APT Session 1: Unix

Laurence
Tratt

Software Development Team
2018-10-12

House rules

1. **Respect each other**

House rules

1. **Respect each other**
   - Be on time.
   - No phones.
   - No talking over others.

House rules

1. **Respect each other**
   - Be on time.
   - No phones.
   - No talking over others.
2. **Respect yourself**

House rules

1. **Respect each other**
   - Be on time.
   - No phones.
   - No talking over others.
2. **Respect yourself**
   - There are no silly questions so...
   - ...*ask questions*.

House rules

1. **Respect each other**
   - Be on time.
   - No phones.
   - No talking over others.
2. **Respect yourself**
   - There are no silly questions so...
   - ...*ask questions*.
3. **Respect the subject**

House rules

1. **Respect each other**
   - Be on time.
   - No phones.
   - No talking over others.
2. **Respect yourself**
   - There are no silly questions so...
   - ...*ask questions*.
3. **Respect the subject**
   - Participate when asked.

House rules

1. **Respect each other**
   - Be on time.
   - No phones.
   - No talking over others.
2. **Respect yourself**
   - There are no silly questions so...
   - ...*ask questions*.
3. **Respect the subject**
   - Participate when asked.
   - Ask questions of yourself: what? why? how?
   - Be prepared to correct me (politely).

Who am I?

- Reader at King's (where I did my BSc and PhD).
- Leader, Software Development Team.
- Research into programming language design and implementation.

Who am I?

- Reader at King's (where I did my BSc and PhD).
- Leader, Software Development Team.
- Research into programming language design and implementation.
- Designed and implemented the Converge programming language.
- Released lots of software in C, Java, Objective-C, Python, Rust and others including: email_merger, extsmail, Converge, multitime, srep, supuner, xcage.

Who am I?

- Reader at King's (where I did my BSc and PhD).
- Leader, Software Development Team.
- Research into programming language design and implementation.
- Designed and implemented the Converge programming language.
- Released lots of software in C, Java, Objective-C, Python, Rust and others including: email_merger, extsmail, Converge, multitime, srep, supuner, xcage.
- Homepage: http://tratt.net/laurie/ Twitter: @laurencetratt

The volunteers

Each week we're lucky that two members of the Software Development Team have given up their time to help out with APT.

The volunteers

Each week we're lucky that two members of the Software Development Team have given up their time to help out with APT.

Edd Barrett.

- Led work on the Unipycation, PyHyp, and Krun systems.
- Currently working on a hardware-tracer accelerated meta-tracer.

The volunteers

Each week we're lucky that two members of the Software Development Team have given up their time to help out with APT.

Edd Barrett.

- Led work on the Unipycation, PyHyp, and Krun systems.
- Currently working on a hardware-tracer accelerated meta-tracer.

and Lukas Diekmann:

- Leads work on the Eco language composition editor.

What is APT?

What is APT?

- Advanced Practical Topics (APT).

What is APT?

- Advanced Practical Topics (APT).
- Optional:

What is APT?

- Advanced Practical Topics (APT).
- Optional: doesn’t count towards your degree.

What is APT?

- Advanced Practical Topics (APT).
- Optional: doesn’t count towards your degree.
- Why be here?

What is APT?

- Advanced Practical Topics (APT).
- Optional: doesn't count towards your degree.
- Why be here?
  - You are truly interested in computers and software.
  - You want to get a picture of the wider landscape.

What is APT?

- Advanced Practical Topics (APT).
- Optional: doesn't count towards your degree.
- Why be here?
  - You are truly interested in computers and software.
  - You want to get a picture of the wider landscape.
  - You want a wide range of experience for your future life/jobs.

What is APT?

- Advanced Practical Topics (APT).
- Optional: doesn't count towards your degree.
- Why be here?
  - You are truly interested in computers and software.
  - You want to get a picture of the wider landscape.
  - You want a wide range of experience for your future life/jobs.
- How does it work?

What is APT?

- Advanced Practical Topics (APT).
- Optional: doesn't count towards your degree.
- Why be here?
  - You are truly interested in computers and software.
  - You want to get a picture of the wider landscape.
  - You want a wide range of experience for your future life/jobs.
- How does it work?
  - Every session has a topic e.g.: Unix, Python, C, version control.

What is APT?

