RTS Project 1

Part I: Linux Kernel Building

Part II: Linux Scheduling Policy Testing

Advisor: Prof. Tei-Wei Kuo

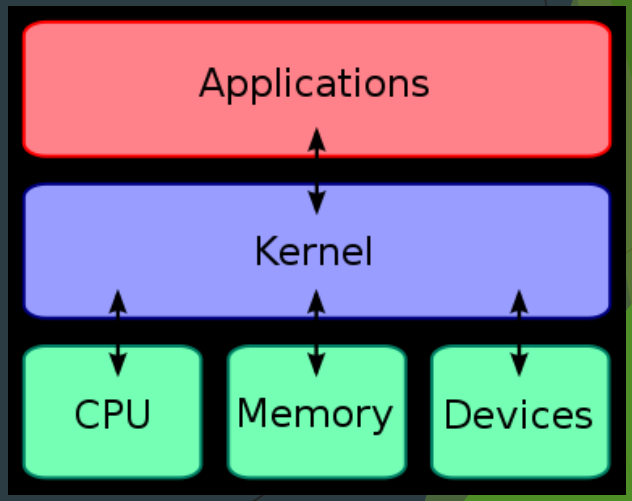
TA: Han-Yi Lin

Outline

- What is “Kernel”
- Environment Setup
- Build Linux Kernel
- Test Linux Scheduling Policy
- 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

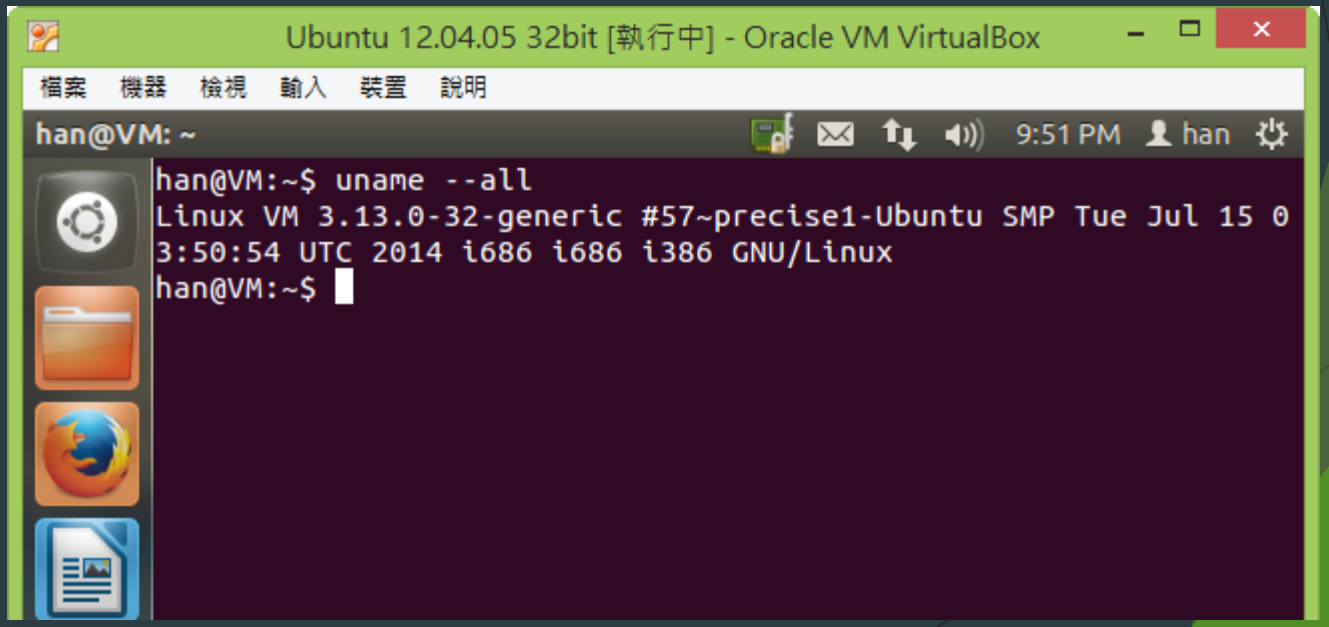


Environment Setup

- 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 (1/5)

