# Introduction to Linux



Martin Read

# C#

- C# development driven more by Microsofts' business need to reinvigorate Windows desktop application development against perceived competitive threats like Java
  - designed to maximize application developer productivity
- Microsoft origins, so
  - usually targeted towards Windows
  - benefits from the .NET framework
- High-level programming language
  - originally very much like Java, now very different

# C#

- High-level programming language
  - object-oriented, event-driven with large systems libraries
- Provides a managed memory model that adds a higher level of abstraction & protects programmers
  - does a lot of what could be done in C for you e.g. memory management
  - simpler syntax is less demanding (& error prone) & has a shallower learning curve for new programmers
  - adds convenience and improves development times
  - complicates access to lower level APIs
- Similarities between C & C# are few
  - syntax is similar, but languages very different behind the scenes

# C#

- Provides a number of features to make it easier to code
  - dynamic typing, implicit variable declarations, & local functions
  - error handling (type checking, bounds checking, uninitialized variable checking)
  - exception handling & garbage collection
- Generally interpreted - most commonly compiled into byte-code (not machine code) & executes on a virtual machine that converts into machine code on the fly

# C#

- Higher level languages are not efficient
  - C# code is much slower & useless for high performance scientific computing
- C# remains a common choice for internal/enterprise applications & games development, but is less common for commercial software
- Rapid client application development & high performance Server development

# What about C?

- Low-level programming language
  - easier to understand and implement than object oriented programming
  - usually compiles to assembly language
  - performs almost as efficiently as assembly code
  - provides base-level access to memory, & requires very little runtime support
  - lets you do almost anything provided correct syntax
  - can cause some real damage to the OS

# What about C?

- Complete binary data transparency
- Consecutive data is placed consecutive in memory
  - an int, followed by a double, followed by a char, is exactly 15 bytes long



- Data allocated in a function gets allocated on the stack exactly as you declare it (usually in the same order)
  - memory layout of data is completely under your control

# Lecture aims

- To introduce the Unix/Linux Operating System.



# Lecture Objectives





- To examine the file & directory structures used.
- To look at a few basic Unix commands.

- Practical in Linux command line

# The UNIX Operating System

- Common Operating System – can be found on Sun, Vax computers, as well as being ported into Intel based computers.
- UNIX has been around since 1969. At one stage it was available free….
- The entire source code was once available…
- Most commonly found as a scientific/engineering Operating System, but…



Commonly found on ‘servers’ to, for example, the web

  – because it handles concurrency (pseudo concurrency) easily

# History

Bell labs (AT&T)

| | | |
|---|---|---|
| 1969 Multics | - | slow, limited interaction. Originally written in assembler and ‘B’. |
| 1972 | - | Rewritten in C |
| Version 6 | | 1973 – University only |
| Version 7 | | 1979 – first sold commercially |
| System V: | | |
| Release 1 | | 1983 – improved commercial appeal |
| Release 4 | | 1989 - 1.2 million Unix installations |
| 1999 | | Linux 2.2 kernel released. |



There are 2 ‘main’ versions of UNIX, referred to as

| | |
|---|---|
| System V | - Bell labs or AT&T UNIX |
| BSD | - Berkley UNIX (mostly universities) |

# Why UNIX?

- It is a ‘Real’ Operating System – supporting virtual memory management, multitasking, multiuser, etc…
- Probably only OS available from several sources
- Almost completely written in a high level language. Thus it is portable to new processors (if they support a compiler and OS to run it on). Used on micros to supercomputers .
- Resource requirements are relatively small & don’t require very specialised hardware

**BUT**



- UNIX has been around for some time & much of it ‘evolved’.



- Is UNIX really a business OS? It was never intended to be.

- Although UNIX is available from many sources, each source is slightly different



