

# Ateneo de Manila University

## Introduction to Unix/Linux



Department of Information Systems and  
Computer Science

S.Y. 2001-2004

<http://cng.ateneo.net/cng/wyu/>

[wyu@ateneo.edu](mailto:wyu@ateneo.edu)

## Contents

- ★ History
- ★ Unix
- ★ Unix and Programming
- ★ Unix and C

## Section I

# History

## UNIX

- ★ is a time sharing operating system, but has evolved to be much more
- ★ ported to a large number of platforms
  - i386 (Intel Platforms)
  - SPARC
  - PA-RISC
  - RS6000/S390
  - PPC
  - etc. . .
- ★ simple and functional operating system
- ★ known for its durability, adaptability and reliability

## MULTICS

*"... tried to be everything to everyone"*

*"... ended up being nothing to anyone."*

- ★ concept of time-sharing system was NOVEL in the 1960s
- ★ developed started on 1962 by John McCarthy (of lisp fame)
- ★ aimed to become the premier multitasking operating system
- ★ development stalled due to politics and internal conflicts

## UNIX

- ★ intended as a pun MULTICS
- ★ emerged with much more modest goals as a programmers OS
- ★ focused on simplicity in both design and implementation
- ★ developed with a portable programming language called C
  - ★ scalability
  - ★ modularity
  - ★ portability

## Bell Labs

- ★ developed in 1969 by Ken Thompson at Bell labs on a PDP-7
- ★ later joined by Dennis Ritchie
- ★ first public release on 1971
- ★ in 1978, it was rereleased as 7th Ed. designed to run on multiple platforms

## UC Berkeley

- ★ source code of Unix made its way to the University of California at Berkeley when Ken Thompson taught there
- ★ Berkeley Standard Distribution (BSD) was born
- ★ researchers, teachers and students contributed to the development of BSD
- ★ the most notable of Berkeley contributions:
  - various operating system features
  - TCP/IP network stack
  - Internetworking tools

## Commercialization of Unix: The Unix Wars

- ★ in 1985, AT&T was split up due to anti-trust legislation
- ★ Unix was commercialized
- ★ Two major development streams:
  - AT&T System V Release 4 (SVR4)
  - Berkeley Standard Distribution (4.3 Reno)
- ★ Three major political streams:
  - Open Standards Foundation (OSF) - IBM, DEC and others
  - Unix International - SUN and AT&T
  - Advanced Computing Environment (ACE)

## Commercialization of Unix: The Unix Wars

- ★ A number of competing commercial unix flavors have emerged:
  - SUN Solaris/SunOS
  - SCO Unix (holds the Unix Copyright)
  - IBM AIX
  - SGI IRIX
  - HP-UX
  - and many more ...
- ★ Unix standards fragmented
- ★ Unix development stalled
- ★ Microsoft, Intel and IBM dominate the market

## Free BSDs

- ★ free flavors of Unix that descended from the original BSD:
  - FreeBSD - original branch of the free BSDs
  - NetBSD - catered to work in a large number of platforms
  - OpenBSD - aimed to becoming a very secure operating system
- ★ source code is publicly available under the BSD license
- ★ very strict development process

## Linux

- ★ one of the most popular Unix clones
- ★ IDC shows Linux as the fastest growing server platform in the market with about 20% of the market
- ★ supports a Unix-like interface and subscribes to Unix design philosophies
- ★ born at the University of Helsinki
- ★ developed by Linus Torvalds and a large number of programmers from around the world
- ★ source code is publicly available license under the GNU GPL

## What's Wrong with Unix

*...it is better to solve the right problem the wrong way than the wrong problem the right way - Dick Hamming of Bell Labs*

*...UNIX was not designed to stop its users from doing stupid things, as that would also stop them from doing clever things - Doug Gwyn of the US Army*

- ★ Unix has an open shop mentality
- ★ must provide the end user with ability to solve problems here and now
- ★ catered to the technical user (geeky and 3l1t3)

## What's Right with Unix

*...Unix is simple. It just takes a genius to understand its simplicity.* - Dennis Ritchie of Bell Labs

*...A good magician never reveals his secret; the unbelievable trick becomes simple and obvious once it is explained. So too with UNIX.* - Noahal A. Mundt author of Kermit

- ★ Open Source Software and Community
- ★ Cross Platform Portability and Standards Compliance
- ★ The Internet and the World Wide Web
- ★ Flexibility in Depth
- ★ Fun to Hack