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



Build Linux Kernel (2/5)

- $ sudo apt-get install fakeroot build-essential kernel-package libncurses5 libncurses5-dev
- $ cd /usr/src
- $ sudo wget https://cdn.kernel.org/pub/linux/kernel/v2.6/longterm/v2.6.32/linux-2.6.32.68.tar.xz
- $ sudo tar xvf linux-2.6.32.68.tar.xz
- $ cd linux-2.6.32.68
- $ sudo make mrproper

Build Linux Kernel (3/5)

- $ sudo make menuconfig
- $ 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

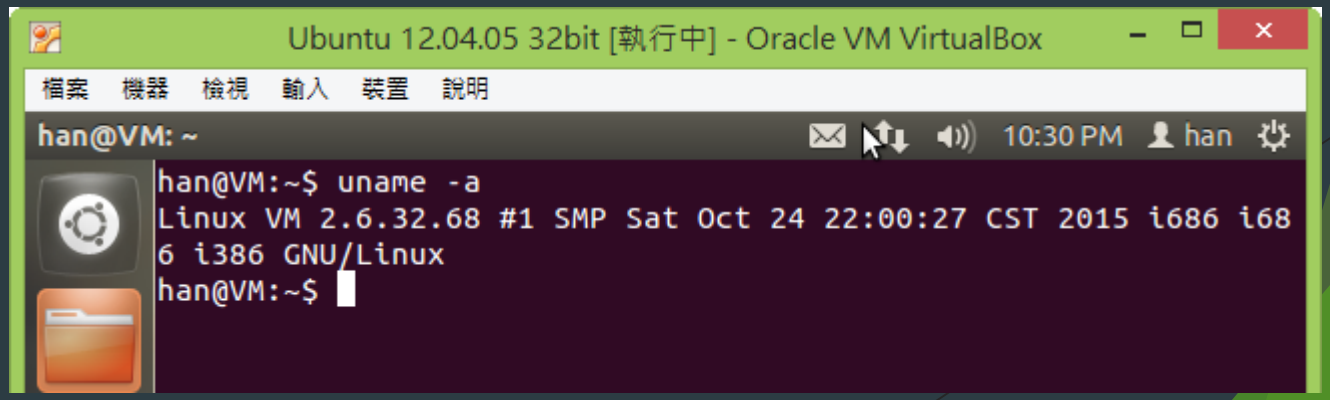
Build Linux Kernel (4/5)

- $ sudo make install
- $ sudo vim /etc/default/grub
  - Add “#” to comment the following 2 lines
    - #GRUB_HIDDEN_TIMEOUT=10
    - #GRUB_HIDDEN_TIMEOUT_QUIET=true
- $ sudo update-grub2
- $ sudo shutdown -r now

Build Linux Kernel (5/5)

- Now, you can select the version 2.6.32.68 kernel in the GNU grub to boot your Ubuntu.

