A practical hands-on approach. We will build new servers and configure them with a variety of packages.

Expertise is distributed: system administration happens everywhere from your PC to large servers, and system administration is generally collaborative. Other people will help you solve problems and learn more.
The systems we will emphasize are Linux. We will also talk some about BSD, Solaris, and Windows.
CNT 5605, Fall 2090: Introduction

You may see occasionally see these abbreviations in course materials:

- **LAH** → Linux Administration Handbook, 2nd Ed (Nemeth et al)
- **W2K8** → Windows Server 2008: Networking Foundations (Minasi et al)
ASSIGNMENT

❖ Read chapters 1, 2, and 3 in LAH
❖ Read chapters 1, 2, and 3 in W2K8
Introduction to Unix

Unix is now more than 30 years old

It began in 1969

See The Evolution of the Unix Time-sharing System by Dennis Ritchie

http://cm.bell-labs.com/cm/cs/who/dmr/hist.html
Introduction to Unix

Started at AT&T’s Bell Labs, derived from MULTICS.

Initial hardware was a DEC PDP-7, and the filesystem was hierarchical but did not have pathnames

(i.e., there was no equivalent to a pathname such as /etc/hosts, it would just be hosts; directory information was kept in a special file called dd)
Introduction to Unix

Rather than a product from a manufacturer, Unix began as collaboration with these goals:

- Simplicity
- Multi-user support
- Portability
- Universities could get source code easily
- Users shared ideas, programs, bug fixes
Introduction to Unix

The development of early Unix was user-driven rather than corporate-driven.

Note that Linux and the BSDs (FreeBSD, OpenBSD, NetBSD) now flourish in similar “open source” environments (http://www.freebsd.org, http://www.openbsd.org, http://www.netbsd.org)
Introduction to Unix

The first meeting of the Unix User Group was in May, 1974; this group would later become the Usenix Association.

You can watch the ever-varying open source world of Linux/BSD distributions at http://www.distrowatch.com
A History of Unix

In the beginning, processes were very different

Originally

- Each terminal only could have one active process
- When creating a “child” process, the parent first closed all of its open files
- Then the parent linked to the executable and opened it
A History of Unix

Then the parent copied a bootstrap to the top of memory and jumped into the bootstrap.

The bootstrap copied the code for the new process over the parent’s code and then jumped into it.

When the child did an exit, it first copied in the parent process code into its code area, and then jumped back into the parent code at the beginning.
Old Unix

Today the parent uses a fork/exec/wait model:

- `fork(2)` (to create a new child process)
- `exec*(2)` (to have the child process start executing a new program)
- `wait*(2)` (to wait on the child (or at least check on its status if non-blocking))
Linux: a complete Unix-compatible operating system

- Runs on huge array of hardware, from IBM’s biggest machines down to commodity routers such as the Linksys WRT routers using DD-WRT

- Based on Linux Torvalds’ kernel (he is still in charge of kernel development (2.6.28 is the current stable release), though now many people work on the kernel)
Linux: a complete Unix-compatible operating system

The Linux distributions we will use are CentOS and Fedora; each includes a full development environment, X-Windows, NFS, office environment products (word processors, spreadsheets, etc), C, C++, Fortran, several mail systems (exim, postfix, and sendmail) and whole lot more (a full install is around 5 to 7 gigabytes)

Linux is mostly POSIX.1 compliant
Linux: a complete Unix-compatible operating system

FAQ: http://www.opengroup.org/austin/papers/posix_faq.html
Solaris

Solaris is a vendor Unix. It came from Sun Microsystems (which is now part of Oracle.)

It runs on Sun’s custom hardware and commodity PC hardware – and in virtual environments, which is how we will run it.
Microsoft Windows

Windows 2008 Server is a proprietary system that follows in the NT and Windows Server 2003 family.

Windows 2008, like Windows 2003, has several varieties:

- “Standard Edition” (Supports “Server Core”)
- “Enterprise Edition” (Supports “Server Core”)
- “Datacenter Edition” (Supports “Server Core”)
- “Windows Web Server”
- “Windows Storage Server”
Microsoft Windows

The Minasi book is an enjoyable, and no longer voluminous, read

Windows has the advantage that it provides a single vendor solution, completely controlled by Microsoft. Generally you use some sort of interface to configure Windows rather than a text editor.
System administration responsibilities (AIC)

- Availability of services and data
- Integrity of services and data
- Confidentiality
System administration duties

- Installing new hardware and software
- Updating and upgrading hardware and software
- Monitoring for problems
- Resolving problems
System administration duties

