School of Computing

Edinburgh Napier UNIVERSITY

CSN07101 Operating Systems

Block 2

1 OPERATING SYSTEMS

Units in this section

1. OS types & Features
2. Multitasking
3. File systems
4. File system management
5. Memory management

Why is an O.S. needed?

An operating system is needed to make the hardware usable. It provides an interface that the user can use, is responsible for running software, and controls the hardware of the computer. The O.S. should isolate the user from the details of the computer. If you have learned how to use Windows, you should be able to sit down at any PC running that system and be able to get it to work, no matter who made the PC. Note that programs that the user wants to run (spreadsheets, word processors, databases etc.) are not part of the O.S. They are referred to as applications.

1.1 Types of OS

- Desktop ____
- Server ____
- Mobile ____
- Embedded ____

Desktop Operating Systems

These are generally designed to run on a single computer: e.g. Windows, Linux or Apple OS. They will have some common features:

- Allowing software to run.
- Providing a disk file system.
- Handling hardware devices.
- Providing a user interface.

Most have support for multiple users using the same computer: Controlling access to resources e.g. confidential files can only be accessed by certain people. There should be a way of allowing users to store files in their own folders (home folder). They will also have network support e.g. for connecting to remote resources such as Websites, Printers, Email, servers and storage. Usually there will be support for automated updates and mobile profiles where user information (e.g. Desktop info) may be moved from machine to machine allowing users to login on different machines, but see the same desktop and settings. Other features include remote logon/remote desktop and shells/user interfaces such as Linux shell/Xwindows

Server Operating Systems

Servers are computers dedicated to producing services to users within an organisation or across the internet. Typical server roles are to provide web, email, file, database, print, gaming and security services. They are generally aimed at providing features such as high-reliability, fast data-streaming, large data storage, or large computing power, but with no need for a good graphics response (typically there is no user sitting at the server, and normally no monitor).

Mobile Operating Systems

Typically used on Smartphones PDAs etc. Current offerings include Android (Google), WindowsPhone7/8 and iOS (Apple). Such OS are optimised for the limited hardware resources available to them. They may lack some of the more sophisticated features available to a desktop OS.

Real Time Operating Systems

Approximately 1% of processors are in computers. The rest are embedded in systems (cameras, cars, washing machines). Many embedded processors run special Real Time Operating Systems (OS-9, Windows CE, VXWorks). These tend to use less memory, provide good response to Real Time signals, and don’t require disks, keyboards & screens. The two Mars exploration rovers, as well as the Mars Reconnaissance orbiter both use VxWorks

OS Versions

Most operating systems have long family lives. Many different versions may be issued. Different versions of the same OS may appear on different platforms. An OS (e.g. Windows/Linux) may be configured for many differing hardware setups. Different versions of Linux/Windows/iOS may run on desktop, server or mobile devices

OS Configuration

An operating system may preconfigured by the supplier to install on a mobile device or to be optimised for server use. An operating system may configure itself during install or ongoing use: as new hardware is "discovered" or as updates are downloaded from the vendor. The user may configure the OS using the "control panel" or config files

1.2 Main Features of an Operating System

Running Applications Software

In a simple system, running software means running one program after another. As a minimum, the operating system must provide a way of loading a program (typically on disk) into RAM, then running the program in a controlled way. More sophisticated OSes (such as Unix or Windows) can run more than one piece of software at a time (*multitasking*). For instance you may be typing into a word processor, while a spreadsheet is doing a calculation

Multitasking

Controlling multiple tasks involves a number of things: making sure that each task runs for some of the time and making sure that resources are shared (e.g. so that tasks take turns to use the printer). The task that the user is currently engaged with usually runs faster (in the 'foreground', other tasks running in the background).

Time-slicing & multiple processors

Some computers work by having a number of application share one processor. The OS runs each application for a short time (a time-slice). If this is done in rotation, it appears as if the computer responds to each user. A modern OS will cope with multiple processors (or multiple processor cores on a single chip), distributing the work between them to achieve higher performance than is possible with a single processor.

Handling the hardware

The OS is responsible for controlling the hardware. A number of standard features may be built into an OS such as: reading and writing to disk drives; sharing RAM between tasks; reading the keyboard/mouse/touchscreen. The range of possible hardware supported by a mobile OS will probably be less. Adding new hardware may mean adding extra software (device drivers) so that the OS can use the new device.

An abstract view of Hardware

The user doesn't want to be concerned with the details of the hardware. A USB memory stick, a CD-ROM and a hard disk all work in quite different ways. The hardware will need totally different instructions to do things like store data. The user just wants to be able to do things like drag a file from the USB drive to the hard disk. It is the Operating system's responsibility to provide an abstracted/ idealised/ logical view of the hardware. The OS effectively does the same for software applications: providing a simplified view of the HW.

Security

The OS is also responsible for providing some aspects of the security within a system. For instance, Windows provides a firewall that can be used as part of the protection within a network. Some of the security is about controlling access: who gets access to files and with what rights. In a large organisation, this can be a complex area: the OS is only part of the system. Networking hardware also has a role to play.

