APT Session 1: Unix

Laurence
Tratt

KING'S College LONDON

Software Development Team
2015-10-14

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 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 and others including: email_merger, extsmail, Converge, multitime, srep, supuner, xcage.
- Homepage: http://tratt.net/laurie/ Twitter: @laurencetratt

The TAs

Joshua Simpson

- 4th Year MSci Computer Science
- Vice President at KCL Tech
- Student Advocate for HackCampus

Sam White

- 4th Year MSci Computer Science
- Strong interest in security & cryptography
- Software Engineer & Technical Architect at Red Badger

Ask both about summer interships!

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.
- 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.

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.
  - ***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.
  - ***Participation is mandatory***. [Optionally in teams of upto 3.]
- When is it?

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.
  - ***Participation is mandatory***. [Optionally in teams of upto 3.]
- When is it?
  - Wednesdays 14:00-17:00, K4U.13/14, starting 2015-10-14

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. Getting a GUI.

What to expect from this session: Unix

1. What is Unix?
2. Install OpenBSD using VirtualBox.
3. Basic command-line techniques.
4. Getting a GUI.
5. Starting a web server.

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/
3. Downloaded either:
   - http://mirror.bytemark.co.uk/OpenBSD/5.7/i386/install57.iso
     (if you're running a 32 bit OS)
   - http://mirror.bytemark.co.uk/OpenBSD/5.7/amd64/install57.iso
     (if 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 manages 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 manages 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 manages 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 42 year old operating system?

Why bother with a 42 year old operating system?

- It’s turned out to be extremely flexible.

Why bother with a 42 year old operating system?

- It’s turned out to be extremely flexible.
- It’s running your Android phone (or your iPhone (sort of); but not your Windows phone). It’s the core of OS X.

Why bother with a 42 year old operating system?

- It’s turned out to be extremely flexible.
- It’s running your Android phone (or your iPhone (sort of); but not your Windows phone). It’s the core of OS X.
- It’s been the #1 choice for serious software developers for over 30 years.

Why bother with a 42 year old operating system?

- It’s turned out to be extremely flexible.
- It’s running your Android phone (or your iPhone (sort of); but not your Windows phone). It’s the core of OS X.
- It’s been the #1 choice for serious software developers for over 30 years.
- Unix users adapt easily to other OSs. The reverse is not true.

Why bother with a 42 year old operating system?

- It’s turned out to be extremely flexible.
- It’s running your Android phone (or your iPhone (sort of); but not your Windows phone). It’s the core of OS X.
- It’s been the #1 choice for serious software developers for over 30 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

- Now we'll set up a VirtualBox image.

Setting up VirtualBox

- Now we'll set up a VirtualBox image.
- [Note carefully the message which tells the key that gives mouse & keyboard control back to the host OS.]

Setting up VirtualBox

- Now we'll set up a VirtualBox image.
- [Note carefully the message which tells the key that gives mouse & keyboard control back to the host OS.]

Exercises:

1. Install OpenBSD. When prompted for the 'root password', enter `aptpass`. [This is an awful password, but please use it so that we can more easily help you in this session.]
2. When prompted to reboot, instead type `halt`.
3. Eject the CD from VirtualBox. Then reboot the VM.
4. Login as root (i.e. username `root`, password `aptpass`).

What you get after login

- After you login, you're in the *shell*.
- Execute commands, see their results etc.

What you get after login

- After you login, you're in 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.

What you get after login

- After you login, you're in 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 curosr 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.

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. Execute `visudo`, which asks `vi` to edit a special system file.
2. Add a new line `%wheel ALL=(ALL) SETENV: ALL` (spacing and capitalisation are important!), save, and exit.

Users

- `root` is all powerful. Accidentally executing `rm -rf /` will wipe your whole system. And Unix has no undelete!
- Other users are ‘safer’, so 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 never login as `root`!
- If you need to run commands as `root`, run them from your user account and prefix them with `sudo`.

Users

- `root` is all powerful. Accidentally executing `rm -rf /` will wipe your whole system. And Unix has no undelete!
- Other users are 'safer', so never login as `root`!
- If you need to run commands as `root`, run them from your user account and prefix them with `sudo`.

Exercises:

1. Run `adduser` to add a non-root user. Choose your favourite username but use password `aptpass` again. Login group `users`; invite to group `wheel`.
2. Logout as root with `exit`. Login as your new user.
3. What happens if you do `ls /root`? And what if you do `sudo 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 Run `ssh -l` *`user`* `calcium.inf.kcl.ac.uk` where *`user`* is your Departmental username.

Power control

- `reboot` reboots OpenBSD.
- `halt -p` turns the machine off.

Packages

- Run `startx`. What do you think?

Packages

- Run `startx`. What do you think?
- 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

- Run `startx`. What do you think?
- 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. Exit FVWM. Run `sudo vi /etc/pkg.conf` and add the line:
   `installpath=http://mirror.bytemark.co.uk/OpenBSD/5.7/packages/i386`
   (change `i386` to `amd64` if you're on a 64 bit OS).
2. Run `sudo pkg_add enlightenment` and select version `1.0.9` when asked.
3. Run `echo e16 >> .xinitrc`
4. Run `startx`
5. Run `sudo pkg_add chromium` then run `chrome`.

Server processes

- Unix is often used for server processes.
- OpenBSD comes with a number of such programs; they’re nearly all turned off by default.

Server processes

- Unix is often used for server processes.
- OpenBSD comes with a number of such programs; they're nearly all turned off by default.

Exercises:

1. Install the `apache-httpd-openbsd` package (a web server).
2. Start Apache with `sudo apachectl start`. View the webpage `http://127.0.0.1/` in Chromium.
3. From `/var/www/conf/httpd.conf`, work out what directory you'd need to put files in them for to appear on your website. Create a file called `hello.html` and then view it in Chromium.
4. From your host OS (i.e. not OpenBSD!), can you view the webpage that's been created? You'll need to put VirtualBox into 'bridged adapter' mode, (but note that it won't work well on every OS / network).

Post-session exercises

Try these (roughly in order):

- Install a better shell. [Try zsh. Try configuring it.]
- 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.]
- Install an advanced editor. [Try VIM, though you will need to configure it. At a minimum start with something like vim-sensible. The good stuff starts with plugins like ctrlp.]
- How to handle mail? [Try an SMTP server like Postfix or OpenSMTPD.]