MONASH University
Information Technology

FIT9134
Computer architecture and operating systems

Week 2

Operating Systems I:
Introduction to Operating Systems

Introduction to Operating Systems

- FIT9134 will use *Unix* and *Linux* as case-studies to illustrate how operating systems work.
  - We will learn how to install & manage a typical *Linux* system on a *Virtual Machine*. The Virtual Machine will run under *Microsoft Windows*.

- This week, we will look at what operating systems are and how they are used within the computing environment, and take a brief look at the history of the Unix O/S.

Reminder : Lectures Vs Labs

- for this unit, the lecture materials are not normally related to the lab tasks. However, there are a few lectures (listed in Week 1's lecture) which contain materials helpful/essential to the lab tasks – you will be reminded during those lectures. The website will also have reminders on these lectures.

Introduction

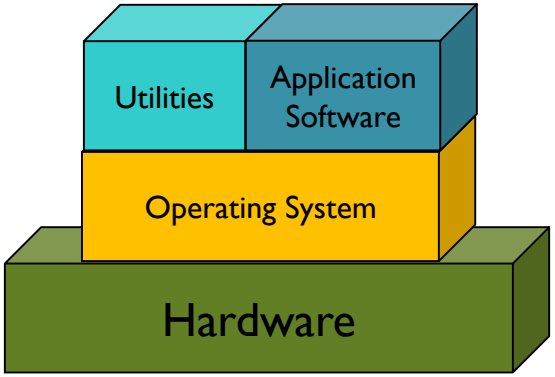
- What is an *Operating System* (O/S)?
  - A typically large piece of software that allows users of computer hardware to:
    - run various sorts of software applications.
    - develop their own programs.
    - have the resources of the computer system managed effectively on their behalf.

- Examples of popular Operating Systems :
  - Unix/Linux
  - Microsoft Windows
  - Mac OS
  - ….

Operating Systems

- Operating systems control the underlying computer hardware.
- Provide an environment within which other programs can be run.
- Can also control how the CPU and other computer resources are allocated to individual users or programs.
- In Unix/Linux, the ***kernel*** is the most important part of the operating system.

We can generally categorize the various software sitting on top of the O/S into "*Utilities*" and "*Application Software*", depending on the complexity and purpose of the software.



Operating Systems (users & files)

- Many modern operating systems (including Unix) are multi-user systems.
  - more than one user can log on to the same computer simultaneously.
- This means the operating system must provide some means of preventing users interfering with each other.

- The operating system should provide separate file spaces for different users (typically called *"home directories"*) so that files can be kept separate and private.
- There should be some security mechanism in place that prevents users from reading or altering the files of other users (unless permissions are explicitly granted).

Operating Systems (processes)

- A multi-user operating system is generally also a multi-tasking operating system
  - individual users can be running more than one program simultaneously.
- On a single-CPU machine, this is actually just an illusion - since if there is only one physical CPU, there can only be one program running on the CPU at any given instant.
- This illusion is achieved by rapidly switching the CPU between the different programs in memory, in turn giving them some "time-slices" on the CPU.

- Unix is a multi-user, multi-tasking operating system.

Major Functions of an O/S

1) ***Process*** **Management**

2) ***File*** **Management**

3) ***Memory*** **Management**

1) Process Management

- A *process* is normally defined as "a *program in execution*".

- The operating system typically needs to provide the following functionality with regard to processes:
  - Creating and destroying processes
  - Controlling the execution/progress of processes
  - Acting on exceptional conditions arising during the execution of processes (eg. errors, interrupts, etc)
  - Allocating hardware resources (fairly) among processes
  - Providing some form of inter-process communication

2) File Management

- The operating system is key to the control of data on secondary storage. Operating systems typically operate on the concept of a *file* – a collection of logically related data.

- Unix takes the file concept to an extreme – *everything in Unix is treated as a file.* (more about this concept later…)

- The file management system within the operating system should hide any device specific details from the applications.
- Typical tasks of the File Management System:
  - Controlling transfer of data to and from secondary storage.
  - Controlling security of file access.
  - Keeping track of storage space and maintaining file directories.
  - Provide sharing mechanisms for files
  - (Possibly) provide recovery and restoration mechanisms

3) Memory Management

- The operating system is also key to the control of data in *primary* storage (*main memory*).
- The main memory is a finite (small, and usually more expensive) resource that must be carefully allocated.
- Memory management is closely related to process management as the degree of multiprogramming that can be accommodated is dependent on allocating enough memory to all processes.

- In modern operating systems, memory allocation may be non-contiguous (a single logical object may be spread over several disjointed memory areas).
- A major function of modern memory management is the implementation of *virtual memory*.