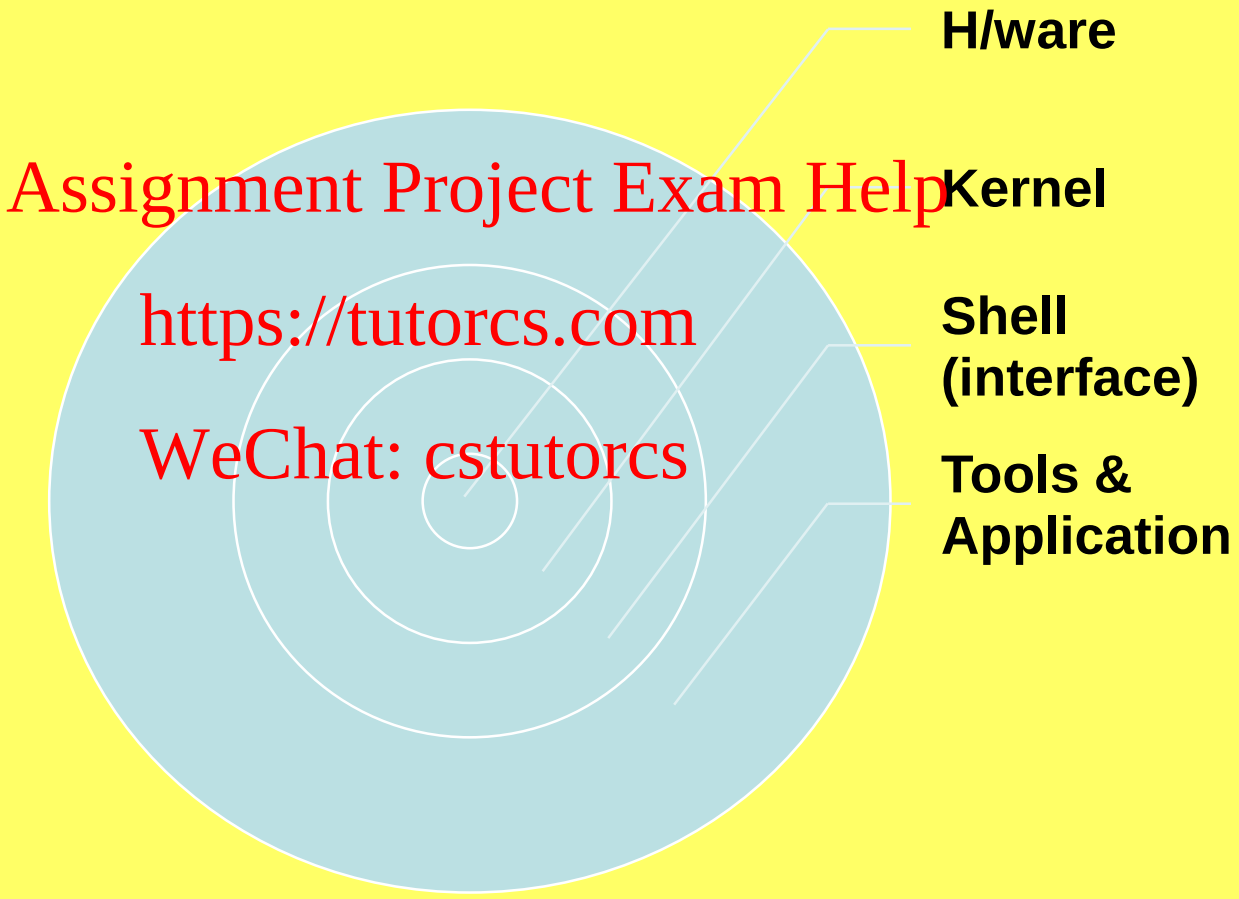
- It is not a real-time OS





- Machines like ICL29000 hardware & the VME Operating System were designed around each other
  - > better optimised system?

# The ‘Onion’ Model



# Tools

- **Tools and applications**

  There are hundreds of tools available to users. Email, word processing, business applications, or programming.



- Account and invoicing of system users
- Computer aided instruction



- On line manual (very comprehensive)
- Programme debugger



- Pattern scanning and processing language
- Languages – C, Basic, FORTRAN
- System status info
- Much file management, inc encryption & database

  – and more – a vast software library

# Interfaces

Most versions work with a windowing system (GUI)

- Every Unix/Linux system has a command-line interface
- Command-line more flexible for advanced use
  - programmes used together in combinations impossible under windows
- Shell scripts automate jobs which would have to be done manually and repetitively using windows
- Text-based interfaces easier than GUIs for sight- & hearing-impaired users

# Linux

- Linux is a free Unix-type OS
- It is a clone of Unix
- Linux supports most of the popular Unix software



- Linux is mostly System V, mostly BSD compatible & mostly POSIX compliant

# Linux

It has all the features you would expect in a modern fully-fledged Unix:



- Multitasking
- Virtual memory
- Shared libraries
- Proper memory management
- TCP/IP networking

# Linux

- Linux is easily portable to most general-purpose 32- or 64-bit architectures as long as they have:



- a paged memory management unit (PMMU)



- a GNU C-compiler
- GNU - recursive acronym for "GNU's Not Unix"

