System Administration for Beginners

Week 4 Notes

March 12, 2008

1 Lecture

In the previous weeks, we have covered much of the basic groundwork needed in a UNIX environment. In the upcoming weeks, we will spend time getting intimately close with making things work. Simply put, we are shifting over to the administrative side of things as opposed to only using the tools that have been provided to us.

1.1 Super-Users

Until now, you have been working with regular user accounts on the instructional and OCF servers. Beginning this week, each project group will be granted an administrator account on their own private virtual server. These administrator accounts are called *super-user* accounts as they have more privileges compared to regular user accounts.

On most UNIX system, the most powerful super-user account is called `root`. A user with root access can do practically anything on a system; thus it is important to take great care when performing tasks as a root user. As said earlier, UNIX will not yell at you or provide any warnings if you tell it to do something stupid. For this reason, system administrators tend to create normal user accounts for themselves, and switch to root only when necessary.

The following is a simple list of tasks that root can do. This list is only a sampling of the changes that root can make and is by no means limited:

- change the password for any user
- create and delete user accounts
- read, write, or execute any file on the system
- change the permissions for any file or directory on the system
- install software globally
- reboot or shutdown the system

1.2 UNIX Filesystem Layout

With root access, you will be able to manipulate files outside your home directory. Therefore, it is important to have a general idea of where files are stored on UNIX-based operating systems.

Like most other operating systems, UNIX uses a hierarchical directory structure. Each mounted filesystem has a *root*, or top level, directory which contains files and sub directories. On Windows, for example, each filesystem mounted on the system is assigned its own drive letter (like `C:`, `D:`) and the filesystem is mounted under it.

UNIX handles filesystem mounts in a different fashion. Everything, if you haven't noticed, is accessed under a single filesystem, with individual filesystems mounted at *mount points*. These mount points are directories or files that are in the main file system where you attach the root directory of another filesystem. Let us now take a look at its directory structure, based on the convention known as the *Filesystem Hierarchy Standard*.

/bin Binaries, or programs, which are accessible by all users are located here. Most of the commands entered without specifying a path are located in a `bin` directory.

/sbin System binaries are programs used for maintenance or administrative tasks. They are generally accessible by super-users because many of these binaries have the capabilities to make changes on a system-wide level

/home Each user's home directory is stored here. Each user gets their own personal directory to store individual data and user-specific settings.

/usr Known as User Shareable, this is a secondary hierarchy similar to the root directory. It contains many similar subdirectories like the root directory, but most of its contents are not as essential to system operations.

/dev This directory contains the links to all the devices connected to your system. The UNIX-filesystem treats devices as files and are located here.

/var Variable data is stored here. Important things like print jobs and system logs can be found in specific subdirectories.

/tmp This temporary spot is useable by any user. Since it is temporary, the data stored here can be cleaned out and lost at any time, so it is important that nothing you want to save is kept in this directory.

/boot The data stored in this directory is used for booting the system. Kernel images and boot loader configuration (e.g., GRUB or LILO) is found here.

/proc A virtual filesystem is mounted here, containing information about system hardware and running processes.

/etc This is where all the system configuration files are stored.

/root Not standard, but generally the home directory for root.

1.3 Server Daemons

Last week, we described a model of the Internet that was composed of three layers. The topmost one, the application layer, refers to various *languages* spoken on the Internet (e.g., HTTP, FTP, BitTorrent, etc). These languages are often used by applications called server daemons.

Server daemons are programs that run on a server and wait for something to happen (in UNIX this is called a background process). Generally, these "somethings" are requests for or to transmit information. For example, a web server is a daemon that waits for someone to visit the website it is hosting (a request for information). Other server daemons, such as email servers, usually wait for a request to transmit information.

To communicate over the Internet, server daemons *listen* for requests on *ports*. A port is like a special door on a server that is accessible to the Internet, and each server daemon gets their own door. Ports are numbered 1 through 65535 and there is an agency that establishes a standard set of ports server daemons should use. However, system administrators are under no obligation to use these standard ports.

1.4 How to Get Server Daemons

In the UNIX world, the most popualr server daemons are the product of open-source projects that provide their software for free. Apache, MySQL, and Postfix are examples of very popular web, database, and email servers, respectively, that are completely free to obtain, use, and customize.

To obtain a server daemon, you go to the website for that server project and download the software. Most popular server daemons are available for a variety of computer platforms (e.g., UNIX, Windows, Mac OS X).

Server daemons for Microsoft Windows platforms are generally distributed as compiled executables, while server daemons for UNIX-based platforms are generally distributed as source code packages. Source code is the program code that defines a program; you have to compile the source code to use the program. The reason why server daemons are distributed as source code packages for UNIX-based platforms is because there are so many different types of UNIX that it would be impractical to provid pre-compiled binaries for all the available UNIX versions. Of course, you will probably be able to find pre-compiled binraies for popular versions of UNIX, but don't be surprised if you don't.

The source code is generally distributed as a compressed tar archive (commonly referred to as tarballs), so you will need to uncompress the archive and extract it.

1.5 ./configure, make, make install

In most cases, extracting the tar archive for a server daemon will create a directory with the name of the server daemon and the version you downloaded. Chagning into this directory, you will normally find files named either `README` or

`INSTALL` that describe how to compile and install the software. It is a good idea to read these files before compiling and installing the package to make sure that you've configured any system-specific settings that your server may require.

Commonly, UNIX software is compiled and installed using three commands. The first command, `./configure`, customzies the source code for your UNIX version and allows you to specify and special configuration parameters that you may need. The second, `make`, actually compiles the program code. The last command, `make install`, installs the software into the UNIX file system.

`./configure` also takes parameters that allow you to customize the software. For example, you might want to disable certain features for performance or security reasons. Another reason to pass a parameter to `./configure` is to change the default installation directory using the `--prefix` parameter.

Some server daemons require additional installation steps, so be sure to read the `README` or `INSTALL` files if available (if they're not, check the place where you got the software for documentation).

1.6 Configuration

After compiling and installing a server daemon, you need to configure it before you can use it. Most server daemons read their configuration from a plaintext configuration file with various directives. For example, Apache 1.3 reads a file named `httpd.conf` that specifies the websites it should host. Under Apache 2, the configuration can be found in a file named `apache2.conf`.

With respect to configuration files, there are two camps of server daemons: those who include ready-to-use configuration files and those who believe that administrators should first become familiar with a piece of software by writing a configuration file from scratch. Regardless of which you choose, it is important to read through a server daemon's documentation prior to deploying it even if you have a ready-to-use configuration file. Failure to read through a server daemon's documentation is often the reason for security breaches.

2 Final Project Overview

For the final project, you will be setting up a production-quality web server running multiple server daemons (web, database, FTP) and with support for common web programming languages. You will also secure the server, set up some tools to automate system administration, and set up some form of backup.

As a test of your work, you will create some sample user accounts and load them with popular web software. At the moment, the final project specifications have not been finished, but this semester I am thinking of having a small list of feature sets that can be implemented on a server and it is up to you to implement them.

You will be using the same private virtual server for the final project. The week before the final project is assigned, we will reset your virtual servers. Make

note of anything you learn sinace all your changes will be wiped out for the final project.