CSC 222

System Programming
Lecture 1: Class info & Intro
Dr. Box Leangsuksun

www.latech.edu/~box

The Original materials were prepared by Dr. Christian Duncan & from Silberchatz's instructor guide

Box's 1 minute Bio

PhD in CS (1995):

- PhD Thesis: Resource management/allocation in Heterogeneous Parallel Distributed Computing

7 years in industry labs (Bell-Labs, Lucent Technologies)

- Highly Reliable Software/system (IN, Service Management)
- Architect, PM, Tech lead (15-30 team size)
- R&D -> 4 major network management products

Associate Professor in CS since 2002.

- 15 graduate students (4 PhD)

Research Interest

- Cloud/Cluster computing, Fault Tolerance OS/Runtime Software Engineering

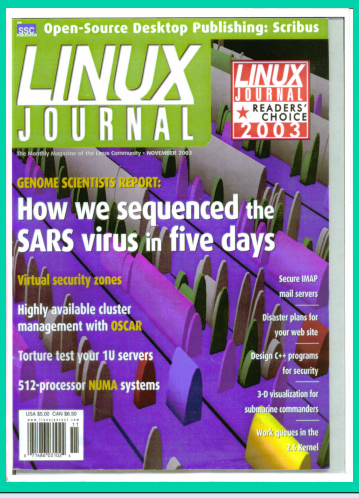
Services

- IEEE Cluster Computing Program committee member 2004-2005
- A founder and CO-Chair: High Availability and Performance Computing
- 2003 Outstanding Teach Award, COES, Louisiana Tech U.

- Creator of www.searchkatrina.org

Appeared in a front cover in two major Linux magazines, various technical papers, research exhibitions.
web site: http://xcr.cenit.latech.edu/ha-oscar

HA-OSCAR beta was released to open source community



Class & Contact Info

TR: 10:00 – 11:50

Room NH 153

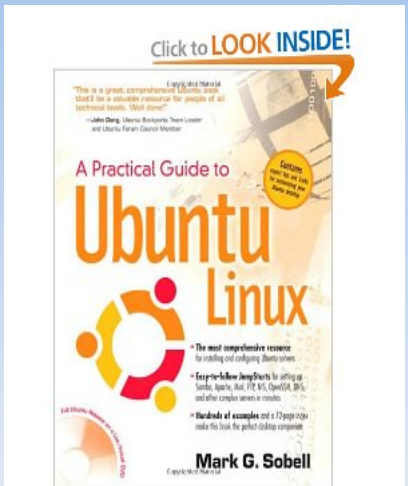
Dr. Box Chokchai Leangsuksun
phone: (318) 257-3291
Nethken Hall 243

E-mail: box@latech.edu or naibox@gmail.com

Facebook: naibox@gmail.com

Books & References

A Practical Guide to Ubuntu Linux by Mark Sobell



Books & References

C and Unix: Tools for Software Design by Barrett and Wagner

The C Programming Language by Kernighan and Ritchie: Referred to as K&R this is *the* book on C. Ritchie was the (co-)creator and (co-)developer of C and Unix.

A Book on C by Kelley and Pohl: Not *the* book but *a* book.

The C++ Programming Language by Stroustrup: the creator of C++.

C++ for Java Programmers by Mark Allen Weiss

Books & References

Some Wikibooks:

- http://en.wikibooks.org/wiki/Linux_For_Newbies: This WikiBook is rather abbreviated with lots of stubs but some material present does seem promising.
- http://en.wikibooks.org/wiki/Guide_to_Unix
- http://en.wikibooks.org/wiki/Bourne_Shell_Scripting
- http://en.wikibooks.org/wiki/C++_Programming

Purpose

to provide the students with an introduction to system-level programming. Although not the primary focus of this course, instruction shall be done within the context of C/C++ and Linux/ Unix.

- *Prepare students for OS class*

Course Outcomes

- To work effectively in a UNIX-style environment.
- To explain the basic operations that are performed from the time a computer is turned on until a user is able to execute programs.
- To write medium to large C/C++ programs for a range of applications.
- To use systems tools for C/C++ programming.
- To write C/C++ programs that use the UNIX system call interface.
- To write small to medium size scripts, in various scripting languages, for a range of applications.