# Linux

- Much of Linux is really an aggregation of hundreds of independently developed utilities that make up the typical Unix system
- Linux offers a windows-based shell interface, commonly known as X-Windows or simply as X

# Linux

- Unlike Windows 10, there's no single version of Linux
- Linux distributions take the Linux kernel
- Combined with other software like the GNU core utilities, X.org graphical server, a desktop environment, web browser, etc
- Each distribution unites some combination of these elements into a single OS you can install

# Which Linux am I running?

- The *uname* command lets you check some basic information about the copy of Linux you're running:

  > *uname -a*

- will display all the information, including the Linux release and CPU information

# *What is Debian?*

- The Debian Project is an association to create a free OS using the Linux kernel
- Most of the basic OS tools come from the GNU project - usually called GNU/Linux
- GNU/Linux - a complete Unix-like operating system
- Debian comes with more than 50,000 packages

# Ubuntu

- Ubuntu is a cross-platform, open-source OS based on Debian
- Ubuntu builds on the Debian architecture & infrastructure, but there are important differences
- Ubuntu is a good place to start for former Windows users

# What do you need to know?

- Print your working directory
- Change directories & list all contents
- Create a file & a directory
- Redirect the output of a file
- View the full/partial contents of file
- Search for something
- Delete files/directories
- Run a programme in the background
  - Then end that process

# *root* super user

- In a multi-user environment, you must be able to protect files from hackers, etc
- There is a single super user-id called *root* which has the power to do anything
- Normally the *root* user prompt is different e.g. #

# Linux naming convention

- A name may be up to 14 characters long
  - recent versions have increased this to 28 characters
- A name cannot contain spaces
- Any other characters may be used
- However, because a number of characters have special uses, it is suggested that only the following are used:
  - Lowercase & uppercase letters (a-z)
  - Numbers (0-9)
  - Underscore ( _ ), period ( . ) and comma ( , )
- Linux is case sensitive – Readme, readme, ReadMe, readMe are all different files

# Naming conventions

- .c - ‘C’ programme source file
- .h - header file
- .o - compiled and assembled code (object file)
- .ps - Postcript source code
- .sh - Shell programme
- .z - compressed file using pack command
- .Z - compressed file using compress compile

# Permissions

| | | | | | |
|---|---|---|---|---|---|
| -rw-rw-r-- | 1 pbg | staff | 31200 | Sep 3 08:30 | intro.ps |
| drwx------ | 5 pbg | staff | 512 | Jul 8 09.33 | private/ |
| drwxrwxr-x | 2 pbg | staff | 512 | Jul 8 09:35 | doc/ |
| drwxrwx--- | 2 pbg | student | 512 | Aug 3 14:13 | student-proj/ |
| -rw-r--r-- | 1 pbg | staff | 9423 | Feb 24 2003 | program.c |
| -rwxr-xr-x | 1 pbg | staff | 20471 | Feb 24 2003 | program |
| drwx--x--x | 4 pbg | faculty | 512 | Jul 31 10:31 | lib/ |
| drwx------ | 3 pbg | staff | 1024 | Aug 29 06:52 | mail/ |
| drwxrwxrwx | 3 pbg | staff | 512 | Jul 8 09:35 | test/ |

# Files and directories

- Any file or directory has three sets of access permissions - r, w, x



- Any file has 3 classes of people who can access it:



| | | |
|---|---|---|
| – Owner | u | (for user) |
| – Group | g | (owners group) |
| – Other | o | sometimes rest, world, etc |

- The *root* user overrides all settings and can access any file



- Groups can be set up by the system administer – to find out which ones you belong to, type





> *groups*

- You can also determine which groups another user is a member of

E.g. *groups martin* (or other username)

# Access permissions

| **Ordinary file** | **Directory** |
| --- | --- |
| r (read)<br>can look at contents of a file | Allows listing of files |
| w (write)<br>can change or delete contents | Create/move/delete files in the directory |
| x (execute)<br>can execute the file | Can search the directory |

# Access permissions

| -rw-r----- | 1 | root | system | 3953 | Jan 12 2018 | martin4 |
|---|---|---|---|---|---|---|
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |



(1) File/directory permission bits
(2) Link count
(3) User name of person who owns the entry
(4) Group that owns the file/directory
(5) Size (in bytes)
(6) Date file last modified
(7) Name of file or directory

# Access permissions

- You can change access permissions of files and directories you own using *chmod*

