Data Analysis in Geophysics
ESCI 7205

Class 1

Bob Smalley

Basics of the Computer Environment

Course Description

Introduction and Operating Systems

What is an operating system (OS or O/S)?

- Interface between Hardware and User.

See: http://en.wikipedia.org/wiki/Operating_system

What is an operating system (OS or O/S)?

- Interface between Hardware and User.
- It is a program (software) designed to manage and coordinate activities and resources of the computer.

See: http://en.wikipedia.org/wiki/Operating_system

What is an operating system (OS or O/S)?

- Controls the hardware (physical part of the computer - memory allocation, fan control, internal and external drive input/output, keyboard and mouse interactions, etc.) and other software.
- Controls how other applications (=programs) are implemented.

See: http://en.wikipedia.org/wiki/Operating_system

OS's at CERI

- Mac OS X (Darwin/UNIX)
  UNIX plus Mac GUI

- 10 Macs in Student Comptuer Lab in Long Building
  - many faculty offices.

OS's at CERI

- Various flavors of Linux

Popular, open source version of UNIX (often described as "UNIX-like", but is UNIX).

Found on a number of machines at CERI, but not officially supported at CERI.

OS's at CERI

- Solaris 9 UNIX

House 3 Sun Lab, many faculty offices

2 Graphical Desktop Environment options:

- Common Desktop Environment (traditional)
- GNOME 2.0 (more PC-like)

OS's at CERI

- Windows (XP, Vista or 7?)

Student Computer Lab in Long Building, many student offices, UM computer labs and other un-enlightened places.

Why learn Unix/Linux?

- Designed to be multi-user (from the dark ages when all computers were shared), interactive (as opposed to "batch"), and multi-tasking (sharing again).

- Invented by and for computer scientists/ system programmers (not users or scientific programmers, unfortunately).


TELETYPE
DATA DYNAMICS

Why learn Unix/Linux?

- Powerful, flexible, and small
- Hardware independent

(these two points are much more important to manufacturers and designers than general users, i.e. us)

Why learn Unix/Linux?

- "Free" (this is why is it still around) from Bell Labs and Berkley.
- Open source – "free" – applications, including compilers.
  - Most common free applications designed as part of the GNU Project (GNU's Not Unix)
- It is what is running in most geoscience (both university and corporate) labs.

The real reason why to learn Unix/Linux?

- Because you have no choice

("Resistance is futile", The Borg, Star Trek).

- It is what is running in most geophysics departments.

- Most geophysics tools (SAC, GMT, GAMIT/GLOBK, etc.) only run on Unix (although there is a Windows version of GMT).

(~89% of the worlds computers run some form of Windows, ~10% run some form of the Mac OS, and ~1% run some flavor of Unix.)

Why learn Unix/Linux?

- "Free" in the sense you don't buy it from AT&T or Berkley
- But there is no such thing as a "Free Lunch". Not "Free" in the sense that you must hire a system programmer/manager otherwise known as a UNIX Wizard or Guru.

(another UNIX myth shot down)

A bit of history

- Originally developed at AT&T in the late 60s/early 70s.
  - Freely given to universities in the 70s.
- Berkeley scientists continued to develop the OS as BSD Unix in parallel with AT&T (AT&T eventually licensed it for commercial use).
- Much development, branching, and combining has led to the most common variants of Unix ("flavors" or "distributions" in Unix speak).

- See http://www.bell-labs.com/history/unix/

Common flavors

- Solaris 9 Unix
  - Distributed by Sun Microsystems, runs on Sun Hardware, PC hardware.
  - Derived from Unix System V release (AT&T) on a Unix kernel.
- Mac OSX
  - Distributed by Apple, runs on Mac Hardware.
  - Derived from BSD Unix OS on a Mach kernel - Darwin.
- Linux
  - Free* and commercial# versions available built on a Linux kernel.
  - Flavors most likely to hear about are RedHat#, Ubuntu*, Fedora*, Debian*, Suse*,....

