Day 1 - Get to know your server

- Lesson video

INTRO

You should now have a remote server setup running the latest Ubuntu Server LTS (Long Term Support) version. You alone will be administering it. 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 derived from the UNIX Operating System - 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 to your server, preferably using a SSH client
- Run a few simple commands to check the status of your server - like this demo

USING A SSH CLIENT

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 your server is a local VM or WSL, you could skip this section by simply using the server console/terminal if you want.** We will explore SSH more in detail at the server side on Day 3 but knowing how to use a ssh client is a basic sysadmin skill, so you might as well do it now.

In MacOS and Linux

On an MacOS 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 the terminal under "Applications menu -> Accessories -> Terminal", "Applications menu -> System -> Terminal" or "Menu -> System -> Terminal Program (Konsole)"- or you can simply search for your terminal application. In many cases `Ctrl+Alt+T` will also bring up a terminal windows.

Once you open up a "terminal" session, you can use your command-line **ssh** client like this:

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

For example:

```
ssh support@192.123.321.99
```

If the remote server was configured with a SSH public key (like AWS, Azure and GCP), then you'll need to point to the location of the private key as proof of identity with the `-i` switch, typically like this:

```
ssh -i ~/.ssh/id_rsa support@192.123.321.99
```

A very slick connection process can be setup with the `.ssh/config` feature - see the "SSH client configuration" link in the EXTENSION section below.

In Windows

On recent Windows 10 versions, the same command-line client is now available, but must be enabled (via "Settings", "Apps", "Apps & features", "Manage optional features", "Add a feature", "OpenSSH client").

There are various SSH clients available for Windows (PuTTY, Solar-PuTTY, MobaXterm, Termius, etc) but if you use Windows versions older than 10, the installation of PuTTY is suggested.

Alternatively, you can install the Windows Subsystem for Linux which gives you a full local command-line Linux environment, including an SSH client - *ssh*.

Regardless of which client you use, the first time you connect to your server, you may receive a warning that you're connecting to a new server - and be asked if you wish to cache the host key. Yes, you do. Just type/click **Yes**.

But don't worry too much about securing the SSH session or hardening the server right now; we will be doing this in Day 3.

For now, just login to your server and remember that Linux is case-sensitive regarding user names, as well as passwords.

You'll be spending a lot of time in your SSH client, so it pays to spend some time customizing 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).

It's also very handy to be able to cut and paste text between your remote session and your local desktop, so spend some time getting confident with how to do this in your SSH client and terminal.

Perhaps you might now try logging in from home and work - even from your smartphone! - using an ssh client app such as Termux, Termius for Android or Termius for iPhone. As a server admin you'll need to be comfortable logging in from all over. You can also potentially use JavaScript ssh clients like consolefish and ShellHub, but these options involve putting more trust in third-parties than most sysadmins would be comfortable with when accessing production systems.

To log out, simply type `exit` or close the terminal.

LOGIN TO YOUR SERVER

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 (but we will dive into this difference in Day 3 as well).

Here's a short vid on using ssh in a work environment.

GENERAL INFORMATION ABOUT THE SERVER

Use `lsb_release -a` to see which Linux distro and version you're using. lsb_release may not be available in your server, as it's not widely adopted, but you will always have the same information available in the system file os-release. You can check its content by typing `cat /etc/os-release`

`uname -a` will also print the system information and it can show some interesting things like kernel version, hardware platform, etc.

`uptime` will show you how long the system has been running. It kinda makes the weird numbers you get from `cat /proc/uptime` a lot more readable.

`whoami` will print the user name you logged on with, `who` will show who is logged on and `w` will also show what they are doing.

HARDWARE INFORMATION

`lshw` can give some detailed information on the hardware configuration, and there's a bunch of switches we can use to filter the information we want to see, but it's not the only tool we use to check hardware with. Some of the used commands are:

- `lscpu` to display information about the CPU architecture
- `lsblk` to list block devices
- `lspci` to list all PCI devices
- `lsusb` to list USB devices

MEASURE MEMORY AND CPU USAGE

Don't worry! Linux won't eat your RAM. But if you want to check the amount of memory used in the system, use `free -h` . `vmstat` will also give some memory statistics.

`top` is like a Task Manager for Linux, it will display the processes and the consumption of resources. `htop` is an interactive, prettier version.

MEASURE DISK USAGE

Use `df -h` to see disk space usage, but go with `du -h` if you want to estimate the size of your folders.

MEASURE NETWORK USAGE

You will have a general idea of your network interfaces and their IP addresses by using `ifconfig` or its modern substitute `ip address`, but it won't show you bandwidth usage.

For that we have `netstat -i` in a more static view and `ifstat` in a continuous view. To interrupt `ifstat` just use `CTRL+C`.

But if you want more info on that traffic, `sudo iftop -i eth0` is a nice display. *Change `eth0` for the interface you wish to capture traffic information.* To exit the monitor view, type `q` to quit.

POSTING YOUR PROGRESS

Regularly posting your progress can be a helpful motivator. Feel free to post to the subreddit/community or to the discord chat a small introduction of yourself, and your Linux background for your "classmates" - and notes on how each day has gone.

Of course, also drop in a note if you get stuck or spot errors in these notes.

EXTENSION

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

- What is swap and swap space?
- How Linux deals with out-of-memory?
- How Do I Find Out Linux CPU Utilization and Usage?
- How do I find out Linux Disk utilization and I/O usage?
- 20 Best Linux Bandwidth Monitoring Tools for Network Analysis

RESOURCES

- Comparing CENTOS and Ubuntu for servers
- How to Use PuTTY on Windows
- Puttygen command line on Linux - SSH key generator
- SSH client configuration
- A Beginners Guide to SSH
- Linux command to display your hardware information
- See if your hardware is Linux-compatible on Hardware for Linux and DistroWatch
- What is Load Average in Linux?

Day 2 - Basic navigation

- Lesson video

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 skilled in it.

When you use a graphic desktop such as Windows or Apple's macOS (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 structure" of "folders" (e.g. *C:\Users\Steve\Desktop* on Windows or */Users/Steve/Desktop* on macOS - and on a Desktop Linux system */home/steve/Desktop*)

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

- Find the documentation for the commands we used so far - demo
- Navigate between directories, then create, list, move and delete files - demo

RTFM

This is a good time to mention that one of the many advantages of Linux is that it's designed to let you know the system, to let you learn how to use it. The documentation available in form of text manuals, guides and forums is where you will spend most of your time during this journey.

Whereas proprietary systems have some free documentation, you see much more frequently the use of paid customer support to fix issues or find how a particular task can be executed. Although you can also do this with Linux (Canonical, RedHat and SuSE are examples of companies that offer support in the same fashion), this is most likely not the case. And you are here to learn, so...

