Advanced Unix System Administration
Fall 2008
Project 1

This project is due by 11:59 PM on **Monday, November 3**, though depending on how things go, an extension may be offered. Writeups should be sent via email to <sluo+decal@ocf.berkeley.edu>; project setups need to be complete by this time.

Half of your project grade is the quality of your setup; half of it is the quality of your writeup. Document all the steps you take in setting things up, even if only briefly. Where you make a choice on how something should be set up, explain the rationale for that decision.

Because of the way the virtual containers for this project are set up, I don't have a way of logging in to them unless you give me access. Therefore, please leave the root password to your container in a file in your login server home directory (don't forget to set appropriate permissions!) and note the location of this file in your writeup.

You may work in groups of up to three, though groups of two are preferred. I expect one writeup per group. Please let me know by Friday, October 23 whom you will be working with; if you don't let me know by that time, I will assume you will be working alone and will create VMs accordingly. Feel free to use the class mailing list (<sluo+decalfa08@ocf.berkeley.edu>) to find a partner, if you'd like.

Scenario

You are the lead system administrator for Foobar Software Solutions, Inc., a company distributing and providing support for Linux distributions which are enhanced versions of the products of other leading Linux vendors. Your company is small, but the quality of its support and its cut-rate prices mean that it is growing in popularity rapidly. With growth, of course, comes growing pains; your computing infrastructure is straining to handle the load placed on it, and your low-cost business model means no money for getting new hardware anytime soon. (Rumors are floating around that management is negotiating a big contract with a Tier 1 server vendor, but even if that pans out, it'll be a few quarters before the money really starts rolling in.)

Tasks

1. Back in the days when the company was based in a garage, development was mostly done on personal laptops, and ad-hoc arrangements were made for source code management. This hasn't scaled well as the company has grown, so you've been asked to set up a central development box with source code version control infrastructure.

    a. Begin by installing Linux on the designated machine. (You have a virtual machine for this purpose; see the notes below for some instructions.) Configure the system as you see fit, with an eye towards stability, performance, and long-term maintainability of the setup. Make sure you install some development tools; having `make` and compilers would be a good start.

    b. Set up some user accounts on the system. They fall roughly into two different categories: developers, who need full access to the system's resources; and other staff, who need to be able to see what the developers are doing, but should have their permissions restricted. At a minimum, configure appropriate process limits, memory limits, and disk quotas for non-developer accounts; for more credit, configure other appropriate resource limits.

2. As part of the campaign to restructure your developers' workflow, you've been asked to evaluate some version control systems on the basis of performance. (Don't worry about feature set; the developers are still figuring out their new workflow, and hopefully management will choose a system that accommodates their needs . . .)

    a. Install at least two of Subversion, Git, and Mercurial on the system. Using a copy of the Linux kernel source, benchmark some common operations with each system: initially importing a source code tree, checking out a working copy, and checking in changes, at least. (The more ambitious may try comparing merges and other more complex operations.) Identify the bottleneck(s) in each operation, and suggest appropriate performance improvements. (For more credit, implement some of these suggestions.)

    b. For each system, document for your developers how to set up access control on a source tree, so that only certain developers have commit privileges, but everyone may check out a copy.

    c. As part of your corporate commitment to open source, management has decided that your version control system's contents should be available to the world. For each version control system you set up, configure remote anonymous access via the standard client, and set up a web interface. Benchmark

the performance of anonymous checkouts and browsing using the web interface, identify the bottleneck(s), and suggest appropriate performance improvements.

d. Based on your findings, suggest hardware upgrades which would give the company the best value for its money.

3. One of your customers has filed the following bug report against your Debian etch derived product, DebianFoo Linux:

   "When I set `LogLevel DEBUG` in my OpenSSH configuration on DebianFoo, I don't see any details of what's happening during an attempt to log in, just whether the attempt succeeded or failed. On some of my other systems, such as my OpenBSD-based router or my Debian Lenny system, `LogLevel DEBUG` shows me details of the connection handshake and authentication process for each connection, not just whether login succeeded. In both cases, my OpenSSH and syslog configuration is the default shipped with the system, except that of course I've changed the `LogLevel` setting to `DEBUG`.

   "This is an extremely useful feature for figuring out what's going wrong with logins, so it would be nice if it worked on DebianFoo."

   As you're the resident sysadmin guru and troubleshooting expert, and your front-line support folks are stumped, you've been tasked to resolve this bug report. Find out why the debug logging the customer wants isn't working, and how to get it to work.

   Note: You're installing Debian Lenny on your project systems. Your homework VMs run Debian Etch.

System Setup Details

For this project, you have the use of a Xen "domU", which in most respects behaves like an actual physical system. Each domU is configured with 256 MB of RAM and one 10 GB hard disk (I realize this wouldn't exactly be high-end hardware if it were real, but we don't have quite enough resources on the host system to increase these figures). You may use any network addresses in the range `10.20.10.x0`-`10.20.10.x9`, where `x` your group number. The netmask to use is `255.255.255.0`, the default gateway is `10.20.10.1`, and your DNS servers are `192.58.221.242`, `128.32.136.9`, and `128.32.206.12`.

To access the domU console, SSH in to `coupdetat.ocf.berkeley.edu` port 2022 with your group's username and password. You will be presented with a menu from which you

can boot, shut down, or reboot the system, attach to its console, or boot the Debian installer.

A few hints on installing the operating system:

- `linux.csua.berkeley.edu` is on campus, and therefore by far the fastest Debian mirror for this installation.
- *DO NOT* forget the root password for your system! It takes a great deal of work to reset it. Remember that you need to give me the root password for the system, so don't set a password that you use for any other purpose.
- The VM setup assumes that your GRUB configuration is in `/boot/grub/menu.lst` on the first partition of your hard disk. If you're using a separate `/boot` partition, this means that it needs to be the first partition, and you may need to create a `/boot` symlink pointing to the partition root to get the GRUB menu to show up.
- Take the option to install GRUB to the MBR (which is the default).
- For performance reasons, it's recommended that you install the `libc6-xen` package and reboot after it's installed.