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



References

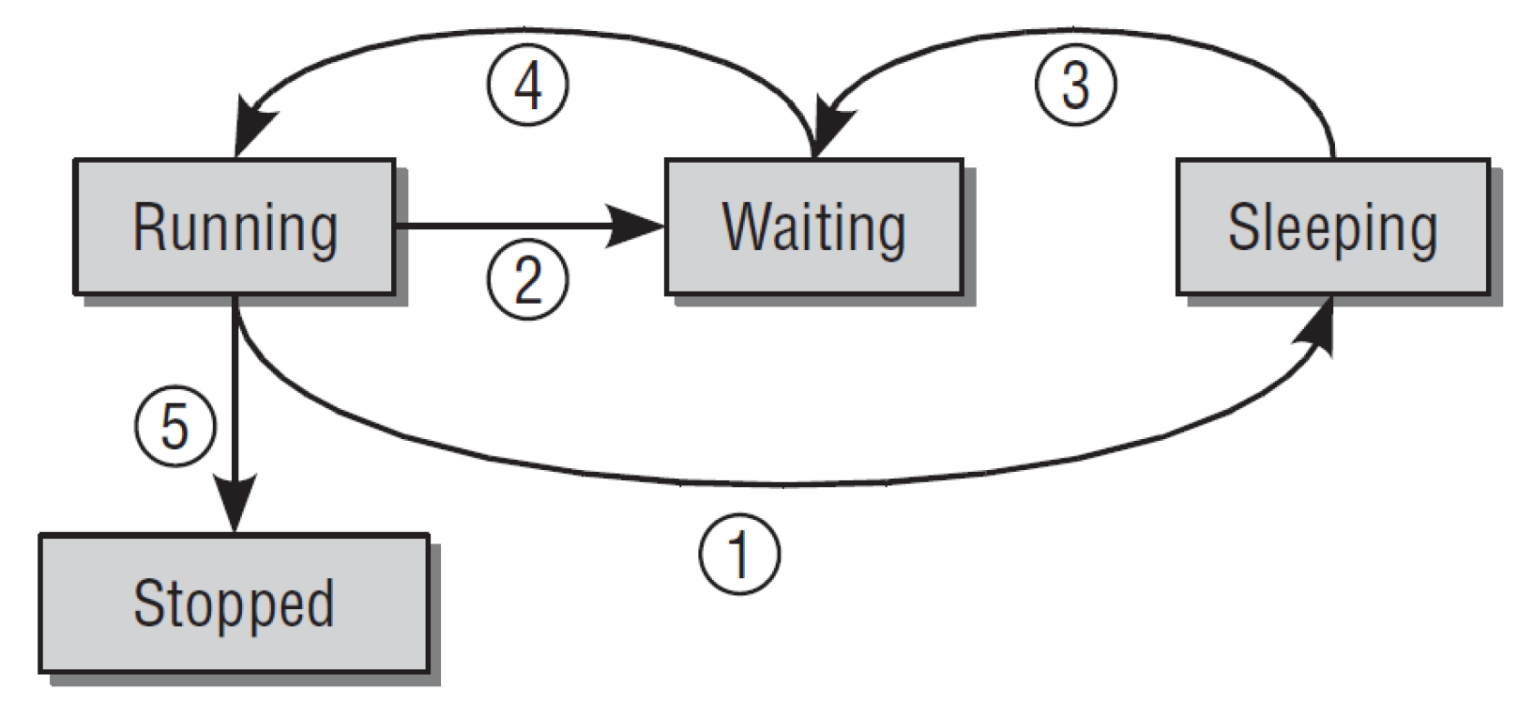
- [1] Wikipedia http://en.wikipedia.org/wiki/Kernel_(computing)
- [2] Oracle VM VirtualBox https://www.virtualbox.org/
- [3] Ubuntu http://www.ubuntu.com/
- [4] Linux Kernel in a Nutshell http://www.kroah.com/lkn/

SCHEDULING IN LINUX (補充資料)

Process Life Cycle

- A process is not always ready to run.
- The scheduler must know the status of every process in the system when switching between tasks.
- A process may have one of the following states:
  - Running — The process is executing at the moment.
  - Waiting — The process is able to run but is not allowed to because the CPU is allocated to another process. The scheduler can select the process at the next task switch.
  - Sleeping — The process is sleeping and cannot run because it is waiting for an external event. The scheduler cannot select the process at the next task switch.
- The system saves all processes in a process table.

Transitions between Process States



The Need of the Scheduler

- A unique description of each process is held in memory and is linked with other processes by means of several structures.
- This is the situation facing the scheduler, whose task is *to share CPU time between the programs to create the illusion of concurrent execution*.
- This task is split into two different parts –
  - One relating to the scheduling policy and
  - The other to context switching

Scheduling in Linux (1/2)

- The schedule function is the starting point to an understanding of scheduling operations.
- It is defined in “kernel/sched.c” and is one of the most frequently invoked functions in the kernel code.
- Not only priority scheduling but also two other soft real-time policies required by the POSIX standard are implemented.
- E.g., completely fair scheduling, real-time scheduling and scheduling of the idle task, etc.

