Ubuntu Practice and Configuration

Post Installation Exercises

intERLab at AIT

Bangkok, Thailand

1. Get used to using `sudo`
2. Create an *inst* account
3. Learn how to install software
4. Update /etc/apt/sources.list
5. Install `gcc` and `make`
6. Learn how to control services
7. Use the `ip` tool
8. Create the `locate` database
9. So, you wanna be root...
10. Install Apache Web Server and PHP
11. Install OpenSSH, public key and new sshd_config
12. Install Gnome 2.x
13. Configure XWindow

1.) Get used to using `sudo`

Ubuntu and Debian approach system administration a bit differently than other Linux distributions. Instead of logging in as the "*root*" user to do system tasks, or becoming *root* by using the `su` command you are encouraged to do your system administration using `sudo`. By default your user has privileges to do this. Let's practice this by running some privileged commands from your user account.

First, log in if you have not done so. Once you are logged in you'll see something like this:

```
user@pcn:~$
```

We'll represent this prompt with the abbreviation "$".

Now try to look at the system password file with actual encrypted passwords:

```
$ less /etc/shadow
```

The first time you attempt this it will fail. Instead do the following:

```
$ sudo less /etc/shadow
```

You will be prompted for a password. This is your user's password. Type it in and you should see the contents of the protected file /etc/shadow (press "q" to exit the output on the screen).

If you wish to issue a command that requires system privileges, use the `sudo` command. For instance,

if you are interested in seeing what groups your account belongs to you can type:

```
$ sudo vigr
```

You are now in the vi editor (you have a handout to help you with this editor). Type:

/yourUserid

Then press the “n” key for “next” to see each group you belong to. Notice that you are in the “adm” group. To exit vi type:

:q!

Get used to using “sudo” to do your system administration work. Exercise number 9, will give you a couple of other options for using system privileged commands as well.

2.) Create an *inst* account

If you are used to many Linux distributions, then you think of the `adduser` and the `useradd` commands as being equivalent. One is simply a link to the other. In Debian/Ubuntu this is not true. They are distinct commands with different capabilities. If you are interested in the differences type:

```
$ man adduser
$ man useradd
```

As you can see the `adduser` command is considerably more powerful. This is what we will use to add a new user and to manipulate user accounts later on.

At this point we would like you to create an account named *inst* with a password given in class. This allows your instructors, your fellow students or yourself a way to access your system if necessary. If you created the *inst* account while installing Ubuntu, then you can either skip this exercise, or create your own account name.

To do this type:

```
$ sudo adduser --shell /bin/bash inst
```

You may be be prompted for your user password to use the `sudo` command.

You will be prompted for a password. Use what the instructor gives in class. Please be sure to use this password. Your session will look like this:

```
user@pcn:~# adduser --shell /bin/bash inst
Adding user `inst' ...
Adding new group `inst' (1001) ...
Adding new user `inst' (1001) with group `inst' ...
Creating home directory `/home/inst' ...
Copying files from `/etc/skel' ...
```

cont: ==>

```
Enter new UNIX password:              <ENTER pw given in class>
Retype new UNIX password:             <ENTER pw given in class>
passwd: password updated successfully
Changing the user information for inst
Enter the new value, or press ENTER for the default
        Full Name []:                       <Press ENTER for default>
        Room Number []:                     <Press ENTER for default>
        Work Phone []:                      <Press ENTER for default>
        Home Phone []:                      <Press ENTER for default>
        Other []:                           <Press ENTER for default>
Is the information correct? [y/N] y         <Press ENTER for default>
user@pcn:~#
```

At this point you are done and the user *inst* now exists on your machine.

In order to allow the new *inst* user to use the `sudo` command it must be a member of the *adm* group. To do this you can type:

```
$ sudo usermod -G adm inst
```

And, to verify that *inst* is now a member of the *adm* group:

```
$ groups inst
```

3.) Learn how to install software

This is a large topic. Your instructor should have discussed this with you previously. In general you can use `apt-get` to install software, clean up software installs, remove software and update your repositories. You can use `aptitude` as a meta-installer to control `apt`. The `dpkg` command extracts and installs individual Debian packages and is called by `apt`. In addition, `synaptic` is a graphical interface to `apt` that can be used in Gnome or KDE. Finally, `apt-cache` allows you to view information about already installed packages.