Providing the user interface

Early computers accepted only commands typed at keyboards. Modern computers interact with the user using graphical displays, touch screens or a mouse. A mobile O.S. may support keyboard and screen. The Graphical interface is easy to use, but typed commands can be saved to a file that can be used again and again: command files/batch files/scripts.

1.3 Components of an OS

OSs are typically built around a software component known as a "Kernel". The Kernel is principally responsible for handling: communication ; sharing resources; scheduling (picking the next task to run); memory management features

MEMORY MANAGEMENT

Managing the computer's memory (RAM) is a part of the OS's function. Each task must have its own section of memory, and be protected from other tasks. Often there is not enough RAM to hold all the software: a section of the hard disk can be used as an extension of RAM (virtual memory).

MEMORY MANAGEMENT IN WINDOWS

Although many computers now run 64 bit versions of the operating system, much of the software still uses 32 bit addresses. 32 bit applications can refer to 2 Gb of memory, and you may have many applications running, each using up to 2GB. This could mean tens of GBs, but most computers have less RAM. The sections of software that are currently being used are held in RAM, other sections are held on disk in what is know as *virtual memory* (Pagefile.sys in Windows). If software refers to addresses that aren't in RAM, O.S. fetches from disk. Often this means swapping data from RAM to disk to make room. To simplify the process, the sections are all a fixed size, referred to as a page (4kbyte for 32 bit Windows, 8kbyte for 64 bit Windows, different for other OSs).

DEVICE DRIVERS

Every device needs software to run it. The commands to work one printer will be different from another printer. The application software and the user shouldn't have to know all these details. Device drivers handle the low-level details; they make the hardware look the same to the application software.

THE USER INTERFACE

Accepts commands from the user, and displays information. The UI may be a Graphical User Interface (GUI): MS-Windows or Unix/Linux X-Windows, or it may be a text based command line interface (CLI): Windows uses a CLI derived from the DOS prompt; UNIX also has various CLI "shell"s (C shell, Bourne)

THE WINDOWS COMMAND LINE

```
D:\>cd temp                              Move to the TEMP subdirectory
D:\temp>del *.*                          Delete all the files in the directory
D:\temp\*.*. Are you sure (Y/N)? y
D:\temp>cd ..                            Move up one level in the directories
D:\
```

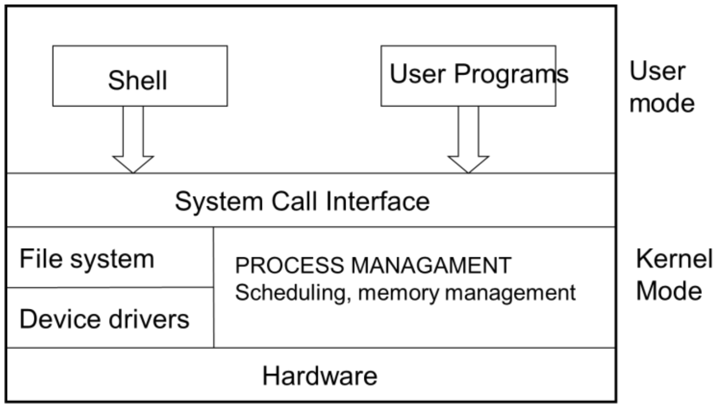
THE SYSTEM CALL INTERFACE

The Operating System will also provide facilities to programs and programmers. If you are writing a new application, you want to be able use the graphics display, disks and so on without having to know the details of the hardware. The System Call Interface provides a standard way of accessing these facilities via the operating system. For instance, Windows provides a standard 'API' (Application Programming Interface), usually called Win32 or Win64.

A WIN API EXAMPLE

To take an example, the **FillRect** function fills a rectangle by using the specified brush. This function includes the left and top borders, but excludes the right and bottom borders of the rectangle.

```
int FillRect(
    HDC hDC,              // handle to DC
    CONST RECT *lprc,     // rectangle
    HBRUSH hbr            // handle to brush
    );
```



OS STRUCTURES: THE 'TRADITIONAL' UNIX SYSTEM

1.4 LINUX…

WHAT IS LINUX?

Linus Torvalds wrote a kernel for an operating system with the same sort of functionality as Unix. This is now maintained by the open source community. Strictly speaking, Linux is only the kernel. The GNU project ('GNU is Not Unix') adds many essential features (e.g. the GNU compiler: gcc). It runs on all sorts of hardware, and free versions are available. Many versions are available as 'distributions': complete packages with applications included.

WHY RUN LINUX?

To make use of a PC you don't have a Windows licence for. To act as a server for email/web hosting etc. without having to pay the high price of a Server version of Windows (£300-500). Most commercial clouds (Amazon's EC2 cloud, Google Compute Engine, and the various OpenStack implementations) run on LinuxSecurity & reliability (fewer viruses/ malware). Specialist applications are available.

REQUIREMENTS

Usually not as demanding as the latest version of Windows. Typically runs well on a machine that is a few years old. Needs disk space (typically 2-5 Gbyte, but can be less). Can also be run from a 'live' CD or USB stick: nothing installed on the hard disk.