## Section II

# UNIX

## **UNIX Philosophy summarized by Eric S. Raymond**

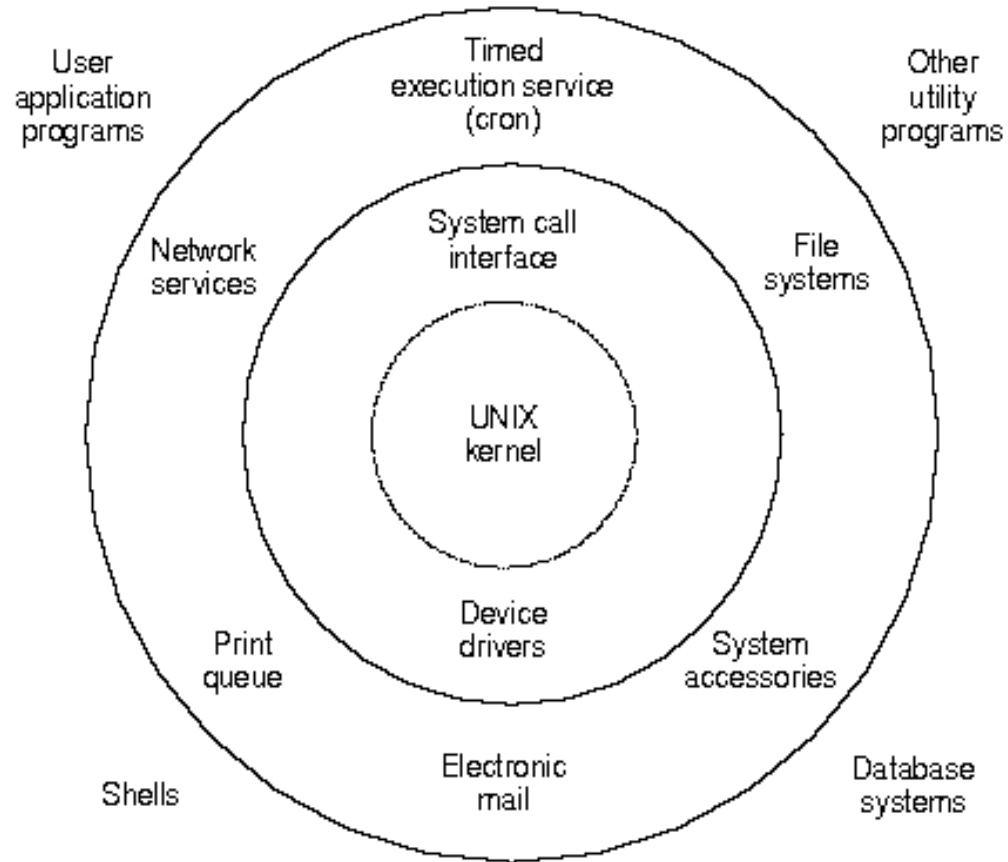
1. Rule of Modularity: Write simple parts connected by clean interfaces.
2. Rule of Composition: Design programs to be connected to other programs.
3. Rule of Clarity: Clarity is better than cleverness.
4. Rule of Simplicity: Design for simplicity; add complexity only where you must.
5. Rule of Transparency: Design for visibility to make inspection and debugging easier.
6. Rule of Robustness: Robustness is the child of transparency and simplicity.
7. Rule of Least Surprise: In interface design, always do the least surprising thing.
8. Rule of Repair: When you must fail, fail noisily and as soon as possible.
9. Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.

10. Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.
11. Rule of Representation: Fold knowledge into data so program logic can be stupid and robust.
12. Rule of Separation: Separate policy from mechanism; separate interfaces from engines.
13. Rule of Optimization: Prototype before polishing. Get it working before you optimize it.
14. Rule of Diversity: Distrust all claims for one true way.
15. Rule of Extensibility: Design for the future, because it will be here sooner than you think.

## **Unix Philosophy In a Nutshell**

**KISS: Keep It Straight and Simple!**

# UNIX Architecture



## UNIX Architecture

### ★ Applications

- are programs that end users intend to use
- serves as an interface between the user and the system
- shells, vi and etc ...

### ★ System Services

- user-level programs utilize functionality provided by these services
- tools and daemons that provide essential functionality to a Unix system
- sendmail, cron and etc ...

