References

- [1] Wikipedia http://en.wikipedia.org/wiki/Kernel_(computing)
- [2] Oracle VM VirtualBox https://www.virtualbox.org/
- [3] Ubuntu http://www.ubuntu.com/
- [4] Linux Kernel in a Nutshell http://www.kroah.com/lkn/

Outline

- Linux Kernel Building
- System Call Implementation
- Project Requirements
- Submission Rules

What is System Call?

- System call is how a program requests services from the kernel of an operating system
- System call provides an essential interface between processes (user) and the operating system (kernel)
- System calls can be roughly grouped into five major categories:
  1) Process control
  2) File management
  3) Device management
  4) Information maintenance
  5) Communication

System Call: Start

Once a system call occurs,

- The processor is switched to the system execution mode (or privileged execution mode)
- Key parts of the current thread context (e.g., the program counter and the stack pointer) are saved
- Then the thread context is then changed:
  - The program counter is set to a fixed (determined by the hardware) memory address, which is within the kernel's address space
  - The stack pointer is pointed at the top of a stack in the kernel's address space

System Call: Execute

Then,

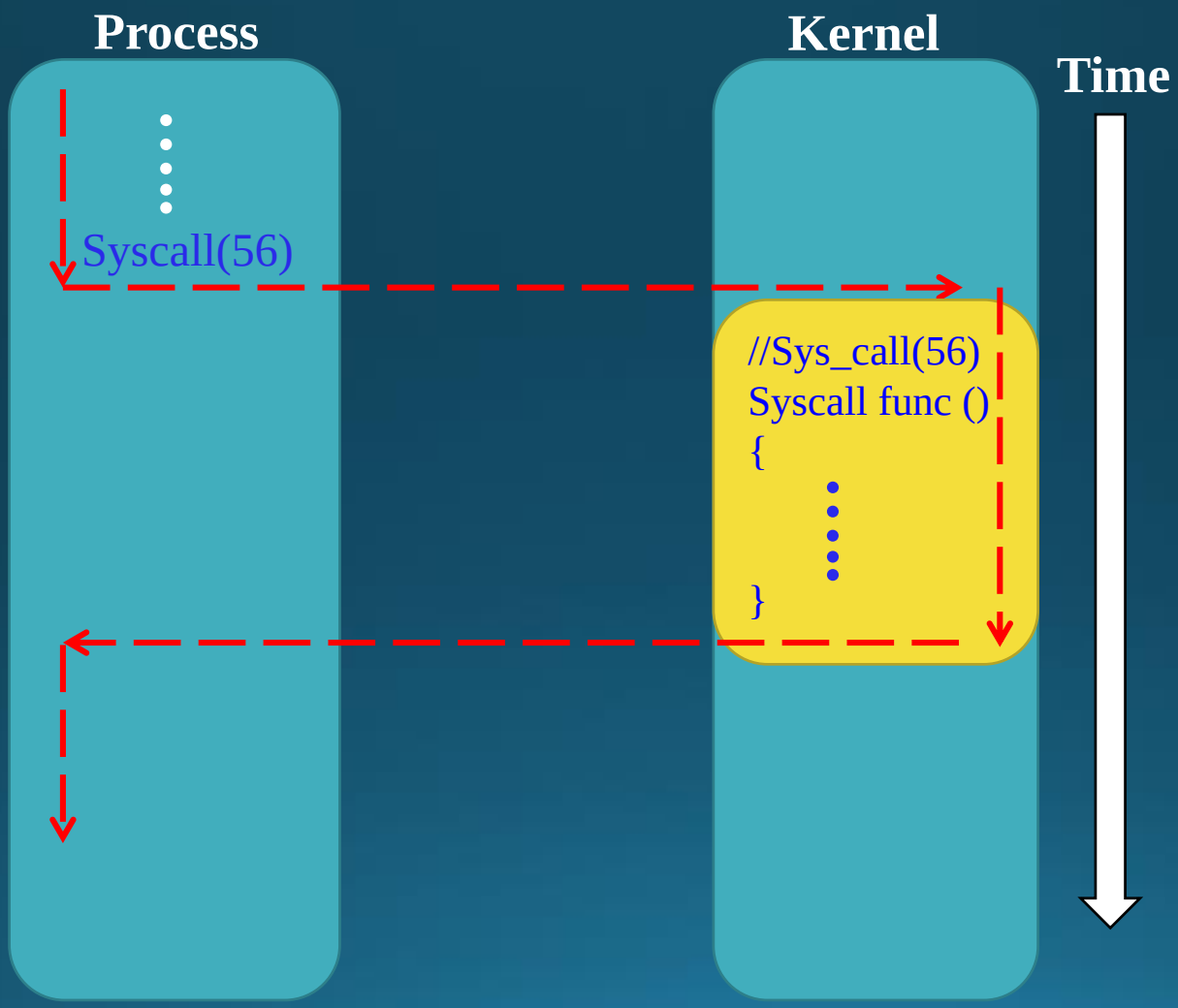
- The calling thread will be executing a system call handler, which is part of the kernel, in system mode
- This kernel's system call handler determines which service the calling process wanted and then performs that service