Does this matter?

- No, the differences between the various flavors of the Unix operating system should not severely affect your work in this class or even much of your research at CERI.

BUT

Does this matter?

- Yes, you need to be aware of OS differences
  - When file sharing with others (this is more of a hardware, rather than an OS, issue).
  - When compiling source code (the executable file is married to hardware).
    - If sharing programs, shell scripts, etc. with others.
    - Or if moving between the different systems at CERI.

Relation to Windows

None.

Windows XP

Built on MS-DOS (which is not really an operating system, it is a file system), which has nothing to do with Unix and everything to do with Microsoft.

Cygwin – unix/linux like environment for windows. Have to build everything from source.

Relation to Windows

The differences between the Unix Philosophy and the Windows Philosophy … can be boiled down into a question of smarts … .

Unix and Windows store the smarts in different places.

Unix stores the smarts in the user.
Windows stores the smarts in the OS.

Learning curves

Enter the concept of the “Learning Curve”. …

A "steep" learning curve generally refers to something that requires a lot of initial learning to do anything, even something very simple.

A "shallow" learning curve is exactly the opposite; can do simple stuff easily immediately.

Learning curves

Armed with those definitions, it's fairly simple to then go ahead and say that Unix has an inherently steep learning curve, and Windows has a very shallow one.

Windows

Our Microsoft brethren have taken the approach of making the shallowest possible learning curve.

Windows

To take a cue from the fast food industry, Windows is the "under-3" toy of the OS world.

The ultimate goal is to flat-out destroy any barrier to entry by removing any requirement for initial knowledge or learning of how and why, and of making the system simplistic enough that it can be used without any understanding of how it works.

Unix

The Unix crowd has taken the opposite approach.

Unix

Unix has a steep learning curve; it doesn't shield the user from complexity; rather, it revels in the complexity.

It recognizes that a general-purpose computer is a fiendishly complicated device capable of doing an unbelievable assortment of things.

Unix

It recognizes that the computer is a tool of the user, and so takes a tool-building philosophy.

Make a lot of tools, and make each tool specific, and let the user select the tool they think appropriate, and let the user combine the tools however they want.