Outline (1)

- Using standard Linux command line in user & system environments, file systems, and tools.
- Writing programs in a scripting language.
- Writing programs in the C programming language, including using pointers and memory management.
- Writing programs in the C++ programming language, including using classes and templates.

Outline (2)

- Using standard C/C++ libraries for various programming tasks.
- Using various tools to enhance programming, such as makefiles, profilers, lint, and debuggers.
- Examining what happens during program compilation, linking, and loading.
- Interacting directly with the operating system by making system calls for file management, file execution, process control, and interprocess communication.

Grading

A is a 90 or higher

B is 89-80

C is 79-70

D is 69-60

- 20% Lab Exercises/Quizzes
- 30% Programming Assignments
- 20% Midterm Exam
- 30% Final Exam

Important Dates:

- Dec 19: No class
- Jan 23: Mid Term exam
- Feb 25: Final exam

Introduction to OS

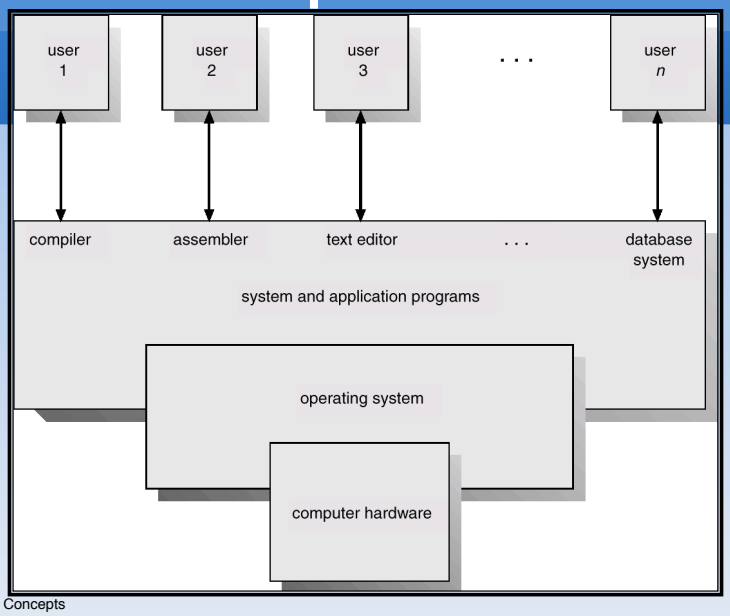
What is an Operating System?

- An operating system's primary purpose is to **facilitate the use of hardware and sofware resources** of a computer system in an efficient, fair, and secure way.
  - Fairness and security are important in multiuser systems.
- It allows users to use **application software**

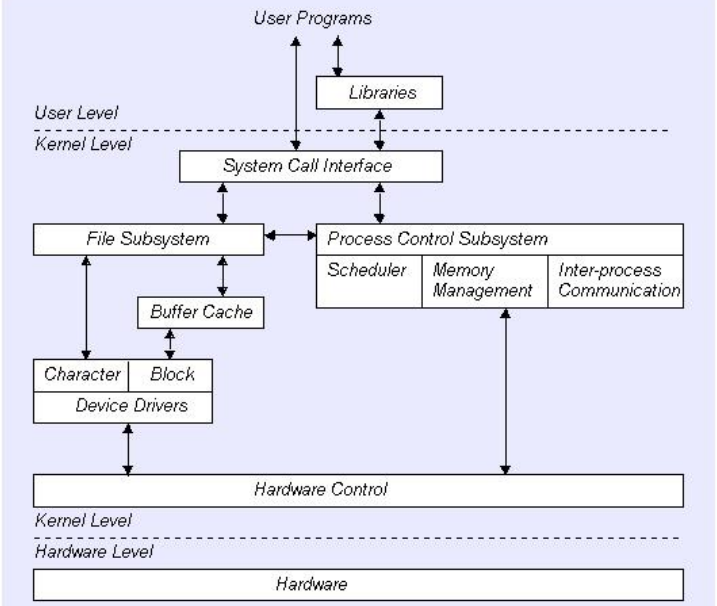
Computer System Components

1. Hardware – provides basic computing resources (CPU, memory, I/O devices).
2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.
3. Applications programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
4. Users (people, machines, other computers).