We are going to concentrate on the `apt-get` method of software installation. But you should spend some time reading about and learning about how `apt` (in general), `aptitude`, `dpkg`, `apt-cache`, and `synaptic` work. To do this read the man pages for each:

```
$ man dpkg
$ man apt
$ man apt-get
$ man aptitude
$ man apt-cache
```

You don't need to read each man page in detail as this could take a while, but review them enough to understand the basics of each command and how they differ.

After reading try a few commands:

```
$ dpkg
$ dpkg –help | more       [space for next page, or CTRL-C to exit more screen]
$ apt-get | more
$ sudo apt-get check      [what does the “check” option do?]
$ aptitude                [Look around at what is installed.]
$ apt-cache | more
$ apt-cache stats
$ apt-cache search nagios2
$ apt-cache showpkg nagios2 | more
```

4.) Update /etc/apt/sources.list

When using `apt`, `apt-get`, `aptitude` and/or `synaptic` there is a master file that tells Ubuntu where to look for software you wish to install. This file is /etc/apt/sources.list. You can update this file to point to different repositories (third party, local repositories, remove the cdrom reference, etc...). In our case we are now going to do this. We'll edit this file and we are going to edit out any reference to the Ubuntu 7.10 cdrom, which is left from the initial install.

To edit the file /etc/apt/sources.list do:

```
$ sudo vi /etc/apt/sources.list
```

In this file we want to comment out any references to the Ubuntu cd-rom. You'll see the following lines at the top of the file:

```
#
# deb cdrom:[Ubuntu-Server 7.10 _Gutsy Gibbon_ - Release i386 (20071016)]/ gutsy main restricted

deb cdrom:[Ubuntu-Server 7.10 _Gutsy Gibbon_ - Release i386 (20071016)]/ gutsy main restricted
```

Update this by simply commenting out the one line (see your vi reference sheet for help):

```
#
# deb cdrom:[Ubuntu-Server 7.10 _Gutsy Gibbon_ - Release i386 (20071016)]/ gutsy main restricted

# deb cdrom:[Ubuntu-Server 7.10 _Gutsy Gibbon_ - Release i386 (20071016)]/ gutsy main restricted
```

Now the `apt` command (`apt-get`) won't attempt to read your cd-rom drive each time you install software.

Change your sources list:

We won't be doing this, but take a closer look at the file /etc/apt/sources.list. You should see multiple entries along the line of “http://CC.archive.ubuntu.com/” where the “CC” is a country code. If you installed and said that your location was Thailand, then the entry would read, “http://th.archive.ubuntu.com/”, and so forth.

If you make changes to this file, then you should remember to run:

```
$ sudo apt-get update
```

To make sure that all your local repository lists are up to date.

5.) Install libc, gcc, g++ and make

Two items missing from a default Debian/Ubuntu installation are `gcc` and `make` plus their associated bits and pieces. This can be quite disconcerting if you are used to compiling software under other versions of Linux. Luckily there is an easy way to install all the bits and pieces you need to use `gcc` and/or `make`. Simply do:

```
$ sudo apt-get install build-essential
```

and respond with a "Y" when asked if you "...want to continue". Once the installation process finishes you should have both `gcc` and `make` installed on your machine.

This is an example of installing software using a "meta-package." If you type in the command:

```
$ sudo apt-cache showpkg build-essential
```

You will see a descriptive list of all the various pieces of software that are installed for this package.

