

CS 378 (Spring 2003)

Linux Kernel Programming

Yongguang Zhang

(ygz@cs.utexas.edu)

Read Me First

- Everything is in the Web
 - Syllabus, lecture notes, reading materials, projects, ...
 - <http://www.cs.utexas.edu/users/ygz/378-03S>
 - Just listen and participate, need not take notes

Who Am I?

- Yongguang Zhang
- Ph.D. in Computer Sciences, 1994, Purdue
- 2nd time teaching this course
 - Last time: Spring 2002
 - Taught 4 undergrad, 3 grad courses before
- 1994-2000: Senior Research Scientist at HRL
 - Doing basic research for Boeing/Raytheon/GM
- My web site: <http://www.cs.utexas.edu/users/ygz>

Your TA

- Chun-Chi *Jonathan* Chen
 - Graduate Student in UTCS
 - Many years of experience in Linux
 - Will help your learning and grade your projects
 - E-mail: ccchen@cs.utexas.edu
 - You will meet him this Friday in class

What is this Course About?

- Linux Kernel Programming
 - How to work in an example modern OS kernel
- Positioned as the laboratory component in the CS undergrad OS curriculum
 - Taken after CS372 (Introduction to OS)
- Comparison to CS372
 - CS372: concepts and principles of an OS
 - CS378: an example of how they are actually done

What Are We Going to Learn?

- Understanding linux kernel structure
 - Know how the kernel works
 - Know how to customize kernel
- Writing kernel code
 - Experience developing code for OS kernel
- System programming skill
 - Ability to deal with large, complex systems.
 - Very different from application programming (e.g., using Java)
- New s/w development model: open community

Why Take This Course?

- Linux is increasingly important – it is a good skill to have
- You want to become a system programmer
- You want to find a job as a system programmer
- You want to go to graduate school and do systems research

What is This Course NOT for?

- How to use Linux
 - You are CS seniors, you should have known by now
 - If not, there are lots of books and online resources
 - Still no? there are dummy books and training courses
- How to program in Linux
 - See above
- Linux certificates
 - Those are for technicians
 - You are a science student from a top-10 CS department, you don't want those kind of jobs

Prerequisite

- CS372 (introduction to OS)
 - CS372: concepts and principles of an OS
 - CS378: an example of how they are actually done
- Taking CS372 and CS378 together?
 - Highly discouraged
 - Statistics from last Spring
 - Those who had CS372: average grade = 3.2
 - Those who did CS372 concurrently: average grade = 0.4
- Will UTCS repeat this course?
 - This is a special topics course, no definite plan

Course Structure

- Hands-on, Project-oriented
 - Lots of programming assignments (9 totals)
 - You learn by doing the projects and consulting the books (textbooks, references, online resources)
 - No exam
- Lectures
 - Cover the basics and overview parts of Linux kernel
 - Not possible to cover everything in depth
 - There will be guest lectures too (by Linux Gurus from the industry, mainly IBM Linux Technology Center)

Projects

- 3 Individual Assignments (week 1-3)
 - One per week, on basic topics of Linux kernel (that everyone should know)
- 5 Group Projects (week 4-8)
 - Each cover different part of the kernel
 - 2-3 students per group
 - 8 projects will be announced before week 4
 - You pick 5 out of 8 projects, and do one per week
 - You can change groups for each project
- 1 Final Project (week 10-15)

The Final Project

- A larger project to put all your skills in use
 - On an interesting topic, may have practical value, may make contribution to Linux community
 - Project list (and group size) will be announced after week 6
 - You form a group and pick a project
- Opportunity to do projects with IBM Linux Technology Center
 - Many final projects are suggested by IBM LTC
 - You can work with mentors from IBM LTC

Lectures

- Part-I: First 4 weeks
 - The basics of Linux kernel
- Part-II: Week 5-11
 - Overview of different parts of the kernel
 - May not cover everything in depth (you will learn them through your projects)
- Part-III: Week 12-15
 - Special topics, guest lectures, etc.

Student Strategy

- Part-I: Master the basics
 - Lectures, assignments
- Part-II: Come to the lecture and learn the basics
- Part-II: Learn the details through projects
 - Start from the project description
 - Identify the parts of kernel that the project covers
 - Learn those parts by reading the textbooks yourself
 - Use what you learn in the project
 - Bring your questions to class
- Part-III: See above

Example Bad Strategies

- Read the textbooks from page i to page 765
 - This is not a reading class
- Attempt to memorize the code
 - In system programming, the skill is more important
 - It is good to know the code, but only through practice
- Skip the lectures
 - Your choice of projects does not cover every topics, but you will learn them in class
 - Class participation counts as 5% of your grade

