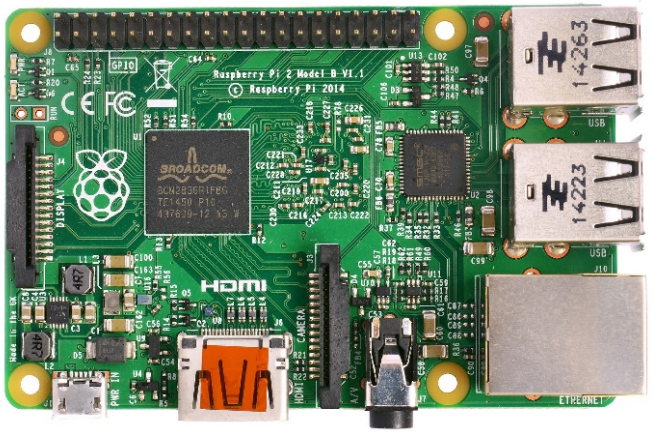
Multimedia Technieken

Introduction to Linux

katholieke hogeschool
associatie KU Leuven
vives

Introduction

Introduction



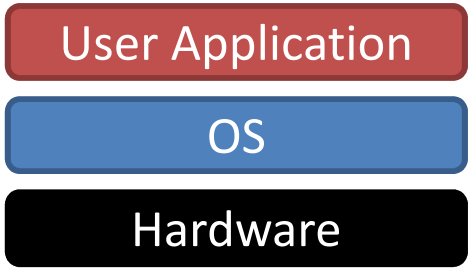
- The Raspberry PI 2 embedded system
  - A 900MHz quad-core ARM Cortex-A7 CPU
  - 1GB RAM
  - 4 USB ports
  - 40 GPIO pins
  - Full HDMI port
  - Ethernet port
  - Combined 3.5mm audio jack and composite video
  - Camera interface (CSI)
  - Display interface (DSI)
  - Micro SD card slot
  - VideoCore IV 3D graphics core

- How to use all these complex features ?

katholieke hogeschool associatie KU Leuven vives

Introduction

- How to use all these complex features ?
  - Typically by installing an Operating System (OS) which provides access to the hardware as a service



- You can actually write programs for the Pi without using an OS but this is very complex
  - These are also called Bare-metal applications

- But which one ?

Introduction

- What OS can we use ?
  - The ones that support the Raspberry Pi 2
    - Guess what, it's a short list
      - Linux based
        - » Raspbian
        - » Ubuntu Mate
        - » OSMC and OpenElec (Media Centers)
      - Windows based
        - » Windows 10 IoT core
      - Other
        - » Risc OS (Real-time OS)

- Initially we will use the Raspbian linux distribution

Linux and it’s kernel

Linux Distribution

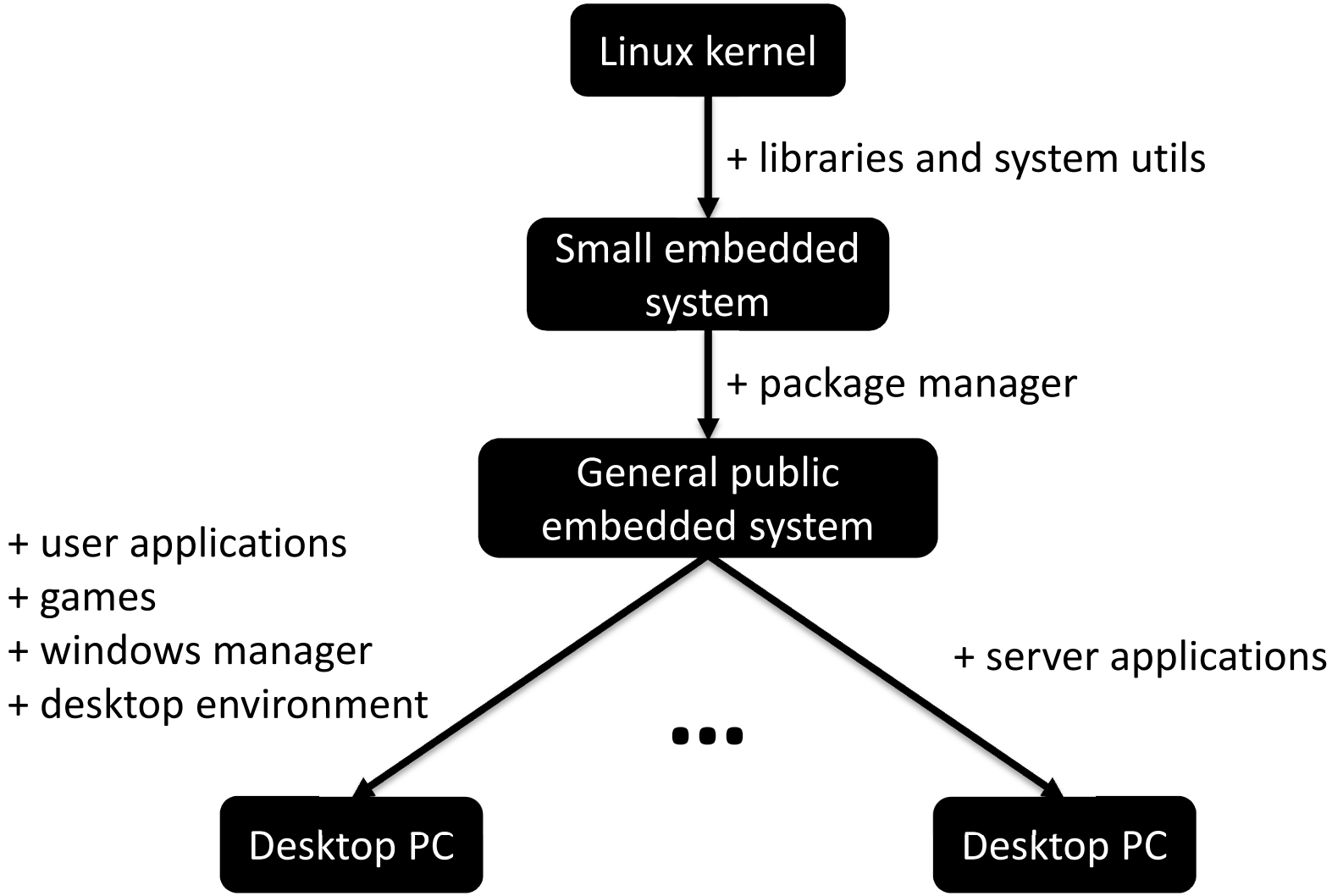
- Linux was originally developed by as a free operating system for personal computers based on the Intel x86 architecture, but has since been ported to more computer hardware platforms than any other operating system.
  - Desktop PC's, servers, mainframe's, supercomputers, smart phones, tablets, TV's, embedded systems, …