Abstract View of System Components



Unix/Linux Kernel interface



Resources

- Hardware Resources include:
  - RAM (main/primary memory)
  - Disk drive (secondary memory/storage)
  - CPU (processing)
  - Ethernet card (communication)
  - Screen (output)
  - Keyboard (input)
  - Printer

Resources

- Software Resources include:
  - Word processors
  - Spreadsheets
  - Web browsers
  - Mail readers
  - Chat Programs
  - Games
  - Utilities to access hardware resources
  - System Libraries

Operating System Definitions

Resource allocator – manages and allocates resources.

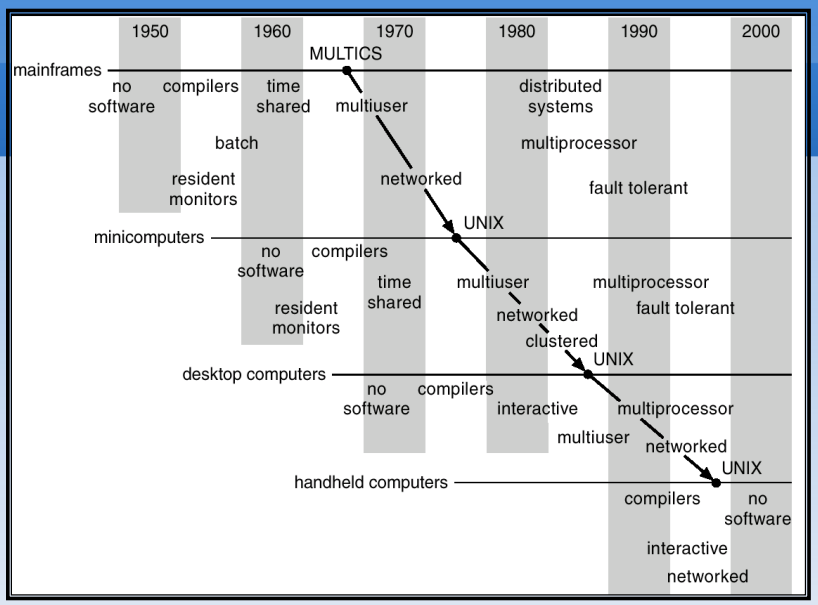
Control program – controls the execution of user programs and operations of I/O devices .

Kernel – the one program running at all times (all else being application programs).

Application User Interface

- The **kernel** is the heart of the operating system
- The Applications are connected to the kernel by language libraries and the system call interface.
- We shall be covering the use of some of these libraries and system calls in this class to see how an application talks to the kernel.
  - In CSC345, the focus is more on the kernel itself.
- We shall discuss it in the context of UNIX-variant systems...

Migration of Operating-System Concepts and Features



Modern
Linux
Window
Andriod
Ios
Chrome
etc

Basic Features of OS/Kernel

User Interface

Process Management

Memory Management

Storage/File Management

I/O subsystem

Protection

USER Interface

- Command Line Interface
  - Shell commands
  - C-shell, tsh-shell, bourne shell etc..
- Graphic User Interface
  - GNOME, KDE etc..

Process Management Activities

- The operating system is responsible for the following activities in connection with process management:
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and dirs
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (g**roup ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights

What is UNIX?

- Originally UNIX was a single operating system developed at AT&T's (Lucent) Bell Labs.
- Today it really refers to a class of operating systems which include:
  - Unix
  - Linux
  - FreeBSD/NetBSD/OpenBSD
  - MacOSX (Darwin)
  - Solaris

Some Flavors of OS

- Operating Systems come in many flavours:
  - Single-user, single-process (single task)
    - Old DOS for example
  - Single-user, multi-process (multi-tasking)
    - Windows 95/98
  - **Multi-user, multi-process (multi-tasking)**
    - **Most OSes these days including UNIX-variants**
    - **How does a single processor actually multi-task?**