Which leads us to the famous acronym RTFM. Reading the manual is **the first thing** you should do when you're learning a command. We will go through the

many ways to obtain that information but if at the end of that search you need more insight, you can always ask a well written question in forums and other communities.

Starting with the `man` command. Each application installed comes with its own page in this manual, so that you can look at the page for `pwd` to see the full detail on the syntax like this:

```
man pwd
```

You might also try:

```
man cp
man mv
man grep
man ls
man man
```

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!

And that's why `tldr` is such a powerful tool! You can easily install it with `sudo apt install tldr` or follow this demo.

```
$ tldr pwd
pwd
Print name of current/working directory.More information: https://www.gnu.org/software/coreu

 - Print the current directory:
   pwd

 - Print the current directory, and resolve all symlinks (i.e. show the "physical" path):
   pwd -P
```

If you know a keyword or some description of what the command is supposed to do, you can try `apropos` or `man -k` like this:

```
$ apropos "working directory"
git-stash (1)        - Stash the changes in a dirty working directory away
pwd (1)              - print name of current/working directory
pwdx (1)             - report current working directory of a process

$ man -k "working directory"
git-stash (1)        - Stash the changes in a dirty working directory away
pwd (1)              - print name of current/working directory
pwdx (1)             - report current working directory of a process
```

But you'll soon find out that not every command has a manual that you can read with `man`. Those commands are contained within the shell itself and we call them builtin commands.

There are some overlaping (i.e. builtin commands that also have a man page) but if `man` does not work, we use `help` to display information about them.

```
$ man export
No manual entry for export

$ help export
export: export [-fn] [name[=value] ...] or export -p
    Set export attribute for shell variables.

    Marks each NAME for automatic export to the environment of subsequently
    executed commands.  If VALUE is supplied, assign VALUE before exporting.

    Options:
      -f        refer to shell functions
      -n        remove the export property from each NAME
      -p        display a list of all exported variables and functions

    An argument of `--' disables further option processing.

    Exit Status:
    Returns success unless an invalid option is given or NAME is invalid.
```

The best way to know if a command is a builtin command, is to check its `type`:

```
$ type export
export is a shell builtin
```

And lastly, `info` reads the documentation stored in info format.

NAVIGATE THE FILE STRUCTURE

- Start by reading the manual: `man hier`
- `/` is the "root" of a branching tree of folders (also known as directories)
- At all times you are "in" one part of the system - the command `pwd` ("print 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` 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 commands, 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 them. 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 or 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" as the first character on the line are directories (folders) rather than files. They may also be shown in a different color or font - if not, then adding the "–color=auto" switch should do this (i.e. `ls -ltra --color=auto`)

BASIC DIRECTORY MANIPULATION

- 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.
- You can create even more directories, nesting inside directories, and then navigate between them with the `cd` command.
- When you want to move that directory inside another directory, you use `mv` and specify the path to move.
- To delete (or remove) a directory, use `rmdir` if the directory is empty or `rm -r` if there still any files or other directories inside of it.

BASIC FILE MANIPULATION

- You can make new empty files with the `touch` command, so you can explore a little more of the `ls` command.
- When you want to move that file to another directory, you use `mv` and specify the path to move.
- To delete (or remove) a file, use `rm`.

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".

EXTENSION

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

- Learn about `pushd` and `popd` to navigate around multiple directories easily. Running `pushd /var/log` moves you to to the `/var/log`, but keeps track of where you were. You can `pushd` more than one directory at a time. Try it out: `pushd /var/log`, `pushd /dev`, `pushd /etc`, `pushd`, `popd`, `popd`. Note how `pushd` with no arguments switches between the last two *pushed* directories but more complex navigation is also possible. Finally, `cd -` also moves you the last visited directory.

RESOURCES

- Difference between help, info and man command
- GNU Texinfo
- Explore the Linux file system
- Linux File System
- Simple Terminal Commands on Ubuntu
- Solaris Unix Commands

Day 3 - Power trip!

- Lesson video

INTRO

You may have been logging in as an ordinary user 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 a typo in a command could potentially cripple your server. As a sysadmin you're typically working on systems that are both important and remote, so avoiding such mistakes is *A Very Good Idea*.

In ancient times, sysadmins used to login as `root` in production systems, 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.

YOUR TASKS TODAY

- Change the password of your `sudo` user
- Change the hostname
- Change the timezone

Check out the demo

LOCAL CHANGES VS GLOBAL CHANGES

Global: programs/environments that any user can use, used across the system. A global change affects all users. **Local** or **By user**: programs/environments that a particular user runs, not available to other users. A local change affects only one user.

WHO ARE YOU AND WHAT CAN YOU DO?

There are 3 types of users in a Linux system:

- `root` - the powerful superuser that can execute any command at any level in the system. They can do all global changes as well as local changes for any user.
- `sudoers` - regular users that are allowed to use `sudo`, i.e., they can execute commands in one or more levels in the system, can do some or all global changes. It's common to have at least one sudoer that has the same powers as root, but the amount of priviledges other sudoers have can vary.
- `regular users` - users that can use the system but can only do local changes, i.e., can only deal with their own files/directories and environment variables.

We will get into more detail about users and their permissions on Day 13 and Day 14.

STOP USING ROOT

If you created a VM with one of the big VPS providers, `root` is already "disabled" and your default user (ubuntu, azureuser, etc) already has `sudo` powers.

However, if you really, *really* want to use `root`, there are ways to do it in AWS, Azure and GCP. **But do it at your own risk!**

However, if you created a VM locally or with other VPS providers, it is very likely that you have your `root` user readily available.

Stop using root. If you followed the guides, you should have created a regular user and adding it to a sudoers group, like this:

```
adduser snori74
```

```
usermod -a -G sudo snori74
```

Adding a regular user to a group with `sudo` priviledges is the easiest way to do it, as the `sudo` group is pretty standard in Ubuntu. But this can also be accomplished by modifying the `/etc/sudoers` using the command `visudo`.

Login with this new user from now on.

CHANGE PASSWORD

If you're using a password to login (rather than public key), then now is a good time to ensure that this is very strong and unique - i.e. at least 10 alphanumeric characters - because your server is fully exposed to bots that will be continuously attempting to break in. 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. In a production system of course, public keys and/or two factor authentication would be more appropriate.

A NOTE ON "HARDENING"

Your server is protected by the fact that its security updates are up to date, and that you've set *Long Strong Unique* passwords - or are using public keys. While exposed to the world, and very likely under continuous attack, it should be perfectly secure. Next week we'll look at how we can view those attacks, but for now it's simply important to state that while it's OK to read up on "SSH hardening", things such as changing the default port and `fail2ban` are unnecessary and unhelpful when we're trying to learn - and you are perfectly safe without them.