6.) Learn how to control services

The first thing to remember is that if you install a new service, say a web server (Apache), then Ubuntu will automatically configure that service to run when you reboot your machine ***and it will start the service immediately!***

This is quite different from the world of Red Hat, Fedora, CentOS, etc. In order to configure and control services the core tool available to you is `update-rc.d`. This tool, however, may not be the easiest to use. Still, you should read and understand a bit about how this works by doing:

```
$ man update-rc.d
```

There are a couple of additional tools available to you that you can install. These are `sysvconfig` and `rcconf`. Both of these are console-based gui tools. To install them do:

```
$ sudo apt-get install sysvconfig rcconf
```

Did you notice that we specified two packages at the same time? This is a nice feature of `apt-get`. Try both these commands out. You'll notice that the `sysvconfig` command is considerably more powerful. Please don't make any changes to your services at this time.

```
$ sudo rcconf
$ sudo sysvconfig
```

Finally, there is a nice Bash script that has been written which emulates the Red Hat `chkconfig` script. This is called `rc-config`. We have placed this script on our “noc” box. Let's download the script and install it for use on your machine:

```
$ cd
$ wget http://noc/workshop/scripts/rc-config
$ chmod 755 rc-config
$ sudo mv rc-config /usr/local/bin
```

At this point the script is installed. You should be able to just run the script by typing:

```
$ rc-config
```

Try viewing all scripts and their status for all run-levels:

```
$ rc-config -l
```

Now trying viewing the status of just one script:

```
$ rc-config -ls anacron
```

You can see how this script works, if you understand enough of bash scripts, by taking a look at it's code:

```
$ less /usr/local/bin/rc-config
```

7.) Use the `ip` tool

The `ip` command is a powerful network debugging tool available to you in Ubuntu. You may have already used this tool in other Linux distributions. But, if not, start by reading:

```
$ man ip
```

As you can see this tool is designed to, “show/manipulate routing, devices, policy routing and tunnels.”

For instance, if you are wondering what your default route is (or are) you can simply type:

```
$ ip route
```

This is actually short for “`ip route show`”. Maybe you are wondering out which interface packets will go to a certain address? A quick way to find out is:

```
$ ip route get 128.223.32.35
```

Clearly you can substitute any IP address you wish above. This is useful for boxes that have multiple network interfaces defined.

Maybe you want to be able to sniff packets coming across an interface on your box. To do this you may wish to place your interface in promiscuous mode. Often this requires updating a kernel parameter. With the ip command you can do:

```
$ sudo ip link set eth0 promisc on
```

Note the use of "sudo" here as setting an interface requires admin privileges. Now you can snoop the packets on the eth0 interface by doing:

```
$ sudo tcpdump -i eth0
```

Be sure to read the man page for tcpdump if you want further information.

8.) Create the locate database

One of the easiest ways to find files on your system is to use the `locate` command. For details, as usual, read the man pages:

```
$ man locate
```

We assume you are familiar with this command, but building the `locate` database is a bit different on different Linux and Unix versions.

Locate uses a hashed database of f filenames and directory paths. the command searches the database instead of the file system to find files. While this is *much* is much more efficient it has two downsides:

1. If you create the `locate` database as root then users can see files using `locate` that they otherwise would not be able to see. This is considered a potential security hole.
2. The `locate` command is only as precise as the `locate` database. If the database has not been recently updated, then newer files will be missed. Many systems use an automated (cron) job to update the `locate` database on a daily basis.

To create an initial `locate` database, or update the current one do:

```
$ sudo updatedb
```

Once this process completes (it may take a few minutes) try using the command:

```
$ locate ssh
```

Quite a few files go past on the screen. To find any file with "ssh" in it's name or it's path and which has the string "conf" you can do:

```
$ locate ssh | grep conf
```

Read about "`grep`" using "`man grep`" for more information. The `locate` command is very powerful and useful. For a more exacting command you can consider using "`find`". This is harder to use and works by brute-force. As usual do "`man find`" for more information.

9.) So, you wanna be root...

As you have noticed Ubuntu prefers that you do your system administration from a general user account making use of the `sudo` command.

If you must have a root shell to do something you can do this by typing:

```
$ sudo bash
```

This is useful if you have to look for files in directories that would otherwise be unavailable for you to see. Remember, be careful. As root you can move, rename or delete any file or files you want.

