DAY - Creating Your Own Server - with Liquid Web

INTRO

First, you need a server. You can't really learn about administering a remote Linux server without having a one of your own - so today we're going to buy one!

Through the magic of Linux and virtualisation, it's now possible to get a small Internet server setup almost instantly - and at very low cost. Technically, what you'll be doing is creating and renting a VPS ("Virtual Private Server"). In a datacentre somewhere a single physical server running Linux will be split into a dozen or more Virtual servers using the KVM (Kernel-based Virtual Machine) feature that's been part of Linux since early 2007. There are many hundreds of hosting companies offering low cost VPS deals - and sites like http://lowendbox.com/ that compare them.

As well as a hosting provider, we also need to choose which "flavour" of Linux to install on our server. If you're new to Linux then the range of "distributions" available can be confusing - but the latest LTS ("Long Term Support") version of Ubuntu Server is a popular choice.

These instruction will walk you through using Liquid Web (http://www.liquidweb.com) as your VPS hosting provider.

Signing up with Liquid Web.

http://www.liquidweb.com/StormServers/index.html

Under the Box that reads Storm® VPS Servers Fully Managed VPS Hosting click the button that reads from $50/mo.

Set the slider to the 1G option and click the green order now button

Most of Liquid Webs servers run CentOS but this course was written for Ubuntu so choose the most current Ubuntu option. Many of the things you learn in this course will apply across all Linux distributions.

Give your server a unique hostname and a strong password. You should only need the default single IP.

Click next and fill out your personal information.

Once your server finishes creation you should be ready to go. You now have the IP address for your server which you alone are responsible for administering!

DAY 1 - Accessing your server

INTRO

You should now have received an email with the address of your brand-new server. This is yours to use during the course - and not shared with anyone else. You alone will be administering it.

Your server will be running the latest Ubuntu Server LTS (Long Term Support) version.

To become a fully-rounded Linux server admin you should become comfortable working with different versions of Linux, but for now Ubuntu is a good choice.

Once you have reached a level of comfort at the command-line then you'll find your skills transfer not only to all the standard Linux variants, but also to Android, Apple's OSX, OpenBSD, Solaris and IBM AIX. Throughout the course you'll be working on Linux - but in fact most of what is covered is applicable to any system in the "UNIX family" - and the major differences between them are with their graphic user interfaces such as Gnome, Unity, KDE etc - none of which you'll be using!

YOUR TASKS TODAY:

- Connect and login remotely your server
- Run a few simple simple commands to check the status of your servers
- Change your password

INSTRUCTIONS

Remote access used to be done by the simple *telnet* protocol, but now the much more secure SSH ("Secure SHell) protocol is always used.

If you're using any Linux or Unix system, including Apple's OS X, then you can simply open up a "terminal" session and use your command-line *ssh* client like this:

```
ssh user@<ip address>
```

For example:

```
ssh support@192.123.321.99
```

On an OSX machine you'll normally access the command line via Terminal.app - it's in the Utilities sub-folder of Applications. On Linux distributions with a menu you'll typically find it under "Applications menu -> Accessories -> Terminal", "Applications menu -> System -> Terminal" or "Menu -> System -> Terminal Program (Konsole)"- or you can simply search for your terminal application.

On Microsoft Windows there is no suitable client built-in, or available from Microsoft - so the "standard" is a free program called PuTTy. It is written and maintained by Simon Tatham, so get it directly from his site at: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html You can download and install the full suite with "Windows installer" version. If you don't have permission to install software on the computer you are using then you can run PUTTY.exe directly from its download page without installing it. There are other SSH clients, and any should be suitable for the course.

The first time you connect to your server, you'll receive a warning that you're connecting to a new server - and be asked if you wish to "cache the host key". Do this. Now, if you get a warning in future connections it means that either: (a) you are being fooled into connecting to a different machine or (b) someone may be trying a "man in the middle" attack.

So, now login to your server with the provided login details - and remember that Linux is case-sensitive regarding user names, as well as passwords.

Once logged in, notice that the "command prompt" that you receive ends in "$" - this is the convention for an ordinary user, whereas the "root" user with full administrative power has a # prompt.

Try these simple commands:

```
ls
uptime
free
df -h
uname -a
```

Now use the *passwd* command to change your password. To do this, think of a new, secure password, then simply type *passwd*, press "Enter" and give your current password when prompted, then the new one you've chosen, confirm it - and then WRITE IT DOWN somewhere.

A common Linux convention is that simply selecting text in the terminal session does a "copy", and within the terminal a RightClick is considered a paste. Test this out by copying some text from Windows or OSX and right-clicking into your terminal session - and also try the reverse: selecting text in the terminal and pasting into windows. Getting the hang of this is a useful skill.

Log out by typing *exit*.

POSTING YOUR PROGRESS

Regularly posting your progress is an important part of the GREP101 process.

In future days, you'll mark your progress by creating files or folders on your server as directed (and which we'll be monitoring), but for now simply drop a quick note to tutor@grep101.com to let us know that you're underway.

If you wish, post a small introduction of yourself for your "classmates" onto the the web forum. No need to be too specific: something like "mid 20s Apple guy, not very technical" or "retired old-timer programmer looking to keep the brain active" is fine.

WRAP

You now have the ability to login remotely to your own server. Perhaps you might now try logging in from home and work - even from your smartphone! As a server admin you'll need to be comfortable logging in from all over. (Note you can run PUTTY.exe directly from its download page without installing it - just google for "putty.exe download" to get to the page. You can also potentially use JavaScript ssh clients (search for consolefish), or from a cybercafe; but these options involve putting more trust in third-parties than most sysadmins would be comfortable with when accessing production systems.

You'll be spending a lot of time in your SSH client, so it pays to spend some time customising it. At the very least try "black on white" and "green on black" - and experiment with different monospaced fonts, ("Ubuntu Mono" is free to download, and very nice).

EXTENSION

If this is all too easy, then spend some time reading up on:

- "SSH Tunneling"
- "Password-less SSH login"

RESOURCES

- "Using PuTTY for SSH (Windows)"
- Comparing CENTOS and Ubuntu for servers
- A Beginners Guide to SSH

DAY 2 - Basic navigation

INTRO

Most computer users outside of the Linux and Unix world don't spend much time at the command-line now, but as a Linux sysadmin this is your default working environment - so you need to be as skilled in it.

When you use a graphic desktop such as Windows or Apple's OS X (or even the latest Linux flavors), then increasingly you are presented with simple "places" where your stuff is stored - "Pictures" "Music" etc but if you're even moderately technical then you'll realize that underneath all this is a hierarchical "directory structures" of "folders" (e.g. *C:\Users\Steve\Desktop* on Windows or */Users/Steve/Desktop* on OS X).

From now on, the course will point you to a range of good online resources for a topic, and then set you a simple set of tasks to achieve. It's perfectly fine to google for other online resources, refer to any books you have etc - and in fact a fundamental element of the design of this course is to force you to do a bit of your own research. Even the most experienced sysadmins will do an online search to find advice for how to use commands - so the sooner you too get into that habit the better!

YOUR TASKS TODAY

- Use the provided resources check out the basic commands and concepts and commands
- Login to your server via SSH and move about the directory structure at the command-line
- Take note of how your "prompt" changes as you change directory
- Be sure to understand how *cd* on it's own takes you back to your "home directory"
- Understand what *cd ~* and *cd ..* do
- Use the *ls* command to list the contents of directories, and try several of the "switches" - in particular *ls -ltr* to show the most recently altered file last
- Use the *mkdir* command to create a new directory (folder) *feedback* in your home folder (e.g */home/support/feedback*)

STEP-BY-STEP

- login to your server using ssh
- */* is the "root" of a branching tree of folders (also known as directories)
- at all time you are "in" one part of the system - the command *pwd* ("present working directory") will show you where you are
- generally your prompt is also configured to give you at least some of this information, so if I'm "in" the */etc* directory then the prompt might be "steve@202.203.203.22: /etc>$" or simply "/etc: $"
- *cd* or *chdir* moves to different areas - so *cd /var/log* will take you into the */var/log* folder - do this and then check with *pwd* - and look to see if your prompt changes to reflect your location.
- you can move "up" the structure by typing *cd ..* ("cee dee dot dot ") try this out by first *cd*'ing to

/var/log/ then *cd ..* and then *cd ..* again - watching your prompt carefully, or typing pwd each time, to clarify your present working directory.
- a "relative" location is based on your present working directory - e.g. if you first *cd /var* then pwd will confirm that you are "in" */var*, and you can move to */var/log* in two ways - either by providing the full path with *cd /var/log* or simply the "relative" path with the command *cd log*
- a simple *cd* will always return you to your own defined "home directory", also referred to as ~ (the "tilde" character) [NB: this differs from DOS/Windows]
- what files are in a folder? The *ls* (list) command will give you a list of the files, and sub folders. Like many Linux command, there are options (known as "switches") to alter the meaning of the command or the output format. Try a simple *ls*, then *ls -l -t* and then try *ls - l -t -r -a*
- by convention, files with a starting character of "." are considered hidden and the *ls*, and many other commands, will ignore then. The *-a* switch includes them. You should see a number of hidden files in your home directory.
- a note on switches: Generally most Linux command will accept one of more "parameters", and one or more "switches". So, when we say *ls -l /var/log* the "*-l*" is a switch to say "long format" and the "*/var/log*" is the "parameter". Many commands accept a large number of switches, and these can generally be combined (so from now on, use *ls -ltra*, rather than *ls -l -t -r -a*
- in your home directory type *ls -ltra* and look at the far left hand column - those entries with a "d" are directories (folders) rather than files. They may also be shown in a different color or font - if not, then adding the "-G" switch should do this (ie *ls -ltraG*)
- you can make a new folder/directory with the *mkdir* command, so move to your home directory, type *pwd* to check that you are indeed in the correct place, and then create a directory, for example to create one called "test", simply type *mkdir test*. Now use the *ls* command to see the result.