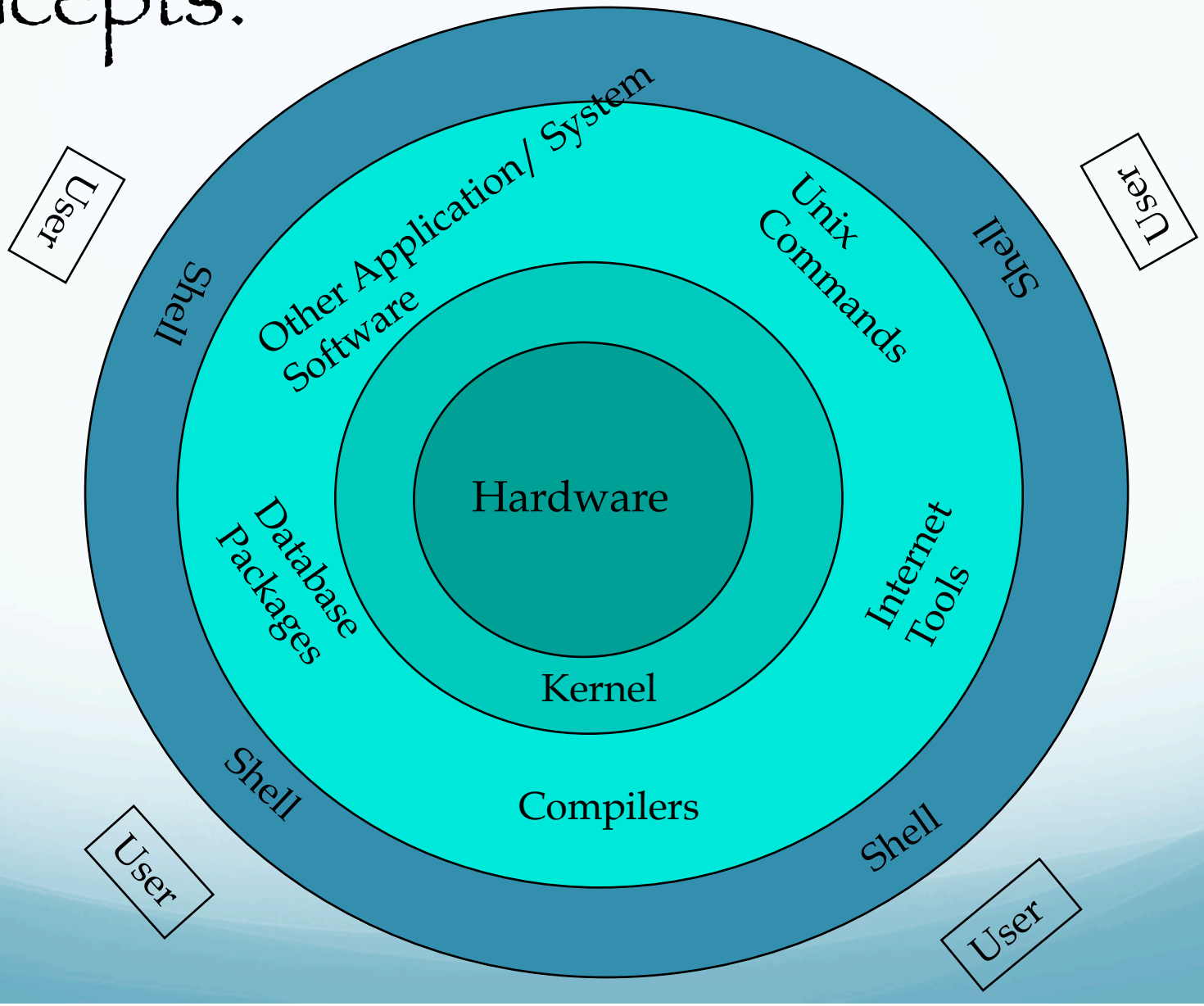
It's not aimed at making things easy; it's aimed at making things possible.

UNIX Philosophy

(Mac) (Unix)

"Dilbert" by, Scott Adams, Sep 30, 1994.

Backing up a bit to illustrate some concepts.



- Hardware – the physical computer.

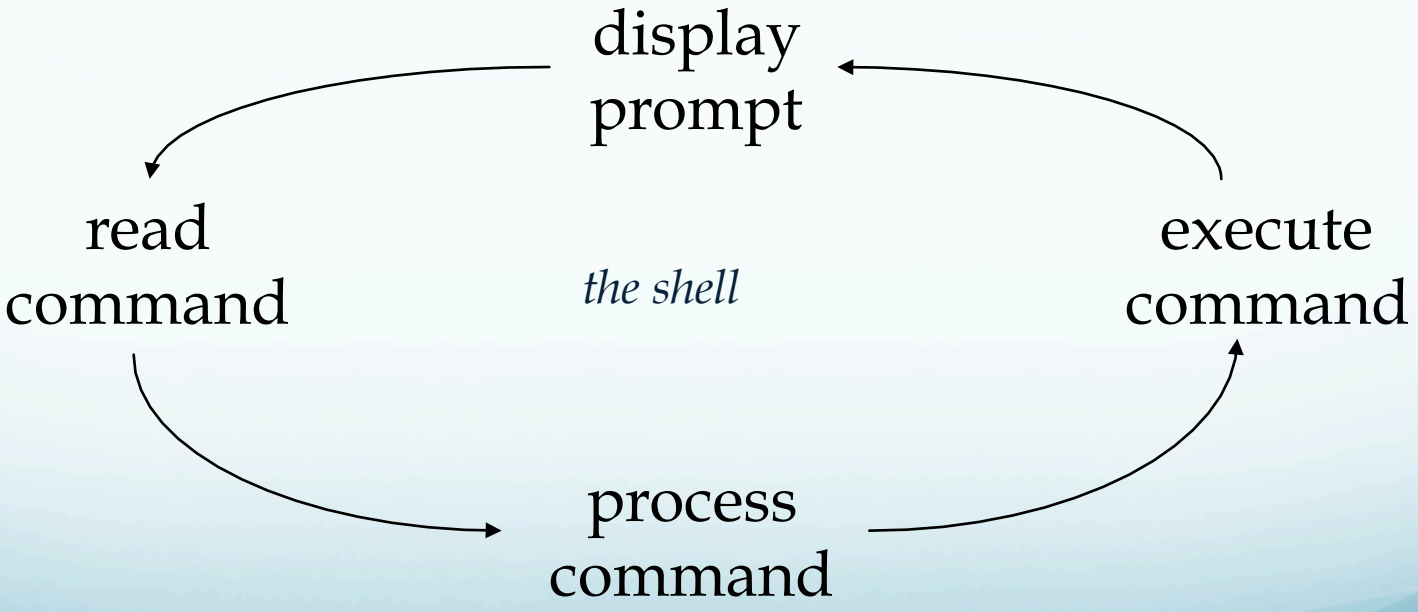
- Kernel – program, usually hardware dependent, that runs the core or key components of the operating system (process, memory, file, device, and network management).