THE POWER OF SUDO

- Use the links in the "Resources" section below to understand how `sudo` works
- Use `ls -l` to check the permissions of `/etc/shadow` - notice that only `root` has any access. Can you use `cat`, `less` or `nano` to view it?
- This file is where the hashed passwords are kept. It is a prime target for intruders - who aim to grab it and use offline password crackers to discover the passwords.
- Now try with `sudo`, e.g. `sudo cat /etc/shadow`
- Test running the `reboot` command, and then via `sudo` (i.e. `sudo reboot`)

Once you've reconnected back:

- Use the `uptime` command to confirm that your server did actually fully restart
- Test fully "becoming root" by the command `sudo -i`. This can be handy if you have a series of commands to do "as root". Note the change to your prompt.

- Type `exit` or `logout` to get back to your own normal "support" login.
- Use `less` to view the file `/var/log/auth.log`, where any use of `sudo` is logged
- You could "filter" this by typing: `grep "sudo" /var/log/auth.log`

Normally invoking the `sudo` command will ask you to re-confirm your identity with your password. However, this can be changed in the sudoers configuration file so it does NOT prompt for a password.

ADMINISTRATIVE TASKS

We will go into detail of the many things you can do to your server, but here are some examples of simple administrative tasks that require `sudo`.

If you wish to, you can now rename your server. Traditionally you would do this by editing two files, `/etc/hostname` and `/etc/hosts` and then rebooting - but the more modern, and recommended, way is to use the `hostnamectl` command; like this:

```
sudo hostnamectl set-hostname mylittlecloudbox
```

No reboot is required but if you want to see the new name in the prompt, just open a new session with `bash` (or logoff and login again, same effect).

For a cloud server, you might find that the hostname changes after a reboot. To prevent this, edit `/etc/cloud/cloud.cfg` and change the "preserve_hostname" line to read:

```
preserve_hostname: true
```

You might also consider changing the timezone your server uses. By default this is likely to be UTC (i.e. GMT) - which is pretty appropriate for a worldwide fleet of servers. You could also set it to the zone the server is in, or where you and your headquarters are. For a company this is a decision not to be taken lightly, but for now you can simply change as you please!

First check the current setting with:

```
timedatectl
```

Then get a a list of available timezones:

```
timedatectl list-timezones
```

And finally select one, like this:

```
sudo timedatectl set-timezone Australia/Sydney
```

Confirm:

```
timedatectl
```

The major practical effects of this are (1) the timing of scheduled tasks, and (2) the timestamping of the logs files kept under `/var/log`. If you make a change, there will naturally be a "jump" in the dates and time recorded.

WRAP

As a Linux sysadmin you may be working on client or custom systems where you have little control, and many of these will default to doing everything as `root`. You need to be able to safely work on such systems - where your only protection is to double check before pressing `Enter`.

On the other hand, for any systems where you have full control, setting up a "normal" account for yourself (and any co-admins) with permission to run `sudo` is recommended. While this is standard with Ubuntu, it's also easy to configure with other popular server distros such as Debian, CentOS and RHEL.

EXTENSION

- How To Edit the Sudoers File
- Hardening SSH
- SSH Tunneling
- How To Set Up Multi-Factor Authentication for SSH on Ubuntu 20.04

What's difference between "sudo -i" and "sudo -s"?

Both `sudo -i` and `sudo -s` are commands that allow a user to obtain root privileges on a Unix-based system. However, they have some differences in how they function.

- `sudo -i` stands for "sudo interactive" and it launches a new login shell for the root user. This means that it creates a new environment for the root user with the root user's home directory and shell configuration files. This makes it similar to logging in directly as the root user, and any commands executed from this shell will have the privileges of the root user.
- `sudo -s` stands for "sudo shell" and it launches a new shell for the root user, but it does not create a new login shell. This means that it does not change the environment or shell configuration files of the current user. Any commands executed from this shell will have the privileges of the root user, but the environment will still be that of the current user.

In summary, `sudo -i` is more powerful and creates a new shell with the full environment of the root user, while `sudo -s` is less powerful and only launches a new shell with the root user's privileges but with the same environment as the current user.

RESOURCES

- This cartoon explains it nicely!

- Sudo in Ubuntu
- How to use “sudo”
- This is how password cracking is done
- Password-less SSH login

Day 4 - Installing software, exploring the file structure

- Complementary video

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 search "midnight commander"
```

This will show a range of matching “packages”, and we can then install them with `apt install` command. So to install package `mc` (Midnight Commander) on Ubuntu:

```
sudo apt 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. You can also read the official manual on this hierarchy by typing `man hier`.

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`, `/etc/ssh/sshd_config` 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 use `apt search` to search for and 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

- Post your progress, comments and questions to the forum.

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

- Difference Between apt and apt-get Explained
- DNF vs APT: Similarities and Differences Analyzed!
- Ubuntu Server Guide - Package Management
- Midnight Commander vs Ranger
- Linux directory system explained

Day 5 - More or less. . .

- Lesson video

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.

YOUR TASKS TODAY

- Use tab completion
- Search in the command history
- Read a dot file using more and less
- Change / customize your prompt

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. 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. You can also press `Ctrl + r`, then start typing any part of the command that you are looking for. You'll see an autocomplete of a past command at your prompt. If you keep typing, you'll get more specific options appear. You can either run it by pressing return, or editing it first by pressing arrows or other movement keys. You can also keep pressing `Ctrl + r` to see other instances of the same command you used with different options.
- 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 in your home directory and type up a summary of how the last five days have worked for you.

EXTENSION

We're using `bash` as our terminal shell for now (it is standard in many distros) but it is not the only one out there. If you want to test out zsh, fish or oh-my-zsh, you will see that there are a few differences and the features are usually the main differentiator. Try that, poke around.

After that, you can go up a notch and try to have several shell sessions open at the same time in the same terminal window with a **terminal multiplexer**. Try screen - that's a little simpler and maybe too terse in the beginning - or tmux, that have many features and colors. There are so much material out there on "how to customize your tmux", have fun.

RESOURCES

- Unix Less Command: 10 Tips for Effective Navigation
- How To Use Bash History Commands and Expansions...
- BASH Shell commands less
- Tab completion
- What are dotfiles?
- Nano editor tutorials
- Bash Shell: Take Control of PS1, PS2, PS3, PS4 and PROMPT_COMMAND
- Bash Shell PS1: 10 Examples to Make Your Linux Prompt like Angelina Jolie

Day 6 - Editing with "vim"

- Complementary video

INTRO

Simple text files are at the heart of Linux, so editing these is a key sysadmin skill. There are a range of simple text editors aimed at beginners. Some more common examples you'll see are `nano` and `pico`. These look as if they were written for DOS back in the 1980's - but are pretty easy to "just figure out".