- Advanced Practical Topics (APT).
- Optional: doesn't count towards your degree.
- Why be here?
  - You are truly interested in computers and software.
  - You want to get a picture of the wider landscape.
  - You want a wide range of experience for your future life/jobs.
- How does it work?
  - Every session has a topic e.g.: Unix, Python, C, version control.
  - BYOL (Bring Your Own Laptop).
  - You will sometimes be asked to preinstall some software, but otherwise you can go at your own pace during the class, following the slides. *We're here to help when you're stuck on, or want to know more about, one of the topics / questions.*

What is APT?

- Advanced Practical Topics (APT).
- Optional: doesn't count towards your degree.
- Why be here?
  - You are truly interested in computers and software.
  - You want to get a picture of the wider landscape.
  - You want a wide range of experience for your future life/jobs.
- How does it work?
  - Every session has a topic e.g.: Unix, Python, C, version control.
  - BYOL (Bring Your Own Laptop).
  - You will sometimes be asked to preinstall some software, but otherwise you can go at your own pace during the class, following the slides. *We're here to help when you're stuck on, or want to know more about, one of the topics / questions.*
  - ***Participation is mandatory***.

What is APT?

- Advanced Practical Topics (APT).
- Optional: doesn't count towards your degree.
- Why be here?
  - You are truly interested in computers and software.
  - You want to get a picture of the wider landscape.
  - You want a wide range of experience for your future life/jobs.
- How does it work?
  - Every session has a topic e.g.: Unix, Python, C, version control.
  - BYOL (Bring Your Own Laptop).
  - You will sometimes be asked to preinstall some software, but otherwise you can go at your own pace during the class, following the slides. *We're here to help when you're stuck on, or want to know more about, one of the topics / questions.*
  - ***Participation is mandatory***. [Optionally in teams of upto 3.]

What to expect from this session: Unix

1. What is Unix?
2. Install OpenBSD using VirtualBox.

What to expect from this session: Unix

1. What is Unix?
2. Install OpenBSD using VirtualBox.
3. Basic command-line techniques.

What to expect from this session: Unix

1. What is Unix?
2. Install OpenBSD using VirtualBox.
3. Basic command-line techniques.
4. Customising the OS.

Prerequisites

You should have:

1. Turned virtualization support on in your laptop's BIOS. The option may be called "virtualization technology", "VT-X" or "AMD-V". [Most Macs have this enabled already; so do Win8/10 machines; and so do AMD machines. Other Intel machines need checking.]
2. Downloaded and installed VirtualBox https://www.virtualbox.org/wiki/Downloads
3. Downloaded either:
   - http://mirror.bytemark.co.uk/OpenBSD/6.3/i386/install63.iso (if you're running a 32 bit OS)
   - http://mirror.bytemark.co.uk/OpenBSD/6.3/amd64/install63.iso (if you're running a 64 bit OS — if in doubt, assume you're running a 64 bit OS)
4. Ensured your laptop can connect to one of the College's wireless networks.

What is Unix? (simplified)

- Unix is an operating system.
- It abstracts away from most hardware details and helps manage the software we run.
- e.g. manages your files; runs programs for you; provides a connection to the internet.

What is Unix? (simplified)

- Unix is an operating system.
- It abstracts away from most hardware details and helps manage the software we run.
- e.g. manages your files; runs programs for you; provides a connection to the internet.
- Other operating systems include Windows.

What is Unix? (simplified)

- Unix is an operating system.
- It abstracts away from most hardware details and helps manage the software we run.
- e.g. manages your files; runs programs for you; provides a connection to the internet.
- Other operating systems include Windows.
- First public release 1973. [Windows first release: 1985]

Why bother with a 45 year old operating system?

Why bother with a 45 year old operating system?

- It’s turned out to be extremely flexible.

Why bother with a 45 year old operating system?

- It’s turned out to be extremely flexible.
- It’s running your Android phone; it’s the core of OS X (and iOS).

Why bother with a 45 year old operating system?

- It’s turned out to be extremely flexible.
- It’s running your Android phone; it’s the core of OS X (and iOS).
- It’s been the #1 choice for serious software developers for most of those 45 years.

Why bother with a 45 year old operating system?

- It’s turned out to be extremely flexible.
- It’s running your Android phone; it’s the core of OS X (and iOS).
- It’s been the #1 choice for serious software developers for most of those 45 years.
- Unix users adapt easily to other OSs. The reverse is not true.

Why bother with a 45 year old operating system?

- It’s turned out to be extremely flexible.
- It’s running your Android phone; it’s the core of OS X (and iOS).
- It’s been the #1 choice for serious software developers for most of those 45 years.
- Unix users adapt easily to other OSs. The reverse is not true.
- Unix is free to download, use, and adapt.

Unix and Linux and BSD and ...