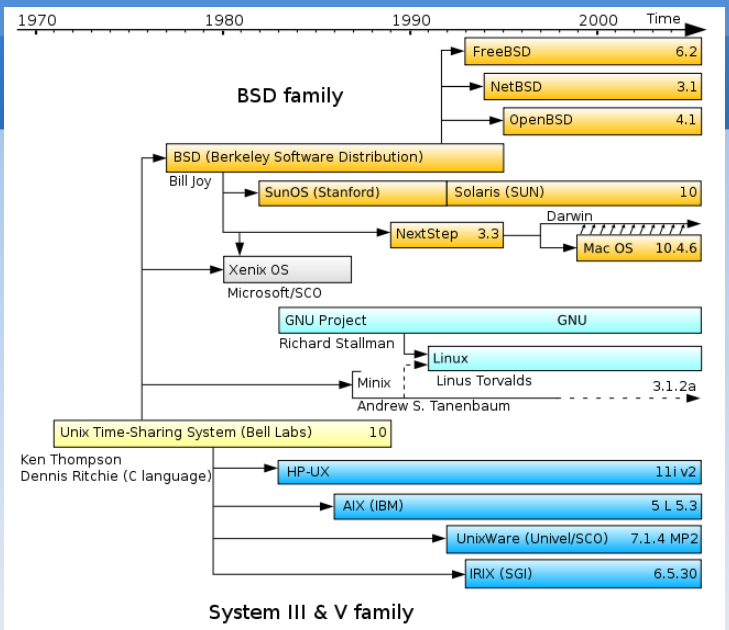
UNIX: The Myth

- UNIX is a heavy-weight operating system for big computers
- UNIX is complicated
- UNIX is hard to use
- UNIX has been created by large companies
- UNIX is monolithic

UNIX: The Legend

- UNIX was developed on small machines and became popular on killer micros. Dialects now run on everything from a PDA to supercomputers.
- UNIX is based on simple and elegant principles (but has added some bulk over the years).
- UNIX is not particularly hard to use (compared to the power it provides) but has a steep learning curve.
- UNIX has been created in a research environment with many individuals and companies contributing to its success.
- Many UNIX kernels are monolithic but typically a UNIX system is extremely modular.

UNIX: Genealogy



Source: http://en.wikipedia.org/wiki/Image:Unix_history.en.svg

UNIX: Genealogy

- In the 1960s, Bell Labs worked on a multiuser operating system called MULTICS
  - **MULT**iplexed **I**nformation and **C**omputing **S**ervice
  - It failed in large part because of the complexity of the software and hardware required to accomplish simple tasks for multiple users
  - Read Handout from *Just Enough Unix 3rd edition (page 9)*
- Researchers at Bell Labs then started work on **UNIX**
  - **Key feature:** Easily allowed a single user to create a process
- For a more detailed visualization check out:
  - http://www.levenez.com/unix/

C and Unix Link

- C was created for the purpose of building the UNIX operating system.
  - Derived from the language known as B.
- A high-level language that allows access to low-level system features.
- UNIX became one of the first operating systems with a kernel built in something other than assembly language.
  - Led to an increase in portability of the kernel.
  - Made it far easier to write, debug, maintain the kernel.

C and Unix Link

- Can handle characters, numbers, and memory addresses
- Small core lang., most functionality provided by libraries
- Typed-language but not strongly typed
  - Every variable has a type but the data can switch type at the whim of the programmer (dangerous!)
- Adequate (but limited) support for abstractions
  - Structures, unions, enumerations
  - Arrays and pointers
  - Ability to define new types
  - Procedural and imperative programming

Major Components of UNIX

- **Kernel**: Master control program
  - Technically, this IS the Operating System
- **Shell**: Interprets (user) commands and passes them on to the kernel. E.g.,
  - Bourne shell (sh), Bourne-again shell (bash), Korn Shell (ksh), and C shell (csh/tcsh).
- **File System**: Organization of the information (files and directories)
- **Utilities** (aka commands): Useful software

Additional Layers

- Windows: Typically, the X Window System
  - *A client-server model*
  - X acts as a server for other programs by providing a graphical user interface.
  - Useful for network environments (client/server on different machines)
- Window Manager: Determines what windows look like and how controlled by the user.
  - E.g. Twm, fvwm, mwm
- Common Desktop Environment:
  - To provide a similar look and feel to the different variants of UNIX.

Compliance

- With numerous variations it was important to ensure the code written for one system worked on other systems.
- Developed POSIX:
  - Portable Operating System Interface for Computer Environments
  - POSIX compliant code should work on all POSIX-compliant UNIX variants.

GNU/Linux

- Technically speaking Linux is the operating system, the Linux kernel.
- The software that makes it useful and that comes with all the popular distributions like Ubuntu, Fedora, SuSE, Mandrive, etc. is usually open-source GNU software.
- So, many argue it should be called GNU/Linux to emphasize the software that comes with it.