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

Bill Joy wrote Vi back in the mid 1970's - and even the "modern" *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. Vim is actually a contraction of Vi IMproved and is a direct descendant of Vi.

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

```
vi --version
```

You should see output similar to the following if the `vi` command is actually symlinked to `vim`:

```
user@testbox:~$ vi --version
VIM - Vi IMproved 8.2 (2019 Dec 12, compiled Oct 01 2021 01:51:08)
Included patches: 1-2434
Extra patches: 8.2.3402, 8.2.3403, 8.2.3409, 8.2.3428
Modified by team+vim@tracker.debian.org
Compiled by team+vim@tracker.debian.org
...
```

YOUR TASKS TODAY

- Run *vimtutor*
- Edit a file with vim

WHAT IF I DON'T HAVE VIM INSTALLED?

The rest of this lesson assumes that you have `vim` installed on your system, which it often is by default. But in some cases it isn't and if you try to run the `vim` commands below you may get an error like the following:

```
user@testbox:~$ vim
-bash: vim: command not found
```

OPTION 1 - ALIAS VIM One option is to simply substitute `vi` for any of the `vim` commands in the instructions below. Vim is reverse compatible with Vi and all of the below exercises should work the same for Vi as well as for Vim. To make things easier on ourselves we can just *alias* the `vim` command so that `vi` runs instead:

```
echo "alias vim='vi'" >> ~/.bashrc
source ~/.bashrc
```

OPTION 2 - INSTALL VIM The other option, and the option that many sysadmins would probably take is to install Vim if it isn't installed already.

To install Vim on Ubuntu using the system package manager, run:

```
sudo apt install vim
```

> **Note:** Since Ubuntu Server LTS is the recommended Linux distribution to use for the Linux Upskill Challenge, installing Vim for all of the other various Linux "distros" is outside of the scope of this lesson. The command above "should" work for most Debian-family Linux OS's however, so if you're running Mint, Debian, Pop!_OS, or one of the many other flavors of Ubuntu, give it a try. For Linux distros outside of the Debian-family a few simple web-searches will probably help you find how to install Vim using other Linux's package managers.

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 "normal mode" and "insert mode", and as a beginner, simply remember:

```
"Press Esc twice or more to return to normal mode"
```

The "normal mode" is used to input commands, and "insert mode" for writing text - similar to a regular text editor's default behaviour.

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 "normal 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 normal 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 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, let's 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 normal mode and *33dd* is a command that says "delete 33 lines". Now, you're still in normal 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 "normal mode"
- From normal mode `:q!` will always quit without saving anything you've done, and
- From normal mode `u` will undo the last action

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

- Finding things: From normal mode, type `G` to get to the bottom of the file, then `gg` to get to the top. Let's search for references to "sun", type `/sun` to find the first instance, hit enter, 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 normal mode when you're done.
- Writing your changes to disk: From normal mode type `:w` to "write" but stay in `vim`, or `:wq` to "write and quit".

This is as much as you ever *need* to learn about `vim` - but there's an enormous amount more you *could* learn if you had the time. Your next step should be to run `vimtutor` and go through the "official" Vim tutorial. It typically takes around 30 minutes the first time through. To solidify your Vim skills make a habit of running through the vimtutor every day for 1-2 weeks and you should have a solid foundation with the basics.

> **Note:** If you aliased vim to vi for the excercises above, now might be a good time to install vim since this is what provides the `vimtutor` com-

mand. Once you have Vim installed, you can run `:help vimtutor` from inside of Vim to view the help as well as a few other tips/tricks.

However, if you're serious about becoming a sysadmin, it's important that you *commit* to using `vim` (or `vi`) for all of your editing from now on.

One last thing, you may see reference to is the Vi vs. Emacs debate. This is a long running rivalry for programmers, not system administrators - `vi/vim` is what you need to learn.

WHY CAN'T I JUST STICK WITH NANO?

- In many situations as a professional, you'll be working on other people's systems, and they're often very paranoid about stability. You may not have the authority to just "sudo apt install <your.favorite.editor>" - even if technically you could.
- However, `vi` is always installed on any Unix or Linux box from tiny IoT devices to supercomputer clusters. It is actually required by the Single Unix Specification and POSIX.
- And frankly it's a shibboleth for Linux pros. As a newbie in an interview it's fine to say you're "only a beginner with vi/vim" - but very risky to say you hate it and can never remember how to exit.

So, it makes sense if you're aiming to do Linux professionally, but if you're just working on your own systems then by all means choose `nano` or `pico` etc.

EXTENSION

If you're already familiar with `vi` / `vim` then use today's hour to research and test some customisation via your `~/.vimrc` file. The link below is specifically for sysadmins:

- Getting more out of Vim

RESOURCES

- Here is why `vim` uses the *hjkl* keys as arrow keys
- Graphical vi-vim Cheat Sheet and Tutorial
- Vi - Vim Tutorial (video)
- How to Copy, Cut and Paste in Vim / Vi

Day 7 - The server and its services

- Lesson video

INTRO

Today you'll install a common server 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

YOUR TASKS TODAY

- Install and run apache, transforming your server into a *web server*

INSTRUCTIONS

- Refresh your list of available packages (apps) by: `sudo apt 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 install apache2`
- Confirm that it's running by browsing to *http://[external IP of your server]* - 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 systemctl stop apache2` - check that the webpage goes dead - then re-start it with `sudo systemctl start apache2` - and check its status with: `systemctl status 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/apache2.conf` - you can use `less` to simply view them, or the `vim` editor to view and edit as you wish.
- In `/etc/apache2/apache2.conf` there's the line with the text: "IncludeOptional conf-enabled/*.conf". This tells Apache that the *.conf files in the subdirectory *conf-enabled* should be merged in with those from `/etc/apache2/apache2.conf` at load. This approach of lots of small specific config files is common.
- If you're familiar with configuring web servers, then go crazy, setup some virtual hosts, or add in some mods etc.
- The location of the default webpage is defined by the *DocumentRoot* parameter in the file `/etc/apache2/sites-enabled/000-default.conf`.
- Use `less` or `vim` to view the code of the default page - normally at `/var/www/html/index.html`. This uses fairly complex modern web de-

sign - so you might like to browse to http://165.227.92.20/sample where you'll see a much simpler page. Use View Source in your browser to see the code of this, copy it, and then, in your ssh session `sudo vim /var/www/html/index.html` to first delete the existing content, then paste in this simple example - and then edit to your own taste. View the result with your workstation browser by again going to *http://[external IP of your server]*

- As with most Linux services, 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, but in a later lesson you'll learn how to filter out just what you want. Notice the `error.log` file too - hopefully this one will be empty!