### ★ Kernel

- is the core of the operating system
- manages system resources (CPU, memory, disk)
- schedules applications and programs

## UNIX System Life Cycle

- ★ Power Up
- ★ RIPL, BIOS POST
- ★ Boot Loader
- ★ Kernel
- ★ Init
- ★ Runlevels
  - Runlevel 1 - System Halt
  - Runlevel 2 - Single User
  - Runlevel 3 - Multiuser Mode
  - Runlevel 5 - GUI Mode
  - Runlevel 6 - Reboot
- ★ Shutdown

## Unix/Linux System

### Important Parts of the kernel:

- ★ Process Management
- ★ Memory Management
- ★ File System
- ★ Input/Output
- ★ Networking

## Process Management

- ★ manages snapshots of each process that identifies:
  - what system resources a process is currently using
  - what system resources a process needs
  - Scheduling - what order these processes are to be executed
  - Interrupt Handling - how processes talk to the OS
  - Interprocess Communication - how processes talk to each other

## Memory Management

- ★ memory is very scarce resource
- ★ responsible for the allocation of memory resources in a computing system
- ★ manages the memory virtual subsystem
  - Main Memory
  - Secondary Memory

## File System

- ★ a way in which the system can organize files, directories and other resources in the system
- ★ contains the different system resources
- ★ in Unix, special devices are also available:
  - process map (/proc)
  - devices (/dev)
  - devices status (/proc)
- ★ Functions of a File system:
  - Directories and file nameing
  - File Storage and Shared Files
  - Reliability, Security, Protection

## Input/Output

- ★ is the main interface between the system and all other hardware and software components, peripherals and devices
- ★ concerned with the low-level interaction of I/O devices
- ★ devices drivers are used to interact with these components, peripherals and devices

## Networking

- ★ has recently become important for sharing scarce system resources such as:
  - I/O devices particularly storage and printing devices
  - applications and services
- ★ provides routines and methods to access different resources via the network
- ★ provides support for the different network protocols

## Section III

# Unix and Programming

## UNIX and Programming Languages

- ★ supports a wider variety of application languages than any other single operating system (in history)
- ★ two excellent reasons for this huge diversity:
  - use of Unix as a research and teaching platform
  - matching your application design with the proper implementation language
- ★ use the language best suited for the application
- ★ programming in Unix is more than just C:
  - Scripting Languages (Bash, Perl)
  - Embedded Programming Languages (C, Assembly)
  - General Purpose Programming Languages (C, C++, Java)
  - Application Specific Languages (SQL)

## UNIX and Programming

- ★ Portability – UNIX, or a variety of UNIX, is available on many machines. Programs written in standard UNIX and C should run on any of them with little difficulty.
- ★ Multiuser / Multitasking – many programs can share a machines processing power.
- ★ File handling – hierarchical file system with many file handling routines.
- ★ Shell Programming – UNIX provides a powerful command interpreter that understands over 200 commands and can also run UNIX and user-defined programs.
- ★ Pipe – where the output of one program can be made the input of another. This can done from command line or within a C program.
- ★ UNIX utilities – there over 200 utilities that let you accomplish many routines without writing new programs. e.g. make, grep, diff, awk, more ....

- ★ System calls – UNIX has about 60 system calls that are at the heart of the operating system or the kernel of UNIX. The calls are actually written in C. All of them can be accessed from C programs. Basic I/O, system clock access are examples. The function `open()` is an example of a system call.
- ★ Library functions – additions to the operating system.

## Section IV

# Unix and C

## UNIX and C

- ★ C is the native language of Unix
- ★ still the dominant systems programming language
- ★ fine grain resource management (power to define)
  - can be a good thing or a bad thing
  - Do It Yourself Memory Management
  - small and lightweight (adheres to Unix Design Philosophy)
  - 30%-40% of C bugs are memory related
- ★ because C was made for Unix!

## References

- W. Richard Stevens, “Advanced Programming in the Unix Environment”
- W. Richard Stevens, “Unix Network Programming”
- Eric S. Raymond, “The Art of Unix Programming,”  
<http://www.catb.org/esr/writings/taoup/>
- David A. Wheeler, “Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!,”  
[http://www.dwheeler.com/oss\\_fs\\_why.html](http://www.dwheeler.com/oss_fs_why.html)
- Richard M. Stallman, “GNU Project,”  
<http://www.gnu.org/gnu/the-gnu-project.html>



Copyright © 2001-2004 by William Emmanuel S. Yu. This material may be distributed only subject to the terms and conditions set forth in the Open Content License, v1.0 or later (the latest version is presently available at <http://opencontent.org/opl.shtml>).