- Linux = Linux distribution (distro)
  - Linux distro = Linux kernel + libraries + system utils

- Linux kernel
  - Created in 1991 by Finnish computer science student Linus Torvalds
  - Unix-like
  - Free and open-source
  - Received contributions from nearly 12,000 programmers

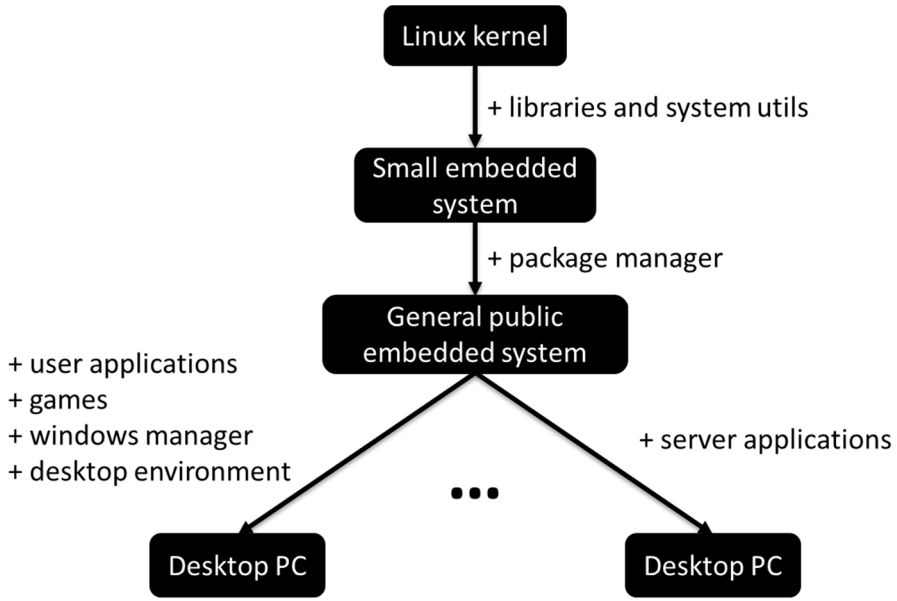
katholieke hogeschool associatie KU Leuven vives

Linux, kernel, distro, …



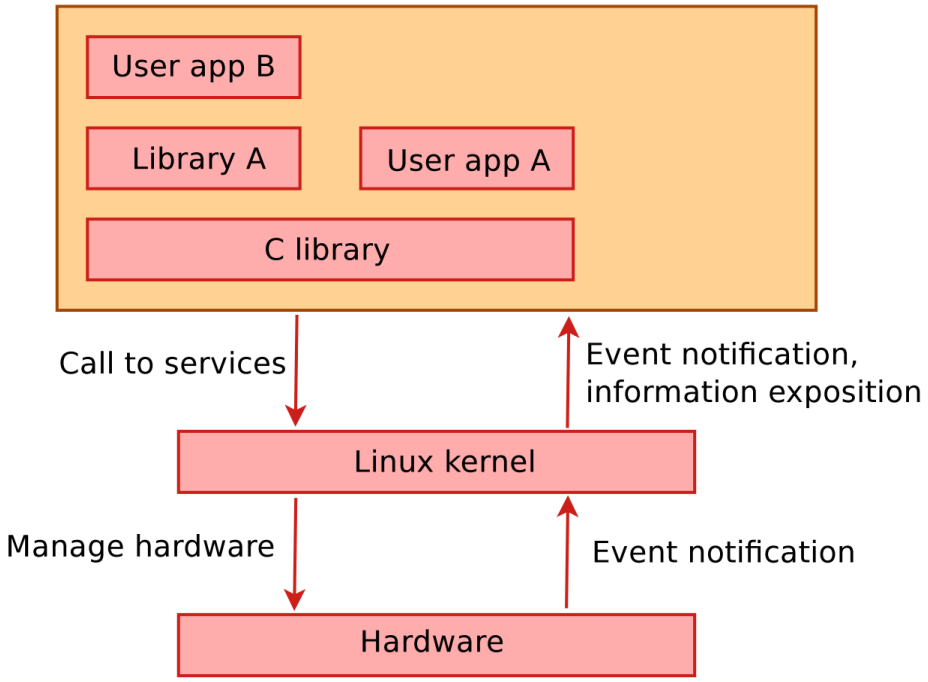
Linux, kernel, distro, …

- When we are talking about embedded Linux we actually are talking about the same kernel code running on millions of other systems.
  - There is no separate code base for embedded systems.
  - When we however build a Linux system for an embedded target we do exclude features we won't be using
  - We are also cross-compiling the kernel to binary code that can run on the target system.