What if you really, really want to log in as root? OK, you can do this as well. First you would do:

```
$ sudo passwd root
```

Then you would enter in a root password – definitely picking something secure and safe, right?! Once you've set a root password, then you can log in as root using that password if you so desire. That's a controversial thing to do in the world of Ubuntu and Debian Linux.

10.) Install Apache Web Server and PHP

During the week we will be using the Apache Web server (version 2.x) as well as the PHP scripting language. In order to install these now you can simply do:

```
$ sudo apt-get install apache2 libapache2-mod-php5
```

If you are wondering how to find something like "libapache2-mod-php5" here is what your instructor did:

```
$ sudo apt-cache search apache | grep php
```

The output was:

```
libapache2-mod-suphp - Apache2 module to run php scripts with the owner permissions
php-auth-http - HTTP authentication
php-config - Your configuration's swiss-army knife
php5-apache2-mod-bt - PHP bindings for mod_bt
suphp-common - Common files for mod suphp
libapache2-mod-php5 - server-side, HTML-embedded scripting language (apache 2 module)
php5-cgi - server-side, HTML-embedded scripting language (CGI binary)
```

Reading the descriptions made it apparent that the version of PHP needed was in the “libapache2-mod-php5” package.

To test whether or not our Apache install has worked we can use the text-based “lynx” web browser. This is not installed by default, so first we must do:

```
$ sudo apt-get install lynx
```

Once the install completes type:

```
$ lynx localhost
```

and you should see the default apache2 directory listed. Press “Q” to quit from lynx. PHP has already been configured to load and Apache has been reconfigured to execute files with extensions of “.php”, but because the PHP module was installed *after* Apache in the command above you must reload/restart the Apache web server for the new configuration to take affect. There are multiple ways to do this, but one that is easy to remember is:

```
$ /etc/init.d/apache2 restart
```

Go ahead and do this now.

11.) Install OpenSSH, public key and new sshd_config

This exercise is to make life easier for workshop management during the week. We are going to install the OpenSSH server, copy over a new configuration file for the server that allows people to log in to the root account using public/private keys and we'll copy over a public key used by your class instructors to push out updates to your machines as needed.

To start do:

```
$ sudo apt-get install openssh-server
$ sudo bash
# cd
# mkdir .ssh
# cd .ssh
# wget http://noc/workshop/conf/authorized_keys
# cd /etc/ssh
# mv sshd_config sshd_config.bak
# wget http://noc/workshop/conf/sshd_config
# /etc/init.d/ssh restart
```

At this point people can log in to your machine either as inst, or as root using the public/private key combination that you installed when you copied the “authorized_keys” file.

Your instructor will tell you whether to go ahead with the next two exercises.

12.) Install Gnome 2.x

It is actually quite simple to install a graphical desktop on Ubuntu. By default Ubuntu uses the Gnome desktop. If you wish to use KDE with Ubuntu there is a separate version of the Ubuntu distribution called Kubuntu that you can find at www.ubuntu.com.

We have configured your workshop lab so that the files for Gnome are on a local machine. The installation requires over 400MB of files to download and over 1GB of total space. Downloading will not take long, but unpacking and installing will take some time.

The Gnome desktop comes with the Ubuntu meta-package called “ubuntu-desktop”.

```
$ sudo apt-get install ubuntu-desktop
```

This will now take quite some time. Feel free to go to lunch if it is time do to that.

13.) Configure XWindow

Ubuntu uses the Xorg XWindow system for the underlying graphics engine that drives the Gnome Desktop. Once the Gnome desktop is installed along with Xorg you need to configure Xorg to work with your hardware and the resolution you have chosen. Luckily Xorg has made this quite easy to do. First do:

```
$ cd
$ sudo Xorg -configure
```

This should create the file xorg.conf.new. To finalize configuring your X Server do:

```
$ sudo cp xorg.conf.new /etc/X11/xorg.conf
```

Now type:

```
$ sudo gdm
```

and your Gnome desktop environment should start. You can log in with your username and password.

Most likely your kernel has been upgraded during these exercises. You may wish to restart your machine at an appropriate time to load the new kernel.