  + = add permissions

  - = remove permissions



*chmod go +rw martin4*

can also use octal or absolute mode

r w x

4 + 2 + 1

*chmod nnn* filename

e.g. *chmod 740 martin4*

# Directories

Directory structures are separated by
**/** (forward slash)

**.** refers to your working directory

**..** refers to the parent directory



***cd***
***mkdir***
***rmdir*** } (DOS copied from Unix)

From other parts of the system
your home directory can be referred to as '~'
*cp bsctemp ~/bsctemp*

# Directories

- Working directory or current directory
- To find out where you are in the system

  *pwd* (Print Working Directory)

  *cd* should take you to your home directory (but takes you to sec204)



- Everything in your home directory you own

# File structure

- Hierarchical structure
- Parent directory/root directory written as “/”
- Subdirectories include bin, etc, tmp & usr – usually containing specific kinds of system files



- */bin* programmes
- */etc* configuration files
- */home* or */user* home directories of users
- */usr* application programmes
- */var* variable files (admin, mail, spool)
- */dev* devices e.g. terminal, printer, drives

# Useful files

- *.login* (executes when you log in)
- *.logout* (executes when you log out)
- *.cshrc* (executes when you invoke a 'C' shell )
- *.plan* (used by 'finger' to display a little message to yourself)
- .profile

# Full & Relative Pathname

- **Full Pathname**
  List each directory, starting from /, down to the file itself. Each directory and the filename must be separated by a “/”

  **OR**

- **Relative Pathname**
  If the file is in a directory near the Working Directory, a relative path may be used



- If the Working Directory is /home/martin, the file myfile.c in that directory may be referred to by:

  Full Pathname: */home/martin/myfile.c*

  Partial Pathname: *myfile.c*

# Full & Relative Pathname

- If the Working Directory is /home/paul
- myfile.c under the directory /home/martin may be referred to by:



Full Pathname: */home/martin/myfile.c*

Relative Pathname: *../martin/myfile.c*

# Basic Command Syntax

- Most commands follow a consistent syntax, of the form:

command [options] [arguments]



- [options] switches that modify the function. Options usually begin with a "-", such as "-f".
- [arguments] usually (but not always) the name of a file or directory to perform the operations on.

# List command

- *ls* list

- *ls –alg* (all files, long format + group id)
- *ls –d* (directory)
- *ls –t* (time order)
- *ls –r* (reverse order)



- *ls –l a**
- *ls –lt martin?*

# Metacharacters

? one character compare

* multiple character compare

[] list of comparative values

[-] range for comparison

[!] negative comparison



!! Execute the previous command

!nn Execute command nn from *history*

!-n Execute the command that was executed n commands ago

# The on-line manual

- To get help on any part of the system
  - use ***man*** or ***info***



*man 1 ls* section number and title



- When the manual entry displays more than one screen, press the space bar for more, or CTL/C to stop, or *q* to quit

# Sections of the manual

1. Commonly used commands & tools
2. System calls (to the Unix kernel). Assumed written in C
3. Higher level subroutines – again in C
4. Special device files. Typically characteristics of disc drives, etc
5. File formats – of system files
6. Games
7. Misc file system hierarchy. Text formatting, macro pages, character sets, etc
8. System admin guide

# on-line manual

- Some commands are in several sections of the manual

  *sleep nn* causes your process to sleep for nn seconds



- Can be used as a Console command



  or

- A call to the operating system from within a programming language



  – Found in sections 1 and 3

- ‘Sleep’ uses the OS functions ‘gettimer’ & ‘sigpause’, which have entries in section 2 of the manual

# Finding files

- Finding files using the find programme

  find directory -options [-print -exec -ok]

  *find . –type f –name “myfile*”*



- Look in working directory (& all subdirectories) for files with names starting ‘myfile’



  *find . -type f –mtime -4*



- Look for all files created or modified in the last 4 days

  *find / -name “[a-z]*”*

# Other commands

- Linux has the GNU locate programme

  *locate alpha*



- date
- who  who is on the system
- whoami  who are you logged in as
- passwd  change your password

- cat filename  (list a file - no pauses)
- more (pg) filename
  (list a file - with pauses - use for large files)
- tail filename  (list the last few lines)

# Processes

- To display all the processes use *ps*

  *ps –ef*

- Displays the user name, PPID (parent process id), start time & full command line
- To stop a process use kill – need to know the process id

  *kill 1481* kill process 1481

- Some processes will not terminate with a normal kill & require a kill absolute to terminate them

  *kill -9 1481*