Introduction to Linux

We're having an EGM!

Presenting today

Jason
Secretary

Current committee

Echo
President

Josh
Publicity

Keshav
Inclusions Officer

Abdelrhman
General Member

Luca
Treasurer

Erik
Competitions

Future committee

Could it be you?

We're electing a second **general member** and a **technical officer...**

We're having an EGM!

- Technical officer
  - Create, maintain, and update our practicals
  - Run technical sessions
  - Work in the website team to maintain our website
- General member
  - Plan, assist with, and run sessions
  - Help with projects
  - Support during sessions
  - Contribute to the newsletter

No experience necessary - learn on the job!
EGM on the 14th October
Sign up with the link on Discord, coming soon

The Legal Bit

- The skills taught in these sessions allow identification and exploitation of security vulnerabilities in systems. We strive to give you a place to practice legally, and can point you to other places to practice. These skills should not be used on systems where you do not have explicit permission from the owner of the system. It is VERY easy to end up in breach of relevant laws, and we can accept no responsibility for anything you do with the skills learnt here.

- If we have reason to believe that you are utilising these skills against systems where you are not authorised you will be banned from our events, and if necessary the relevant authorities will be alerted.

- Remember, if you have any doubts as to if something is legal or authorised, just don't do it until you are able to confirm you are allowed to.

- Relevant UK Law: https://www.legislation.gov.uk/ukpga/1990/18/contents

Code of Conduct

- Before proceeding past this point you must read and agree to our Code of Conduct - this is a requirement from the University for us to operate as a society.
- If you have any doubts or need anything clarified, please ask a member of the committee.
- Breaching the Code of Conduct = immediate ejection and further consequences.
- Code of Conduct can be found at shefesh.com/conduct

What is Linux?

- What is Linux?
    - An easy answer: Linux is an **operating system kernel**.

- What does that actually mean?
    - It is a program which carries out very basic core functionality to make the OS work.
    - It handles the allocation of memory, CPU-time and other system resources to **processes.**
    - It provides an **application programming interface** that those processes can call to do various things, most obviously communicating with system hardware.

You could say it's **a program that exists to support other programs**.
As such, it's not much use on its own...

So we have distributions

- To make Linux do anything useful, you need other software.

- What other software exactly?
  - It depends on what exactly you want to use the OS for!

- You'll almost always install a collection of software that includes Linux *and* everything else to form a fully functioning OS.

- These collections are called **Linux distributions** ("distros").
  - As well as the software they include, distributions tend to have **package management systems** which make it quite simple to add or remove software.

- Usually the term "Linux" actually means "a Linux distribution"
  - some people will argue this is an incorrect use of the term, and I suppose they're technically right, but it doesn't really matter

There are loads of distributions

- Linux is **free and open-source**
  - The authors allow the public to distribute it and make modified versions.
  - This means there are **lots of distributions**, many of them free.
  - It's hard to measure, but Linux is almost certainly the most popular OS kernel in the world - as well as PCs, you'll find it in supercomputers, network routers, mobile phones, digital signs, security cameras, vehicle control systems, fridge-freezers...

- Some common distros:
  - **Ubuntu Desktop** and **Fedora Workstation** - full graphical desktop OSes, suitable for replacing MacOS or Windows.
  - **Alpine** - aims to be as small and simple as possible - for use in embedded systems and containers with limited resources.
  - **Kali - a distro designed for pentesting** (we will use this often in ShefESH).

- Despite differences between distros, skills and experience are usually transferrable.

An aside about naming

- Although the name Linux refers to the kernel, often people will use it to mean the whole operating system. This very rarely causes a problem, because it's almost always obvious from the context whether you are talking about the kernel on its own or a complete OS.

- Nonetheless, some particularly pedantic users will point out that this is incorrect. Your OS isn't Linux, it's "Kali GNU/Linux". This, while technically true, is almost always irrelevant to the discussion at hand and really doesn't need to be pointed out.