System Call: End

When the kernel finishes,

- It returns from the system call and this means to
  - Restore the key parts of the thread context that were saved when the system call was made
  - Switch the processor status back to the user execution mode (or unprivileged execution mode)
- Now the thread is executing the calling-process program again and picks up where it left when it made the system call

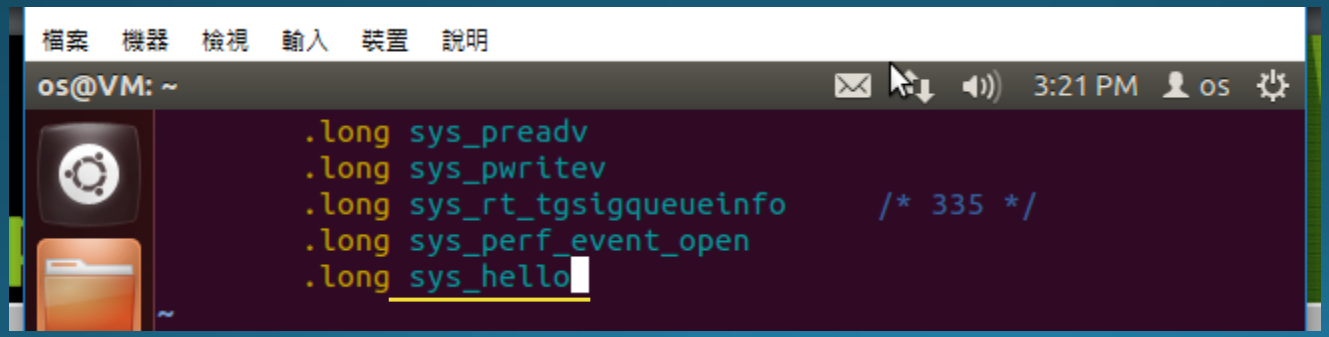
System Call: Diagram



Example: Hello System Call (1/5)

1. Download the linux-2.6.32.60 kernel source code
   - wget https://www.kernel.org/pub/linux/kernel/v2.6/longterm/v2.6.32/linux-2.6.32.60.tar.xz
2. Decompress the kernel source code
   - sudo tar xvf linux-2.6.32.60.tar.xz -C /usr/src
3. Add system call to the system call table
   - Open the file linux-2.6.32.60/arch/x86/kernel/syscall_table_32.S and add the following line.