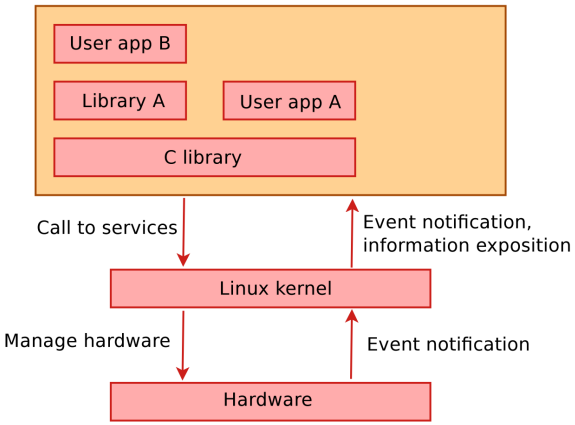


The keThe Linux Kernelrnel

- The Linux kernel provides the core system facilities required for any system based upon Linux to operate correctly.
- It has complete control over everything that occurs in the system.
- Application software relies upon specific features of the Linux kernel such as its handling of hardware devices and its provision of many fundamental abstractions such as virtual memory, sockets, tasks (known as processes), files and many others.



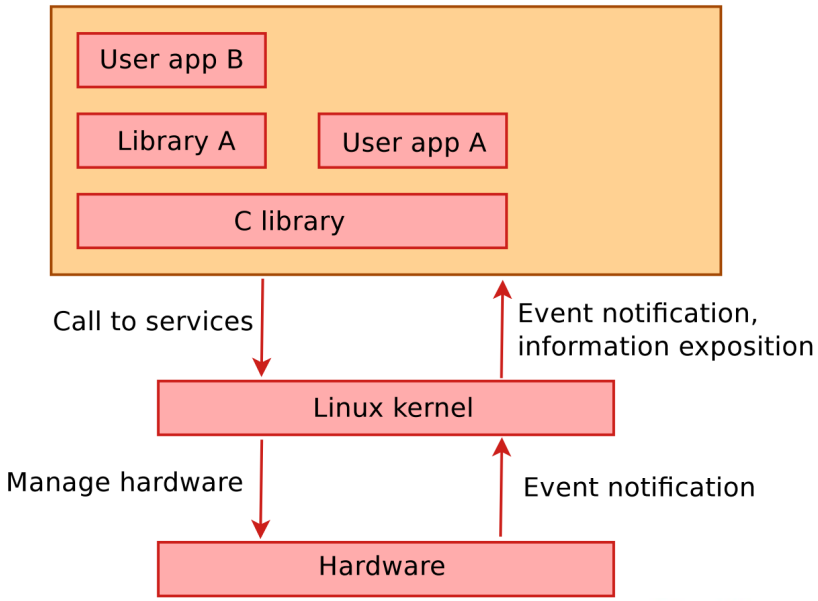
The Linux Kernel



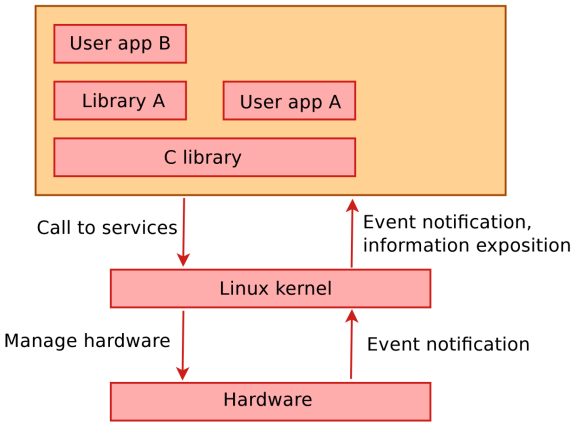
- The main roles of the kernel are
  - Manage all the hardware resources
    - CPU, memory, I/O.
  - Provide a set of portable, architecture - and hardware independent APIs (Application Programmable Interface) to allow user space applications and libraries to use the hardware resources.
  - Handle concurrent accesses and usage of hardware resources from different applications.
    - Example: a single network interface is used by multiple user space applications through various network connections. The kernel is responsible for "multiplexing" the hardware resource.

The Linux Kernel

- The main interface between the kernel and user space is the set of system calls that are provided by the kernel (about 300 system calls that provide the main kernel services).
- These services include
  - file and device operations
  - networking operations
  - inter-process communication
  - process management
  - memory mapping
  - timers
  - threads
  - synchronization primitives
  - …



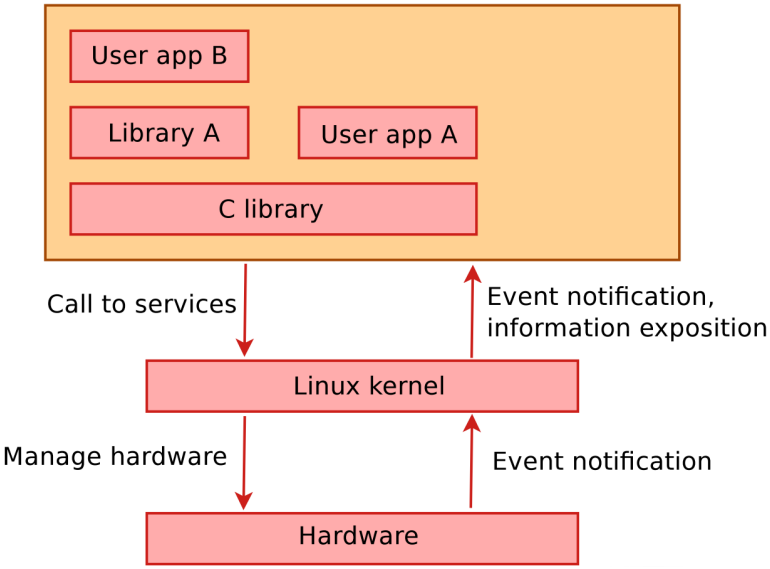
The Linux Kernel



- Some key features of the kernel are
  - Portability and hardware support
    - It runs on most architectures.
  - Scalability
    - Linux can run on super computers as well as on tiny devices (4 MB of RAM is enough).
  - Compliance to standards and interoperability
  - Exhaustive networking support
  - Security
    - It can't hide its flaws. Its code is reviewed by many experts.
  - Stability and reliability
  - Modularity
    - Can include only what a system needs even at run time.
  - Easy to program
    - You can learn from existing code. Many useful resources on the net.