Unix and Linux and BSD and ...

- There is no one true Unix: UNIX, Linux, FreeBSD, OpenBSD, Solaris, and many others.
- When someone says ‘Unix’ they mean the generic family of related OSs.

Unix and Linux and BSD and ...

- There is no one true Unix: UNIX, Linux, FreeBSD, OpenBSD, Solaris, and many others.
- When someone says ‘Unix’ they mean the generic family of related OSs.
- The best known ‘pure’ Unix is Linux.
- OS X has a (slightly out-of-date) Unix core, but adds a non-Unix GUI.

Unix and Linux and BSD and ...

- There is no one true Unix: UNIX, Linux, FreeBSD, OpenBSD, Solaris, and many others.
- When someone says ‘Unix’ they mean the generic family of related OSs.
- The best known ‘pure’ Unix is Linux.
- OS X has a (slightly out-of-date) Unix core, but adds a non-Unix GUI.
- Today we will be looking at OpenBSD because:
  - It’s small, highly consistent, and well documented.
  - It has a strong emphasis on security.

Unix and Linux and BSD and ...

- There is no one true Unix: UNIX, Linux, FreeBSD, OpenBSD, Solaris, and many others.
- When someone says ‘Unix’ they mean the generic family of related OSs.
- The best known ‘pure’ Unix is Linux.
- OS X has a (slightly out-of-date) Unix core, but adds a non-Unix GUI.
- Today we will be looking at OpenBSD because:
  - It’s small, highly consistent, and well documented.
  - It has a strong emphasis on security.
  - It’s been my favourite OS since 1999.

How to run it

How to run it

Four main options:

1. Wipe your machine and install Unix on the disk.
2. Have Unix installed alongside another OS.

How to run it

Four main options:

1. Wipe your machine and install Unix on the disk.
2. Have Unix installed alongside another OS.
3. Buy a Unix ‘virtual machine’ on the internet.
4. Install your own virtual machine and install it there.

[‘Virtual Machine’ is an overloaded term. In this context: software which can run a full OS inside another OS.]

OS-level virtual machines

- There are several OS-level VMs e.g.: qemu, VirtualBox.
- They run as normal(ish) software on a *host* OS.

OS-level virtual machines

- There are several OS-level VMs e.g.: qemu, VirtualBox.
- They run as normal(ish) software on a *host* OS.
- Each can run multiple *guest* OSs.
- Each is given RAM to work in and disk space.
- We can also take an ISO file (i.e. a rip of a CD) and make it appear to the guest OS as a CD.
- After that, the OS-inside-an-OS should work as ‘normal’.

Setting up VirtualBox (1)

- Setting up VirtualBox looks intimidating, but it's simple if you follow these steps (helped by the fact that, in most cases, VirtualBox selects sensible defaults).

Setting up VirtualBox (1)

- Setting up VirtualBox looks intimidating, but it's simple if you follow these steps (helped by the fact that, in most cases, VirtualBox selects sensible defaults).
- Load VirtualBox.
- Click 'New' to create a new VM. Give it the name 'openbsd' (lower case), which will automatically configure some other options for you. Click 'Next'. Give your VM at least 1024MiB/1GiB RAM (if your laptop has more than 4GiB RAM, give the VM 2048MiB/2GiB). Click 'Next'.
- You want to 'Create a virtual hard disk now' so click 'Create', select 'VDI (VirtualBox Disk Image)', click 'Next', select 'Dynamically allocated', click 'Next' and give 16GiB to the virtual disk. Click 'Next'.

Setting up VirtualBox (2)

- You should be back at the main VirtualBox window, with an OpenBSD VM on the left hand side.
- Right click on the OpenBSD VM and select ‘Settings’. Click on ‘Storage’ and click on the CD icon (which should say ‘Empty’). On the right hand side it should now say ‘Optical drive: IDE Secondary Master’ with a little CD icon to the right of that text. Click on the CD icon and select ‘Choose Virtual Optical Disk’. Now select the `install63.iso` file you downloaded earlier (this is, virtually, equivalent to putting a CD with the `install63.iso` burnt onto it into a real CD drive). Click ‘OK’.

Setting up VirtualBox (2)

- You should be back at the main VirtualBox window, with an OpenBSD VM on the left hand side.
- Right click on the OpenBSD VM and select ‘Settings’. Click on ‘Storage’ and click on the CD icon (which should say ‘Empty’). On the right hand side it should now say ‘Optical drive: IDE Secondary Master’ with a little CD icon to the right of that text. Click on the CD icon and select ‘Choose Virtual Optical Disk’. Now select the `install63.iso` file you downloaded earlier (this is, virtually, equivalent to putting a CD with the `install63.iso` burnt onto it into a real CD drive). Click ‘OK’.
- You’ve now configured VirtualBox!