Expectation from You

- You work hard (approximately 10 hours/week)
- You take initiatives
 - You learn the details by yourself
 - Don't wait till the last minute to start your project
- You work well with your teammate
 - Or you may have no one to work with for next project
- Everything in Linux
 - No project, document written with MS Windows

Grades

- 5% Class participation (not just attendance)
- 15% Community building
 - We are trying something new this semester, to follow the open community software development model
 - More on this in a later lecture
- 80% Projects:
 - 3 individual assignments: 18% (6% each)
 - 5 group projects: 30% (6% each)
 - 1 final project: 32%

Course Matters

- **Class Meeting Time: Mon/Fri 12:00pm-1:15pm**
 - No class on Wednesday
- **Professor Office Hour: Mon/Fri 2:30pm-3:30pm**
 - Other time please make appointment by e-mail (ygz@cs.utexas.edu)
- **TA Office Hour: TBA**
 - Questions about your projects, Linux kernel: see your TA first
- **Course web site: all about this course**

Textbook

- Textbooks: one is required
 - Understanding the Linux Kernel, second edition
 - Linux Kernel Programming, third edition
- Reference books: ask me
- Online resources
 - Many many linux websites
 - A good start: www.linux.org, www.kernel.org
 - Use google

Where Machines Can You Use?

- Any public linux machine in UTCS
 - Run "cshosts -a publinux" – 88 to choose from
 - Use your normal CS account
 - You will learn how to run/compile linux in your student account in a moment
- At home
 - You can remote login to a public linux machine
 - You could try use your personal computer, but your project must run on UTCS so TA can grade it
- Non-CS student: get a CS class account

Today's Short Topics

- System programming
- Linux history
- Using user-mode-linux

Systems Programming

- What is System Programming?
 - Dealing with large complex and usually monolithic system software (e.g., OS kernel)
- Why large, complex, and monolithic software?
 - Structure often limits flexibility and performance
 - Working such system requires extraordinary skill
 - It is often an art more than a science or an engineering
- In industry
 - System programmers usually have more “respect” than application programmers

Modern Operating Systems

- Architecture
 - Kernel: OS core running in privileged mode
 - User-space: compiler, editor, GUI, ..., system and application programs
- Unix
 - Many flavors (most not “free”)

What is Linux

- Full featured, Unix-like OS
- Open source: entire kernel source available free
- Community: created by a loose worldwide community of programmers collaborating mainly through the internet
- Portable from mainframe to hand-helds
- Estimated users: over 18 millions (counter.li.org)

GNU: Open Source before Linux

- The concept of “free” software
 - Through 70s: Richard Stallman advocates “free” software
 - “free” as in freedom (not zero cost):
 - free to use, distribute (for a profit), and modify
- 1984: Richard Stallman founded GNU
 - Goal: to produce free software.
 - GPL: ensure software freedom by copyright terms
 - GNU software: Unix-like programs (no kernel)

Linux Right on the Start

- 1991: Finnish student Linus Torvalds started working on update of Minix
- From early start, Linus asked for volunteers on the Internet to help him develop Linux
 - People started using and publicizing Linux
 - A number of programmers joined the project
- From the start, the source code has been freely available on the Internet
- GNU has lots of user-space programs but no kernel