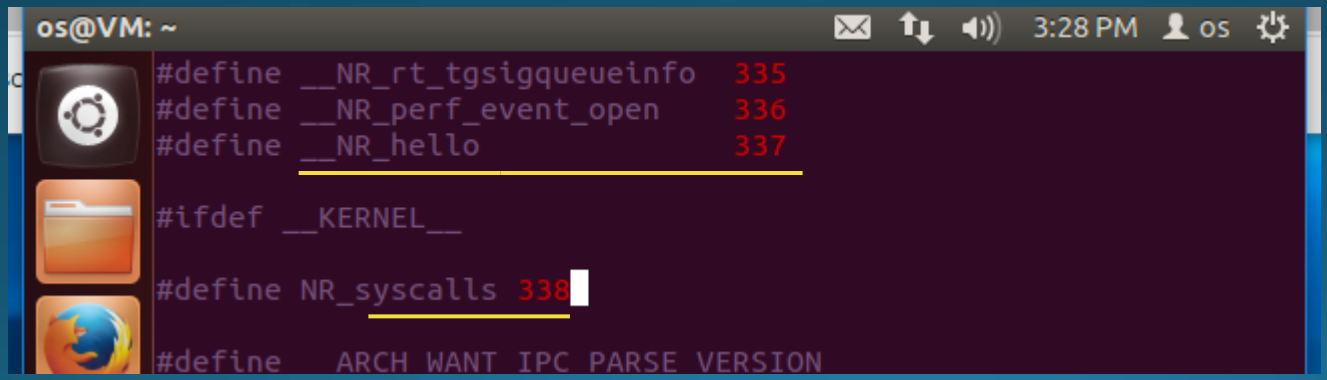
- Add ".long sys_hello"



Example: Hello System Call (2/5)

4. Define macros associated with system call
   - Open the file linux-2.6.32.60/arch/x86/include/asm/unistd_32.h
   - You will notice that a macro is defined for each system call. At the end of the huge macro definition, add a definition for our new system call and accordingly incremented the value of the macro NR_SYSCALLS

- Add "#define __NR_hello 337"
- Update "#define NR_syscalls 338"



Example: Hello System Call (3/5)

5. Define macros associated with system call
   - Now to the file linux-2.6.32.60/arch/x86/include/asm/syscalls.h, add the prototype of the system call.
   - Add the prototype of the system call "asmlinkage long sys_hello(void);"

```
asmlinkage long sys_mmap(unsigned long, unsigned long, unsigned
 long,
                         unsigned long, unsigned long, unsigned
 long);
asmlinkage long sys_uname(struct new_utsname __user *);

asmlinkage long sys_hello(void);
```

   - Now, in the directory of the kernel sources linux-2.6.32.60/kernel/, create a file "hello.c" with the following contents:

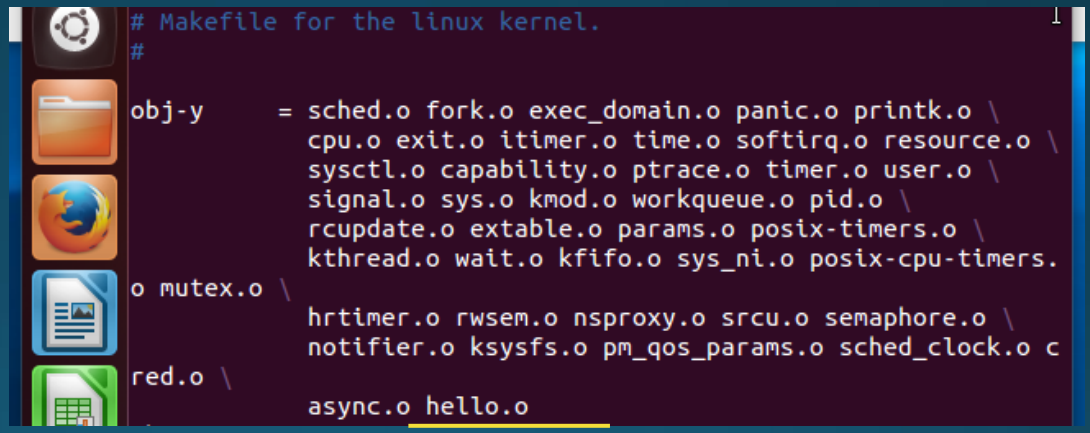
```
#include<linux/kernel.h>
#include<linux/linkage.h>

asmlinkage long sys_hello(void);
{
        printk("HELLO SYSTEM CALL \n");
        return 0;
```

Example: Hello System Call (4/5)

6. After you create the function definition, we have to modify the Makefile (linux-2.6.32.60/kernel/Makefile) so as to compile the new system call to merge it into the kernel

Add “hello.o” to “obj-y”