Installing OpenBSD (1)

- You should be back at the main VirtualBox window.
- Select the OpenBSD VM and click ‘start’. OpenBSD will now boot inside your VM. If you click the mouse inside VirtualBox then the VM will ‘take over’ the mouse. **Note carefully the message that tells you which button you need to press to return the mouse back to your main OS!**

Installing OpenBSD (1)

- You should be back at the main VirtualBox window.
- Select the OpenBSD VM and click ‘start’. OpenBSD will now boot inside your VM. If you click the mouse inside VirtualBox then the VM will ‘take over’ the mouse. **Note carefully the message that tells you which button you need to press to return the mouse back to your main OS!**
- OpenBSD’s installer is a simple text-only affair. There are quite a few stages to go through in the installer, and they can seem arbitrary and confusing. Fortunately, the defaults are nearly always what we want, and this is a one-off task, so don’t worry *too* much if some of what you’re about to do seems arbitrary.

Installing OpenBSD (1)

- You should be back at the main VirtualBox window.
- Select the OpenBSD VM and click ‘start’. OpenBSD will now boot inside your VM. If you click the mouse inside VirtualBox then the VM will ‘take over’ the mouse. **Note carefully the message that tells you which button you need to press to return the mouse back to your main OS!**
- OpenBSD’s installer is a simple text-only affair. There are quite a few stages to go through in the installer, and they can seem arbitrary and confusing. Fortunately, the defaults are nearly always what we want, and this is a one-off task, so don’t worry *too* much if some of what you’re about to do seems arbitrary.
- Once OpenBSD has booted, it will say `(I)nstall, (U)pgrade, (A)utoinstall, (S)hell`. The letters between brackets can be used instead of typing the whole word. Press ‘I’ for install and then press ‘return’.

Installing OpenBSD (2)

- You are now in ‘install’ mode. The first question asks you to select a keyboard. You can generally get away with just pressing ‘return’ here, but if you have an unusual keyboard, try typing in the appropriate 2 character country name in lower case (I type ‘uk’ as I have a UK keyboard) and then press ‘return’.

Installing OpenBSD (2)

- You are now in ‘install’ mode. The first question asks you to select a keyboard. You can generally get away with just pressing ‘return’ here, but if you have an unusual keyboard, try typing in the appropriate 2 character country name in lower case (I type ‘uk’ as I have a UK keyboard) and then press ‘return’.
- When asked for the system hostname enter ‘test’ (this name isn’t very important for us, but it does make a bit of difference for servers on the internet).

Installing OpenBSD (2)

- You are now in ‘install’ mode. The first question asks you to select a keyboard. You can generally get away with just pressing ‘return’ here, but if you have an unusual keyboard, try typing in the appropriate 2 character country name in lower case (I type ‘uk’ as I have a UK keyboard) and then press ‘return’.
- When asked for the system hostname enter ‘test’ (this name isn’t very important for us, but it does make a bit of difference for servers on the internet).
- You’ll now be asked to configure the network. The defaults (between square brackets), in this order, are all acceptable: `em0` (making your VM thinking it’s connected via ethernet to the outside world), `dhcp` (automatically obtaining a connection to the internet; you can see the installer finding you an IP address), `none` (we don’t need IPv6 today), `done` (we only need 1 network adapter), and `my.domain` (this isn’t important for us).

Installing OpenBSD (3)

- You'll now be prompted for the root password. *Don't enter arbitrary stuff here or we can't help you later!* Use `aptpass` as the password (you'll be asked to confirm it a second time). [This is an awful password, but please use it so that we can more easily help you in this session.]

Installing OpenBSD (3)

- You'll now be prompted for the root password. *Don't enter arbitrary stuff here or we can't help you later!* Use `aptpass` as the password (you'll be asked to confirm it a second time). [This is an awful password, but please use it so that we can more easily help you in this session.]
- Answer 'yes' to 'Do you want to start `sshd`' and to using the 'X Window System'. Also answer 'yes' (notice that this isn't the default!) to 'Do you want the X Window System to be started by xenodm(1)' (a very simple GUI login manager).