WRAP

Being able to move confidently around the directory structure at the command line is important, so don't think you can skip it! However, these skills are something that you'll be constantly using over the twenty days of the course, so don't despair if this doesn't immediately "click".

EXTEND YOURSELF

If this is already something that you're very familiar with, then:

- Take the time today to understand how the environment variable PS1 etc work (this article: Bash Shell: Take Control of PS1, PS2, PS3, PS4 and PROMPT_COMMAND is a good start).
- Set youself up with a custom prompt using the information in Bash Shell PS1: 10 Examples to Make Your Linux Prompt like Angelina Jolie
- ...but also ensure you've created the new directory *feedback* in your home folder (e.g */home/support/feedback*) - we'll be using this in the next steps.

RESOURCES

- Explore the Linux file system with "cd"
- Linux File System
- Simple Terminal Commands on Ubuntu
- Solaris Unix Commands

Day 3 - Power trip!

INTRO

You have been logging in as "support" at your server, yet you're probably aware that "root" is the power user on a Linux system. This administrative or "superuser" account, is all powerful - and it's easy to mistakenly cripple your server.

On many production systems all sysadmins login as "root", but it's now common Best Practice to discourage or disallow login directly by "root" - and instead to give specified trusted users the permission to run root-only commands via the *sudo* command.

This is the way that your server has been set-up, with your "support" login given the ability to run any root-only command - but only if you precede it with *sudo*, and re-confirm your identity with your password.

YOUR TASKS TODAY:

- Understand how to tell whether you have superuser powers or not
- Test running the *reboot* command, and then via *sudo*(ie *sudo reboot*)
- Use the *uptime* command to confirm that your server did actually fully restart

RESOURCES

- This cartoon explains it nicely!
- Sudo in Ubuntu
- How to use "sudo"

EXTENSION

- Test fully "becoming root" by the command *sudo -i* and then type *exit* or *logout* to get back to your own normal "support" login.
- Read Server Best Practices

Day 4 - Installing software, exploring the file structure

INTRO

As a sysadmin, one of your key tasks is to install new software as required. You'll also need to be very familiar with the layout of the standard directories in a Linux system.

You'll be getting practice in both of these areas in today's session.

Your tasks today

- Install a new application from the online repositories
- Become familiar with some of the standard directories
- Look at the format and content of some configuration files.

If you've used a smartphone "app store " or "market", then you'll immediately understand the normal installation of Linux software from the standard repositories. As long as we know what the name or description of a package (=app) is, then we can search for it:

```
apt-cache search "midnight commander"
```

This will show a range of matching "packages", and we can then install them with the *apt-get* command. So to install package *mc* (Midnight Commander) on Ubuntu:

```
sudo apt-get install mc
```

(Unless you're already logged in as the *root* user you need to use *sudo* before the installation commands - because an ordinary user is not permitted to install software that could impact a whole server).

Now that you have *mc* installed, start it by simply typing *mc* and pressing *Enter*.

This isn't a "classic" Unix application, but once you get over the retro interface you should find navigation fairly easy, so go looking for these directories:

```
/root
/home
/sbin
/etc
/var/log
```

...and use the links in the Resources section below to begin to understand how these are used.

Most key configuration files are kept under */etc* and subdirectories of that. These files, and the logs under */var/log* are almost invariably simple text files. In the coming days you'll be spending a lot of time with these - but for now simply use F3 to look into their contents.

Some interesting files to look at are: */etc/passwd* and */var/log/auth.log*

Use F3 again to exit from viewing a file.

F10 will exit *mc*, although you may need to use your mouse to select it.

(On an Apple Mac in Terminal, you may need to use ESC+3 to get F3 and ESC+0 for F10)

Now search for an install some more packages: Try searching for “hangman”. You will probably find that an old text-based version is included in a package called *bsdgames*. Install and play a couple of rounds...

Posting your progress

- Use *mc* to create a Day4 folder under */home/support*

Extension

- Use *mc* to view */etc/apt/sources.list* where the actual locations of the repositories are specified. Often these will be “mirror” sites that are closer to your server than the main Ubuntu servers.
- Read Repositories - CommandLine for more of the gory details.

Resources

- Ubuntu and Red Hat/CentOS package management comparison
- Ubuntu Server Guide - apt-get

Day 5 - More or less...

INTRO

Today we'll end with a bang - with a quick introduction to five different topics. Mastery isn't required today - you'll be getting plenty of practice with all these in the sessions to come!

Don’t be misled by how simplistic some of these commands may seem - they all have hidden depths and many sysadmins will be using several of these every day.

TASKS

Use the links in the Resources section to complete these tasks:

- Get familiar with using *more* and *less* for viewing files, including being able to get to the top or bottom of a file in *less*, and searching for some text
- Test how “tab completion” works - this is a handy feature that helps you enter commands correctly. It helps find both the command and also file name parameters (so typing *les* then hitting “Tab” will complete the command *less*, but also typing *less /etc/serv* and pressing “Tab” will complete to *less /etc/services*. Try typing *less /etc/s* then pressing “Tab”, and again, to see how the feature handles ambiguity.
- Now that you've typed in quite a few commands, try pressing the “Up arrow” to scroll back through them. What you should notice is that not only can you see your most recent commands - but even those from the last time you logged in. Now try the *history* command - this lists out the whole of your cached command history - often 100 or more entries. There are number of clever things that can be done with this, but the simplest is to repeat a command - pick one line to repeat (say number 20) and repeat it by typing !20 and pressing “Enter”. Later when you'll be typing long, complex, commands this can be *very* handy.
- Look for “hidden” files in your home directory. In Linux the convention is simply that any file starting with a "." character is hidden. So, type *cd* to return to your "home directory" then *ls -l* to show what files are there. Now type *ls -la* or *ls -ltra* (the "a" is for "all") to show all the files - including those starting with a dot. By far the most common use of "dot files" is to keep personal settings in a home directory. So use your new skills with *less* to look at the contents of *.bashrc* , *.bash_history* and others.
- Finally, use the *nano* editor to create a file */home/support/feedback/Day5* and type up a summary of how the last five days have worked for you. (Yes, we’ll be “magically” retrieving and reading this!)

RESOURCES

- Unix Less Command: 10 Tips for Effective Navigation *Managing command history
- BASH Shell commands less
- Tab completion

- Linux homedir dotfiles
- Nano editor tutorials

Day 6 - Installing Apache

INTRO

For the first day of the second week, you'll install a common application - the Apache2 web server - also known as *httpd* - the "Hyper Text Transport Protocol Daemon"!

If you're a website professional then you might do things slightly differently, but our focus with this is not on Apache itself, or the website content, but to get a better understanding of:

- application installation
- configuration files
- services
- logs

TASKS

- Refresh your list of available packages (apps) by: *sudo apt-get update* - this takes a moment or two, but ensures that you'll be getting the latest versions.
- Install Apache from the repository with a simple: *sudo apt-get install apache2*
- Confirm that it's running by browsing to *http://[your server DNS name or IP]* - where you should see a confirmation page.
- Apache is installed as a "service" - a program that starts automatically when the server starts and keeps running whether anyone is logged in or not. Try stopping it with the command: *sudo service apache2 stop* - check that the webpage goes dead - then re-start it with *sudo service apache2 start* check its status with: *sudo service apache2 status* - and see what other options are available with: *sudo service apache2*.
- As with the vast majority of Linux software, configuration is controlled by files under the */etc* directory - check the configuration files under */etc/apache2* especially */etc/apache2/apache.conf* - you can use *less* to simply view them, or an editor like *nano* to view and edit as you wish
- The location of the default webpage is defined by the *DocumentRoot* parameter in the file */etc/apache2/sites-available/default*
- Use *nano* (or another editor like *vim* if you prefer) to edit the default page - normally at */var/www/index.html* - even if you've never seen HTML before you should see how to change the displayed body text, and then reload your the webpage in your browser to see the result.
- Notice the directory */etc/apache2/conf.d* - the settings in the files in this folder are merged in with those from */etc/apache2/apache2.conf* at load. This approach of lots of small specific config files is common. The ".d" extension has no special meaning to the operating system, but is a common convention to denote a directory.
- If you're familiar with configuring web servers, then go crazy, setup some virtual hosts, or add in some mods etc.
- As with the vast majority of Linux software, Apache keeps its logs under the */var/log* directory - look at the logs in */var/log/apache2* - in the *access.log* file you should be able to see your session

from when you browsed to the test page. Notice that there's an overwhelming amount of detail - this is typical, and later you'll see we'll see how to filter out what we want. Notice the *error.log* file too - hopefully this one will be empty!

Posting your progress

As usual, practice your text-editing skills, and allow us to judge your progress - but this time do it by editing */var/www/index.html* - so it should be visible on your new website. (It doesn't have to be pretty!)