The Linux Kernel

- Very important to know is that the kernel interface is stable over time.
  - This basically means that only new system calls can be added by the kernel developers and no old calls can be removed.
  - This means that applications running on an older kernel version should always work on a newer one.
- The system call interface is actually wrapped by the C library and user space applications usually never make a system call directly but rather use the corresponding C library functions.



Linux Basics

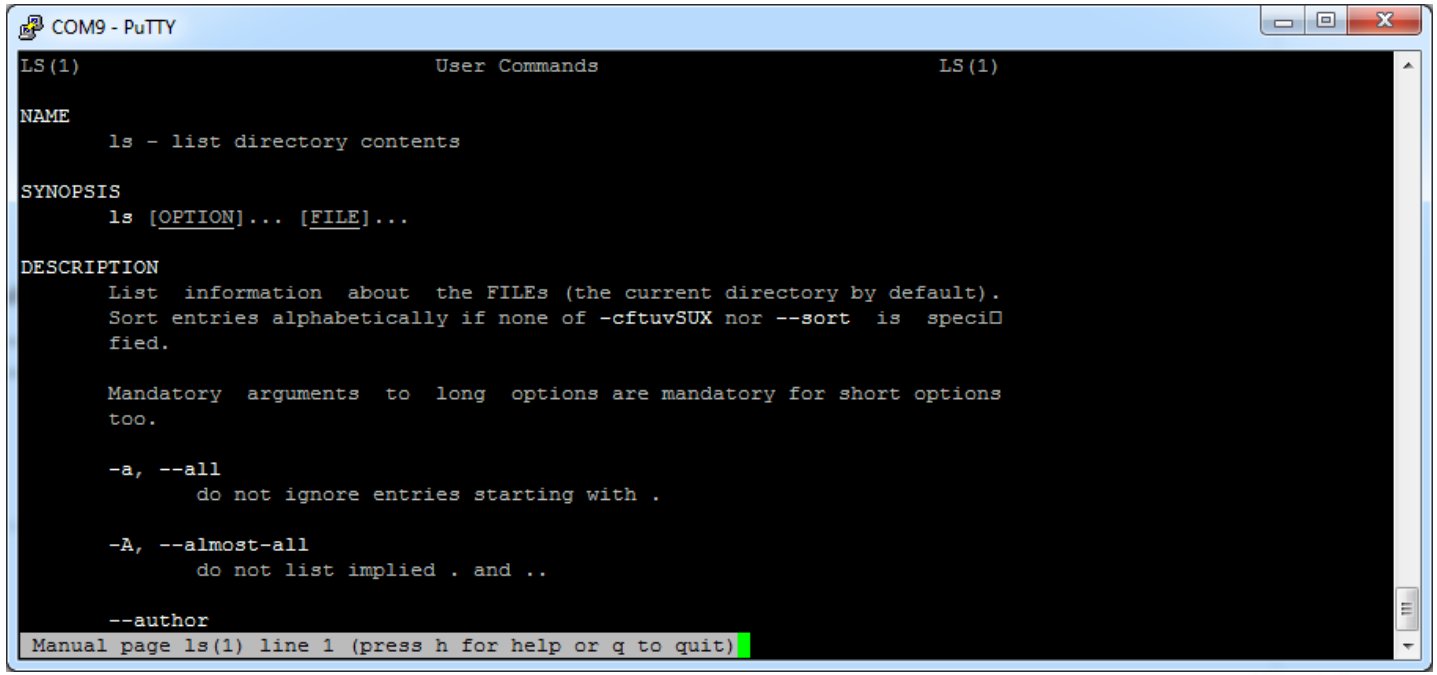
The Man Pages

While the information and examples presented here will be valid for most Linux distro's, some information and commands may only work for Debian based distro's.

The Man Pages

- The most important command you need to know is the "man" command,
  - Provides an interface to the reference manuals.
- By adding a command after the man command you can consult the man-pages for the particular command.
  - Example:

```
$ man ls
```



The Man Pages

- You can scroll through the man-pages using the arrow keys.
- Searching the current man-page can be done by first typing a slash ("/"), followed by your search term.
  - Jumping to the next hit can be done by hitting the "n" key, while jumping back is done with "SHIFT-n".
- Exiting the man-pages is achieved using the "CTRL-c" combination.

- **Some exercises:**
  - What does the "cat" command do ? How can it be used to output the content of a file ? Try to read the file "/proc/cpuinfo"
  - What does the "dmesg" command do ?
  - The "free" command shows system memory usage.How can you make the numbers "human readable" ?

Linux Basics

The Linux File System

While the information and examples presented here will be valid for most Linux distro's, some information and commands may only work for Debian based distro's.

Some History

- Linux inherits many of its concepts of filesystem organization from its Unix predecessors.
  - As far back as 1979, Unix was establishing standards to control how compliant systems would organize their files.
  - The Linux Filesystem Hierarchy Standard or FHS for short

- One important thing to mention when dealing with these systems is that Linux implements just about everything as a file.
  - This means that a text file is a file
  - a directory is a file (simply a list of other files)
  - a printer is represented by a file (the device drivers can send anything written to the printer file to the physical printer)
  - …
  - Although this is in some cases an oversimplification

Exploring the File System

- Where are thou ?
  - Use the "pwd" (print working dir) command
  - This simply returns the directory you are currently located in.

```
bioboost@NDWMINT ~ $ pwd
/home/bioboost
```

- List the content of a directory using the "ls" command
  - This will tell you all directories and files in your current directory.

```
bioboost@NDWMINT / $ ls
bin    dev   initrd.img  lost+found  opt   run   sys  var
boot   etc   lib         media       proc  sbin  tmp  vmlinuz
cdrom  home  lib64       mnt         root  srv   usr
```

Exploring the File System

- The two most common used flags of "ls" are probable -l and -a.
  - The "-l" forces the command to output information in long-form

```
bioboost@NDWMINT ~ $ ls -l
total 88
...
drwxr-xr-x  3 root root  4096 Jun 20 07:48 home
-rw-r--r--  1 root root     0 Sep 24 13:37 log.txt
drwx------  2 root root 16384 Jun 20 07:34 lost+found
...
```