- The needlessly-long-form name "Kali GNU/Linux" comes from the fact that the OS contains:
    - the **Linux** kernel
    - the **GNU Project,** a widely used collection of system software and libraries
    - various things provided by the **Kali** maintainers

- Final bit of trivia: the acronym GNU is recursive, standing for "GNU's Not UNIX!"
    - UNIX is another family of OSes, which Linux borrowed heavily from.

SSH

- You could **replace your current OS with Linux** - but we won't be covering that in this session
- Instead, we'll use the **secure shell (SSH)** protocol
- SSH allows you to log in as a user and access the command line of a remote computer over the network

Basic syntax of an SSH command:

```
ssh username@hostname
```

Let’s get connected

For details on how to connect, please see the screen in the room.

Let's get connected

You can now see a Bash prompt that you can type commands at.

Try typing **`help`** (then press enter)

This isn't the only way to use Bash - you can also write a sequence of commands in a file to create a **script** that you can run.

Filenames and paths

- There is a directory/folder tree structure, which may already be familiar to you from other operating systems, especially other Unix-like ones (incl. MacOS)

- There are a few more differences from Windows
  - The whole system has only one tree, with one root - there are no “drive letters”
  - We never use \ as a path separator, always **/**
  - Almost any character is *technically* allowed in a filename.

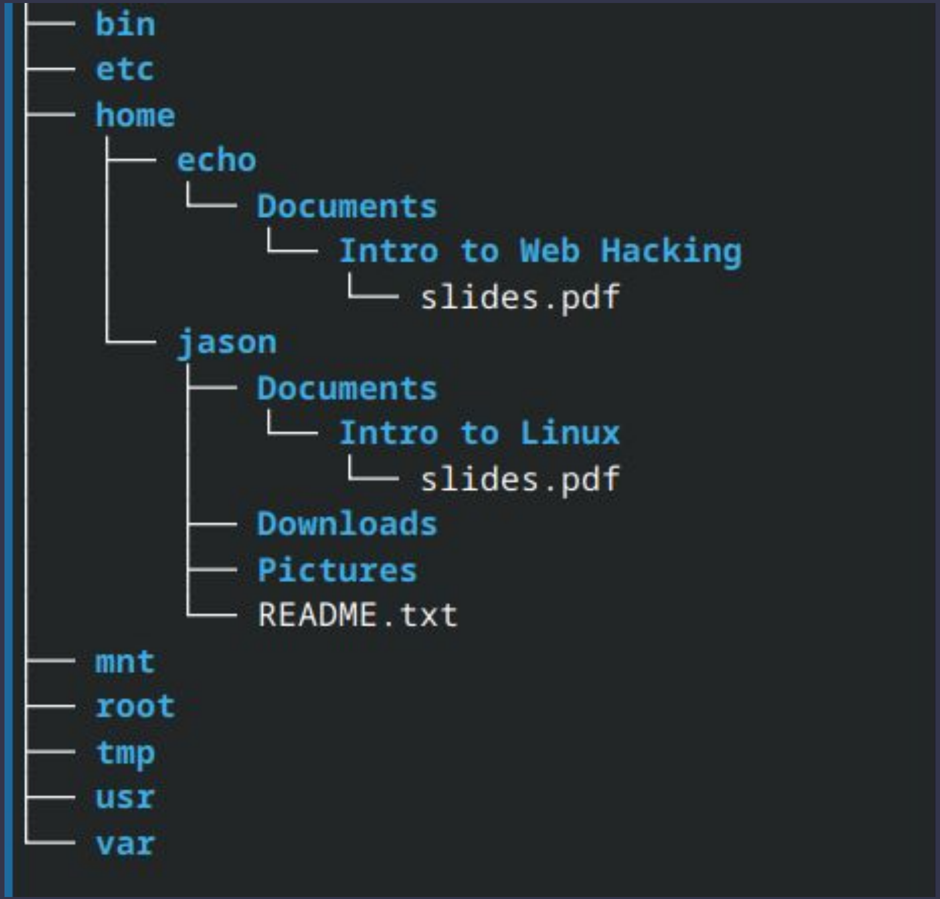
- **Working directory** (**.**) = “the directory we’re in at the moment”