Security

- As the sysadmin of this server, responsible for its security, you need to be aware that you've now increased the "attack surface" of your server. The logs may reveal access from a wide range of visiting search engines - and attackers - but that's perfectly normal.
- If you run the commands: *sudo apt-get update*, then: *sudo apt-get upgrade*, and accept the suggested upgrades, then you'll be up to date, and secure enough for a test environment - but you should re-run this regularly.

RESOURCES

- HTTPD - Apache2 Web Server
- The Apache HTTP Server

EXTENSION

Read up on:

- The traditional SYS V INIT system
- Find the Apache service script under */etc/init.d* and have a read - this is a "shell script"
- Check to see how it's started and stopped at various runlevels. (The command: *runlevel* will show you what runlevel you started from and are currently running at, and the contents of */etc/rc3.d* show what will be started or killed when you move to runlevel 3)
- Google for information on the shiny new init systems coming to Linux: *upstart* for Ubuntu and *systemd* for CentOS and others

Day 7 - the infamous "grep"...

INTRO

Your server is now running two services: the *sshd* (Secure Shell Daemon) service that you use to login; and the Apache2 web browser. Both of these services are generating logs as you and others access your server - and these are text files which we can analyse using some simple tools.

Plain text files are a key part of "the Unix way" and there are many small "tools" to allow you to easily edit, sort, search and otherwise manipulate them. Today we'll use *grep*, *cat*, *more*, *less*, *cut*, *awk* and *tail* to slice and dice your logs.

The *grep* command is famous for being extremely powerful and handy, but also because its "nerdy" name is typical of Unix/Linux conventions. The GREP101 course takes its name from this command because it nicely sums up both the power and challenge of working at the command line.

TASKS

- Dump out the complete contents of a file with *cat* like this: *cat /var/log/apache2/access.log*
- Use *less* to open the same file, like this: *less /var/log/apache2/access.log* - and move up and down through the file with your arrow keys, then use "q" to quit.
- Again using *less*, look at a file, but practice confidently moving around using *gg*, *GG* and */*, *n* and *p* (to go to the top of the file, bottom of the file, to search for something and to hop to the next "hit" or back to the previous one)
- View recent logins and *sudo* usage by viewing */var/log/auth.log* with *cat* and *less*
- Look at just the tail end of the file with *tail /var/log/apache2/access.log* (yes, there's also a *head* command!)
- Follow a log in real-time with: *tail -f /var/log/apache2/access.log* (while accessing your server's web page in a browser)
- Dump out a file with *cat* , pipe that output to *grep* with a search term - like this: *cat /var/log/auth.log | grep "Failed password"*
- Simplify this to: *grep "Failed password" /var/log/auth.log*
- Piping the output of one command to another allows you to narrow your search so try something like: *grep "Failed password" /var/log/auth.log | grep "root"*
- Use the *cut* command to select out most interesting portions of each line by specifying "-d" (delimiter) and "-f" (field) - like: *grep "Failed password" /var/log/auth.log | grep "root" | cut -f 4- -d ":"* (field 4 onwards, where the delimiter between field is the ":" character). This is very useful in extracting useful information from log data.

The output of any command can be "redirected" to a file with the ">" operator. The command: *ls -ltr > listing.txt* wouldn't list the directory contents to your screen, but instead redirects it into the file "listing.txt" (creating that file if it didn't exist, or overwriting the contents if it did).

POSTING YOUR PROGRESS

Re-run the command to list all the IP's that have unsuccessfully tried to login to your server as root - but this time, use the the ">" operator to redirect it to the file:
/home/support/feedback/attackers.txt

RESOURCES

- Text processing commands (http://www.youtube.com/watch?v=xdiZpPsw7mQ)
- Drew's grep tutorial (http://www.uccs.edu/~ahitchco/grep/)

EXTENSION

- See if you can extend your filtering of *auth.log* to select just the IP addresses, then pipe this to *sort*, and then further to *uniq* to get a list of all those IP addresses that have been "auditing" your server security for you.
- Investigate the *awk* and *sed* commands. When you're having difficulty figuring out how to do something with *grep* and *cut*, then you may need to step up to using these. Googling for "linux sed tricks" or "awk one liners" will get you many examples.
- Aim to learn at least one simple useful trick with both *awk* and *sed*

Day 8 - Ports, open and closed

INTRO

Your server is now providing two services - SSH for remote login, and WWW for web access. These services are by default provided on specific well-known TCP/IP “ports” - 22 and 80.

As a sysadmin you need to understand what ports you have open on your servers, because each open port is also a potential focus of attacks. You need to be be able to put in place monitoring and controls on them as appropriate.

INSTRUCTIONS

First we'll look at a couple of ways of determining what ports are open on your server:

- *netstat* - this is a standard utility
- *nmap* - this "port scanner" won't normally be installed by default

There are a wide range of options that can be used with *netstat*, but first try: *netstat -lnp*

The output lines show which ports are open on which interfaces:

```
$ sudo netstat -lnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*                   LISTEN
3808/apache2
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
599/sshd
tcp6       0      0 :::22                   :::*                    LISTEN
599/sshd
Active UNIX domain sockets (only servers)
```

The lines above show ports 80 and 22 open, and because we’ve example used the "-p" switch, so we can see which process and program is servicing each port.

Now install *nmap* with apt-get. This works rather differently, actively probing 1,000 or more ports to check whether they're open. It's most famously used to scan remote machines - please don't - but it's also very handy to check your own configuration, by scanning your server:

```
$ nmap localhost

Starting Nmap 5.21 ( http://nmap.org ) at 2013-03-17 02:18 UTC
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00042s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
```

```
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds
```

Port 22 is providing the *ssh* service, which is how you're connected, so that will be open. If you have Apache running then port 80/http will also be open. Every open port is an increase in the "attack surface", so it's Best Practice to shut down services that you don't need.

Host firewall

The Linux kernel has built-in firewall functionality called "netfilter". We configure and query this via various utilities, the most low-level of which is the *iptables* command. This is powerful, but also complex - so we'll use a more friendly alternative - *ufw* - the "uncomplicated firewall".

First let's list what rules are in place by typing *sudo iptables -L*

You will see something like this:

```
Chain INPUT (policy ACCEPT)
target  prot opt source               destination

Chain FORWARD (policy ACCEPT)
target  prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target  prot opt source               destination
```

So, essentially no firewalling - any traffic is accepted to anywhere.

Using *ufw* is very simple. First we need to install it with:

```
sudo apt-get install ufw
```

Then, to allow SSH, but disallow HTTP we would type:

```
sudo ufw allow ssh
sudo ufw deny http
```

(BEWARE - do *not* "deny" ssh, or you'll lose all contact with your server!)

and then enable this with:

```
sudo ufw enable
```

Typing *iptables -L* now will list the detailed rules generated by this - one of these is:

```
"DROP       tcp  --  anywhere             anywhere             tcp dpt:http"
```

The effect of this is that although your server is still running Apache, it's no longer accessible from the "outside". Test for yourself!

In practice, ensuring that you're not running unnecessary services is often enough protection, and a host-based firewall is unnecessary, but this very much depends on the type of server you are configuring. Regardless, hopefully this session has given you some insight into the concepts.

Using non-standard ports

Occasionally it may be reasonable to re-configure a service so that it's provided on a non-standard port. This will reduce the attacking traffic that you see, which may be useful, but would be scoffed at by experts as "security by obscurity" - equivalent to moving the keyhole on your front door in an unusual place rather than improving the lock itself.

POSTING YOUR PROGRESS

- Redirect the output of "netstat -lnp" to */home/support/feedback/Day8*

RESOURCES

- UFW - Uncomplicated Firewall (https://help.ubuntu.com/community/UFW)
- Collection of basic Linux Firewall iptables rules (http://linuxconfig.org/collection-of-basic-linux-firewall-iptables-rules)
- 10 Netstat Command Example (http://www.thegeekstuff.com/2010/03/netstat-command-examples/)
- UFW Uncomplicated Firewall (http://www.youtube.com/watch?v=nc3A5Dy4xE0&feature=relmfu) (video)

EXTENSION

Even after denying access, it might be useful to know who's been *trying* to gain entry. Check out these discussions of logging and more complex setups:

- How to Log Linux IPTables Firewall Dropped Packets to a Log File
- Firewalling with iptables - One approach

Day 9 - Getting the computer to do your work for you

INTRO

Linux has a rich set of features for running scheduled tasks. One of the key attributes of a good sysadmin getting the computer to do your work for you (sometimes misrepresented as laziness!), and a well configured set of scheduled tasks is key to keeping your server running well.

INSTRUCTIONS

Each user potentially has their own set of scheduled task which can be listed with the *crontab* command (list out "support"s crontab entry with *crontab -l* and then root's with *sudo crontab -l*).