The history of Unix

IBM7094

- CTSS (Compatible Time Sharing System) was arguably the first *time-sharing* operating system…ran on an IBM7094 at MIT around 1962 (64Kb main memory core). Users' jobs are swapped to/from tapes.
- Project MAC (Multiple Access Computers) at MIT was commenced in 1962 to discuss the future of computing.
  - Developed the specifications of an O/S called Multics (Multiplexed Information and Computing Service)

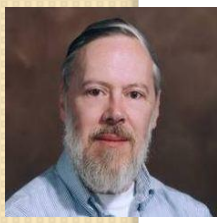
The history of Unix

- Multics initially developed by Bell Labs, MIT and GE.
- First ran on a GE645 computer in 1967. Written in PL/I (an early high-level programming language).
- Bell Labs withdrew from Multics development in 1969.
- GE computer division bought by Honeywell in 1970.
- Honeywell marketed Multics commercially, and sold less than 50 systems.
- New hardware development for Multics systems ceased in mid 1980s. Honeywell sold computer division to Bull.
- Last Multics site (Canadian National Defence) shut down in October 2000.

The history of Unix

Ken Thompson

Dennis Ritchie



Little used PDP-7 in corner of Lab →

- Although Bell Labs had formally withdrawn from Multics development in 1969, Ken Thompson and Dennis Ritchie (amongst others) continued operating system work on a "little used PDP-7 in the corner". The PDP-7 was a much smaller machine than the Multics GE645. Thompson had been also been writing a game (Space Travel) on this machine.
- File system ideas implemented first.
- Full operating system, shell, editor, assembler and simple file utilities follow…

The history of Unix

- Initially referred to as UNICS (as a joke – UNiplexed Information and Computing Service).
- Brian Kernighan suggested the name UNIX in 1970. UNIX was referred to in capitals initially. Dennis Ritchie has suggested this was because the team was very fond of the ability of their new typesetter to produce small caps. Ken Thompson states it should have been Unix.
- A note about the name: UNIX is now a trademark of The Open Group. Only those systems which pay royalties to the Open Group and agree to meet the Open Group specifications of UNIX are allowed to be branded with the UNIX brand. This includes many commercial Unix'es.

The history of Unix

- Initial version of Unix on the PDP-7 was written in assembly language.
- It was recognised by Thompson that portability would provide a key advantage for the O/S.
- What was an appropriate high-level language?
- Time for a bit of programming language history!

The history of Unix

- **ALGOL** was an early programming language (late 1950s) designed for scientific computing.
- **PL/I** was developed at IBM – a complex language that took inspiration from **ALGOL**, **FORTRAN** and **COBOL**. Was intended to provide scientific and business functionality but was not widely adopted. MULTICS written in **PL/I**.
- **CPL** was developed at Cambridge University around 1963-4, **FORTRAN**/**ALGOL** inspired…from that came a language named **BCPL** (distilling the "Basic" features of **CPL**).
- Ken Thompson wrote a language called **B** which was designed to simplify further **BCPL** to fit into limited memory on the PDP-7 and introduce new syntax. **B** was typeless (only had the "word" as a "datatype").
- http://helloworldcollection.de/

The history of Unix

- In 1970 a Bell Labs PDP-11 computer was used as a test bed for porting the Unix system from the PDP-7. There were problems with B on this architecture due to the typeless nature of B.
- Dennis Ritchie began extending B, adding firstly facilities for character, integer and arrays and pointers.
- This newly evolved language was developed during the period 1969-1973 (further major development of the language followed later), and was called

C

The history of Unix

Ritchie and Thompson porting Unix to the PDP-11

- During 1972/1973, Unix was totally rewritten in the C language.
- This was really the start of the Unix revolution. Unix was now written in a robust high-level language.
- The O/S continued to be developed over the years, with portability being an important criteria.