- The very first character tells us what kind of file it is. The three most common types are:
  - -: Regular file
  - d: Directory (a file of a specific format that lists other files)
  - l: A hard or soft link (basically a shortcut to another file)

Exploring the File System

- The two most common used flags of "ls" are probable -l and -a.
  - The "-a" flag lists all files, including hidden files.
  - In Linux, files are hidden automatically if they begin with a dot

```
bioboost@NDWMINT ~ $ bioboost@NDWMINT ~ $ ls -al
total 124
drwxr-xr-x 18 bioboost bioboost 4096 Sep 24 13:04 .
drwxr-xr-x  3 root     root     4096 Sep 21 12:35 ..
drwx------  4 bioboost bioboost 4096 Sep 21 12:40 .cache
drwxr-xr-x 12 bioboost bioboost 4096 Sep 21 12:41 .config
drwxr-xr-x  2 bioboost bioboost 4096 Sep 21 12:40 Documents
...
```

- The first two entries, . and .. are special
  - The . directory is a shortcut that means "the current directory"
  - The .. directory is a shortcut that means "the current directory's parent directory"

Traversing the File System

- To change to a different directory, you issue the "cd" command
  - It stands for "change directory"

- You can follow the "cd" command with either an absolute or a relative pathname
  - An absolute path is a file path that specifies the location of a directory from at the top of the directory tree
    - Absolute paths begin with a "/"
  - A relative path is a file path that is relative to the current working directory.
    - This means that instead of defining a location from the top of the directory structure, it defines the location in relation to where you currently are.

Traversing the File System

- Let's traverse to the absolute path "/home"
  - It doesn't matter what our current working dir is

```
bioboost@NDWMINT ~ $ pwd
/home/bioboost

bioboost@NDWMINT ~ $ cd /home

bioboost@NDWMINT /home $ pwd
/home
```

Traversing the File System

- The lack of the "/" from the beginning tells to use the current directory as the base for looking for the path

```
bioboost@NDWMINT ~ $ ls
Desktop  Documents  Downloads  Music  Pictures
Public  Templates  Videos

bioboost@NDWMINT ~ $ pwd
/home/bioboost

bioboost@NDWMINT ~ $ cd Downloads

bioboost@NDWMINT ~/Downloads $ pwd
/home/bioboost/Downloads
```

Traversing the File System

- This is where the ".." directory entry comes in handy.
  - It can be used as a shortcut to move to the parent directory of the current dir

```
bioboost@NDWMINT ~ $ pwd
/home/bioboost

bioboost@NDWMINT ~ $ cd ..

bioboost@NDWMINT /home $ pwd
/home
```

- The "." directory represents the current working dir.
  - So changing to "." has no effect
    - Try this for yourself

Traversing the File System

- You can immediately jump to your personal home directory by using the "~" as destination

```
bioboost@NDWMINT / $ pwd
/
bioboost@NDWMINT / $ cd ~

bioboost@NDWMINT ~ $ pwd
/home/bioboost
```