However, there's also a system-wide crontab defined in */etc/crontab* - use *less* to look at this. Here is example, along with an explanation:

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.monthly )
```

Lines beginning with "#" are comments, so *# m h dom mon dow user command* defines the meanings of the columns. The first line says that at 17mins after every hour, on every day, the credential for "root" will be used to run any scripts in the /etc/cron.hourly folder - and similar logic kicks off daily, weekly and monthly scripts.

This is a tidy way to organise things, and many Linux distributions use this approach. It does mean we have to look in those */etc/cron.** folders to see what's actually scheduled.

On your system type: *ls /etc/cron.daily* - you'll see something like this:

```
$ ls /etc/cron.daily
apache2  apt  aptitude  bsdmainutils  locate  logrotate  man-db  mlocate  standard
sysklog
```

Each of these files is a script or a shortcut to a script to do some regular task, and they're run in alphabetic order by *run-parts*. So in this case *apache2* will run first. Use *less* to view some of the

scripts on your system - many will look very complex and are best left well alone, but others may be just a few lines of simple commands.

Look at the articles in the resources section - you should be aware of *at* and *anacron* but are not likely to use them in a server. Google for "logrotate", and then look at the logs in your own server to see how they've been "rotated".

POSTING YOUR PROGRESS

List out the contents of your */etc/cron.daily* - redirecting it to */home/support/Day9*

RESOURCES

- Job scheduling with "cron" and "at"

EXTENSION

- Research *systemd* and how it may change the way scheduled tasks are done.

Day 10 - Finding things...

INTRO

For the last day of week two, we'll look at how you find files, and text inside these files, quickly and efficiently.

It can be very frustrating to know that a file or setting exists, but not be able to track it down! Master today's commands and you'll be much more confident as you administer your systems.

Today you'll look at four useful tools:

- locate
- find
- grep
- which

INSTRUCTIONS

locate

If you're looking for a file called *access.log* then the quickest approach is to use "locate" like this:

```
$ locate access.log
/var/log/apache2/access.log
/var/log/apache2/access.log.1
/var/log/apache2/access.log.2.gz
```

As you can see, by default it treats a search for *"something"* as a search for *"*something"*. It's very fast because it searches an index, but if this index is out of date or missing it may not give you the answer you're looking for. This is because the index is created by the *updatedb* command - typically run only nightly by *cron*. It may therefore be out of date for recently added files, but it can be worthwhile updating the index by manually running: *sudo updatedb*.

find

The *find* command searches down through a directory structure looking for files which match some criteria - which could be name, but also size, or when last updated etc. Try these examples:

```
find /var -name access.log
find /home -mtime -3
```

The first searches for files with the name "access.log", the second for any file under */home* with a last-modified date in the last 3 days.

These will take longer than *locate* did because they search through the filesystem directly rather from an index. Also, because *find* uses the the permissions of the logged-in user you'll get "permission denied" messages for many directories if you search the whole system. Starting the command with *sudo* of course will run it as *root*.

These examples are just the tip of a very large iceberg, check the articles in the RESOURCES section and work through as many examples as you can - time spent getting really comfortable with *find* is not wasted.

grep -r

Rather than asking "grep" to search for text within a specific file, you can give it a whole directory structure, and ask it to recursively search down through it. This trick is particularly handy when you "just know" that an item appears "somewhere" - but are not sure where.

As an example, you know that "PermitRootLogin" is an ssh parameter in a config file somewhere under /etc, but can't recall exactly where it is kept:

```
grep -r -i "PermitRootLogin" /etc/*
```

Because this only works on plain text files, it's most useful for the */etc* and */var/log* folders. (Notice the *-i* which makes the search "case insensitive", finding the setting even if it's been entered as "Permitrootlogin"

You may now have logs like */var/log/access.log.2.gz* - these are older logs that have been compressed to save disk space - so you can't read them with *less*, or search them with *grep*. However, there are *zless* and *zgrep*, which do work, and on ordinary as well as compressed files.

which

It's sometimes useful to know where a command is being run from. If you type *nano*, and it starts, where is the *nano* binary coming from? The general rule is that the system will search through the locations setup in your "path". To see this type:

```
echo $PATH
```

To see where *nano* comes from, type:

```
which nano
```

Try this for *grep*, *vi* and *service* and *reboot*. You'll notice that they're typically always in subfolders named *bin*, but that there are several different ones.

POSTING YOUR PROGRESS

Create and edit a file */home/support/feedback/Day10* with your comments on how this second week of the course has gone for you.

RESOURCES

- Using the find command
- 10 Tips for using "find"
- Five simple recipes for "grep"

EXTENSION

The "-exec" feature of the "find" command is extremely powerful. Test some examples of this from the RESOURCES links.

DAY 11 - Copying with SFTP

INTRO

You've now had a working Internet server of your own for some time, and seen how you can create and edit small files there. You've created a web server where you've been able to edit a simple web page.

Today we'll be looking at how you can move files between your other systems and this server - tasks like:

- Taking a copy of some files from your server onto your desktop machine
- Copying up some text to your server to put on your webpage
- Uploading some photos and logos for your webpage

PROTOCOLS

There are a wide range of ways a Linux server can share files, including:

- SMB: Microsoft's file sharing, useful on a local network of Windows machines
- AFP: Apple's file sharing, useful on a local network of Apple machines
- WebDAV: Sharing over web (http) protocols
- FTP: Traditional Internet sharing protocol
- scp: Simple support for copying files
- rsync: Fast, very efficient file copying
- SFTP: file access and copying over the SSH protocol (Despite the name, the SFTP protocol at a technical level is completely unrelated to traditional FTP)

Each of these have their place, but for copying files back and forth from your your local desktop to your server, SFTP has a number of key advantages:

- No extra extra setup is required on your server
- Top quality security
- Allows browsing through the directory structure
- You can create and delete folders

If you're successfully logging in via *ssh* from your home, work or a cybercafe then you'll also be able to use SFTP from this same location because the same underlying protocol is being used.

By contrast, setting up your server for any of the other protocols will require extra work. Not only that, enabling extra protocols also increases the "attack surface" - and there's always a chance that you'll mis-configure something in a way that allows an attacker in. It's also very likely that restrictive firewall policies at a workplace will interfere with or block these protocols. Finally, while old-style FTP is still very commonly used, it sends login credentials "in clear", so that your flatmates, cafe buddies or employer may be able to grab them off the network by "packet sniffing". Not a big issue

with your "classroom" server - but it's an unacceptable risk if you're remotely administering production servers.

SFTP client software

What's required to use SFTP is some client software. A command-line client (unsurprisingly called *sftp*) comes standard on every Apple OSX or Linux system. If you're using a Linux desktop, you also have a built-in GUI client via your file manager. This will allow you to easily attach to remote servers via SFTP. (For the Nautilus file manager for example, press ctrl + l to bring up the 'location window" and type: *sftp://username@myserver-address*).

Although Windows and Apple OSX have no built-in GUI client there are a wide range of third-party options available, both free and commercial. If you don't already have such a client installed, then choose one such as: * WinSCP or FileZilla - for Windows users * CyberDuck or FileZilla - for Mac OSX users

Download locations are under the RESOURCES section.

Configuring and using your choice of these should be straightforward. The only real potential for confusion is that these clients generally support a wide range of protocols such as scp and FTP that we're not going to use. When you're asked for SERVER, give your server's IP address, PORT will be 22, and PROTOCOL will be SFTP or SSH.

INSTRUCTIONS

- Configure your chosen SFTP client to login to your server as "support"
- Copy some files from your server down to your local desktop (try files from your */home/support* folder, and from */var/log*)
- Create an "*images*" folder under your server's */home/support folder*, and upload some images to it from your desktop machine
- Go up to the root directory. You should see */etc*, */bin* and other folders. Try to create an "*images*" folder here too - this should fail because you are logging in as "support", so there are files you won't be able to download, and places where you won't have permission to create new files or folders. In your own "home" directory you of course have full permission.

Once the files are uploaded you can login via *ssh* and use *sudo* to give yourself the necessary power to move files about.

POSTING YOUR PROGRESS

- Make sure you've created a */home/support/images* folder with some image files in it.

RESOURCES

- CyberDuck (http://cyberduck.ch/)

- FileZilla (http://filezilla-project.org/download.php?type=client)
- Reason to use SFTP over FTP or SCP (http://sysadminspot.com/linux/sftp-vs-ftp-vs-scp-advantages/)
- sftp File From One Server To Another (http://www.cyberciti.biz/faq/sftp-file-from-server-to-another-in-unix-linux/)

DAY 12 - Who has permission?

INTRO

Files on a Linux system always have associated "permissions" - controlling who has access and what sort of access. You'll have bumped into this in various ways already - as an example, yesterday while logged in as "support" you could not upload files directly into */var/www* or create a new folder at */*.

The Linux permission system is quite simple, but it does have some quirky and subtle aspects, so today is simply an introduction to some of the basic concepts.

This time you really *do* need to work your way through the material in the RESOURCES section!

OWNERSHIP

First let's look at "ownership". All files are tagged with both the name of the user and the group that owns them, so if we type "ls -l" and see a file listing like this:

```
-rw-------  1 steve  staff      4478979  6 Feb  2011 private.txt
-rw-rw-r--  1 steve  staff      4478979  6 Feb  2011 press.txt
-rwxr-xr-x  1 steve  staff      4478979  6 Feb  2011 upload.bin
```

Showing that all these files are owned by user "steve", and the group "staff".

PERMISSIONS

Looking at the '-rw-r--r--" at the start of a directory listing line, ignore the first "-" for now, and see these as potentially three groups of "rwx": the permission granted to the user who owns the file, the "group", and "other people".

For the example list above:

- *private.txt* - Steve has "rw" (ie Read and Write) permission, but neither the group "staff" nor "other people" have any permission at all
- *press.txt* - Steve can Read and Write to this file too, but so can any member of the group "staff" - and anyone can read it
- *load.bin* - Steve can write to the file, all others can read it. Additionally all can "execute" the file - ie run this program

You can change the permissions on any file with the *chmod* utility. Create a simple text file in your home directory with the nano editor (e.g. *tuesday.txt*) and check that you can list its contents by typing: *cat tuesday.txt* or *less tuesday.txt*.

Now look at its permissions by doing: *ls -ltr tuesday.txt*