Installing OpenBSD (3)

- You'll now be prompted for the root password. *Don't enter arbitrary stuff here or we can't help you later!* Use `aptpass` as the password (you'll be asked to confirm it a second time). [This is an awful password, but please use it so that we can more easily help you in this session.]
- Answer 'yes' to 'Do you want to start `sshd`' and to using the 'X Window System'. Also answer 'yes' (notice that this isn't the default!) to 'Do you want the X Window System to be started by xenodm(1)' (a very simple GUI login manager).
- You'll now be asked to set up a user. Choose whatever username you like (keep it clean!), but use `aptpass` for the password again.

Installing OpenBSD (3)

- You'll now be prompted for the root password. *Don't enter arbitrary stuff here or we can't help you later!* Use `aptpass` as the password (you'll be asked to confirm it a second time). [This is an awful password, but please use it so that we can more easily help you in this session.]
- Answer 'yes' to 'Do you want to start `sshd`' and to using the 'X Window System'. Also answer 'yes' (notice that this isn't the default!) to 'Do you want the X Window System to be started by xenodm(1)' (a very simple GUI login manager).
- You'll now be asked to set up a user. Choose whatever username you like (keep it clean!), but use `aptpass` for the password again.
- Answer 'no' to 'Allow root SSH login' (it's a security problem waiting to happen). Your timezone is 'GB'.

Installing OpenBSD (4)

- We now have to configure disk storage. The root disk is `wd0`, you want to use the `Whole` disk, and then accept the `Auto layout`. The installer will then format your virtual hard disk appropriately.

Installing OpenBSD (4)

- We now have to configure disk storage. The root disk is `wd0`, you want to use the `Whole` disk, and then accept the `Auto layout`. The installer will then format your virtual hard disk appropriately.
- We now have to copy the operating system from the virtual CD to the virtual hard disk. The sets are stored on `cd0`, the default pathname (e.g. `6.3/amd64`) is OK, and we want to install all the sets (so `done` is what we need).
- You will be warned that `Directory does not contain SHA256.sig. Continue without verification?` – answer `yes` (OpenBSD doesn't use crypto on the CD installs for historical reasons). The installer will then start copying files across. Once that's complete, we're `done` with set locations.

Installing OpenBSD (4)

- We now have to configure disk storage. The root disk is `wd0`, you want to use the `Whole` disk, and then accept the `Auto layout`. The installer will then format your virtual hard disk appropriately.
- We now have to copy the operating system from the virtual CD to the virtual hard disk. The sets are stored on `cd0`, the default pathname (e.g. `6.3/amd64`) is OK, and we want to install all the sets (so `done` is what we need).
- You will be warned that `Directory does not contain SHA256.sig. Continue without verification?` – answer `yes` (OpenBSD doesn't use crypto on the CD installs for historical reasons). The installer will then start copying files across. Once that's complete, we're `done` with set locations.
- You might be asked to set the time of the VM: `yes`.

Installing OpenBSD (4)

- Congratulations – OpenBSD is installed! You now need to `Halt` and wait for the message `The operating system has halted to appear.` You can then close the VirtualBox window, selecting ‘Power off the machine’ (this is safe, as OpenBSD’s installer has shut the OS down, but can’t quite ‘power down’ the VM).

Installing OpenBSD (4)

- Congratulations – OpenBSD is installed! You now need to `Halt` and wait for the message `The operating system has halted to appear.` You can then close the VirtualBox window, selecting 'Power off the machine' (this is safe, as OpenBSD's installer has shut the OS down, but can't quite 'power down' the VM).
- We now need to eject the virtual CD from VirtualBox, otherwise when we run the VM again, we'll go back to the installer. Right click on the OpenBSD VM in VirtualBox , go to 'Settings', and then 'Storage'. Select `install63.iso` then go to the right hand CD icon and select 'Remove from Virtual Drive'.

Installing OpenBSD (4)

- Congratulations – OpenBSD is installed! You now need to `Halt` and wait for the message `The operating system has halted to appear.` You can then close the VirtualBox window, selecting 'Power off the machine' (this is safe, as OpenBSD's installer has shut the OS down, but can't quite 'power down' the VM).
- We now need to eject the virtual CD from VirtualBox, otherwise when we run the VM again, we'll go back to the installer. Right click on the OpenBSD VM in VirtualBox , go to 'Settings', and then 'Storage'. Select `install63.iso` then go to the right hand CD icon and select 'Remove from Virtual Drive'.
- Start the OpenBSD VM. You're now running 'normal' OpenBSD!

Logging in and immediately afterwards

- Your OpenBSD VM will boot into `xenodm`, a simple login manager. Use the username and password you set up earlier.