SETUP

Order a CD or USB stick, or download a distribution. Run from a USB stick only when it is plugged in. Or run a tool to re-organise your hard disk to make a free partition to put Linux on. Install Linux on this partition. Configure a number of options. Set up the system so that it will boot to Linux (or give you a choice of which O.S. to boot to: 'dual-boot' or 'multi-boot' systems).

WHAT DO THEY INCLUDE?

Details vary, but typically they will include: Linux kernel ; GCC (GNU C Compiler) & utilities ; Desktop system (usually based on KDE or Gnome) ; Web browser/email (e.g. Firefox/Chrome); Office suite (usually Openoffice); Multimedia (e.g. mp3/mp4 playback)

WHICH VERSION OF LINUX?

The website: distrowatch.com is devoted to reviewing and comparing distros, and giving news about Linux.

The website: www.thelinuxshop.co.uk has dozens of versions for sale on CD or USB

In 2013, Ubuntu & Mint were possibly the most popular version. Distrowatch keep a 'top 100' chart: an indication of how fragmented the Linux scene has become

1.5 MOBILE OPERATING SYSTEMS

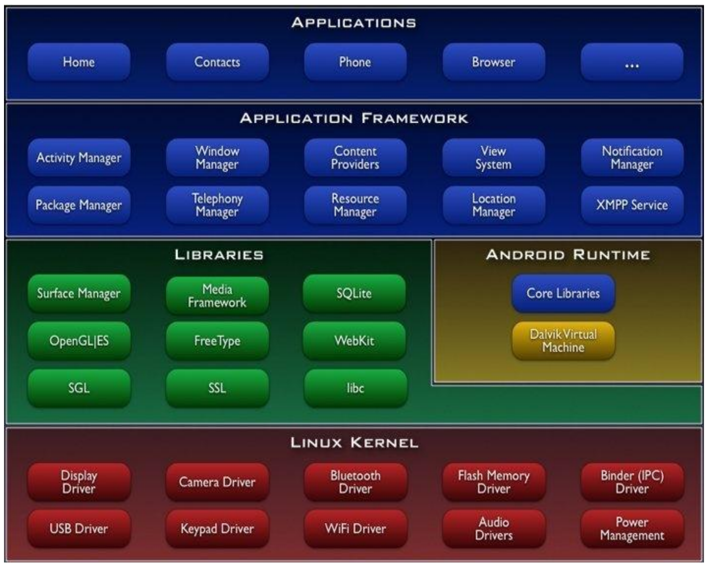
Operating systems that are designed to run on mobile devices. Typically they have access to fewer hardware resources. They normally support a limited range of applications software. They are communications oriented

ANDROID

Developed initially by Android Inc. Android Inc now a subsidiary of Google. Android is used on phones as well as tablet devices. Based on a Linux Kernel. Runs Java Apps

ANDROID

Apps are written in Java using a freely available SDK. Not all of the standard Java libraries are supported. Although a Linux Kernel is used Android is not regarded as Linux compatible.



ANDROID STRUCTURE

(Creative commons licence: http://developer.android.com/images/system-architecture.jpg)

2 MULTI-TASKING

Contents

INTRODUCTION TO MULTITASKING

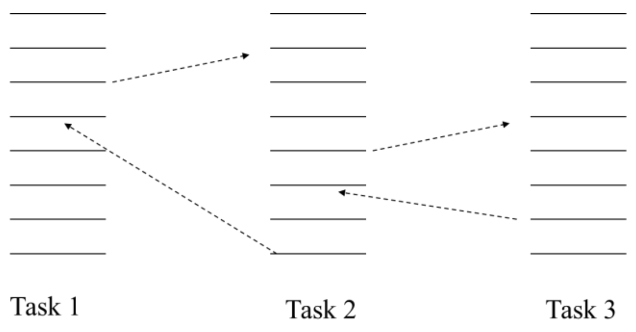
Large computers (e.g. Unix machines) have used multitasking for years. Partly this is to let multiple users run appications on a common, shared computer. Early PCs running DOS and Apple Macs didn't multitask. From the mid 90's versions of Windows and Mac-OS have multitasked.

2.1 TIMESHARING & ROUND-ROBIN

By switching the processor between different tasks (typically every few milliseconds), it can act as if it was running more than one thing at once. This used to be done a lot on multi-user computers. Jim's code would run for 10 ms, then Frank's would run for 10 ms, then Al's, then back to Jim's. As long as the gaps are short, it appears like the computer is responding to each user. This is referred to as a 'Round-Robin' system.

2.2 PC MULTITASKING

Now that PCs are cheap enough for every user to have one, there is less need for the 'timesharing'. For a single user system, the need is to run multiple applications. While the word-processor is sending a large document to the printer, the web browser can be downloading pages from the web, and the spreadsheet doing a calculation. Normally, this is done to get the best out of a single processor, but increasingly there may be more than one processor for these tasks to be shared over.



Questions: How many processors can you get in a PC nowadays?

How many tasks are there to share out in Windows?