```
-rw-r--r-- 1 support support   12 Nov 19 14:48 tuesday.txt
```

So, the file is owned by the user "support", who is the only one that can write to the file, but any member of the group "support" can read it - as in fact can any other user.

Now let's remove the permission of the user to write their own file:

```
chmod u-w tuesday.txt
```

...and remove the permission for "others" to read the file:

```
chmod o-r tuesday.txt
```

Do a listing to check the result:

```
-r--r----- 1 support support   12 Nov 19 14:48 tuesday.txt
```

...and confirm by trying to edit the file with *nano* or *vim*. You'll find that you appear to be able to edit it - but can't save any changes. You can of course easily give yourself back the permission to write to the file by:

```
chmod u+w tuesday.txt
```

GROUPS

On most modern Linux systems there is a group created for each user, so user "support" is a member of the group "support". However, groups can be added as required, and users added to several groups.

To see what groups you're a member of, simply type: *groups*

On an Ubuntu system 'support' should be a member of: support, sudo and adm - and if you list the */var/log* folder you'll see your membership of the "adm" group is why you can use *less* to view the contents of */var/log/auth.log*

The "root" user can add a user to an existing group with the command:

```
usermod -a G group user
```

so your "support" user can do the same simply by prefixing the command with "sudo" - use this to add your support user to the group "root", and then use the "group" command to check that it has

worked.

POSTING YOUR PROGRESS

Just for fun, create a file: *secret.txt* in your home folder, take away all permissions from it for the user, group and others - and see what happens when you try to edit it with *nano*.

RESOURCES

- http://tldp.org/LDP/intro-linux/html/sect_03_04.html
- http://catcode.com/teachmod/
- http://www.trainsignal.com/blog/linux-file-permissions
- http://www.youtube.com/watch?v=vKTg1ATHI4E

EXTENSION

- Research *umask* and test to see how it's setup on your server
- Research and get familiar with the classic *octal* mode of describing and setting file permissions. (e.g. *chmod 664 myfile*)

Day 13 - Your first staff member...

INTRO

Today you're going to set-up another user on your system. You're going to imagine that this is your first employee, trusted to do just a few simple tasks:

- check that the system is running
- check disk space with: df -h

...but you also want them to be able to reboot the system, because you believe that "turning it off and on again" resolves most problems :-)

You'll be covering a several new areas, so have fun!

ADDING A USER

Choose a name for your new user - we'll use "fred" in the examples, so to add this new user:

```
sudo adduser fred
```

(Names are case-sensitive in Linux, so "Fred" would be a completely different user)

The "adduser" command works very slightly differently in each distro - if it didn't ask you for a password for your new user, then set it manually now by:

```
sudo passwd fred
```

You will now have a new entry in the simple text database of users: */etc/passwd* (check it out with: *less*), and a group of the same name in the file: */etc/group*. The encrypted password for the user is in: */etc/shadow* (you can read this too if you use "sudo" - check the permissions to see how they're set. For obvious reasons it's not readable to just everyone).

If you're used to other operating system it may be hard to believe, but these simple text files are the whole Linux user database and you could even create your users and groups by directly editing these files - although this isn't normally recommended.

Additionally, *adduser* will have created a home directory, */home/fred* for example, with the correct permissions.

Login as your new user to confirm that everything works. Now while logged in as this user try to run *reboot* - then *sudo reboot*.

CLEVER SUDO TRICKS

Your new user is just an ordinary user and so can't use *sudo* to run commands with elevated privileges - until we set them up. We could simply add them to a group that's pre-defined to be able to use sudo to do *anything* as root - but we don't want to give 'fred" quite that amount of power.

Use *ls -l* to look at the permissions for the file: */etc/sudoers* This is where the magic is defined, and you'll see that it's tightly controlled, but you should be able to view it with: *sudo less /etc/sudoers* You want to add a new entry in there for your new user, and for this you need to run a special utility: *visudo*

To run this, you can temporarily "become root" by running:

```
sudo su -
```

Notice that your prompt has changed to a "#"

Now simply run *visudo* to safely edit */etc/sudoers* with your normal editor - in this case *nano*.

All lines in */etc/sudoers* beginning with "#" are optional comments. You'll want to add some lines like this:

```
# Allow user "fred" to run "sudo reboot"
# ...and don't prompt for a password
#
fred ALL = NOPASSWD:/sbin/reboot
```

You can add these line in wherever seems reasonable. The *visudo* command will automatically check your syntax, and won't allow you to save if there are mistakes - because a corrupt sudoers file could lock you out of your server!

Type *exit* to remove your magic hat and become simple "support" again - and notice that your prompt again becomes: $

TESTING

Test by logging in as your test user and typing: *sudo reboot*

POSTING YOUR PROGRESS

Send an email to tutor@GREP101.com to let us know how these last two lessons went for you.

EXTENSION

If you find this all pretty familiar, then you might like to check and update your knowledge on a

couple of related areas:

- Restricting shell access (http://www.cyberciti.biz/tips/howto-linux-shell-restricting-access.html)
- The history of /etc/passwd and /etc/shadow (http://linuxers.org/article/history-etcpasswd-and-etcshadow-files)
- Linux user login management (https://www.ibm.com/developerworks/mydeveloperworks/blogs/58e72888-6340-46ac-b488-d31aa4058e9c/entry/the_linux_user_login_management_etc_passwd_and_etc_shadow_files19?lang=en)

RESOURCES

- Advanced /etc/sudoers examples (http://www.gratisoft.us/sudo/sample.sudoers)
- A cartoon that should now make sense! (http://xkcd.com/149/)
- Basic Linux Permissions: sudo and sudoers (http://www.youtube.com/watch?v=YSSIm0g00m4) (video)

Day 14 - Editing with *vim*

INTRO

Simple text files are at the heart of Linux, so editing these is a key sysadmin skill. There are a range of simple editors aimed at beginners such as: *nano*, *pico*, *joe* or *jed* . These all look horribly ugly, and as if they were written for DOS back in the 1980's - but are pretty easy to "just figure out".

The Real Sysadmin however, uses *vi* - this is the editor that's always installed - and today you'll get started using it.

Bill Joy wrote *vi* back in the mid 1970's - and even the "modern" descendant *vim* that we'll concentrate on is over 20 years old, but despite their age, these remain the standard editors on command-line server boxes. Additionally, they have a loyal following among programmers, and even some writers.

Very often when you type *vi*, what the system actually starts is *vim*. To see if this is true of your system type:

```
vi --version
```

to check.

THE TWO THINGS YOU NEED TO KNOW

- There are two "modes" - with very different behaviours
- Little or nothing onscreen lets you know which mode you're currently in!

The two modes are "command mode" and "editing mode", and as a beginner, there is one simple technique to remember - simply:

"Press Esc twice or more to return to command mode"

INSTRUCTIONS

So, first grab a text file to edit. A copy of */etc/services* will do nicely:

```
cd
pwd
cp -v /etc/services testfile
vim testfile
```

At this point we have the file on screen, and we are in "command mode". Unlike *nano*, however, there’s no onscreen menu and it's not at all obvious how anything works!

Start by pressing *Esc* once or twice to ensure that we are in command mode (remember this trick from above), then type *:q!* and press *Enter*. This quits without saving any changes - a *vital* first skill when you don't yet know what you're doing! Now let's go in again and play around, seeing how powerful and dangerous *vim* is - then again, quit without saving:

```
vim testfile
```

Use the keys *h j k* and *l* to move around (this is the traditional *vi* method) then try using the arrow keys - if these work, then feel free to use them - but remember those *hjkl* keys though because one day you may be on a system with just the traditional *vi* and the arrow keys won't work.

Now play around moving through the file. Then exit with *Esc Esc :q!* as discussed earlier.

Now that you've mastered that, lets get more advanced.