Logging in and immediately afterwards

- Your OpenBSD VM will boot into `xenodm`, a simple login manager. Use the username and password you set up earlier.
- After you login, you're in an `xterm`, which shows you the *shell*.
- Execute commands, see their results etc.

Logging in and immediately afterwards

- Your OpenBSD VM will boot into `xenodm`, a simple login manager. Use the username and password you set up earlier.
- After you login, you're in an `xterm`, which shows you the *shell*.
- Execute commands, see their results etc.
- There are many shells. OpenBSD's default is `ksh`.
- We'll treat it as a given for the time being.

Logging in and immediately afterwards

- Your OpenBSD VM will boot into `xenodm`, a simple login manager. Use the username and password you set up earlier.
- After you login, you're in an `xterm`, which shows you the *shell*.
- Execute commands, see their results etc.
- There are many shells. OpenBSD's default is `ksh`.
- We'll treat it as a given for the time being.
- Minimal keyboard shortcuts:

↑ / ↓	Cycle through previous commands
Ctrl+A	Move cursor to beginning of line
Ctrl+E	Move cursor to end of line

Unix filesystem

- A *path* gives the location of a directory/file.
- The root path is at '`/`'. [Every other directory/file is a subdirectory/file of the root.]
- Paths starting with `/` are *absolute*; all other paths are *relative*.
- The shell always knows what the 'current working directory' is.
- Useful commands:

`cd x`	changes directory to `x`.
`ls`	displays the current directories contents (`ls -l` for detailed output).
`less x`	displays the contents of `x` ('*q*' quits `less`).
`pwd`	prints out the current working directory.

Unix filesystem

- A *path* gives the location of a directory/file.
- The root path is at '`/`'. [Every other directory/file is a subdirectory/file of the root.]
- Paths starting with `/` are *absolute*; all other paths are *relative*.
- The shell always knows what the 'current working directory' is.
- Useful commands:

`cd x`	changes directory to `x`.
`ls`	displays the current directories contents (`ls -l` for detailed output).
`less x`	displays the contents of `x` ('*q*' quits `less`).
`pwd`	prints out the current working directory.

Exercises:

1. What is your current working directory after logging in?
2. How many entries are there in the root directory?

Standard filesystem layout

There is a semi-standard layout:

`/bin/`	program binaries
`/dev/`	special files for interacting with hardware
`/etc/`	configuration files
`/lib/`	libraries
`/usr/local/`	locally installed software
`/tmp/`	temporary directory for everyone
`/var/`	storage area for long-running server programs

Help

- OpenBSD is extensively documented.
- `man x` gives you the documentation for command `x` (the viewer is `less`, so you can quit it with '`q`').

Help

- OpenBSD is extensively documented.
- `man x` gives you the documentation for command `x` (the viewer is `less`, so you can quit it with '`q`').
- `man -k s` searches program names and brief descriptions for the string '`s`'.
- Never rule out Google as a source of help.

Help

- OpenBSD is extensively documented.
- `man x` gives you the documentation for command `x` (the viewer is `less`, so you can quit it with '`q`').
- `man -k s` searches program names and brief descriptions for the string '`s`'.
- Never rule out Google as a source of help.

Exercises:

1. Have a look at the man page for `ls`.
2. List all entries in the root directory in 'long format' (i.e. with dates and times).

Manipulating directories and files

Useful commands:

`cp x y`	copies the file `x` to `y` (overwriting `y` if it existed).
`cp -r x y`	copies the directory `x` to `y` (putting `x` into `y` if the latter already existed).
`mkdir x`	creates a directory called `x`.
`mv x y`	renames the file/directory `x` to `y` (putting `x` into `y` if the latter already existed).
`rm x`	deletes a file called `x`.
`rm -r x`	deletes a directory called `x`.
`touch x`	creates a blank file called `x`.

Manipulating directories and files

Useful commands:

`cp x y`	copies the file `x` to `y` (overwriting `y` if it existed).
`cp -r x y`	copies the directory `x` to `y` (putting `x` into `y` if the latter already existed).
`mkdir x`	creates a directory called `x`.
`mv x y`	renames the file/directory `x` to `y` (putting `x` into `y` if the latter already existed).
`rm x`	deletes a file called `x`.
`rm -r x`	deletes a directory called `x`.
`touch x`	creates a blank file called `x`.

Exercises:

1. Create a blank file in `/tmp/` called `apt`.
2. What happens if you try and create a directory of the same name?
3. Rename `apt` to `apto`.
4. Delete `apto`.

Chaining Unix commands

- How did you count #files in the root directory?