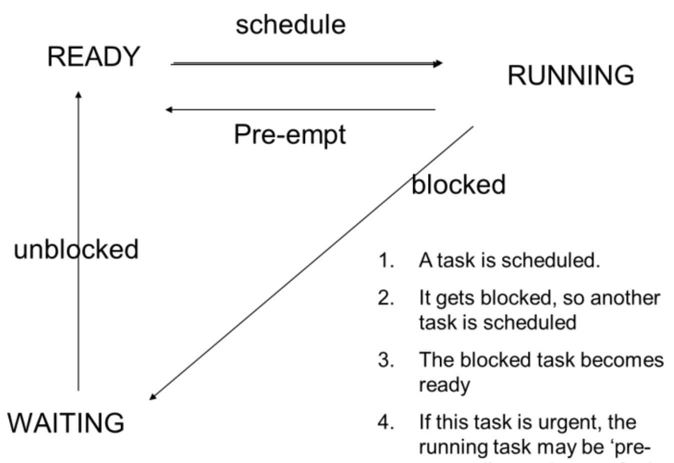
2.3 A TIMESLICE

Efficiency is gained from sharing out the processor time between tasks. If one task gets blocked (perhaps waiting for data), another one can be found to run instead. The length of time each task runs for can vary, depending on whether it gets blocked. Even if it doesn't get blocked, it will usually get interrupted after a few milliseconds to let other tasks run. The timeslice has to be long enough to run a significant amount of software: it is the Operating System that organizes this.

2.4 TASK STATES

In a typical system, one task will be running. Some tasks may be ready to run, and are just waiting to get on the processor. Others are likely to be blocked: waiting for data or some event. When a task finishes running (for whatever reason), another task has to be picked to run. It will be chosen from the list of ready tasks. Choosing which task to run next is known as 'scheduling' and affects the performance.

Question: How many tasks can be in each state?



1. A task is scheduled.
2. It gets blocked, so another task is scheduled
3. The blocked task becomes ready
4. If this task is urgent, the running task may be 'pre-empted' to make way for it.

2.5 SCHEDULING

Picking which task to run next is important. It may be done for fairness when there is more than one user (run each user's code for 10 ms slots). Commonly it is done by priority: some tasks will be more urgent than others, so the most urgent is picked next. Often tasks that interact with peripherals are urgent: data may have to be fetched from, say, the network interface card quickly. If it is not read quickly the data may be lost.

Windows

Current versions of Windows allow for multitasking in a much more advanced form than the early versions of Windows such as 3.1, which used a cooperative system. Windows allows for multiple processes to run on multiple processors using shared memory. Typically, each application is run as a process in its own memory space. Each process can consist of a number of threads, each of which can run concurrently. For instance a word processor might have threads for file I/O, user interface, spell-checking, and so on.

Priority levels

In Windows, processes have 32 priority levels (0 is the lowest priority, and 31 is the highest) divided into two classes: real-time and variable. Priorities of 16-31 are used for real-time processes, such as the kernel, that are not swapped to page file. They have a fixed priority and are scheduled ahead of other threads. Priorities of 0-15 are used for dynamic applications: user applications, and non-critical bits of the O.S. that could be written to the page file.

User control of thread priority in Windows

The user can provide some control when starting a task by giving it one of three priority levels (low = 4, normal = 8, high = 12). The command *start /normal myprog* would start myprog with a base priority of 8. If the user has Administrator privileges, she can also start a process with base priority = realtime (24). Realtime threads have a fixed priority. Variable priority threads have a base priority, and a dynamic priority that can vary about the base priority of the parent process. If a thread has to wait on I/O, its dynamic priority will rise. Scheduling is pre-emptive: the highest priority thread is always run.

2.6 Threads

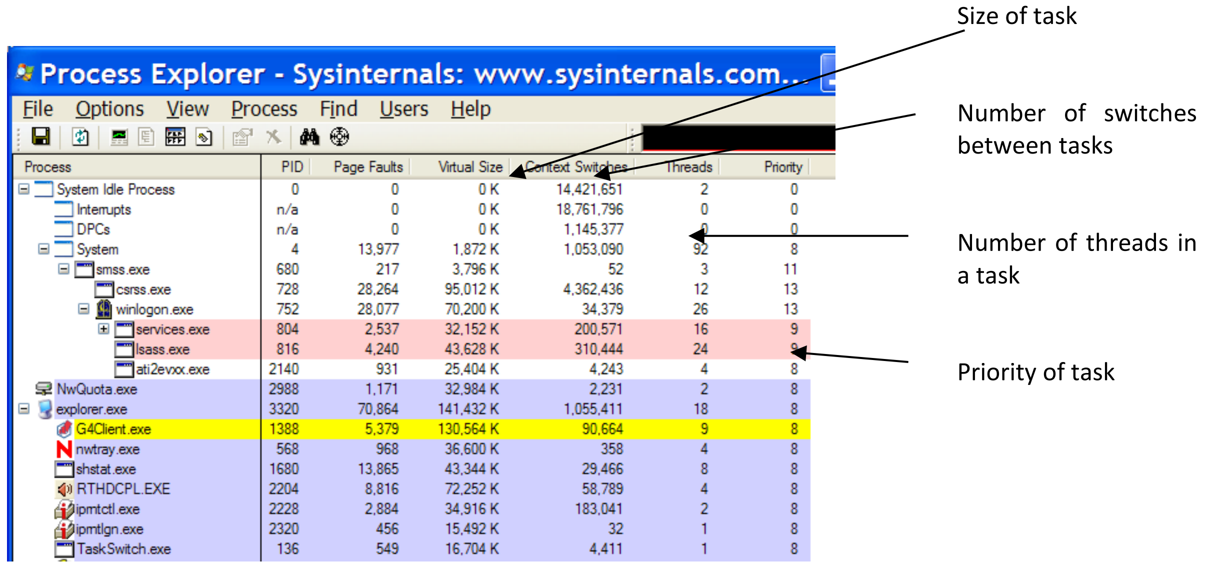
The basic unit, the thread, is not quite the same as a task or process; it is sometimes referred to as a Light Weight Process (LWP). A process may consist of a number of threads which share resources and memory. They may be scheduled concurrently on multiple processors, and time-sliced on single processors. They are called 'Light-Weight' because the kernel only maintains a small context for them; switching between threads within a process requires little overhead. Switching between processes takes a full context-switch.