```
vim testfile
```

This time, move down a few lines into the file and press *3* then *3* again, then *d* and *d* again - and suddenly 33 lines of the file are deleted!

Why? Well, you are in command mode and *33dd* is a command that says "delete 33 lines". Now, you're still in command mode, so press *u* - and you've magically undone the last change you made. Neat huh?

Now you know the three basic tricks for a newbie to *vim*:

- *Esc Esc* always gets you back to "command mode"
- From command mode *:q!* will always quit without saving anything you've done, and
- From command mode *u* will undo the last action

So, here's some useful, productive things to do:

- Finding things: From command mode, type *gg* to get to the top of the file, then *GG* to get to the bottom. Let's search for references to "sun", type */sun* to find the first instance, then press *n* repeatedly to step through all the next occurrences. Now go to the top of the file (*gg* remember) and try searching for "*Apple*" or "*Microsoft*".
- Cutting and pasting: Go back up to the top of the file (with *gg*) and look at the first few lines of comments (the ones with "#" as the first character. Play around with cutting some of these out and pasting them back. To do this simply position the cursor on a line, then (for example), type *11dd* to delete 11 lines, then immediately paste them back in by pressing *P* - and then move down the file a bit and paste the same 11 lines in there again with *P*
- Inserting text: Move anywhere in the file and press *i* to get into "insert mode" (it may show at the bottom of the screen) and start typing - and *Esc Esc* to get back into command mode when you're done.

- Writing your changes to disk: From command mode type *:w* or *:wq* to “write and quit”.

This is much as you ever *need* to learn about *vi* - but there's an enormous amount more you could learn if you had the time - just don't bother going there until after you've finished this course!

One last thing, you may see reference to *"vi versus emacs"*. This is a long running argument for programmers, not system administrators - *vi/vim* is what you need to learn.

POSTING YOUR PROGRESS

Create and edit (with vim!) a file: */home/support/done-with-vim*

EXTENSION

If you're already familiar with *vi* / *vim* then use today's hour to research and test some customisation via your *~/.vimrc* file. Here are a couple of useful advanced resources on this:

- http://amix.dk/vim/vimrc.html
- http://gaveen.owain.org/2009/07/my-vim-configuration.html

RESOURCES

- Here is why *vim* uses the *hjkl* keys as arrow keys (http://www.catonmat.net/blog/why-vim-uses-hjkl-as-arrow-keys/)
- Graphical vi-vim Cheat Sheet and Tutorial (http://www.viemu.com/a_vi_vim_graphical_cheat_sheet_tutorial.html)
- Vi - Vim Tutorial (http://www.youtube.com/watch?v=71YTkxUNwmg) (video)
- Vim for Editing Files in Linux (http://www.youtube.com/watch?v=ImK_dHPOTIE&playnext=1&list=PLmE8MrZA_nN4W28CoLgsEZ5lUrwSxYSHX&feature=results_video) (video)

Y 15 - Deeper into repositories...

INTRO

Early on you installed some software packages to your server using *apt-get*. That was fairly painless, and we explained how the Linux model of software installation is very similar to how "app stores" work on Android, iPhone, increasingly in Mac OSX - and now Windows 8.

Today however, you'll be looking "under the covers" to see how this works; better understand the advantages (and disadvantages!) - and to see how you can safely extend the system beyond the main official sources.

REPOSITORIES AND VERSIONS

Any particular Linux installation has a number of important characteristics:

- Version - e.g. Ubuntu 9.04, CentOS 5, RHEL 6
- "Bit size" - 32-bit or 64-bit
- Chip - Intel, ARM, PowerPC

The version number is particularly important because it controls the versions of application that you can install. When Ubuntu 9.04 was released (in April 2009 - hence the version number!), it came out with Apache 2.2.11. So, if your server runs 9.04, then even if you installed Apache with *apt-get* five years later that is still the version you would receive. This provides stability, but at an obvious cost for web designers who hanker after some feature which later versions provide. (Security patches *are* made to the repositories, but by "backporting" security fixes from later versions into the old stable version that was first shipped).

WHERE IS ALL THIS SETUP?

We'll be discussing the "package manager" used by the Debian and Ubuntu distributions, and dozens of derivatives. This uses the *apt-get* command, but for most purposes the competing *yum* command used by Fedora, RHEL, CentOS and Scientific Linux works in a very similar way - as do the equivalent utilities in other versions.

The configuration is done with files under the */etc/apt* directory, and to see where the packages you install are coming from, use *less* to view */etc/apt/sources.list* where you'll see lines that are clearly specifying URLs to a “repository” for your specific version:

```
deb http://archive.ubuntu.com/ubuntu precise-security main restricted universe
```

There's no need to be concerned with the exact syntax of this for now, but what’s fairly common is to want to add extra repositories - and this is what we'll deal with next.

EXTRA REPOSITORIES

While there's an amazing amount of software available in the "standard" repositories (more than 3,000 for CentOS and ten times that number for Ubuntu), there are often packages not available - typically for one of two reasons:

- Stability - CentOS is based on RHEL (Red Hat Enterprise Linux), which is firmly focussed on stability in large commercial server installations, so games and many minor packages are not included
- Ideology - Ubuntu and Debian have a strong "software freedom" ethic (this refers to freedom, not price), which means that certain packages you may need are unavailable by default

So, next you'll adding an extra repository to your system, and install software from it.

ENABLING EXTRA REPOSITORIES

First do a quick check to see how many packages you *could* already install. You can get the full list and details by running:

```
apt-cache dump
```

...but you'll want to press Ctrl-c a few times to stop that, as it's far too long-winded.

Instead, filter out just the packages names using *grep*, and count them using: *wc -l* (*wc* is "word count", and the "-l" makes it count lines rather than words) - like this:

```
apt-cache dump | grep "Package" | wc -l
```

These are all the packages you could now install - but you're now going to make even more available by adding the "Universe" and "Multiverse" repositories - if they are not already installed. These are hosted at Ubuntu, but with less support and Multiverse: *"contains software which has been classified as non-free ...may not include security updates"*. Examples of useful tools in Multiverse include the compression utilities *rar* and *lha*, and the network performance tool *netperf*.

So, check to see whether you can install *rar* (a file compression utility) by typing:

```
sudo apt-get install lha
```

Now enable the "Multiverse" repository, following the guides at: * Repositories/Ubuntu (Community wiki) (https://help.ubuntu.com/community/Repositories/Ubuntu)

- Ubuntu Server Guide (https://help.ubuntu.com/12.04/serverguide/configuration.html)

After adding this, update your local cache of available applications:

```
sudo apt-get update
```

Once done, try again to install *rar*.

EXTENSION - Ubuntu PPAs

Ubuntu also allows users to register an account and setup software in a Personal Package Archive (PPA) - typically these are setup by enthusiastic developers, and allow you to install the latest "cutting edge" software.

Imagine that your server is a development box for a service to allow video files to be uploaded and automatically converted in some way. The plan is to use the command-line tool of the "HandBrake" video transcoder. This is not included in any of the standard repositories - but further, you want to take advantage of new features not yet available in the standard released version - so you'll enable a PPA for "daily builds" of Handbrake. (John Stebbins maintains this at *https://launchpad.net/~stebbins/+archive/handbrake-snapshots*).

Do this with the the following steps:

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:stebbins/handbrake-snapshots
```

Now update your local cache of available applications:

```
sudo apt-get update
```

...and try installing the command-line version of HandBrake:

```
sudo apt-get install handbrake-cli
```

When you next run "sudo apt-get upgrade" you'll likely be prompted to install a new version of handbrake-cli - because the developers are literally making changes every day. (And if it's not obvious, when the developers have a bad day your software will stop working until they make a fix - that's the real "cutting edge"!)

SUMMARY

Installing only from the default repositories is clearly the safest, but there are often good reasons for going beyond them. As a sysadmin you need to judge the risks, but in the example we came up with a realistic scenario where connecting to an unstable working developer's version made sense.

As general rule however you:

- Will seldom have good reasons for hooking into more than one or two extra repositories
- Need to read up about a repository first, to understand any potential disadvantages.

POSTING YOUR PROGRESS

Make sure that you install *lha* - we've set our “magic robot script” to check your server for it.

RESOURCES

- How to use yum - Introduction (http://fedoranews.org/tchung/howto/2003-11-09-yum-intro.shtml)
- Package management with APT (https://help.ubuntu.com/community/AptGet/Howto)
- What do you mean by Free Software? (http://www.debian.org/intro/free)

Day 16 - tar and friends...

INTRO

As a system administrator, you need to be able to confidently work with compressed “archives” of files. In particular two of your key responsibilities; installing new software, and managing backups, often require this.

CREATING ARCHIVES

On other operating systems, applications like WinZip, and pkzip before it, have long been used to gather a series of files and folders into one compressed file - with a .zip extension. Linux takes a slightly different approach, with the "gathering" of files and folders done in one step, and the compression in another.

So, you could create a "snapshot" of the current files in your */etc/init.d* folder like this:

```
tar  -cvf  myinits.tar  /etc/init.d/
```

This creates *myinits.tar* in your current directory.

Note 1: The *-v* switch (verbose) is included to give some feedback - traditionally many utilities provide no feedback unless they fail. Note 2: The *-f* switch specifies that *“the output should go to to the filename which follows”* - so in this case the order of the switches is important.

(The cryptic “tar” name? - originally short for "tape archive")

You could then compress this file with GnuZip like this:

```
gzip myinits.tgz myinits.tar
```

A compressed tar archive like this is known as a "tarball". Although we've given the output filename a *.tgz* extension, this doesn't have any meaning to the system but is helpful to humans.

In practice you can do the two steps in one with the "-z" switch, like this:

```
tar  -cvzf  myinits.tgz  /etc/init.d/
```

This uses the *-c* switch to say that we're creating an archive; *-v* to make the command "verbose"; *-z* to compress the result - and *-f* to specify the output file.

TASKS FOR TODAY

- Check the links under "Resources" to better understand this - and to find out how to extract files from an archive!
- Use *tar* to create an archive copy of some files and check the resulting size
- Run the same command, but this time use *-z* to compress - and check the file size
- Copy your archives to */tmp* (with: *cp*) and extract each there to test that it works

POSTING YOUR PROGRESS

Nothing to post today - but make sure you understand this stuff, because we'll be using it for real in the next day's session!