Chaining Unix commands

- How did you count #files in the root directory? `ls | wc -l`

Chaining Unix commands

- How did you count #files in the root directory? `ls | wc -l`
- Unix command-line programs read text from `stdin` and write to `stdout` (errors go to `stderr`). Chain one command's `stdout` to the next's `stdin` with '|'.

Chaining Unix commands

- How did you count #files in the root directory? `ls | wc -l`
- Unix command-line programs read text from `stdin` and write to `stdout` (errors go to `stderr`). Chain one command's `stdout` to the next's `stdin` with '|'.

Useful commands:

`cat x`	writes the contents of the file `x` to `stdout`.
`less`	without a pathname specified, displays the contents of `stdin`.
`sort`	read and sorts `stdin`'s contents, writing them to `stdout`.
`wc -l`	writes the number of lines `stdin` contains to `stdout`.

Exercises:

1. How many words are in `/usr/share/dict/words`?
2. Sort the contents of `/etc/passwd` and scroll through them.

Other `stdin` / `stdout` manipulators

`x	y`	chain `x`'s `stdout` to `y`'s `stdin`.
`x > y`	`x`'s `stdout` is written to a file called `y` (and not to the terminal). `y` is overwritten if it previously existed.
`x >> y`	`x`'s `stdout` is appended to a file called `y` (and not to the terminal). `y` is created if it did not exist.
`x < y`	`x`'s `stdin` now reads from a file called `y` (and not from the terminal).

Process manipulation

- What happens if you execute `cat /dev/zero`?

Process manipulation

- What happens if you execute `cat /dev/zero`?
- This is a *foreground process* that's out of control.
- We can ask a foreground process to exit by pressing `Ctrl-C`.

Process manipulation

- What happens if you execute `cat /dev/zero`?
- This is a *foreground process* that's out of control.
- We can ask a foreground process to exit by pressing `Ctrl-C`.
- We can suspend a foreground process by pressing `Ctrl-Z`.
- `bg` then puts that process in the background so we can execute others. We can return it to the foreground with `fg`.

Process manipulation

- What happens if you execute `cat /dev/zero`?
- This is a *foreground process* that's out of control.
- We can ask a foreground process to exit by pressing `Ctrl-C`.
- We can suspend a foreground process by pressing `Ctrl-Z`.
- `bg` then puts that process in the background so we can execute others. We can return it to the foreground with `fg`.

Exercises:

1. Execute `cat /dev/zero`, and suspend it.
2. Put it in the background, run `top` ('`q`' quits `top`) to see what processes are active.
3. Put the command back to the foreground then ask it to exit.

What's going on?

- If your machine seems slow, what might be the cause?

What's going on?

- If your machine seems slow, what might be the cause?
- Probably there are processes that are consuming processor time that you're not aware of.

What's going on?

- If your machine seems slow, what might be the cause?
- Probably there are processes that are consuming processor time that you're not aware of.
- The easiest thing is to run `top` and see.

What's going on?

- If your machine seems slow, what might be the cause?
- Probably there are processes that are consuming processor time that you're not aware of.
- The easiest thing is to run `top` and see.

Exercises:

1. Open a new `xterm` and run `top` in it. Which processes are consuming the most CPU? Which are consuming the most RAM?
2. To get more regular updates, press 's' then enter a value (in seconds). I find '1' is a very useful value.

Users

- `root` is all powerful. Accidentally executing `rm -rf /` will wipe your whole system. And Unix has no undelete!
- Other users are ‘safer’, so try to never login as `root`!

Users

- `root` is all powerful. Accidentally executing `rm -rf /` will wipe your whole system. And Unix has no undelete!
- Other users are ‘safer’, so try to never login as `root`!
- However, if you need to, use `su -` to change to `root`.
- If you need to run commands as `root`, run them from your user account and prefix them with `doas`. We’ll set that up shortly…

Users

- `root` is all powerful. Accidentally executing `rm -rf /` will wipe your whole system. And Unix has no undelete!
- Other users are 'safer', so try to never login as `root`!
- However, if you need to, use `su -` to change to `root`.
- If you need to run commands as `root`, run them from your user account and prefix them with `doas`. We'll set that up shortly...

Exercises:

1 What happens if you do `ls /root`? And what if you change to the root user and execute `ls /root`?

Editing files

1. There are many Unix text editors, but `vi` and `emacs` predominate. Every modern Unix has (at least) a simple version of `vi` builtin.

Editing files

1. There are many Unix text editors, but `vi` and `emacs` predominate. Every modern Unix has (at least) a simple version of `vi` builtin.
2. You start `vi` in 'command' mode; move back to it with '*Escape*'.