Hyper-threading

Dual and quad core processors are relatively easy to understand: complete processors are implemented on a single chip. Hyper-threading has been used by Intel as a sort of half way stage to multiple cores. In a hyper-threaded core, there is a single processor and limited duplication of resources. A hyper-threaded core might have 2 sets of registers and 2 program counters. To the operating system it would look like there were 2 processors, but there might only be one Arithmetic unit and one cache. If only part of the core is duplicated, it does not take as much silicon as a complete duplication of cores. For example, the PCs in C6 & C27 have 4 cores, each of which can run two threads, so 8 streams of instructions can be executed at any one time.

2.7 Task Manager/Process Explorer

This is a very good tool for showing information about what tasks are running. Confusingly, it sometimes refers to tasks as processes, another name for the same thing. You can get Task Manager to display current tasks, how often the scheduler switches between tasks, what the task priorities are, and so on. An alternative, Process Explorer, gives even more detail.


Process Explorer - Sysinternals: www.sysinternals.com...
File
Options
View
Process
Find
Users
Help
Size of task
Number of switches between tasks
Number of threads in a task
Priority of task

Process	PID	Page Faults	Virtual Size	Context Switches	Threads	Priority
System Idle Process	0	0	0 K	14,421,651	2	0
Interrupts	n/a	0	0 K	18,761,796	0	0
DPCs	n/a	0	0 K	1,145,377	0	0
System	4	13,977	1,872 K	1,053,090	92	8
smss.exe	680	217	3,796 K	52	3	11
csrss.exe	728	28,264	95,012 K	4,362,436	12	13
winlogon.exe	752	28,077	70,200 K	34,379	26	13
services.exe	804	2,537	32,152 K	200,571	16	9
lsass.exe	816	4,240	43,628 K	310,444	24	9
ati2evxx.exe	2140	931	25,404 K	4,243	4	8
NwQuota.exe	2988	1,171	32,984 K	2,231	2	8
explorer.exe	3320	70,864	141,432 K	1,055,411	18	8
G4Client.exe	1388	5,379	130,564 K	90,664	9	8
nwtray.exe	568	968	36,600 K	358	4	8
shstat.exe	1680	13,865	43,344 K	29,466	8	8
RTHDCPL.EXE	2204	8,816	72,252 K	58,789	4	8
ipmtctl.exe	2228	2,884	34,916 K	183,041	2	8
ipmtlgn.exe	2320	456	15,492 K	32	1	8
TaskSwitch.exe	136	549	16,704 K	4,411	1	8

3 FILE SYSTEMS

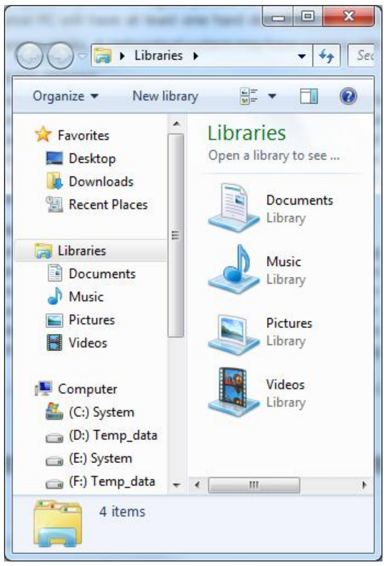
1) Mass storage
2) Files & file names
3) File types &extensions
4) Attributes
5) File operations
6) Directories
7) Paths

INTRODUCTION

The file system is probably the most obvious aspect of the Operating System. Whatever system is being used (Windows, Unix, Linux, Mac-OS), the user has to be able to save and open documents, programs, spreadsheets, and other types of files. Most Modern Operating systems provide facilities for doing this within a hierarchical file system: there will be some mechanism for organising files into folders or directories (and sub-folders/ sub-directories).

3.1 MASS STORAGE

File systems present a logical picture of a complicated storage system using, potentially, many different types of storage units. A typical PC will have at least one hard disk drive, a DVD/CD-ROM drive, and perhaps removable USB memory sticks. A networked system may have servers with large disk drives that can be backed up onto tape. Increasingly cloud-based storage is used.