- Programs/Applications – hardware independent – unix commands, compilers, applications

- Shell – hardware independent - how the user interacts with the Programs/Applications layer.

The Shell

- The UNIX user interface is called the *shell*.
  - The shell does 4 jobs repeatedly:



Final Model


Understanding Computer Technology
Main storage
Floppy disk
Application software
Function key
Users interface
Surge control device
input
Central processing unit
Backup system
De-bugging tool
Overflow (input / output error)
Peripherals (hardware)
Supplementary data

We will now take a short detour to examine the Unix philosophy.

It will keep returning to haunt us, but if you understand it, it will make the process less painful.

What is the "Unix Philosophy"?

(can computer operating systems have a "philosophy"?)

According to Doug McIlroy

(i) Make each program do one thing well.

So, to do a new job, build afresh rather than complicate old programs by adding new features (otherwise known as "bells and whistles").

What is "Unix Philosophy"?

Machine shop vs. appliance

(gives you the tools and you to make appliance)

What is "Unix Philosophy"?

Advantage

- POWERFUL

What is "Unix Philosophy"?

Disadvantages

- Lots of reinventing the wheel
- Requires a more educated user
- Requires more work from the user rather than the developer

What is "Unix Philosophy"?

Typical question: can UNIX do this?

Typical answer: NO, but YOU can write a program!

Unix enthusiasts think this is the answer the average user wants to hear!

Caricature of UNIX vs Windows

If you need a washing machine

Windows gives you a simple washing machine (only one 1 setting, you shouldn't wash your cashmere sweater, but there are no operating instructions [its intiutive] so you probably don't know that and ruined it.)

UNIX gives you a machine shop (you better know 1) how wash clothes and 2) how to design and build a machine to do it.)

"UNIX Philosophy"

(ii) Expect the output of every program to become the input to another, as yet unknown, program.

- Don't clutter the output with extraneous information useful to the user, but not needed by the input for next program.

"UNIX Philosophy"

Unfortunately this may make things confusing for the uninitiated user.

The output is for "next program" (in a "pipe"), not the user.

"UNIX Philosophy"

Idea of "filter" –

Every program takes its input from Standard IN (originally a teletype, now a keyboard),

does something to it ("filters" it) and

sends it to Standard OUT (originally a teletype, now a screen)

(notice that the "user" is not part of this model).

"UNIX Philosophy"

Idea/use of – redirection ("<", "<<" and ">", ">>")

- Take input from a file rather than Standard IN

- Send output to a file rather than Standard OUT

(Unix treats everything like a "file", even hardware)