Note for AWS/Azure/GCP users

Don't forget to add port 80 to your instance security group to allow inbound traffic to your server.

- AWS
- Azure
- GCP

POSTING YOUR PROGRESS

Practice your text-editing skills, and allow your "classmates" to judge your progress by editing `/var/www/html/index.html` with `vim` and posting the URL to access it to the forum. (It doesn't have to be pretty!)

SECURITY

- As the sysadmin of this server, responsible for its security, you need to be very aware that you've now increased the "attack surface" of your server. In addition to *ssh* on port 22, you are now also exposing the *apache2* code on port 80. Over time the logs may reveal access from a wide range of visiting search engines, and attackers - and that's perfectly normal.
- If you run the commands: `sudo apt update`, then `sudo apt upgrade`, and accept the suggested upgrades, then you'll have all the latest security updates, and be secure enough for a test environment - but you should re-run this regularly.

EXTENSION

Read up on:

- Using *systemctl* to manage services

RESOURCES

- HTTPD - Apache2 Web Server
- The Apache HTTP Server

TROUBLESHOOT AND MAKE A SAD SERVER HAPPY!

Practice what you've learned with some challenges at SadServers.com:

- "Cape Town": Borked Nginx

Day 8 - The infamous "grep" and other text processors

- Lesson video

INTRO

Your server is now running two services: the *sshd* (Secure Shell Daemon) service that you use to login; and the Apache2 web server. 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.

YOUR TASKS TODAY

- 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 *N* (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 `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)
- You can take the output of one command and "pipe" it in as the input to another by using the `|` (pipe) symbol
- So, dump out a file with `cat`, but pipe that output to `grep` with a search term - like this: `cat /var/log/auth.log  | grep "authenticating"`
- Simplify this to: `grep "authenticating" /var/log/auth.log`
- Piping allows you to narrow your search, e.g. `grep "authenticating" /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 "authenticating" /var/log/auth.log| grep "root"| cut -f 10- -d" "` (field 10 onwards, where the delimiter between field is the " " character). This approach can be very useful in extracting useful information from log data.
- Use the `-v` option to invert the selection and find attempts to login with other users: `grep "authenticating" /var/log/auth.log| grep -v "root"| cut -f 10- -d" "`

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 redirect into the file "listing.txt" (creating that file if it didn't exist, or overwriting the contents if it did).

WHERE'S MY /VAR/LOG/AUTH.LOG?

If you didn't find the file `/var/log/auth.log` you're probably using a minimal version of Ubuntu (it can be your own local VM or a version in one of the VPS). That minimal image is, well... minimal. It only has the systemd journal available and it didn't come with the old syslog system by default.

But don't worry! To get that back, `sudo apt install rsyslog` and the file will be created. Just give it a few minutes to populate before working on the lesson.

It also be missing a few of the other programs we use in the challenge, but you can always install them.

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: `~/attackers.txt`. You might like to share and compare with others doing the course how heavily you're "under attack"!

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`

RESOURCES

- Text processing commands
- OSTechNix grep tutorial
- Where GREP came from
- SED onliners
- RegExr - a tool to learn, build and test Regular Expressions
- explainshell.com - write down a command-line to see the help text that matches each argument

TROUBLESHOOT AND MAKE A SAD SERVER HAPPY!

Practice what you've learned with some challenges at SadServers.com:

- "Saskatoon": counting IPs.
- "Santiago": Find the secret combination
- "The Command Line Murders"

Day 9 - Diving into networking

- Lesson video

INTRO

The two services your server is now running are *sshd* for remote login, and *apache2* for web access. These are both "open to the world" via the 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 appropriate monitoring and controls.

YOUR TASKS TODAY

- Secure your web server by using a firewall

INSTRUCTIONS

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

- `ss` - this, "socket status", is a standard utility - replacing the older `netstat`
- `nmap` - this "port scanner" won't normally be installed by default

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

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

```
sudo ss -ltp
State   Recv-Q  Send-Q   Local Address:Port     Peer Address:Port  Process
LISTEN  0       4096     127.0.0.53%lo:53        0.0.0.0:*      users:(("systemd-resolve",pi
LISTEN  0       128            0.0.0.0:22           0.0.0.0:*      users:(("sshd",pid=625,fd
LISTEN  0       128               [::]:22              [::]:*      users:(("sshd",pid=625,fd
LISTEN  0       511                  *:80                 *:*     users:(("apache2",pid=1066
```

The network notation can be a little confusing, but the lines above show ports 80 and 22 open "to the world" on all local IP addresses - and port 53 (DNS) open only on a special local address.

Now install `nmap` with `apt install`. 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.

Note that however that "localhost" (127.0.0.1), is the loopback network device. Services "bound" *only* to this will only be available on this local machine. To see what's actually exposed to others, first use the `ip a` command to find the IP address of your actual network card, and then `nmap` that.

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 are the `iptables` command, and the newer `nftables`. These are 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. It is available by default in all Ubuntu installations after 8.04 LTS, but if you need to install it:

```
sudo apt install ufw
```

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

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

BEWARE! Don't forget to explicitly ALLOW `ssh`, or you'll lose all contact with your server! If not allowed, the firewall assumes the port is DENIED by default.

And then enable this with:

```
sudo ufw enable
```

Typing `sudo iptables -L` now will list the detailed rules generated by this - one of these should now be:

```
"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" - all incoming traffic to the destination port of http/80 being DROPed. Test for yourself! You will probably want to reverse this with:

```
sudo ufw allow http
sudo ufw enable
```

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.

BTW: For this test/learning server you should allow http/80 access again now, because those `access.log` files will give you a real feel for what it's like to run a server in a hostile world.

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 is particularly common advice for **ssh/22** - and would be done by altering the configuration in `/etc/ssh/sshd_config`.

Some call this "security by obscurity" - equivalent to moving the keyhole on your front door to an unusual place rather than improving the lock itself, or camouflaging your tank rather than improving its armour - but it *does* effectively eliminate attacks by opportunistic hackers, which is the main threat for most servers.

But, if you're going to do it, remember all the rules and security tools you already have in place. If you are using AWS, for example, and change the SSH port to 2222, you will need to open that port in the EC2 security group for your instance.

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
- Iptables How To

RESOURCES

- 12 ss Command Examples to Monitor Network Connections
- UFW - Uncomplicated Firewall
- Collection of basic Linux Firewall iptables rules
- 10 Netstat Command Example
- UFW Uncomplicated Firewall (video)
- How to install nftables in Ubuntu
- No, moving your ssh port isn't security by obscurity
- Port knocking

TROUBLESHOOT AND MAKE A SAD SERVER HAPPY!

Practice what you've learned with some challenges at SadServers.com:

- “Tokyo”: can’t serve web file
- “Taipei”: Come a-knocking

Day 10 - Getting the computer to do your work for you

- Complementary video

INTRO

Linux has a rich set of features for running scheduled tasks. One of the key attributes of a good sysadmin is 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.

YOUR TASKS TODAY

- Schedule a job to *apt update* and *apt upgrade* everyday

CRON

Each user potentially has their own set of scheduled task which can be listed with the `crontab` command (list out your user crontab entry with `crontab -l` and then that for *root* with `sudo crontab -l`).

However, there’s also a system-wide crontab defined in `/etc/crontab` - use `less` to look at this. Here’s 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.
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.
```

Lines beginning with “#” are comments, so `# m h dom mon dow user command` defines the meanings of the columns.