Editing files

1. There are many Unix text editors, but `vi` and `emacs` predominate. Every modern Unix has (at least) a simple version of `vi` builtin.
2. You start `vi` in 'command' mode; move back to it with '*Escape*'.

Useful commands (note that ':' is significant!):

`a`	move to 'insert mode' (after current character).
`i`	move to 'insert' mode (at current character).
`:q`	exit.
`:q!`	exit without saving.
`u`	undo the last change.
`x`	delete character under the cursor.
`:w`	save.

Exercises:

1. Switch to root with `su -` and then execute `vi /etc/doas.conf` to edit the `doas` config file.
2. Add a new line `permit :wheel` then save and exit.

doas

- `doas` allows you to run individual commands as another user (by default, as root).
- (`sudo` is an older, less secure, version of `doas`; more systems currently use `sudo` than `doas` though).

doas

- `doas` allows you to run individual commands as another user (by default, as root).
- (`sudo` is an older, less secure, version of `doas`; more systems currently use `sudo` than `doas` though).

Exercises:

1 What happens if you do `ls /root`? And what if you change to `doas ls /root`?

Inter-box communication

- Unix boxes are friendly.
- Use `ssh` to login to another machine.

Inter-box communication

- Unix boxes are friendly.
- Use `ssh` to login to another machine.

Exercises:

1. If you haven't done so already, activate your Informatics account.
2. Run `ssh -l` *`user`* `bastion.nms.kcl.ac.uk` where *`user`* is your Departmental username.

Power control

- `doas reboot` reboots OpenBSD.
- `doas halt -p` turns the machine off (safely).

Packages

- OpenBSD is... spartan.
- There are lots of *packages* of other software one can install with `pkg_add`. See the list at http://openports.se/.

Packages

- OpenBSD is… spartan.
- There are lots of *packages* of other software one can install with `pkg_add`. See the list at http://openports.se/.

Exercises:

1. Run `doas vi /etc/installurl` and add the line: `https://mirror.bytemark.co.uk/pub/OpenBSD/`
2. Run `doas pkg_add xfce`
3. Run `echo startxfce4 >> .xsession`
4. Logout or `reboot` then log back in.
5. Run `doas pkg_add firefox` then run `firefox`.

A better shell

- OpenBSD's default `ksh` is too minimalistic for my tastes.
- You can replace it with many others: `zsh` remains my favourite.

A better shell

- OpenBSD's default `ksh` is too minimalistic for my tastes.
- You can replace it with many others: `zsh` remains my favourite.

Exercises:

1. Run `doas pkg_add zsh`.
2. Run `zsh -l` to launch a new instance of `zsh`
3. Run `chsh -s zsh` to change your shell for all subsequent logins.
4. Run `vi ~/.zshrc` and add these contents:

```
autoload -Uz compinit promptinit
compinit
promptinit
prompt walters
```

   Save, logout, and log back in again.

A better text editor

- `vi` is torturous for serious editing. `vim` is `vi`'s big brother: it's highly configurable, has many available extensions etc.

Exercises:

1. Run `doas pkg_add vim--gtk2` (this also installs a gui).
2. Run `doas pkg_add wget`. You can then download my vimrc into your home directory:
   `wget -O$HOME/.vimrc https://tinyurl.com/yb9kqqla`
3. You'll need to install git with `doas pkg_add git` for the next step to work.
4. Run `gvim`. The first time you run it, various things will be installed. Wait until they're finished, quit vim, then reload. You now have a usable text editor!

Start programming

- Every major programming language is available to install and run. Pick your favourite (e.g. Java or Python) and try it.

Exercises:

1. Run `doas pkg_add python` to install Python 3.6 (creating an executable called `python3.6`).
2. Run `doas pkg_add jdk-1.8` to install Java 8. Note that you'll need to execute:
   `export PATH=$PATH:/usr/local/jdk-1.8.0/bin/`
   in your shell for commands like `javac` to work.
3. Try writing "hello world" in one or both languages!

Additional exercises

Try these (roughly in order):

- Configure `zsh` further. Try oh-my-zsh.
- Experiment with file permissions & owners. [Try `chmod` and `chown`.]
- How to terminate arbitrary processes? [Try `kill`, `pkill`]
- Install a modern desktop. [Try KDE or Gnome.]
- How to run a website? [I use OpenBSD's httpd.]
- How to handle mail? [Try an SMTP server like Postfix or OpenSMTPD.]