RESOURCES

- How Can I Do Archiving With Tar? (http://lowfatlinux.com/linux-tar.html)
- Linux TAR Command (http://linuxbasiccommands.wordpress.com/2008/04/04/linux-tar-command/)
- Linux tar command tutorial (https://www.youtube.com/watch?v=CUdwDEKlDrw) (video)

EXTENSION

- What is a .bz2 file - and how would you extract the files from it?
- Research how absolute and relative paths are handled in tar - and why you need to be careful extracting from archives when logged in as root
- You might notice that some tutorials write "tar cvf" rather than "tar -cvf" with the switch character - do you know why?

Day 17 - From the source

INTRO

A few days ago we saw how to authorise extra repositories for *apt-get* to search, when we need unusual applications, or perhaps more recent versions than those in the standard repositories.

Today we're literally going to "go to the source". This is not something to be done lightly - the whole reason for package managers is to make your life easy - but occasionally it is justified, and it is something you need to be aware of and comfortable with.

The applications we've been installing up to this point have come from repositories. The files there are "binaries" - pre-compiled, and often customised by your distro. What might not be clear is that your distro gets these applications from a diverse range of un-coordinated development projects (the "upstream"), and these developers are continuously working on new versions. We'll go to one of these, download the source, compile and install it.

FIRST WE NEED THE ESSENTIALS

Projects normally provide their applications as "source files", written in the C, C++ or other computer languages. We're going to pull down such a source file, but it won't be any use to us until we compile it into an "executable" - a program that our server can execute. So, we'll need to first install a standard bundle of common compilers and similar tools. On Ubuntu, the package of such tools is called "build-essential", so install it like this:

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

GETTING THE SOURCE

First, test that you already have *nmap* installed, and type *nmap -V* to see what version you have. This is the version installed from your standard repositories. Next, type: *which nmap* - to see where the executable is stored.

Now let's go to the "Project Page" for the developers *http://nmap.org/* and grab the very latest cutting-edge version. Look for the download page, then the section "Source Code Distribution" and the link for the "Latest development nmap release tarball" and note the URL for it - something like:

```
http://nmap.org/dist/nmap-5.61TEST2.tar.bz2
```

This is version 5.61TEST2, the latest development release when these notes were originally written, but it will be different now. So now we'll pull this down to your server. The first question is where to put it - we'll put it in your home directory, so change to your home directory with:

```
cd
```

then simply using *wget* ("web get"), to download the file like this:

```
wget -v http://nmap.org/dist/nmap-5.61TEST2.tar.bz2
```

The -v (for verbose), gives some feedback so that you can see what's happening. Once it's finished, check by listing your directory contents:

```
ls -ltr
```

As we've learnt, the end of the filename is typically a clue to the file's format - in this case ".bz2" signals that it's a tarball compressed with the bz2 algorithm. While we could uncompress this then un-combine the files in two steps, it can be done with one command - like this:

```
tar -j -x -v -f   nmap-5.61TEST2.tar.bz2
```

....where the -j means "uncompress a bz2 file first", -x is extract, -v is verbose - and -f says "the filename comes next". Normally we'd actually do this more concisely as:

```
tar -jxvf   nmap-5.61TEST2.tar.bz2
```

So, lets see the results,

```
ls -ltr
```

Remembering that directories have a leading "d" in the listing, you'll see that a directory has been created :

```
drwxr-xr-x 20 steve  steve     4096 2011-10-01 06:06 nmap-5.61TEST2
-rw-r--r--  1 steve  steve  21633731 2011-10-01 06:46 nmap-5.61TEST2.tar.bz2
```

Now explore the contents of this with *mc* or simply *cd nmap-5.61TEST2* - you should be able to find and read the actual source code. Even if you know no programming, the comments can be entertaining reading.

By convention, source files will typically include in their root directory a series of text files in uppercase such as: README and INSTALLATION. Look for these, and read them using *more* or *less*. It's important to realise that the programmers of the "upstream" project are not writing for Ubuntu, CentOS - or even Linux. They have written a correct working program in C or C++ etc and made it

available, but it's up to us to figure out how to compile it for our operating system, chip type etc. (This hopefully gives a little insight into the value that distributions such as CentOS, Ubuntu and utilities such as *apt-get* etc add, and how tough it would be to create your own Linux From Scratch)

So, in this case we see an INSTALL file that says something terse like:

```
Ideally, you should be able to just type:

./configure
make
make install

For far more in-depth compilation, installation, and removal notes
read the Nmap Install Guide at http://nmap.org/install/ .
```

In fact, this is fairly standard for many packages. Here's what each of the steps does:

- *./configure* - is a script which checks your server (ie to see whether it's ARM or Intel based, 32 or 64-bit, which compiler you have etc). It can also be given parameters to tailor the compilation of the software, such as to not include any extra support for running in a GUI environment - something that would make sense on a "headless" (remote text-only server), or to optimize for minimum memory use at the expense of speed - as might make sense if your server has very little RAM. If asked any questions, just take the defaults - and don't panic if you get some WARNING messages, chances are that all will be well.
- *make* - compiles the software, typically calling the GNU compiler *gcc*. This may generate lots of scary looking text, and take a minute or two - or as much as an hour or two for very large packages like LibreOffice.
- *make install* - this step takes the compiled files, and installs them into the correct places, installs documentation and other similar tasks. Until now you've just been playing in your home directory, but this step installs for all users, so requires *root* privileges so you'll need to actually run: *sudo make install*. If asked any questions, just take the defaults.

Now, potentially this last step will have overwritten the *nmap* you already had, but more likely this new one has been installed into a different place.

In general */bin* is for key parts of the operating system, */usr/bin* for less critical utilities and */usr/local/bin* for software you've installed yourself. When you type a command it will search through each of the directories given in your PATH environment variable, and start the first match. So, if */bin/nmap* exists, it will run instead of */usr/local/bin* - but if you give the "full path" to the version you want - such as */usr/local/bin/nmap* - it will run that version instead.

The “locate” command allows very fast searching for files. Install it by:

```
sudo apt-get install locate
```

We'll use this to check which “nmap” commands we have and where they’re located. To update the index of files:

```
sudo updatedb
```

Then to search the index:

```
sudo locate bin/nmap
```

This should find both your old and copies of *nmap*

Now try running each, for example:

```
/usr/bin/nmap -V

/usr/local/bin/nmap -V
```

Type *set* to see all your "environmental variables" - and look for the PATH variable - that's the order in which directories are searched for executables. For example:

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

The *nmap* utility relies on no other package or library, so is very easy to install from source. Most other packages have many "dependencies", so installing them from source by hand can be pretty challanging even when well explained (look at: http://oss.oetiker.ch/smokeping/doc/smokeping_install.en.html for a good example).

POSTING YOUR PROGRESS

Email tutor@GREP101.com with your comments on today's session.

##RESOURCES

- Understanding software Installation (configure, make, make install) (http://www.codecoffee.com/tipsforlinux/articles/27.html)
- Installing From Tarballs (http://linux.byexamples.com/archives/156/installing-from-tarballs/)
- How to rebuild an existing package from source (http://raphaelhertzog.com/2010/12/15/howto-to-rebuild-debian-packages/)
- Compiling things on Ubuntu the Easy Way (https://help.ubuntu.com/community/CompilingEasyHowTo)

EXTENSION

Research some distributions where “from source” is normal:

- What is Linux From Scratch? (http://www.linuxfromscratch.org/lfs/)
- What is Gentoo? (http://www.gentoo.org/main/en/about.xml)
- The Arch Build System (https://wiki.archlinux.org/index.php/Arch_Build_System)

Day 18 - Log rotation

INTRO

When you're administering a remote server, logs are your best friend, but disk space problems can be your worst enemy - so while Linux applications are generally very good at generating logs, they need to be controlled.

The *logrotate* application keeps your logs in check. Using this, you can define how many days of logs you wish to keep; split them into manageable files; compress them to save space, or even keep them on a totally separate server.

Good sysadmins love automation - having the computer automatically do the boring repetitive stuff Just Makes Sense.

##ARE YOUR LOGS ROTATING?

Look into your logs directories - */var/log*, and subdirectories like */var/log/apache2*. Can you see that your logs are already being rotated? You should see a */var/log/syslog* file, but also a series of older compressed versions with names like */var/log/syslog.1.gz*

##WHEN DO THEY ROTATE?

You will recall that *cron* is generally setup to run scripts in */etc/cron.daily* - so look in there and you should see a script called *logrotate* - or possibly *00logrotate* to force it to be the first task to run.

##CONFIGURING LOGROTATE

The overall configuration is set in */etc/logrotate.conf* - have a look at that, but then also look at the files under the directory */etc/logrotate.d*, as the contents of these are merged in to create the full configuration. This is example of what you might see: /var/log/apache2/*.log { weekly missingok rotate 52 compress delaycompress notifempty create 640 root adm }

Much of this is fairly clear: the apache2 .log file will be rotated each week, with 52 compressed copies being kept.

Typically when you install an application a suitable logrotate "recipe" is installed for you, so you'll not normally be creating these from scratch. However, the default settings won't always match your requirements, so it's perfectly reasonable for you as the sysadmin to edit these - for example, the default *apache2* recipe above creates 52 weekly logs, but you might find it more useful to have logs rotated daily, a copy automatically emailed to an auditor, and just 30 days worth kept on the server.