Scheduling in Linux (2/2)

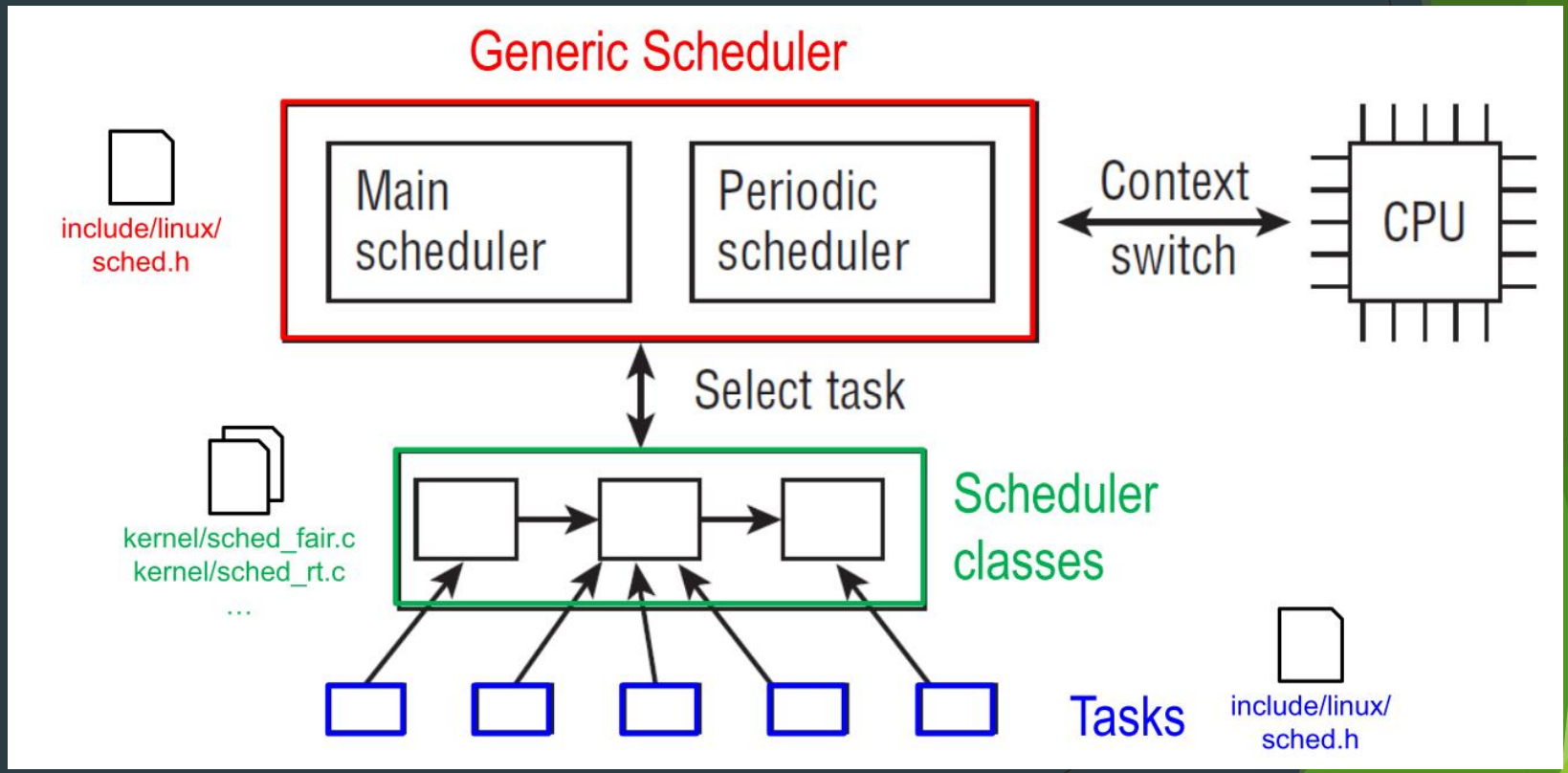
- The scheduler uses a series of data structures to sort and manage the processes in the system.
- Scheduling can be activated in two ways:
  - Main scheduler: Either directly if a task goes to sleep or wants to yield the CPU for other reasons,
  - Periodic scheduler: Or by a periodic mechanism that is run with constant frequency to check from time to time if switching tasks is necessary
- Generic scheduler = Main + Periodic schedulers

