An introduction of operating system project

Jeaho Hwang
• Linux introduction
• Running Linux
• Developing Kernel
• Useful Tools
• Project 0
• Project Policy
• What is an “operating system”?
  – A software that application software operates on
Linux operating system

- An UNIX-like operating system by Linus Torvalds

Linux distributions
Linux Operating System

- Linux distribution =
  - Applications (standard C library + system utilities)
  - Linux (kernel)
- A pre-emptive multi-process monolithic kernel
  - Event driven architecture
  - Supports multiple architectures
  - Kernel modules to extend
Kernel abstraction

- Process, thread
- Virtual memory
- Block device, file
- Frame buffer device
- Character device
- Socket
Kernel source tree

- **Documentation**
- **arch**: architecture dependent codes
  - Boot, interrupt, system call and memory management
- **kernel**: scheduler and synchronization
- **mm**: memory allocation and page caching
- **block**: block device abstraction
- **net**: network stack
- **fs**: virtual file system and file systems
- **drivers**: device drivers (physical and virtual)
SWE3015: Operating System Project

STARTING KERNEL DEVELOPMENT
Running Linux

• Which device can we run?
  – ANY DEVICES!
    • Your desktop
    • Your laptop
    • Or your phone
    • Even or refrigerator!

  – But, we recommend laptop
    • You should demonstrate your work in class
    • If you can, we don’t care if you use desktop.
Running Linux

• How can we run?
  – Native
    • Installing Linux on your own PC
      – Recommended for experience Linux users
    • You should format PC or set up multi-booting
  – Virtual Machine
    • Using any VM tools (virtualbox, VMware, etc.)
    • Virtualbox is recommended because it’s free!
      – [https://www.virtualbox.org/](https://www.virtualbox.org/)
      – Assign at least 8GB storage and 1GB RAM

Because you must be stuck by flood of crashes, don’t boot the PC you are working by your own kernel!
Running Linux

• How can we install Linux?
  – Choose any distribution you want
    • **Ubuntu**, Centos, Gentoo, etc.
    • We are using ubuntu-13.04-server-amd64 for explanation.
      – Server might be better because it provides no graphics.
      » You can see console logs directly
Developing Kernel

- Downloading kernel source code
  - From [https://www.kernel.org/](https://www.kernel.org/)
    - Download tar.xz of your favorite version
  - From Ubuntu archive
    - `apt-get source linux-source-3.8.0`
  - From the git source of the ubuntu kernel
    - `git clone git://kernel.ubuntu.com/ubuntu/ubuntu-precise.git`

- Modify the source
  - As you want 😊
Kernel development

- Coding style / indentation
  - Refer to Documentation/CodingStyle
  - Written in C, but with an object-oriented programming style

- The best documentation is the code itself
  - Most of codes are not documented
  - Or, the documents are stale

- Beware of race conditions and the interrupt context
Developing Kernel

• Building the kernel
  – See https://help.ubuntu.com/community/Kernel/Compile/
  – And http://bagjunggyu.blogspot.kr/2013/12/custom-kernel.html (Korean)
  – Configure
    • cp /boot/config-`uname -r` .config
      – Copying current booted kernel configure
    • make menuconfig
      – Menu style configuration, typical usage
    • make localmodconfig
      – Default setting + compile only modules now loaded
      – Saving compile time
Developing Kernel

• Building the kernel
  – Compile
    • make \(-j8\)
      – \(-j\) stands concurrent building threads
      – Recommend equal or less than the number of CPU core
  – Installing
    • make modules\_install
      – Installing compiled kernel modules
    • make install
      – Installing the kernel
Useful Tools

• Text Editor
  – Vi is common but I respect emacs, too.
  – Of course you can use nano, eclipse, or visual studio 😊

• Ctags, cscope
  – Generates an index of C objects
  – #make tags

• Git, svn
  – Version control systems
  – Unless you want to lose your work, make your repository.
    • **REAL CASE:** gcc –o test.c test.c
(un)Useful Tips

• After installing ubuntu
  
  $sudo sed -i ‘s/us/kr/’ /etc/apt/sourcelist
  (kr or kr2)
  $sudo apt-get update
  $sudo apt-get install vim

• ...and let’s discuss your own tips
Project 0: Start Up!

- Make a development environment (personal, not team!)
  1. Install Linux somewhere
  2. Install your own Linux kernel
     • HINT: make menuconfig -> General setup
  3. Result of `uname –r` should be 3.x.x-your_name
     • Example: 3.8.13.13-Jay
  4. (bonus point) print any string on console log
     • You can see the log by `dmesg`
     • HINT: use printk() on the kernel code

- Due date: Mar 16th (Sun) 24:00
  - Demo presentation will be on 19th class
Project 0: Start Up!

• Example

```bash
hahaman5@ubuntu:~$ uname -r
3.8.13.13-Jay
hahaman5@ubuntu:~$ dmesg | head
[ 0.000000] Initializing cgroup subsys cpuset
[ 0.000000] Initializing cgroup subsys cpu
[ 0.000000] Linux version 3.8.13.13-Jay (hahaman5@ubuntu) (gcc version 4.7.3 (Ubuntu/Linaro 4.7.3-1ubuntu1)) #4 SMP Tue Mar 4 16:00:55 KST 2014 (Ubuntu 3.8.0-19.29-Jay)
[ 0.000000] KERNEL supported cpus:
[ 0.000000] This is Jay speaking!!!
[ 0.000000] Intel GenuineIntel
[ 0.000000] AMD AuthenticAMD
[ 0.000000] Centaur CentaurHauls
[ 0.000000] e820: BIOS-provided physical RAM map:
```

Project 0: Start Up!

• Assignment submission
  – Mail to hahaman5@gmail.com
  – The title should be “[SWE3015] project0 name”
    • If assistants missed mail because of violation, we’d not care it.
  – The mail must contains following
    • Screenshots of ‘uname’
    • Screenshots of ‘printk()’ (optional)
    • A brief report
      – Explain how you did it briefly.
      – Write in your comfortable language (but no Português or Español)
Project Policy

• Team project (personal for proj. 0, 0.5)
  – Max. 3 people for each team
  – Evaluate contributions in report
    • All members MUST participate all projects.

• Schedule
  – Due is Sunday 24:00 (or Monday 00:00)
  – Accept late submissions up before demo class
    • 20% penalty per day
    • Cannot start the next project until the current is finished
  – Demo on the next class
Project Policy

• Token
  – 4 tokens for each team
  – Use tokens freely to avoid penalties
    • Must express on report clearly
    • No advantage for remaining tokens
    • No refund or exchange 😊
  – Examples
    • Late 2 days, use 2 tokens -> no penalty, 4 tokens remain
    • 2 days, use 1 token -> 20% penalty, 5 tokens remain
• Any question?
References

- http://www.makelinux.net/kernel_map
- https://help.ubuntu.com/community/Kernel/Compile/
Kernel feature (1)

- Compatible to POSIX
- Multi-architecture
  - x86, ARM, MIPS, ...
  - Multi-processor, NUMA, ...
- Multi-process, multi-thread
  - Fair, time sharing scheduler
- Synchronization primitives
  - Semaphore, spinlock, RCU, futex, ...
Kernel feature (2)

- Device mapper
  - LVM, software RAID, flash caching, ...

- File system
  - ext4, btrfs, f2fs, ...
  - FUSE

- OSS sound

- Kernel Virtual Machine

- Wide varieties of device drivers
  - Block, network, graphics, sound, tty, ...