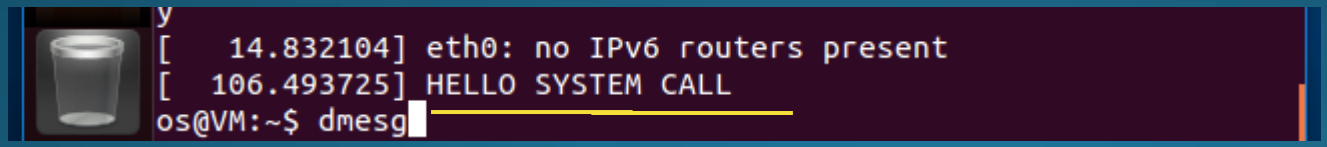
REBUILD the whole kernel and reboot the Ubuntu

Example: Hello System Call (5/5)

- After you REBUILD and REBOOT into the kernel that you just compiled, try to run the following program:
- Test program

```
#include <sys/syscall.h>
#include <unistd.h>
#include <stdio.h>
int main() {
        syscall(337);
        return 0;
}
```

- The output of printk() is written to the kernel log. To view it, type the command “dmesg”



Outline

- Linux Kernel Building
- System Call Implementation
- Project Requirements
- Submission Rules

Requirements of Project 1 (1/2)

- Implement 3 new system calls into your Linux kernel (60%, 20%for each):
- 1. sys_hello (void) *337*\
  - printk “HELLO SYSTEM CALL Student-id1 Student-id2”
- 2. sys_multiply (long, long) *338*\
  - Return the calculation result
- 3. sys_min (long, long) *339*\
  - Return the calculation result

Requirements of Project 1 (2/2)

- Write a test program to test your implemented system calls (20%)
  - Notably, the test program should call the 3 system calls to demonstrate the results
- Report (20%)
  - Implementation details or faced difficulties
  - Your results (please print-screen)
  - At most 2 pages

Outline

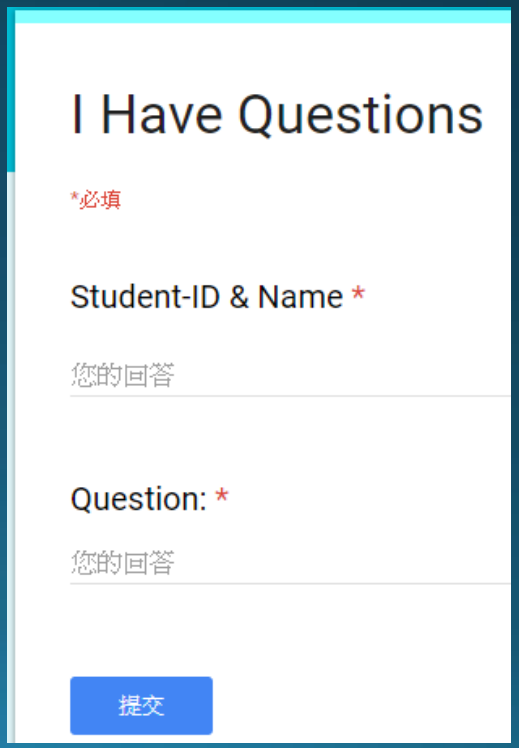
- Linux Kernel Building
- System Call Implementation
- Project Requirements
- Submission Rules

Submission Rules

- Project deadline: 2018/04/14 23:00
  - Delayed submissions yield severe point deduction, -5/day.
  - Upload to FTP Server
  - IP: 140.112.28.143
  - Port: 21
  - Account name: os2018
  - Password: ktw2018os
- The team project should
  - Contain your test program, modified source files (NOT the whole kernel), and your report (PDF format, within 2 pages)
  - Be packed as one file named "OSPJ1_Team##.zip"
- DO NOT COPY THE HOMEWORK

Contact TAs

- If you have any question about the project, please feel free to contact TAs.
- I have questions: https://goo.gl/forms/39eB4ex4w3EX7I4K2
- Han-Yi Lin: d03922006@csie.ntu.edu.tw
- Yu-Chen Lin: f04922077@csie.ntu.edu.tw
- Yi-Shen Chen: d05922009@csie.ntu.edu.tw
- Yu-Chuan Chang: r05922057@csie.ntu.edu.tw



Reference

1. Wikipedia http://en.wikipedia.org/wiki/System_call
2. Linux System Calls http://www.advancedlinuxprogramming.com/alp-folder/alp-ch08-linux-system-calls.pdf