Linux Scheduling Subsystem

Generic Scheduler

Scheduler Classes

Task | Task | Task



Linux Scheduling Policy Testing

Linux Scheduling Policy Testing

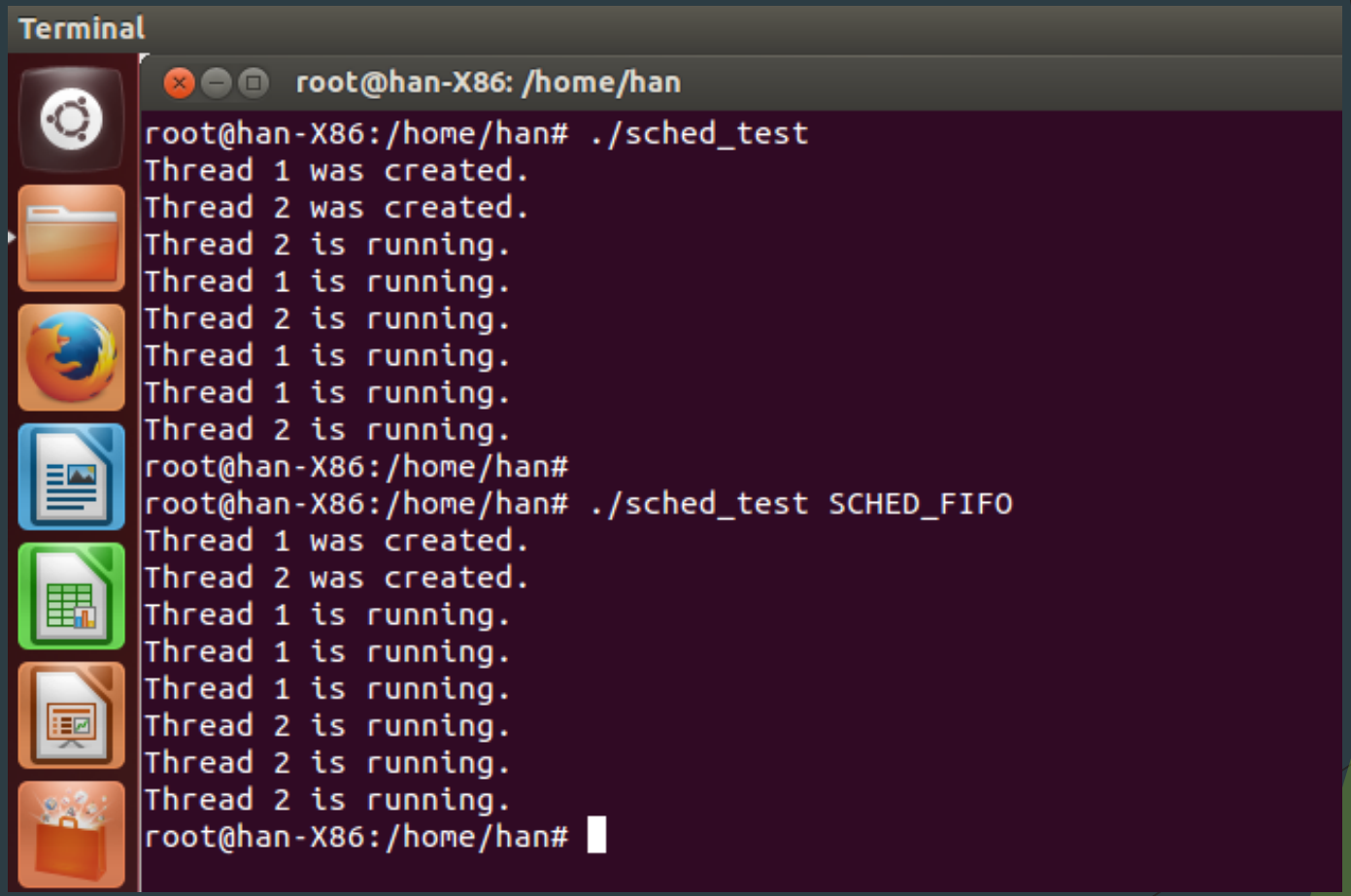
- Linux Scheduling Policy Classes
  - Normal Scheduling policies (Non-real-time)
    - SCHED_OTHER, SCHED_BATCH, SCHED_IDLE.
  - Real-Time policies
    - SCHED_FIFO, SCHED_RR.
- The default scheduling policy is non-real-time.
- In this part, using Linux real-time scheduling policy (FIFO) to schedule threads in a process.