The history of Unix

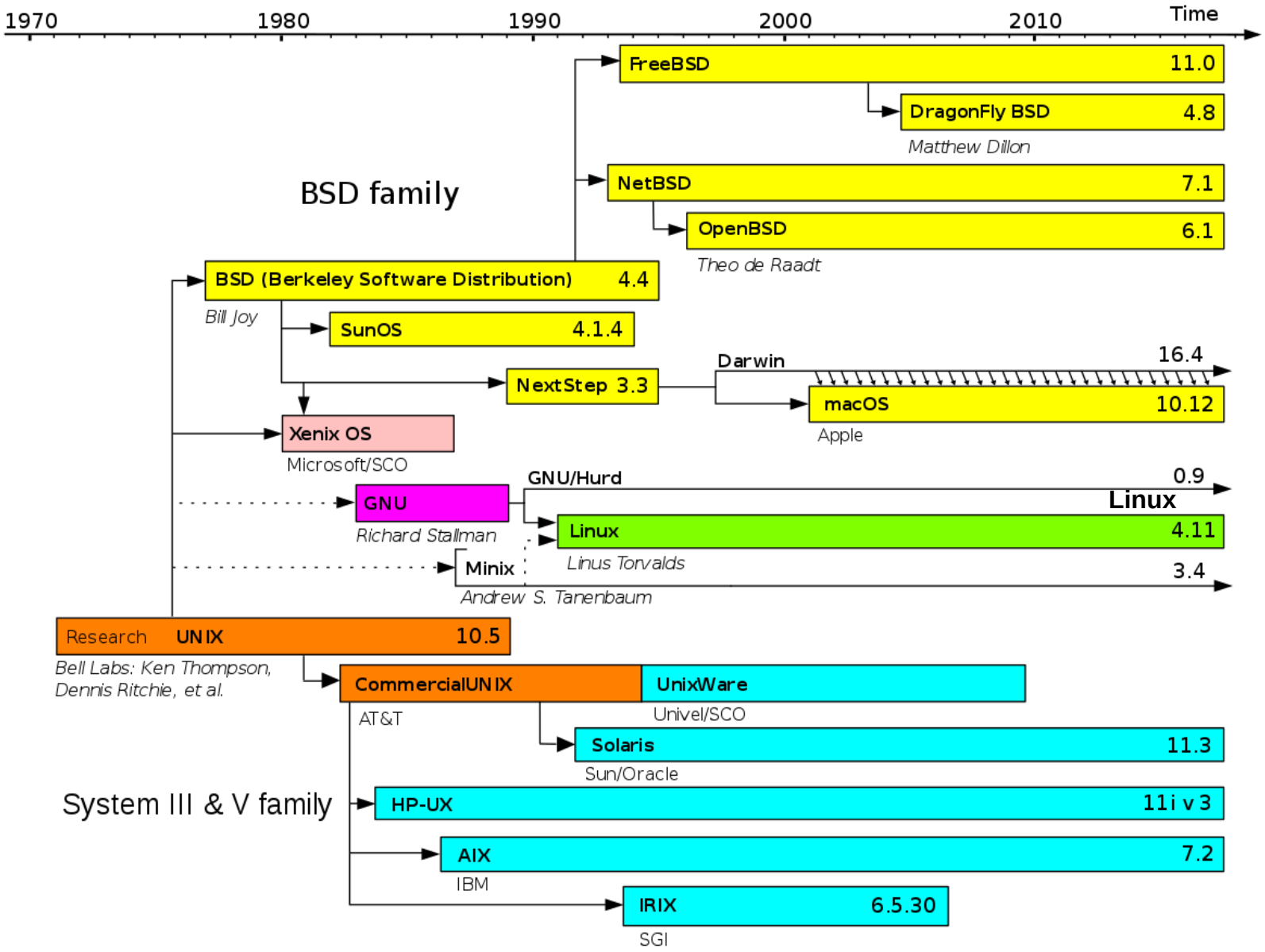
- One of the many institutions (138 by 1976) that had obtained a license to use Unix was the University of California at Berkeley.

- Ken Thompson had studied at Berkeley, and returned as a visiting professor in 1975. Chuck Haley and Bill Joy were graduate students at the time and started working on Unix development with others. They did a lot of work on a Pascal compiler for Unix, but also released a particular version of Unix with enhanced utilities called the *Berkeley Software Distribution (BSD)*. (Bill Joy is also famous for writing the *vi* text editor and was co-founder of Sun)

The history of Unix

- From here, it gets complicated!
- Traditionally, Unix versions have been described either as the BSD flavour or the System V flavour (AT&T).
- However, this has blurred somewhat with the advent of Linux. The Linux codebase (more shortly) was developed from the ground up.
- The time roughly between 1987 and 1993 was a time of intense competition and rivalry between the two flavours known as the "*Unix wars*".

The history of Unix



The history of Unix - GNU

- Richard Stallman (often referred to by his username ***rms***) is the father of the Free Software Foundation which included the GNU (GNU's not UNIX) project.

- Programmer/hacker at MIT from 1971 – produced the GNU manifesto in 1985 (a publication which advocated the creation of a free UNIX-like O/S)

- Software licensed under the GPL (GNU General Public License) essentially allows use, copying and modifying of code only if the same rights are passed on to other recipients.

The history of Unix - MINIX

- Andrew Tanenbaum (PhD UC Berkeley) wrote a UNIX clone from scratch called MINIX in order to support an operating systems course he was teaching. At the time AT&T did not permit the teaching of UNIX V6 internals. Full source code for MINIX was published as an appendix to his textbook *Operating Systems: Design and Implementation* in 1987.

- Current version (free) is MINIX Version 3 - (www.minix3.org)

The history of Unix - Linux

Where it all started…..

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: 1991Aug25.205708.9541@klaava.Helsinki.FI
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix – I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs.It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Unix versus Microsoft Windows

- Almost a religious debate
- Points of comparison:
  - Stability
  - Security
  - Flexibility
  - Interface
- Comparisons between the two operating systems will arise during the course.

UNIX Philosophy

- *Programs are tools*.
- Like any good tool, they should be *specific in function*, but usable for many different purposes.
- Within limits, the output of any program should be usable as the input of another program.
- All of the information needed by a program should either be contained in the data stream passed to it from another program or specified on the command line.
- The UNIX philosophy underpins much of the development of UNIX programs and operating system utilities.

Some Useful Texts/Resources

Refer to Moodle site for FIT9134

IMPORTANT: Week 3+4 Lectures

These are very important for the lab tasks – do not miss them