Although the detail is a bit complex, it’s pretty clear what this does. 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".

SYSTEMD TIMERS

All major Linux distributions now include "systemd". As well as starting and stopping services, this can *also* be used to run tasks at specific times via "timers". See which ones are already configured on your server with:

```
systemctl list-timers
```

Use the links in the RESOURCES section to read up about how these timers work.

RESOURCES

- A good overview of systemd/Timers
- "How to Use Systemd Timers as a Cron Replacement"
- Automate system administration tasks by scheduling jobs
- Using cron to automate maintenance

Day 11 - Finding things...

- Lesson video
- Complementary video

INTRO

Today 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 some useful tools:

- `locate`
- `find`
- `grep`
- `which`

YOUR TASKS TODAY

- Find all files that have the word "Permission" in it

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
```

(If `locate` is not installed, do so with `sudo apt install mlocate`)

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, so 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 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* - or you could filter the errors with `grep` like this: `find /var -name access.log 2>&1 | grep -vi "Permission denied"`.

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, including following all symbolic links (which `-r` does not). 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.

EXTENSION

The `-exec` feature of the `find` command is extremely powerful.

But "finding things" can go so much further than that! You can not only track down the content of a file, but also its **usage** with commands like `lsof` and `fuser`.

Test some examples of this from the RESOURCES links.

RESOURCES

- 25 find command examples...
- 10 Tips for using "find"
- Five simple recipes for "grep"
- How to use the lsof command to troubleshoot Linux
- Learn "fuser", a little-known Linux workhorse command!

TROUBLESHOOT AND MAKE A SAD SERVER HAPPY!

Practice what you've learned with some challenges at SadServers.com:

- "Saint John": what is writing to this log file?

Day 12 - Transferring files

- Lesson video
- Complementary video

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

YOUR TASKS TODAY

- Upload a file to the server
- Download a file from the server
- Synchronize a backup

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 local desktop to your server, SFTP has a number of key advantages:

- No 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 macOS 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 macOS 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 your username
- Copy some files from your server down to your local desktop (try files from your "home" folder, and from `/var/log`)
- Create an "`images`" folder under your "home" folder on the server, 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 an ordinary use, so 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.

RESOURCES

- CyberDuck
- FileZilla
- SFTP – SSH Secure File Transfer Program
- sftp File From One Server To Another

Day 13 - Users and Groups

- Lesson video
- Complementary video

INTRO

Today you're going to set-up another user on your system. You're going to imagine that this is a help-desk person that you trust 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!

YOUR TASKS TODAY

- Create a new user
- Create a new group
- Create a new user and add to an existing group
- Make a new user a sudoer

Follow this demo

ADDING A NEW USER

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

```
sudo adduser helen
```

(Names are case-sensitive in Linux, so "Helen" 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 helen
```

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`. A hash of the 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 systems 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/helen` for example, with the correct permissions.

ATTENTION! `useradd` is not the same as `adduser`. They both create a new user, but they interact very differently. Check the link in the EXTENSION section to see those differences.

ADDING A NEW GROUP

Let's say we want to all of the developers in my organization to have their own group, so they can have access to the same things.

```
sudo groupadd developers
```

On most modern Linux systems there is a group created for each user, so user "ubuntu" is a member of the group "ubuntu". But if you want, you can create a new user directly into an existing group, using the `ingroup` flag. So a new user `fred` would be created like this:

```
sudo adduser --ingroup developers fred
```

ADDING AN USER TO GROUPS

Users can also be part of more than one group, and groups can be added as required.

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

On an Ubuntu system the first user created (in your case `ubuntu`), should be a member of the groups: `ubuntu`, `sudo` and `admin` - and if you list the `/var/log` folder you'll see your membership of the `sudo` group is why you can use `less` to read and 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 `ubuntu` user can do the same simply by prefixing the command with `sudo`.

Because the new user `helen` is not the first user created in the system, they don't have the power to run `sudo` - which *your* user has by being a member of the group `sudo`.

So, to check which groups `helen` is a member of, you can "become helen" by switching users like this:

```
sudo su helen
```

Then:

```
groups
```

If you try to do stuff only a sudo user can do, i.e. read the contents of `/var/log/auth.log`, even using the prefix `sudo` won't work. Helen is not a sudo and has no permissions to perform this action.

Now type "exit" to return to your normal user, and you can add `helen` to this group with:

```
sudo usermod -a -G sudo helen
```

Instead of switching users again, simply run the `groups helen` to check. Try that with `fred` too and check how everything works.

See if any of your new users can `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 (like we did with `helen`) - but we don't want to give `fred` quite that same 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 -i
```

Notice that your prompt has changed to a `#`

Now simply run `visudo` to begin editing `/etc/sudoers` - typically this will use `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 your normal user again - and notice that your prompt reverts to: `$`

TESTING

Test by logging in as your test user and typing: `sudo reboot` Note that you can "become" helen by:

```
sudo su helen
```

If your ssh config allows login only with public keys, you'll need to setup `/home/helen/.ssh/authorized_keys` - including getting the owner and permissions correct. A little challenge of your understanding of this area!

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
- Linux Password & Shadow File Formats
- What's the difference between 'useradd' and 'adduser'?
- How to create users and groups in Linux from the command line
- Learn how to use the $EDITOR environmental variable to set your default editor to `vim`. With this done, `visudo` will use `vim` rather than `nano` for editing.

RESOURCES

- How To Edit the Sudoers File
- Sudo – An Advanced Howto
- A cartoon that should now make sense!
- Basic Linux Permissions: sudo and sudoers (video)

Day 14 - Who has permission?

- Lesson video
- Complementary video

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 your "ordinary" user, 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!

YOUR TASKS TODAY

- Change the ownership of a file to root
- Change file permissions

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
```

Then these files are owned by user "steve", and the group "staff". Anyone that is not "steve" or is not part of the group "staff" is considered "other". Others may still have permissions to handle these files, but they do not have any ownership.