Implement

- Write a C program (sched_test.c) to create two threads.
- Each thread will print who is running and busy for 1 second.
- Run the program by default time-sharing schedule policy and show the result.
  Ex. $ ./sched_test
- Run the program by real-time scheduling policy (FIFO) and show the result.
  Ex. $ ./sched_test SCHED_FIFO

```
1  int main(){
2
3  create thread 1
4  print "Thread 1 was created"
5
6  create thread 2
7  print "Thread 2 was created"
8
9  }
10
11 void thread_function(){
12
13 print "Thread # is running"
14 busy 1 second
15
16 }
```

Result


Terminal
root@han-X86: /home/han
root@han-X86:/home/han# ./sched_test
Thread 1 was created.
Thread 2 was created.
Thread 2 is running.
Thread 1 is running.
Thread 2 is running.
Thread 1 is running.
Thread 1 is running.
Thread 2 is running.
root@han-X86:/home/han#
root@han-X86:/home/han# ./sched_test SCHED_FIFO
Thread 1 was created.
Thread 2 was created.
Thread 1 is running.
Thread 1 is running.
Thread 1 is running.
Thread 2 is running.
Thread 2 is running.
Thread 2 is running.
root@han-X86:/home/han#

Hint

- int sched_setscheduler(pid_t pid, int policy, const struct sched_param *param);
- The policy corresponding value define in /include/linux/sched.h
- Set the priority of real-time process (sched_param *param)
- The permission to run real-time process
- The number of CPU cores on your virtual machine
  - CPU affinity

Project1 Requirements

- The project should
  - Contain your report (PDF format, within 2 pages) and a C program.
  - 報告內容可以是：編譯時遇到的問題、編譯步驟的解釋、關於 Linux Scheduling Policy Testing 與 Linux scheduling trace code 的心得、等等。
  - Please show your name, student ID and E-mail in your report
  - Be packed as one file named “RTS_PJ1_Team##.zip” or “RTS_PJ1_StudentID.zip” (If one member)
  - RTS_PJ1_Team##.zip or (RTS_PJ1_StudentID.zip)
    - RTS_PJ1_report.pdf
    - sched_test.c

Submission Rules

- Project deadline: 2015/11/09 (Monday) 23:59
  - Delayed submissions yield severe point deduction
- Send your project to TA
  - TA's E-mal: d03922006@csie.ntu.edu.tw
  - Mail title: RTS_PJ1_Team##
    or "RTS_PJ1_StudentID" (If one member)
  - If you have new version: RTS_PJ1_Team##v#
    or "RTS_PJ1_StudentIDv#" (If one member)
- DO NOT COPY THE HOMEWORK

References

- Reference Book
  - Professional Linux® Kernel Architecture, Wolfgang Mauerer, Wiley Publishing, Inc.