- Planning for growth and obsolence
- Making backups
- Recovering data from backups
Availability

- Lack of availability can have a large dollar impact on many businesses
- The most common danger to availability is SPOFs (single points of failure)
Availability

SPOFs are generally cured by redundancy. Redundancy models:

- Cold → Good points: “guarantees” availability, but (1) typically most expensive (2) tends to require frequent testing (3) typically slow to bring live
Availability

Warm (dual-use hardware that is powered up and used for related but non-production purposes such as development or q/a) → Typically good availability and less expensive than cold since the hardware has day-to-day use, but (1) tends to be one-offish and (2) tends to require frequent testing of switchover
Availability

Hot (redundant hardware in production) typically least expensive since it is in active production use and testing requirements tend to be less but more likely to have capacity problems in the event of partial failure.
For system administration purposes, often data integrity is implicitly enforced both at the hardware and the operating system level, such as CRCs for disk reads, parity in RAID-5 controllers, and so forth.
Integrity of data

However, it can also be an explicit issue, such as logical names that must be correct.
Integrity of data

Example: you create a soft link from the logical name for a package to its new implementation, say DBB-Current → DBB-1.8; however, inside of an update script, you somehow leave a single instance of a hard-coded DBB-1.7 (the old version) rather the logical name DBB-Current. When the script runs, the update now mixes the two packages DBB-1.8 and DBB-1.7.
Data security

This is a **big** topic, even from a pure system administration point of view.

Passive security: design your systems with security in mind.
Data security

Active security: (1) Searching for problems with tools such as Snort and Nessus (2) Forensically researching problems with tools such as with TCT.
Some of the elements of Unix/Linux system administration: editors, scripting, and compilation

Editors: A lot of Unix/Linux system administration tasks revolve around editing various files. The best two choices for editing are the programs vi and emacs.
Some of the elements of Unix/Linux system administration: editors, scripting, and compilation

Scripting: In addition to simple configuration files, another important subset of tasks that require editors are scripts. These scripts are used for various purposes, and often automating routine tasks such as running programs such as `makewhatis` and `updatedb`. 
Some of the elements of Unix/Linux system administration: editors, scripting, and compilation

Compilation: Generally on Unix/Linux machines, gcc is a good choice for compilation. While it is not perfect, and the various generations of gcc can be somewhat frustrating to deal with, it is hard to beat the price.
Unix/Linux system administration: editors

**vi**: It is almost always found somewhere on a Unix/Linux machine, often in the form of `vim`.

**vi** advantage: It is simple to learn. It has two modes, “motion” and “insert”. In the “motion” mode, the most important keys are just

- **h**: Move the cursor left.
- **l**: Move the cursor right.
- **j**: Move the cursor down.
Unix/Linux system administration: editors

- **k**: Move the cursor up.
- **i**: Go into “insert” mode.
- **“:”**: Execute an internal vi command of some sort. The most popular “w” for “write”, “q” for “quit”, and “
Unix/Linux system administration: editors

- **vi** disadvantage: While original **vi** is quite simple, it also is not very featureful. This isn’t as much of a problem with **vim**, an updated version of **vi**. These extensions are not as logical as those **emacs** since they were created as an afterthought.
Unix/Linux system administration: editors

emacs: Not always installed by default. In fact, in the last few years, default installation of Emacs in Unix/Linux distributions has become less common.
Unix/Linux system administration: editors

emacs advantages: As computer scientists, emacs is quite intuitive: each sequence of keystrokes can be mapped to a Emacs Lisp function. For instance, by default in TeX mode, the key “a” is just mapped to a function which inserts an “a”, but the double quote key is mapped to insert first a pair of “ and then the second time to insert a pair of closing ” in accord with TeX’s expectations.
Unix/Linux system administration: editors

- **emacs** disadvantage: It’s not found on every machine.
- **emacs** disadvantage: It’s big.
Unix/Linux system administration: scripting languages

bash: The “Bourne-Again SHell”. Introduced the “readline” library. A comprehensive overhaul of the original Bourne shell.

perl5: The “Practical Extraction and Report Language”. An amazing language in its own right; originally, it could be mistaken for line noise, much as the old TECO editor commands.
Unix/Linux system administration: scripting languages

Perl 6: Currently, version 5 is very useful, though version 6 still hasn’t jelled. My lecture notes for this class have used that “not jelled” phrase since 2003, it appears!
Unix/Linux system administration: scripting languages

Scripts in the Unix/Linux world are most commonly written in shell, Perl, or occasionally in Python. (We still occasionally find some AWK scripts, but it’s been fading fast over the last few years.)