DEVICE DETAILS

The problem with these drives is that they all work in different ways. Obviously, a command that will tell a CD re-writer to burn a new CD will not work if sent to a USB memory stick. The detailed workings of these devices are very complex. For instance, a simple read from a hard disk involves sending the cylinder, head and sector addresses plus various other parameters. The disk drive typically then sends back the data plus about 6 status bytes.

LOGICAL PICTURE

The operating system should handle this complexity (using 'device driver' software), leaving the user free to concentrate on a logical view of the file system. The user should be free to issue a command (or drag a file icon) in order to copy a file from a hard disk to a USB flash drive without having to know the details of the underlying hardware. Of course, the user will have to learn commands or techniques for working with the O.S., but these should be simple.

Where, for instance, is the 'Music' folder?

3.2 FILES

Files are an abstraction mechanism. They will be collections of ones and zeroes stored on disk or flash chip in a USB stick in different ways. They might be magnetic domains on hard disks or tapes, or pits on CDs that reflect light in two different ways. The Operating system should present a logical picture of these to the user so that they become a convenient mechanism for storing different types of data, then retrieving it later.

FILE NAMES

When a process creates a file, it gives it a name. When the process ends, the file continues to exist, and can be accessed by other processes. File naming rules vary between Operating Systems. Almost all will allow a string of 1-8 characters, e.g.

```
eric        mydoc          report
```

Most will allow some special characters (such as numbers and some punctuation marks):-

```
report2      secret!      Fig.1-12
```

Some file systems (e.g. Unix, Linux) distinguish between upper and lower case letters. The following would all be different filenames:

```
Jane        jane        JANE        JaNe
```

Other systems (e.g. MS-DOS) are not case sensitive; the previous examples would all refer to the same file. The length of file name can vary. Some (e.g. MS-DOS) impose a severe limit, typically of 8 characters or so. Others (Windows) allow huge names of up to 256 characters or so. Confusingly, the system may have to cope with either.

Question: Are Windows file names case sensitive?

3.3 File Types & Extensions

Many systems allow an extension to the file name, e.g **prog.c** The extension, usually after a full stop, is used to denote something about the file (in this case it is a C program). MS-DOS allows an 8 character file name, followed by an optional 3 character extension:

```
prog.c          report1.txt        data.bak
```

Unix allows extensions of various sizes. In fact, there can be two or more extensions. A C program that has been compressed might be: `prog1.c.Z`

Sometimes the extension is a convention, useful as a reminder to the user. However, it may be essential (a C compiler may only be able to compile files ending in **.c**)

Example Extensions

File.bak	Backup file
File.txt	Text file
File.html	WWW document
File.zip	Compressed file
File.c	C program
File.tar.gz	Files collected into tape archive format & then compressed
File.gif	Graphical Interchange Format image
File.jpg	Image encoded using JPEG standard

Question: Does Windows allow multiple extensions?

The 8.3 Limitation & LFN

Windows was originally based on MS-DOS. This restricted file names to the 8.3 character format. Newer versions of Windows use Long File Names of up to 255 characters. These names are case-preserving but not case sensitive. This means that you can use capital letters to make the names clearer when they are displayed (**ResearchJanReport.doc**), but that the O.S. ignores the capital letters (**researchjanreport.doc** is the same file). There are some restrictions. Most characters, such as spaces, can be used, but not / \ : | * ? " < > If necessary, say to run an old MS-DOS program, an 8.3 name can be generated: **RESEAR~1.DOC**

3.4 Attributes

As well as a name and data, every file has other information stored about it. The details vary between operating systems, but typically include the date and time the file was created, and the size of the file. Other attributes relate to whether the file has been archived, when it was last modified, and such aspects as security (who can do what to the file). The following table gives examples of these, though they may not all be present in a particular O.S.

File Attributes

Protection	Who can access the file to do what
Password	Password needed for access
Creator	Id of the person who created the file
Size	Number of bytes in the file
Read-only	Can the file be written to?
Hidden	File may not be shown in listings
System	File contains system information
Archive	Has the file been backed up?
Creation time	When the file was created
Change time	Time when last modified

3.5 File operations

The purpose of files is to store data, and make it available for later use. The user may wish to operate directly on files using commands or a Graphical User Interface (GUI). For instance, a file may be created using **File..New..** in Windows Explorer. Programs too have to be able to manipulate files. This is done by making a number of system calls available that programs can use, for instance to add data to a file. The GUI will actually work by translating mouse and menu commands into the system calls that perform the required actions.

Typical File Operations

1. **CREATE**: The file is created with no data; this sets up the attributes.
2. **DELETE**: File is removed from the system.
3. **OPEN**: Opening a file before using it allows the attributes and disk addresses to be read into main memory for rapid access.
4. **CLOSE**: This frees up memory that has been used to hold attributes and disk addresses.
5. **READ**: Fetch data from the file. It may be necessary to specify how much data, and a buffer to put it in.
6. **WRITE**: Store data. This may overwrite existing data. As with read, a pointer locates the current position n the file.