If you want to change the ownership of a file, use the `chown` utility. This will change the user owner of *file* to a new user:

```
sudo chown user file
```

You can also change user and group at the same time:

```
sudo chown user:group file
```

If you only need to change the group owner, you can use `chgrp` command instead:

```
sudo chgrp group file
```

Since you created new users in the previous lesson, switch logins and create a few files to their home directories for testing. See how they show with `ls -l`

PERMISSIONS (SYMBOLIC NOTATION)

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" - we like to call that UGO.

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*, i.e. "other people", can read it
- *upload.bin* - Steve has `rwx`, he can read, write and execute - i.e. run this program - but the group and others can only read and execute it

You can change the permissions on any file with the `chmod` utility. Create a simple text file in your home directory with `vim` (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-rw-r-- 1 ubuntu ubuntu   12 Nov 19 14:48 tuesday.txt
```

So, the file is owned by the user "ubuntu", and group "ubuntu", who are the only ones that can write to the file - but any other user can only read it.

CHANGING PERMISSIONS

Now let's remove the permission of the user and "ubuntu" group to write their own file:

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

```
chmod g-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 ubuntu ubuntu   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. (In this case, as the owner, you have "permission to override permissions", so can can write with `:w!`). You can of course easily give yourself back the permission to write to the file by:

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

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 `vim`.

EXTENSION

If all of this is old news to you, you may want to look into Linux ACLs:

- How to manage ACLs on Linux
- Linux Access Control Lists

Also, SELinux and AppArmour:

- SELinux man page
- SELinux User's and Administrator's Guide
- SELinux For Mere Mortals
- Securing Ubuntu 18 04 with Apparmor

RESOURCES

- How to Use the chown Command to Change the Owner of a File in Linux
- If chown can change groups, why was chgrp created?
- Linux file permissions explained
- File permissions and attributes
- File Security
- chmod Tutorial
- File and Directory Permissions
- What is "umask" and how does it work?

Day 15 - Deeper into repositories...

- Complementary video

INTRO

Early on you installed some software packages to your server using `apt install`. 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, and increasingly in MacOS and Windows.

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 20.04, CentOS 5, RHEL 6
- "Bit size" - 32-bit or 64-bit
- Chip - Intel, AMD, PowerPC, ARM

The version number is particularly important because it controls the versions of application that you can install. When Ubuntu 18.04 was released (in April 2018 - hence the version number!), it came out with Apache 2.4.29. So, if your server runs 18.04, then even if you installed Apache with `apt` 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` command, but for most purposes the competing `yum` and `dnf` commands used by Fedora, RHEL, CentOS and Scientific Linux work 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. Sometimes there are extra packages available if you enable extra repositories. Most Linux distros have a similar concept, but in Ubuntu, often the "Universe" and "Multiverse" repositories are disabled by default. 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 might include the compression utilities `rar` and `lha`, and the network performance tool `netperf`.

To enable the "Multiverse" repository, follow the guide at:

- Community wiki for command line

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

```
sudo apt update
```

Once done, you should be able to install `netperf` like this:

```
sudo apt install netperf
```

. . . and the output will show that it's coming from Multiverse.

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.

As an example, install and run the `neofetch` utility. When run, this prints out a summary of your configuration and hardware. This is in the standard repositories, and `neofetch --version` will show the version. If for some reason you wanted to be have a later version you could install a developer's Neofetch PPA to your software sources by:

```
sudo add-apt-repository ppa:ubuntusway-dev/dev
```

As always, after adding a repository, update your local cache of available applications:

```
sudo apt update
```

Then install the package with:

```
sudo apt install neofetch
```

Check with `neofetch --version` to see what version you have now.

Check with `apt-cache show neofetch` to see the details of the package.

When you next run "sudo apt upgrade" you'll likely be prompted to install a new version of `neofetch` - because the developers are sometimes 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.

RESOURCES

- Package management command comparison
- How to use yum - Introduction
- Package management with APT
- What do you mean by Free Software?

Day 16 - Archiving and compressing

- Complementary video

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 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.tar
```

... which will create `myinits.tar.gz`. A compressed tar archive like this is known as a "tarball". You will also sometimes see tarballs with a *.tgz* extension - at the Linux commandline this doesn't have any meaning to the system, but is simply 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!

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?

RESOURCES

- 18 Tar Command Examples in Linux
- Linux TAR Command
- Linux tar command tutorial (video)

Day 17 - Build from the source

- Complementary video

INTRO

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

Today we're going one step further - 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.

(Another big part of what package managers like `apt` do, is to identify and install any required "dependencies". In the Linux world many open source apps take advantage of existing infrastructure in this way, but it can be a very tricky thing to resolve manually. However, the app we're installing today from source is relatively unusual in being completly standalone).

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". Install it like this:

```
sudo apt 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:

```
https://nmap.org/dist/nmap-7.70.tar.bz2
```

This is version 7.70, the latest development release when these notes were written, but it may 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 https://nmap.org/dist/nmap-7.70.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-7.70.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-7.70.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 :

```
-rw-r--r--  1 steve  steve  21633731    2011-10-01 06:46 nmap-7.70.tar.bz2
drwxr-xr-x 20 steve  steve  4096        2011-10-01 06:06 nmap-7.70
```

Now explore the contents of this with `mc` or simply `cd nmap-7.70` - you should be able to use `ls` and `less` 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`, `yum` 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 that plus documentation to your system and in some cases will setup services and scheduled tasks etc. Until now you've just been working in your home directory, but this step installs to the system for all users, so requires `root` privileges. Because of this, 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 chosed to manually install 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, but because these files have only just been added, we'll need to manually update the index of files:

```
sudo updatedb
```

Then to search the index:

```
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
```

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 challenging even when well explained (look at: http://oss.oetiker.ch/smokeping/doc/smokeping_install.en.html for a good example).

NOTE: Because you've done all this outside of the `apt` system, this binary won't get updates when you run `apt update`. Not a big issue with a utility like `nmap` probably, but for anything that runs as an exposed service it's important that you understand that you now have to track security alerts for the application (and all of its dependencies), and install the later fixed versions when they're available. This is a significant pain/risk for a production server.

POSTING YOUR PROGRESS

Pat yourself on the back if you succeeded today - and let us know in the forum.