- Your **home directory (~)** - stores your user-specific files

- If a filename begins with a **.** the file is considered “hidden” (this is **not** a security feature)

- Paths can be
  - Relative to the working directory
  - Relative to your home directory (starts with ~)
  - Absolute (starts with /)

Filenames and paths



```
├── bin
├── etc
├── home
│   ├── echo
│   │   └── Documents
│   │       └── Intro to Web Hacking
│   │           └── slides.pdf
│   └── jason
│       ├── Documents
│       │   └── Intro to Linux
│       │       └── slides.pdf
│       ├── Downloads
│       ├── Pictures
│       └── README.txt
├── mnt
├── root
├── tmp
├── usr
└── var
```

1. If **jason** is logged in, what is the absolute path of **~/Documents**?

2. If the working directory is **/home**, what is the relative path to Echo's web hacking slides?

3. If the working directory is **/mnt**, what is the relative path to README.txt?

`pwd` - **p**rint **w**orking **d**irectory

- **`pwd`** you what the working directory currently is
- Immediately after logging in, this will be your **home directory**

`cd path` - **c**hange **d**irectory

- Change the working directory to something else

`ls [path]` - **l**i**s**t directory

- List everything in the directory
- Use the option **`-a`** or **`--all`** to include hidden files

`mkdir path` - **m**a**k**e **dir**ectory

- Create a new directory

mv path_from path_to - **m**o**v**e

- **path_to** can either be a directory or a file
- renaming is moving
- use **-r** to move directories

cp path_from path_to - **c**o**p**y

- Syntax is the same is **mv**

rm path - **r**e**m**ove

- You can’t *easily* recover an **rm**ed file, but this is not a secure erase
- use **-r** to remove directories recursively

Look in Echo’s home directory.

Can you find the title of the next session they are running?

Some interesting directories

`/bin`	“essential user command **bin**aries” - often now just a symbolic link to /usr/bin
`/boot`	for the use of the **boot**loader
`/dev`	**dev**ice files
`/etc`	**e**ditable **t**ext **c**onfiguration files - you can say “etcetera”
`/home`	user **home** directories
`/lib`	shared **lib**raries - often now just a symbolic link to /usr/lib
`/mnt`	filesystem **m**ou**nt** points
`/media`	like `/mnt` but specifically for removable **media**
`/root`	**root**’s home directory
`/sbin`	**bin**aries to be run as root (**s**uperuser)
`/tmp`	**t**e**mp**orary (“programs must not assume that any files or directories in /tmp are preserved between invocations of the program”)
`/usr`	read-only data shareable between **us**e**r**s (e.g. applications installed system-wide)

You don’t have to remember all of this!

Filesystem Hierarchy Standard: https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html

`echo` `text` – “display line to standard output”

- The “standard output” is a term that describes wherever the output from a program goes by default
- For our purposes, it means your SSH shell
- For those of you learning Java: the standard output is represented by **`System.out`**
    - so **`echo`** is effectively **`System.out.println`**

`cat` `file` – “con**cat**enate to standard output”

- lets you read the contents of a file

`nano` `file` - **ano**ther text editor

- A successor to “pico” - hence the strange mnemonic.
- On a few systems, Nano is not available, so try `vi` - but it’s much less user-friendly

Change the standard output

a | b

the stdout from command **a** is used as an input to command **b**

a >> b

the stdout from command **a** is appended to file **b**

a > b

the stdout from command **a** is written to file **b,** replacing the existing content

Useful places to redirect to

- pipe to **`grep`** - to filter the output. Too many options to list here, but try **`man grep`**
- pipe to **`less`** - when you want to read the whole output, but it won't fit on the screen (e.g. when **`cat`**ing a very long file)
- redirect to **/dev/null** - when you don't care about the output at all!

/dev/null is a special "device" that just throws away anything you send to it.

There are other such special files - try **`cat /dev/random`** to look like a cool hacker!!!!!!!1!!!!1!!!!
(or even **`cat /dev/random > /dev/null`** for a thoroughly pointless use of CPU time).