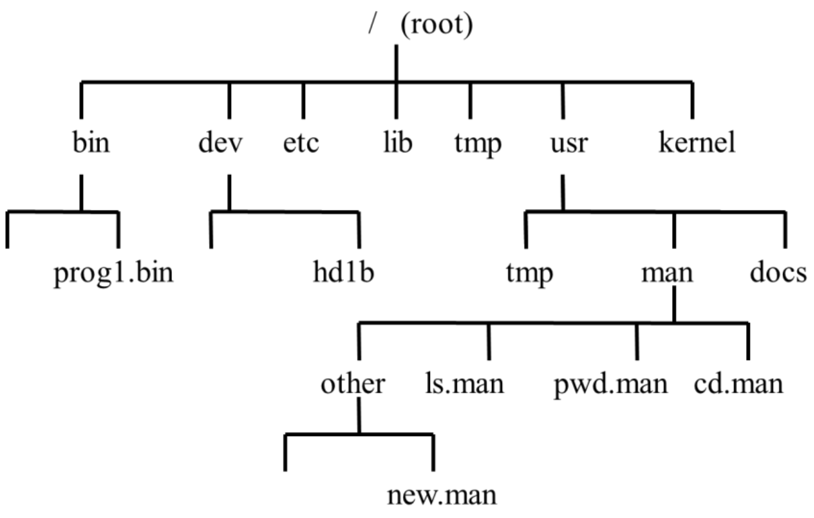
- **APPEND**: Like write, but can only add data to the end of a file, so no data is overwritten.
- **SEEK**: Some files can be accessed at a random position (rather than sequentially). Seek is used to specify where the file is to be accessed.
- **SET ATTRIBUTES**: Some of the attributes are adjustable. For instance, a program may want to change the access control attributes.
- **RENAME**: Not all systems allow this: it may be necessary to make a copy with a new name

3.6 Directories

Early Personal Computers used relatively small hard disks (10 Mb). As these could only hold a limited number of files, they were stored in a single directory. As numbers of files (and users) grew, it made sense to allow for multiple directories. This approach had been used for a long time on multi-user computers. Having multiple directories allows logical organisation of files. For instance, all word-processed documents might be in one directory, programs in another. Having multiple directories does add to the complication of navigating around the directory system.

Hierarchical directories

Example: Unix



Directory Operations

- **CREATE:** Create an empty directory.
- **DELETE**: Delete a directory (only works if empty).
- **OPENDIR**: Analogous to opening a file.
- **CLOSEDIR**: Close to free up memory space.
- **READDIR**: Read the next entry in the directory list.
- **RENAME**: Obvious
- **LINK**: In some systems, files can appear in more than one directory. Link sets this up.
- **UNLINK**: removes the specified link.

3.7 Paths

The added complication that comes with having a multi-levelled directory system is that the user has to navigate around it. This task is facilitated by a graphical interface showing the directory structure pictorially. Programs (and users of text-based interfaces) have to be able to specify the path to files. This is usually done absolutely or relatively. With absolute path names, the path from the root is specified. For instance, in the previous diagram, there are two files that can be referred to as:

```
/bin/prog1.bin
/usr/man/ls.man
```

Question: What is the path to the new.man file?

Working Directory (WD)

Most Operating Systems have the concept of the Working Directory: there is a directory that is assumed if no other directory is being specified. In the diagram, if the current (or present) WD was **man**, then all the files **ls.man, pwd.man, cd.man** could be referred to by name without having to specify the path. You can use both forms. To delete a file using the remove command (**rm**):

```
rm ls.man        rm /usr/man/ls.man
```

If you refer to a file in another directory, you must specify the path to get there. If **man** is the WD:

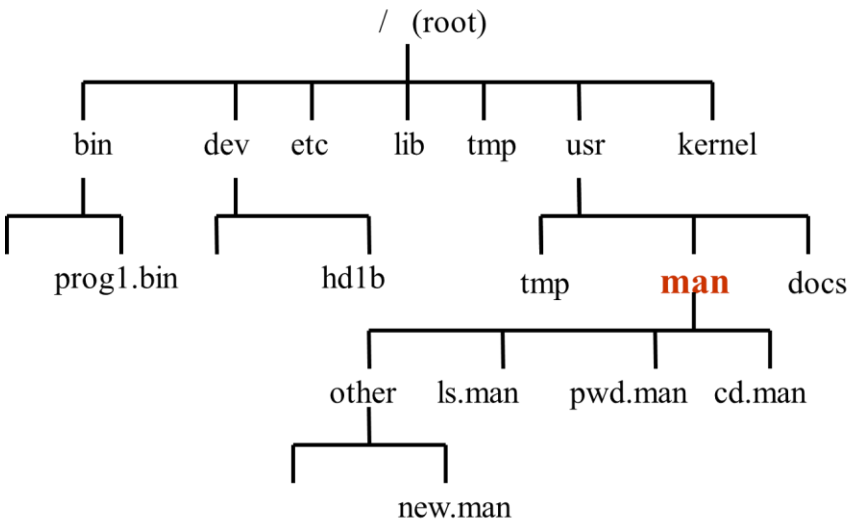
```
rm /bin/prog1.bin
```

Relative Paths

In the previous example, the command: `rm prog1.bin` would not have worked, since the file is not in the WD. Specifying the absolute path from the root is one way of getting to it. The other way is to specify the path relative to the current WD. Unix & Windows use a handy shortcut; the WD can be referred to as **.** The parent directory can be referred to as **..**