##RTFM

This is a good time to mention that Linux comes with a fine on-line manual - invoked with the *man*

command. Each application you install also installs its own page into this manual, so that you can look at the page for *logrotate* to see the full detail on the syntax like this:

```
man logrotate
```

You might also try:

```
man nmap
man tar
man cp
man grep
man man
man kill
```

As you'll see, these are excellent for the detailed syntax of a command, but many are extremely terse, and for others the amount of detail can be somewhat daunting!

##YOUR TASK TODAY

- Edit your logrotate configuration for *apache2* to rotate daily
- Make whatever other changes you wish
- Check the next day to see that it's worked

##RESOURCES

- The Ultimate Logrotate Command Tutorial (http://www.thegeekstuff.com/2010/07/logrotate-examples/)
- Howto: Use logrotate to manage log files (http://linuxers.org/howto/howto-use-logrotate-manage-log-files)
- LINUX: openSUSE and logrotate (http://www.youtube.com/watch?v=UoHmj3ef3Is)
- Use logrotate to Manage Log Files (http://library.linode.com/linux-tools/utilities/logrotate)

Day 19 - Inodes, symlinks and stat

INTRO

Today's topic looks gives a peek “under the covers” at the technical detail of how files are stored.

Linux supports a large number of different “filesystems” - although on a server you’ll typically be dealing with just *ext3* or *ext4* and perhaps *btrfs* - but today we’ll not be dealing with any of these but instead with the layer of Linux that sits *above* all of these - the Linux Virtual Filesystem.

The VFS is a key part of the Linux, and an overview of it and some of the surrounding concepts is very useful in confidently administering a system.

THE NEXT LAYER DOWN

Linux has an extra layer between the filename and the file's actual data on the disk - this is the *inode*. This has a numerical value which you can see most easily in two ways:

The *-i* switch on the *ls* command:

```
ls -li /etc/hosts
35356766 -rw------- 1 root root 260 Nov 25 04:59 /etc/hosts
```

The *stat* command:

```
stat /etc/hosts
File: `/etc/hosts'
Size: 260          Blocks: 8       IO Block: 4096   regular file
Device: 2ch/44d    Inode: 35356766 Links: 1
Access: (0600/-rw-------)  Uid: (  0/  root)   Gid: (  0/  root)
Access: 2012-11-28 13:09:10.000000000 +0400
Modify: 2012-11-25 04:59:55.000000000 +0400
Change: 2012-11-25 04:59:55.000000000 +0400
```

Every file name "points" to an inode, which in turn points to the actual data on the disk. This means that several filenames could point to the same inode - and hence have exactly the same contents. In fact this is a standard technique - called a "hard link". The other important thing to note is that when we view the permissions, ownership and dates of filenames, these attributes are actually kept at the inode level, *not* the filename. Much of the time this distinction is just theoretical, but it can be very important.

TWO SORTS OF LINKS

Work through the steps below to get familiar with hard and soft linking:

First move to your home directory with:

```
cd
```

Then use the *ln* ("link") command to create a “hard link”, like this:

```
ln /etc/passwd link1
```

and now a "symbolic link" (or “symlink”), like this:

```
ln -s /etc/passwd link2
```

Now use *ls -li* to view the resulting files, and *less* or *cat* to view them.

Both hard and symlinks are widely used in Linux, but symlinks are especially common - for example:

```
ls -ltr /etc/rc2.d/*
```

This directory holds all the scripts that start when your machine changes to “runlevel 2” (its normal running state) - but you'll see that in fact most of them are symlinks to the real scripts in */etc/init.d*

It's very common to have something like :

```
prog
prog-v3
prog-v4
```

where the program "prog", is a symlink - originally to v3, but now points to v4 (and could be pointed back if required)

Read up in the links provided, and test on your server to gain a better understanding. In particular, see how permissions and file sizes work with symbolic links versus hard links or simple files

The Differences

Hard links:

- Only link to a file, not a directory
- Can't reference a file on a different disk/volume
- Links will reference a file even if it is moved
- Links reference inode/physical locations on the disk

Symbolic (soft) links:

- Can link to directories
- Can reference a file/folder on a different hard disk/volume
- Links remain if the original file is deleted
- Links will NOT reference the file anymore if it is moved
- Links reference abstract filenames/directories and NOT physical locations.
- They have their own inode

Resources

- Hard and soft links (http://linuxgazette.net/105/pitcher.html)
- What's an inode? (http://www.linux-mag.com/id/8658/)
- UNIX / Linux filesystem Inodes (http://www.cyberciti.biz/tips/understanding-unixlinux-filesystem-inodes.html) Linux symbolic (soft) and hard links

Extension

- Anatomy of the Linux file system (http://www.ibm.com/developerworks/linux/library/l-linux-filesystem/)

Day 20 - Scripting

INTRO

Today is the final session for the course.

You've seen that a continual emphasis for a sysadmin is to automate as much as possible, and also how in Linux the system is very "transparent" - once you know where to look!

Today, on this final session for the course, we'll cover how to write small programs or "shell scripts" to help manage your system.

When typing at the Linux command-line you're directly communicating with "the command interpreter", also known as "the shell". Normally this shell is *bash*, so when you string commands together to make a script the result can be called either a '"shell script", or a "bash script".

Why make a script rather than just typing commands in manually?

- It saves typing. Remember when we searched through the logs with a long string of grep, cut and sort commands? If you need to do something like that more than a few times then turning it into a script saves typing - and typos!
- Parameters. One script can be used to do several things depending on what parameters you provide
- Automation. Pop your script in */etc/cron.daily* and it will run each day, or install a symlink to it in the appropriate */etc/rc.d* folder and you can have it run each time the system is shut down or booted up.

START WITH A SHABANG!

Scripts are just simple text files, but if you set the "execute" permissions on them then the command interpreter will look for a special line starting with the two characters "#" and "!" - referred to as the "shabang" (or "crunchbang") at the top of the file.

This line typically looks like this:

```
#!/bin/bash
```

Normally anything starting with a "#" character would be treated as a comment, but in the first line and followed by a "!", it's interpreted as: *"please feed the rest of this to the /bin/bash program, which will interpret it as a script"*. All of our scripts will be written in the *bash* language - the same as you've been typing at the command line throughout this course - but scripts can also be written in many other "scripting languages", so a script in the Perl language might start with *#!/usr/bin/perl*

YOUR FIRST SCRIPT

You'll write a small script to list out who's been most recently unsuccessfully trying to login to your server, using the entries in */var/log/auth.log*.

Use *vim* to create a file, *attacker*, in your home directory with this content:

```
#!/bin/bash
#
#   attacker - prints out the last failed login attempt
#
echo "The last failed login attempt came from IP address:"
grep "Failed password" /var/log/auth.log|tail -1 | cut -d: -f 4 | cut -d" " -f9
```

Putting comments at the top of the script like this isn't strictly necessary (the computer ignores them), but it's a good professional habit to get into.

To make it executable type:

```
chmod +x attacker
```

Now to run this script, you just need to refer to it by name - but the current directory is (deliberately) not in your $PATH, so you need to do this either of two ways:

```
/home/support/attacker
./attacker
```

Once you're happy with a script, and want to have it easily available, you'll probably want to move it somewhere on your $PATH - and */usr/local/bin* is a normally the appropriate place, so try this:

```
sudo mv attacker /usr/local/bin/attacker
```

...and now it will Just Work whenever you type *attacker*

EXTENDING THE SCRIPT

You can expand this script so that it requires a parameter and prints out some syntax help when you don't give one. There are a few new tricks in this, so it's worth studying:

```
#!/bin/bash
#
##     topattack - list the most persistent attackers
#
if [ -z "" ]; then
echo -e "\nUsage: `basename
```

```
   #!/bin/bash # ## topattack - list the most persistent attackers # if [ -z "$1"
  ]; then echo -e "\nUsage: `basename $0` <num> - Lists the top <num> attackers
  by IP" exit 0 fi echo " " echo "Persistant recent attackers" echo " " echo
  "Attempts IP " echo "----------------------" grep "Failed password for root"
  /var/log/auth.log|cut -d: -f 4 | cut -d" " -f7|sort |uniq -c |sort -nr |head
  -$1

` <num> - Lists the top <num> attackers by IP"
 exit 0
 fi
 echo " "
 echo "Persistant recent attackers"
 echo " "
 echo "Attempts     IP "
 echo "----------------------"
 grep "Failed password for root" /var/log/auth.log|cut -d: -f 4 | cut -d"
" -f7|sort |uniq -c |sort -nr |head -
```

Again, use vim to create *"topattack"*, *chmod* to make it executable and *mv* to move it into */usr/local/bin* when you have it working.

A collection of simple scripts like this is something that you can easily create to make your sysadmin tasks simpler, quicker and less error prone.

And yes, this is the last lesson - so please, write to tutor@GREP101.com to let me know how the course went for you, and what you plan to do with your new knowledge and skills!

##RESOURCES

- Writing a Simple Bash Script (https://www.linux.com/learn/tutorials/284789-writing-a-simple-bash-script-)
- Learn Bash Scripts - Tutorial (video) (http://www.youtube.com/watch?v=QGvvJO5UIs4)
- Bash scripting tutorial (http://linuxconfig.org/Bash_scripting_Tutorial)
- BASH Programming - Introduction HOW-TO (http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html)
- How to be a good (and lazy) System Administrator (http://www.linuxjournal.com/content/how-be-good-and-lazy-system-administrator)