Linux Kernel Evolution

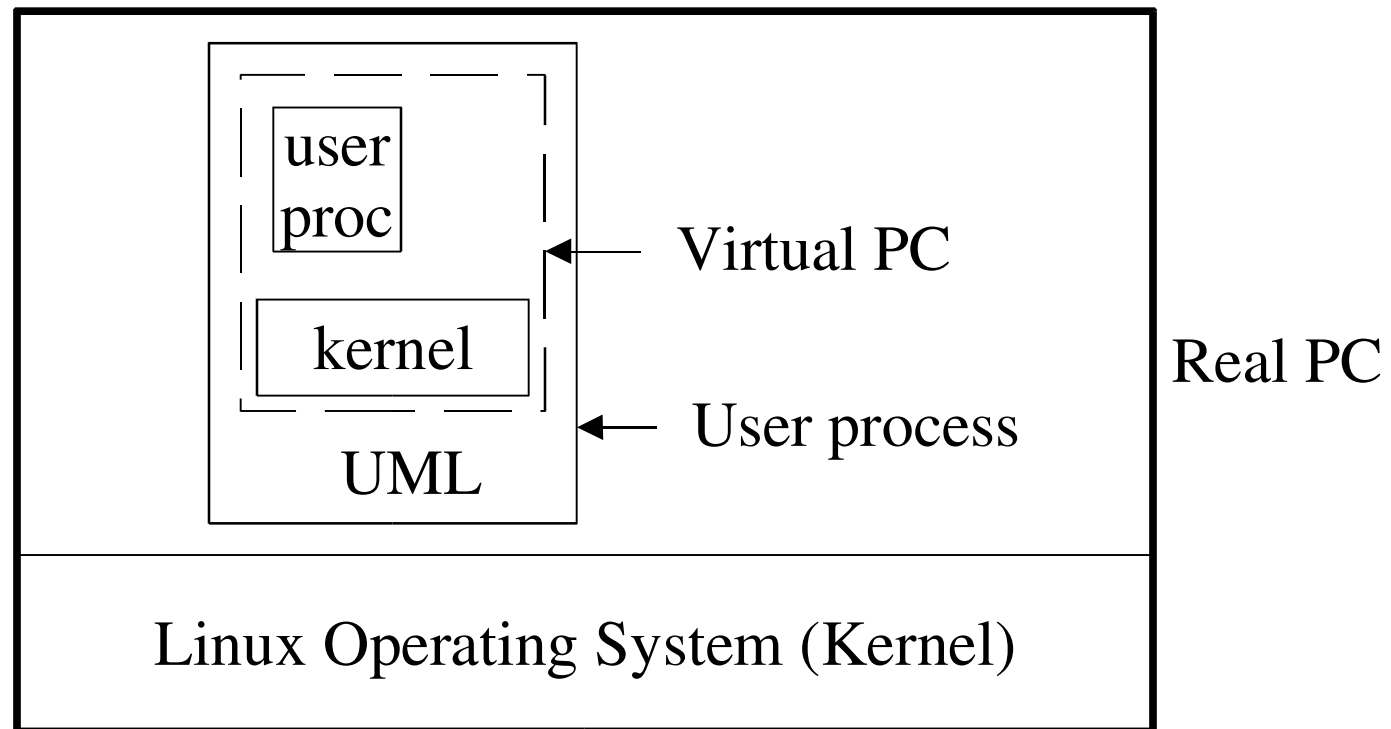
- 0 (Apr 1991): First e-mail from Linus
- 0.01 (Sept 1991)
- 1.0 (March 94), user: 100,000
- 1.2 (March 95), user: 500,000
- 2.0 (June 96), user: 1,500,000
- 2.2 (January 99)
- 2.4 (January 2001)
- Old/Even versioning system
 - 2.4.x: stable kernels
 - 2.5.x: development kernels

The Kernel of Our Study

- Version 2.4.19-46um
 - Stable version 2.4
 - Nineteenth release (2.4.19)
 - With User-Mode-Linux support (Patchlevel 2.4.19-46)
- The entire kernel source code is at
 - `/projects/cs378.ygz/src/linux-2.4.19`
- Huge (160MB)

User-Mode-Linux (UML)

- Just think of it as a virtual PC



Running UML

- Copy the root file system (root_fs_utcs) to your directory
 - `cp /projects/cs378.ygz/fs/root_fs_utcs .`
- To boot up UML:
 - `/projects/cs378.ygz/bin/linux ubd0=root_fs_utcs`
- ubdX: Think as a virtual disk
- Watch all the output until you see the login sign.

Being Root

- Then you can login as root with username "root" and password "root".
- Be careful – you are now the root
 - You can do serious damage to your file system
- Shutting it down
 - Simply log in as "root", and type "halt"
 - Watch all the messages until the original shell prompt (before running UML) returns

Compiling UML

- Create your own source code tree

```
mkdir ~/my-first-linux
cd ~/my-first-linux
ln -s /projects/cs378.ygz/src/linux-2.4.19 .
```
- Configuration

```
cp /projects/cs378.ygz/src/uml.config-2.4.19 .config
make xconfig ARCH=um
```

 - [when the window pops up, select 'Save and Exit']

Compiling UML (2)

- Building the kernel (takes about 5-10 minutes)
 - make dep ARCH=um
 - make linux ARCH=um
- Now the kernel executable file is `./linux`
 - Or, `~/my-first-linux/linux`
- Test the new kernel:
 - `./linux ubd0=../root_fs_utcs`
 - Assuming the `root_fs_utcs` file is one level above
- Disk space requirement: 20M minimum
 - More when you start to include options

Today's Homework

- Practice using Linux (if you are not familiar with it already)
- Practice running UML
 - Also test your system administration skill in UML
- Do Assignment 1
- Reading before coming to class Friday
 - LKP: chapter 1-2
 - ULK: chapter 1, appendix C

Assignment 1

- Compile your own version of UML
- Explore UML configuration
- Learn to debug UML
 - <http://usemode-linux.sourceforge.net/debugging.html>
 - Hint: you need to reconfigure & recompile UML first
- Due: Jan 20, Mon (one week from today)

Next Class

- Guest Lecture from IBM LTC
 - Linux Kernel Overview

Acknowledgement

- This course is in part supported from a grant from IBM Linux Technology Center (Austin)