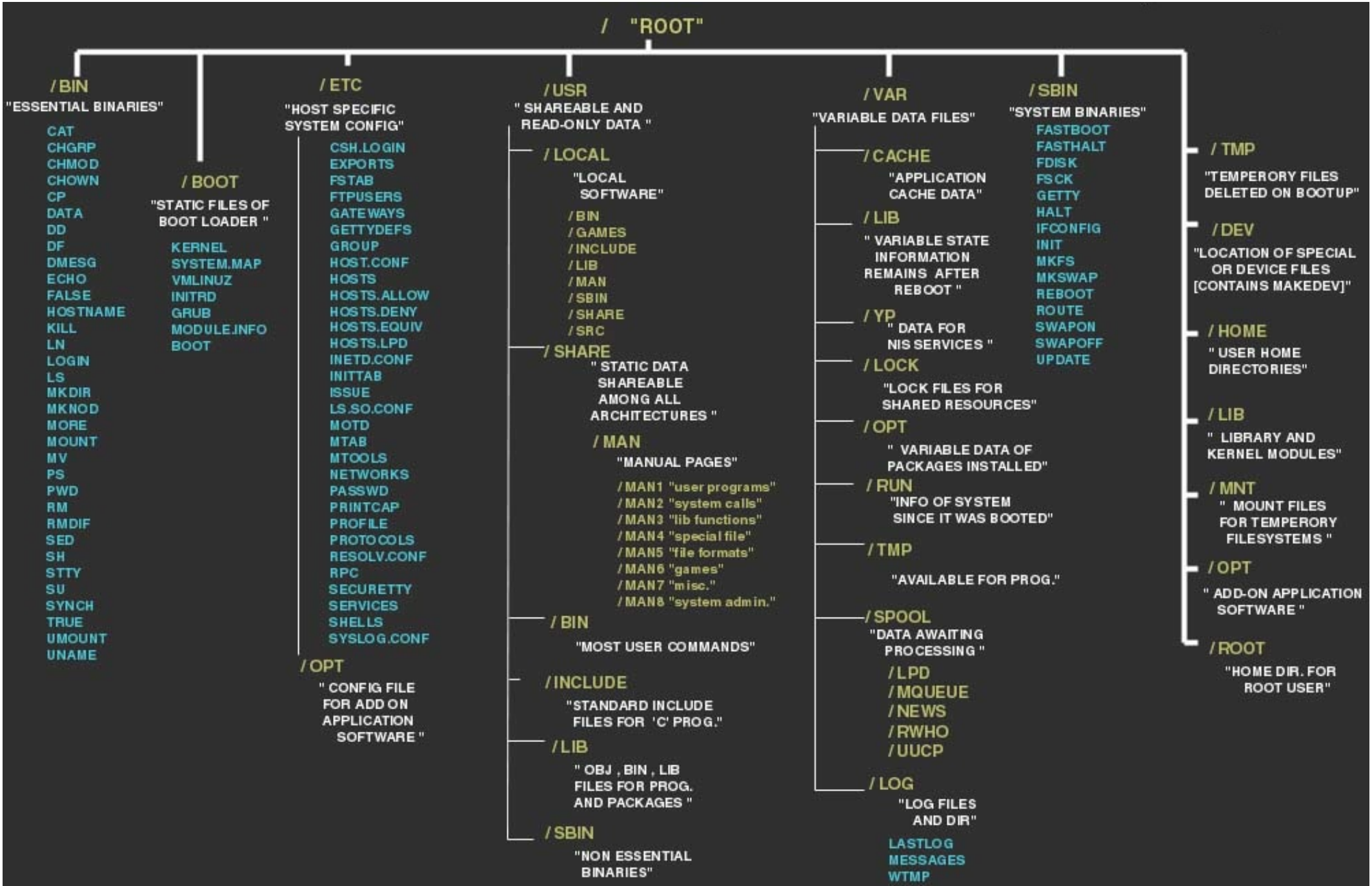
The Linux File System Layout

- The file system is contained within a single tree
  - Regardless of how many devices are incorporated.
- What this means is that all components accessible to the operating system are represented somewhere in the main file system.
- In Windows, each hard drive or storage space is represented as its own file system, which are labeled with letter designations
- In Linux, every file and device on the system resides under the "root" directory, which is denoted by a starting "/".
  - Thus, if we want to go to the top-level directory of the entire operating system and see what is there, we can type

```
bioboost@NDWMINT ~ $ cd /
bioboost@NDWMINT / $
```

- Every file, device, directory, or application is located under this one directory.

The Linux File System Layout



The /proc directory

- The "/proc" directory is actually a pseudo-file system of its own that is mounted to that directory.
  - The proc file system does not contain real files, but is instead dynamically generated to reflect the internal state of the Linux kernel.
- This means that we can check and modify different information from the kernel itself in real time.

- For instance, you can get detailed information about the processor and its cores by typing

```
bioboost@NDWMINT ~ $ cat /proc/cpuinfo
processor       : 0
model name      : ARMv6-compatible processor rev 7 (v6l)
Features        : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
```

Manipulating the File System

- A file can be created using the "touch" command
- You can remove a file using the "rm" command
  - Add "-r" (recursive) to remove a directory and its content
- You can move a file using the "mv" file
  - Also applicable for directory
  - Can also be used for renaming a file or directory
- To copy a file use "cp"
  - Add "-r" (recursive) to copy a directory and its content
- A text editor such as "nano" can be used to manipulate the content of text file

Linux Basics

Linux Permissions and Ownership

While the information and examples presented here will be valid for most Linux distro's, some information and commands may only work for Debian based distro's.

Users

- Linux is a multi-user system.
- Basic understanding of users and groups is required before ownership and permissions can be discussed
  - They are the entities that the ownership and permissions apply to.

- There are two types of users
  - **system users** - which are used to run non-interactive or background processes on a system
  - **regular users** - which are used for logging in and running processes interactively.

- An easy way to view all of the users on a system is to look at the contents of the "/etc/passwd" file.
  - Each line in this file contains information about a single user, starting with its user name (the name before the first :).

Users

```
bioboost@NDWMINT ~ $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
...
syslog:x:101:104::/home/syslog:/bin/false
kernoops:x:106:65534:Kernel Oops Tracking
bioboost:x:1000:1000:Nico De Witte,,,:/home/bioboost:/bin/bash
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
```

Superuser

- In addition to the two user types, there is the superuser, or root user, that has the ability to override any file ownership and permission restrictions.
- In practice, this means that the superuser has the rights to access anything on its own server.
- This user is used to make system-wide changes, and must be kept secure.

- It is also possible to configure other user accounts with the ability to assume "superuser rights".
  - Allowing the user to execute "sudo" commands
    - More on this later

Viewing Ownership and Permissions

- In Linux, each and every file is owned by a single user and a single group, and has its own access permissions.
- The most common way to view the permissions of a file is to use "ls" with the long listing option

```
bioboost@NDWMINT ~ $ ls -l <somefile>
```

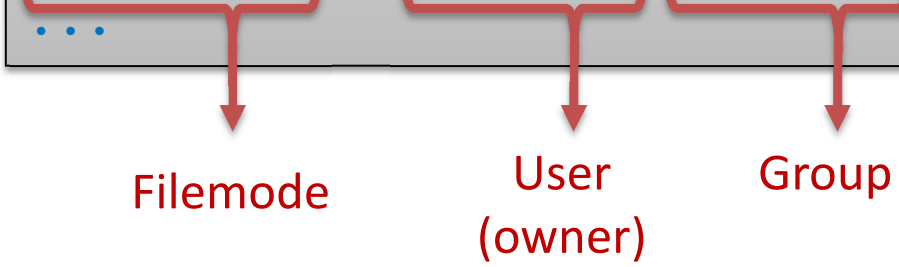
- If you want to view the permissions of all of the files in your current directory, run the command without an argument

```
bioboost@NDWMINT ~ $ ls
```