There’s a message hidden in the Network Manager configuration file, on a line beginning “secret”

The first person to give me the secret word wins a sticker.

Try to **cd** into your neighbour’s home directory. What happens?

Permissions

- You’re already familiar with the concept of a **user.**

- A user can belong to one or more **groups**
  - Groups are defined in the file **`/etc/group`**

- Each file or directory has an **owner** (which is a user) and an associated group.

- There are three basic things you can do to a file:
  - Read
  - Write
  - Execute

- The special user **root** is the “superuser”, and can do anything to any file
  - To avoid accidentally carrying out harmful actions, do not log in as root unless you have to.
  - Often, systems enforce this rule by disabling the root account.

- What others can do depends on the **mode bits** of the file.

`chown user[:group] path` - **ch**ange file **own**er

- Change the user and group that owns **path**.

`chmod mode path` - **ch**ange file **mod**e bits

- Changing the file mode bits changes permissions.

- **mode** can take a few different forms. The most intuitive is the shorthand symbolic form, which consists of 2 to 5 characters
  - Whose permissions you want to change: **u** = owning user, **g** = group, **o** = others, blank = all
  - Operator: **+** to grant, **-** to remove, **=** to set
  - Which permissions - **r** = read, **w** = write, **x** = execute

- Examples of modes:
  - `+rw` grant everyone permission to read and write
  - **`u+x`** grant the owning user permission to execute
  - **`g=rw`** set the permissions for the group to allow reading and writing but not executing
  - **`-x`** remove everyone's permission to execute

```
$ echo “hello” > README
$ chown jason:shefesh_committee README
$ chmod =r README
$ chmod u+w README
```

- Who can read the file “README”?
- Who can write to it?
- Who can execute it?

`man` – read **<u>man</u>**ual page

- Often you'll get the answer you need quicker with a web search - but it's still useful to know about

`find` - search in filesystem

- Lots of different things you can do
- Try typing `man find`

`which command` - get full path of command

- This does the same path-finding process as would happen if you actually tried to run the command - but instead of running whatever it finds, it just outputs the absolute path

`help` - get information on how to use the shell

- Some useful summary documentation, particularly if you are writing scripts

Useful common flags

- **`--version`**
  - Output software version number, and sometimes copyright/licensing information
- **`--verbose`**
  - Output more detailed information than normal (e.g. a line for every file processed)
- **`--recursive`**
  - Operate on files and directories recursively
- **`--force`**
  - Override a restriction, such as a requirement for confirmation
- **`--help`**
  - Provide some sort of information about how to use the program

Often you can use the shortened forms **-r** for recursive and **-f** for force.
This is where the famous command “**`rm -rf /`**”, which means “delete everything on the system”, comes from!

1. Delete the file in your home directory called DELETE_ME.
2. Rename the file in your home directory called RENAME_ME.
3. Find another file called DELETE_ME and delete that too.
4. Use a command to output your name to the console.
5. Use a command to output your name to a new text file in your Documents directory.
6. Edit the file you created in part 5 to include a fact about yourself, such as your favourite food.
7. Modify the file you created in part 5 so that your neighbour is able make a copy of it.
8. Make a copy of your neighbour's file that they created in part 5, and then read their fact.
9. Here are some common utilities. Get the version number and the absolute path to the executable for each.
   a. **wget** - to download files from the internet
   b. **tar** - to create or extract from an "archive", which is one file that bundles together several files
   c. **sha512sum** - to compute and verify SHA512 checksums
10. Write a Bash script that downloads the file at shefesh.com/rp/intro_to_linux_24.tar, checks that its SHA512 checksum matches the one in the file **/usr/share/intro_to_linux/SHA512SUM**, and, only if it does match, outputs the content of the file within the archive called README.
11. Find a way of announcing "I'm done" to everyone else logged into the system.

Don't be afraid to Google things, or use `man` and `help`

Upcoming Sessions

What's up next?

www.shefesh.com/sessions

Open-source intelligence (OSINT)
14th October
(**EGM** during this session)

Password cracking
21st October

Bad USB
28th October

Any Questions?

www.shefesh.com

Thanks for coming!