EXTENSION

Research some distributions where "from source" is normal:

- What is Linux From Scratch?
- What is Gentoo?
- The Arch Build System

None of these is typically used in production servers, but investigating any of them will certainly increase your knowledge of how Linux works "under the covers" - asking you to make many choices that the production-ready distros such as RHEL and Ubuntu do on your behalf by choosing what they see as sensible defaults.

RESOURCES

- The magic behind configure, make, make install
- Installing From Tarballs
- How to rebuild an existing package from source
- Compiling things on Ubuntu the Easy Way

Day 18 - Log rotation

- Lesson video

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.

YOUR TASKS TODAY

- Check the logs for *apache2* that are Severity 3
- Edit logrotate configuration for *apache2* to rotate daily

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. You will probably see one called *apache2*, with contents like this:

```
/var/log/apache2/*.log {
weekly
missingok
rotate 52
compress
delaycompress
notifempty
create 640 root adm
}
```

Much of this is fairly clear: any 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.

RESOURCES

- The Ultimate Logrotate Command Tutorial
- LINUX: openSUSE and logrotate
- Use logrotate to Manage Log Files

TROUBLESHOOT AND MAKE A SAD SERVER HAPPY!

Practice what you've learned with some challenges at SadServers.com:

- "Manhattan": can't write data into database.

Day 19 - Inodes, symlinks and other shortcuts

- Complementary video

INTRO

Today's topic 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; instead with the layer of Linux that sits *above* all of these - the Linux Virtual Filesystem.

The VFS is a key part of 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.

Note that the permissions on a symlink generally show as allowing everthing - but what matters is the permission of the file it points to.

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 also 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 resources 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

EXTENSION

- Anatomy of the Linux file system

RESOURCES

- Hard and soft links
- Linux inodes Explained
- Everything You Ever Wanted to Know About inodes on Linux

Day 20 - Scripting

- Complementary video

INTRO

Today is the final session for the course. Pat yourself on the back if you worked your way through all lessons!

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.

YOUR TASKS TODAY

- Write a short script that list the top 3 IP addresses that tried to login into your server

START WITH A SHEBANG!

Scripts are just simple text files, but if you set the "execute" permissions on them then the system will look for a special line starting with the two characters "#" and "!" - referred to as the "shebang" (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` and one in Python `#!/usr/bin/env python3`

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 -i "disconnected from" /var/log/auth.log|tail -1| cut -d: -f4| cut -f7 -d" "
```

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:

```
#
##   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 "Disconnected from authenticating user root" /var/log/auth.log|cut -d: -f 4 | cut -d"
```

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

(BTW, you can use `whois` to find details on any of these IPs - just be aware that the system that is "attacking" you may be an innocent party that's been hacked into).

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.

If automating and scripting many of your daily tasks sounds like something you really like doing, you might also want to script the setup of your machines and services. Even though you can do this using bash scripting like shown in this lesson, there are some benefits in choosing an orchestration framework like ansible, cloudinit or terraform. Those frameworks are outside of the scope of this course, but might be worth reading about.

And yes, this is the last lesson - so please, feel free to write a review on how the course went for you and what you plan to do with your new knowledge and skills!

RESOURCES

- Learn Bash Scripts - Tutorial (video)
- Bash scripting tutorial
- BASH Programming - Introduction HOW-TO
- How to be a good (and lazy) System Administrator

Day 21 - What next?

- Complementary video

What is this madness – surely the course was for just 20 days?

Yes, but hopefully you'll go on learning, so here's a few suggestions for directions that you might take.

Play with your server

You're familiar with the server you used during the course, so keep working with it. Maybe uninstall Apache2 and install NGINX, a competing webserver. Keep a running stat on ssh "attackers". Whatever. A free AWS will last a year, and a $5/mo server should be something you can easily justify.

Add services that you'll use

You should now be capable of following tutorials on installing and running your own instance of Minecraft, Wordpress, WireGuard VPN, or Mediawiki. Expect to have some problems – it's all good experience!

Take a look at Server World for some inspiration.

Extend your learning

Stop browsing articles on Gnome, KDE or i3 – and start checking out any articles like "*20 Linux commands every sysadmin should know*". Try these out, delve into the options. Like learning a foreign vocabulary, you will only be able to use these "words" if you know them!

Check out Linux Journey if you haven't already, specially if you are still pretty new to Linux and would like to see a different learning approach. Linux 101 Hacks is also a good resource.

Practice what you've learned with some challenges at SadServers.com. There you'll find a collection of scenarios where you have to do, fix or hack something in a Linux server. It's great to exercise your troubleshooting skills without messing with your own server.

To get crazy fast in the command line, try Command Line Challenge, practicelinux.com, learnshell.org and commandlinefu.com.

If your next level goal is to get into DevOps, take a look at the DevOps Roadmap.

Certifications

If you're looking to do Linux professionally, and you don't have an impressive CV or resume already, then you should be aiming at getting a Linux certification. There are really just three certs/tracks that count:

- CompTIA Linux+ - one and done exam, distro independent but doesn't hold much value in the market. Do this if you don't want to get too deep into Linux, or you have other CompTIA tracks going on and an employer is paying for them.
- LPI LPIC-1: Linux Administrator – Very extensive description of the coverage of their various certs/courses. You can go very deep with this exams, they cover everything you can think of pure Linux. Not so popular with employers but the knowledge certainly holds it value.
- Red Hat – You could spend a lot of time and money here, but it might well pay off! Geared to RedHat Enterprise Linux distribution and its particularities, it is a practical exam (the others are multiple question) and it's well known in Enterprise circles, it really pops up in any resume.

Even if you don't want/need certs, the outline of the topics in these references can give you a good idea of areas to focus on in your self-learning.

Affordable professional training

- LinkedIn Learning
- Udemy

- CBT Nuggets

Show your appreciation!

Steve Brorens (@snori74) was a collector of postcards and enjoyed greatly all the "Snail Mail" he received from the students.

But since his passing there's nowhere to send postcards anymore. You can show your appreciation for the course by letting everyone else know how awesome it was! Recommend the course to other people, invite your friends to do the challenge together, have fun! Show the world you finished the challenge by posting about it on social media.

Contribute

Livia Lima is the one currently maintaining the material. But she's only one person and appreciates any help to keep this challenge running consistently every month, and available to everyone.

If you'd like to contribute, here a few things you can do:

- **Answer other students questions** in our channels. Help a friend through the challenge.
- **Correct typos, dead links, etc** by submitting a correction request to the source material.
- **Suggest improvements** by submitting a feature request to the source material.
- **Help moderate** Lemmy, Reddit or Discord. Are you a whiz in one (or more) of those platforms? Help admin them.
- **Support the infrastructure** by donating or sponsoring. The challenge is free but the website servers and the domains costs money, so we appreciate if you can spare a buck.

Thanks for all and happy linuxing!