Viewing Ownership and Permissions

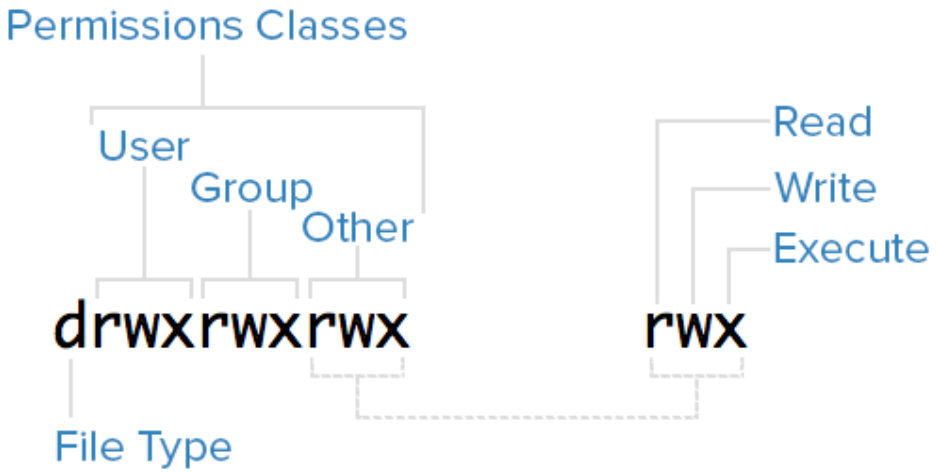
- Example

```
bioboost@NDWMINT ~ $ ls -l
total 36
drwxr-xr-x 2 bioboost bioboost 4096 Sep 21 12:40 Desktop
-rw-r--r-- 1 bioboost bioboost   78 Sep 24 15:24 hello_world.rb
drwxr-xr-x 2 bioboost bioboost 4096 Sep 21 12:40 Music
drwxr-xr-x 2 bioboost bioboost 4096 Sep 21 12:40 Pictures
-rw------- 1 bioboost bioboost    0 Sep 24 16:49 private.txt
-rw-rw-rw- 1 bioboost bioboost    0 Sep 24 16:50 public.txt
drwxr-xr-x 2 bioboost bioboost 4096 Sep 21 12:40 Videos
...
```



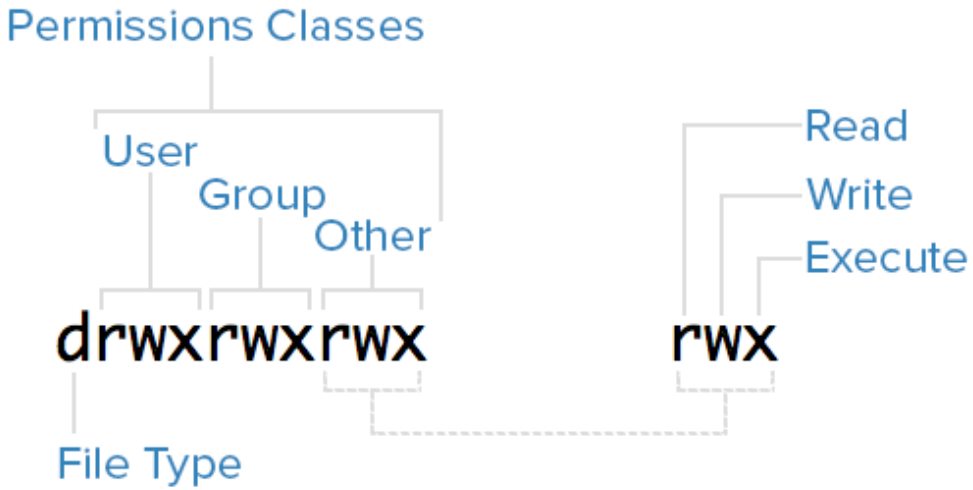
The Filemode

- First character is the file type
  - -: regular file
  - d: directory (a file of a specific format that lists other files)
  - l: a hard or soft link (basically a shortcut to another file)
  - c: a character device file
  - b: a block device file
  - p: a pipe file
  - s: a socket file



Permission Classes

- The next 9 characters make up the permission classes, namely
  - User: The owner of a file belongs to this class
  - Group: The members of the file's group belong to this class
  - Other: Any users that are not part of the user or group classes belong to this class.
- The order of the classes is consistent across all Linux distributions.



Symbolic Permissions

- The next thing to pay attention to are the sets of three characters, or triads, as they denote the permissions, in symbolic form, that each class has for a given file.
  - In each triad, read, write, and execute permissions are represented in the following way:
    - Read: Indicated by an r in the first position
    - Write: Indicated by a w in the second position
    - Execute: Indicated by an x in the third position. In some special cases, there may be a different character here.
  - A hyphen (-) in the place of one of these characters indicates that the respective permission is not available for the respective class.

Read, Write and Execute

- **Read**
  - Normal file: allows a user to view the contents of the file.
  - Directory: allows a user to view the names of the file in the directory.
- **Write**
  - Normal file: allows a user to modify and delete the file.
  - Directory: allows a user to delete the directory, modify its contents (create, delete, and rename files in it), and modify the contents of files that the user can read.
- **Execute**
  - Normal file: allows a user to execute a file (the user must also have read permission). As such, execute permissions must be set for executable programs and shell scripts before a user can run them.
  - Directory: allows a user to access, or traverse, into (i.e. cd) and access metadata about files in the directory (the information that is listed in an ls -l).

Examples

`-rw-------`	`mark`	`customers`	Mark (the owner) can read the file and can write to it. The customers group and others have no permissions.
`-rwxr-x---`	`mark`	`customers`	Mark (the owner) has full file control (including execution). The users of the group customers can read and execute the file. Others have no permissions.
`drwxr-x---`	`mark`	`www-data`	Mark (the owner) of the directory has full control. The www-data group users can read and traverse the directory. Other can not.
`drwx--x---`	`mark`	`customers`	Mark (the owner) of the directory has full control. The customers can 'cd' into the directory but cannot 'ls' to show the content.
`drwx-wx---`	`mark`	`customers`	Mark (the owner) of the directory has full control. The customers can 'cd' into the directory but cannot 'ls' to show the content. They can however create and delete files in the directory. Customer can even delete files he does not have read or write access too as long as he knows the name of the file.
`drwxrw----`	`mark`	`customers`	Customers can list the directory content but cannot cd into the dir or access any of the files.

Linux Basics

Debian Package Manager

While the information and examples presented here will be valid for most Linux distro's, some information and commands may only work for Debian based distro's.

A Repository of Packages

- Every Linux distribution is different in terms of how software is installed.
- Linux distributions use different installation file types, package managers, and commands for installation.
- Even within a single form of Linux, there are different types of package managers.
  - A package manager is software used to handle the installation, removal, configuration, and updating of programs and drivers.

- Debian files are usually downloaded by package managers from a software repository.
  - A repository is a collection of Debian files that typically comes from a server

- Debian files are not required to come from a repository, although most do.
  - Can be manually downloaded too

Debian and it's Packages

- Packages generally contain all of the files necessary to implement a set of related commands or features.

- Two types of Debian packages
  - **Binary packages**
    - Contain executables, configuration files, man/info pages, copyright information, and other documentation.
    - Distributed in a Debian-specific archive format (.deb file)
  - **Source packages**
    - consist of a .dsc file describing the source package
    - a .orig.tar.gz with original unmodified source
    - usually a .diff.gz file that contains the Debian-specific changes to the original source

Debian and it's Packages

- Installation of software by the package system uses "dependencies" which are carefully designed by the package maintainers.
  - Example
    - GNU C compiler (gcc) "depends" on the package binutils which includes the linker and assembler.
    - If a user attempts to install gcc without having first installed binutils, the package management system (dpkg) will send an error message that it also needs binutils, and stop installing gcc.

- Multiple tools exist to manage Debian packages
  - From graphic or text-based interfaces to the low level tools used to install packages.
  - dpkg, APT, aptitude, dselect, …

Advanced Package Tool

- dpkg is the main package management program
  - Low level
- APT is the Advanced Package Tool and provides the apt-get program.
  - apt-get provides a simple way to retrieve and install packages from multiple sources using the command line.
  - Unlike dpkg, apt-get does not understand .deb files, it works with the packages proper name and can only install .deb archives from a source specified in /etc/apt/sources.list.
  - apt-get will call dpkg directly after downloading the .deb archives from the configured sources.

Using APT to Manage Your Packages

- To update the list of package known by your system, you can run

```
bioboost@NDWMINT ~ $ apt-get update
```

  - Always run this command before installing packages to make sure you are using latest versions

- To upgrade all the packages on your system (without installing extra packages or removing packages), run

```
bioboost@NDWMINT ~ $ apt-get upgrade
```

- These commands must be run with root privileges, so use 'sudo'

Using APT to Manage Your Packages

- To install a package you can use

```
bioboost@NDWMINT ~ $ apt-get install <packagename>
```

  - This will also install the dependencies of the package

- To remove the foo package from your system, run

```
bioboost@NDWMINT ~ $ apt-get remove <packagename>
```

- To remove the foo package and its configuration files from your system, run

```
bioboost@NDWMINT ~ $ apt-get --purge remove
<packagename>
```

Using APT to Manage Your Packages

- To upgrade all the packages on your system, and, if needed for a package upgrade, installing extra packages or removing packages, run

```
bioboost@NDWMINT ~ $ apt-get dist-upgrade
```

- Some exercises
  - Install git
  - Install aptitude
  - Install ruby

Aptitude

- Aptitude is a package manager for Debian GNU/Linux systems that provides a frontend to the apt package management infrastructure.
  - Aptitude is a text-based interface using the curses library

- Easy access to all versions of a package.
- easy to keep track of obsolete software by listing it under "Obsolete and Locally Created Packages".
- fairly powerful system for searching particular packages and limiting the package display
- can be used to install the predefined tasks
- Aptitude in full screen mode has su functionality embedded and can be run by a normal user.
  - It will call su (and ask for the root password, if any) when you really need administrative privileges

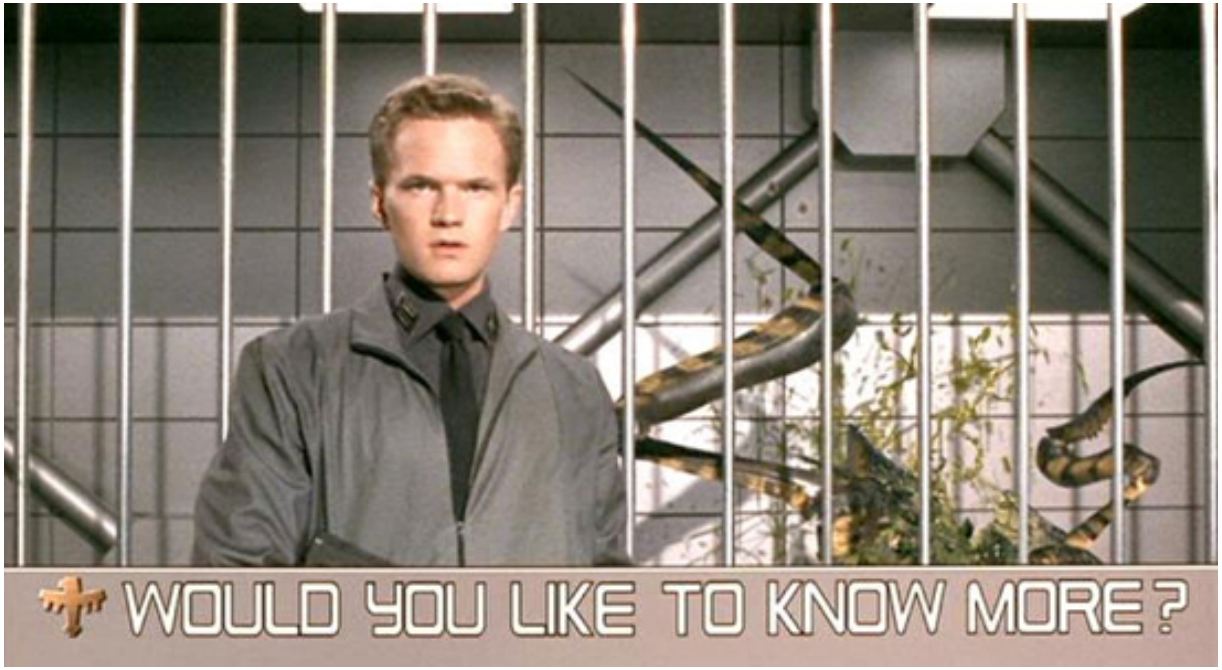
Aptitude

- We will mainly use it for searching for packages

```
bioboost@NDWMINT ~ $ aptitude search <searchterm>
```

- Assignment
  - Install the apache package and find out where the webpages are stored. Make sure you can view your website from your host machine. Change the index.html page (you can use the nano editor for this) and add some cool things to it.

Would you like to know more ?



- Well then