For instance, if I was in the **`/tmp` directory**, I could use the following to remove the **`ls.man`** file:

```
rm  ../man/ls.man
```



More Examples

Assuming that the present WD is man, then the following could be used:

```
cp  ls.man  ../tmp/ls.man
```

(copy the **ls.man** file from the PWD to the **tmp** directory).

```
cp  cd.man  ../tmp/cd.bak
```

(copy the **cd.man** file from the PWD to the **tmp** directory, and rename the copy **cd.bak**).

```
cp  /bin/prog1.bin  prog1.bak
```

(copy the **prog1.bin** file from the **bin** directory to the working directory, and rename the copy **prog1.bak**).

Source & Destination

The previous examples show that, for a copy, you have to say the source and destination:

```
cp  source_file destination_file
```

For the source and destination, you either have to specify the path to get to the files, or the PWD will be used. This pattern is used for lots of commands: if the file is in the Present Directory, just give the name, otherwise you have to specify how to get to it

Changing Directory

A system that uses the idea of the Present (or Current) Working Directory must provide information about it. The Unix command pwd will print the Present Working Directory. In some systems, the present working directory may appear in the prompt.

Also, it must be possible to change the PWD. The command cd can be used to change directory:

cd /bin

It's quite easy to get lost in a complicated directory structure. There is a handy shortcut for your home directory, the tilde sign ~

cd ~ return to your home directory.

3.8 File and Folder Permissions

In a networked system, files and folders have to be made available, but in a controlled manner. It is important to be able to specify access. This is usually done by controlling which groups get what access to an item (in large systems with many users it is messy to give permissions to individuals). Both NTFS (used in NT/2000/XP/2003/Vista/7/8…) and Linux/Unix have systems for controlling access. They differ in the details.

Folders and files

Permissions tend to be slightly different for folders (also called directories) and files. Sometimes the meanings of the permissions are obvious, for instance the Read permission for a file means you can read the contents of the file; Read permission for a folder means you can read the contents of the folder. Other times it is not so clear. Execute permission for a file means you can run that program, but what is execute permission for a folder?

Linux/Unix permissions

Files and directories have 10 bits that determine their properties. The first bit determines whether the file is a directory or not (a directory is just a specialised type of file). The other nine bits are grouped into 3 sets of three. The first set of bits determine the permissions for the owner of the file. The next set of bits determines the permissions for the group assigned to the file. The last set of bits determines the permissions for all other users.

Example: **- r w x r w - r - -**

The first bit is a -, showing this is a file. The next three bits (r w x) show the permissions for the owner/user. The next three bits (r w -) show the permissions for the group associated with the file. The final three bits show the permissions for other users. The permissions may also be shown in binary, with a 1 representing the permission is allowed.

So, **rwx rw- r–** could be **111 110 100** Also it could be shown in decimal **7 6 4**

Directory example: d r w x r - x r - x

The first bit shows that this is a directory. The next three bits show that the owner can list the directory contents, add and remove files and access sub-directories. Both the group and other users can access the directory and list its contents, but not add or remove files from the directory. Sometimes the X permission for a directory is referred to as the access permission. It determines whether you can 'cd' into the directory.

Questions: What would the above Directory permissions be in decimal?

What would the bits look like for a file where the user (owner) could read and write, and everybody else could only read?

Question: How many different permissions are there?

Permission	Meaning for a directory	Meaning for a file
r	List the directory's contents	Read contents
w	Create or remove files in directory	Write contents
x	Access files and sub-directories	Execute program

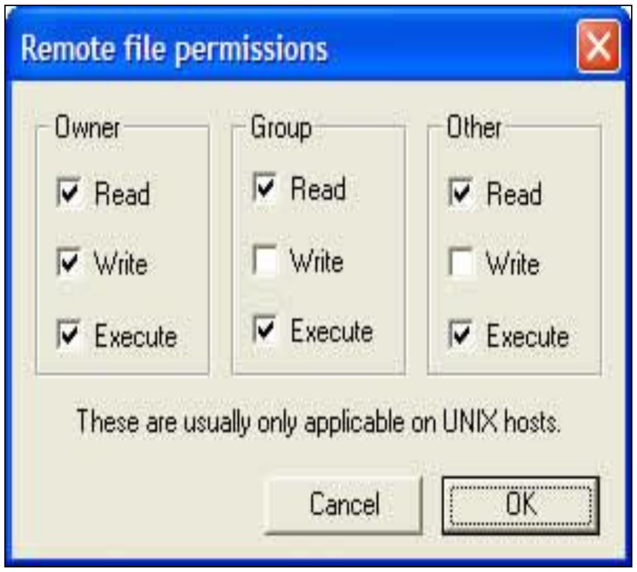
Directory and file permissions

Changing permissions

The *chmod* command can be used to change the permissions of files and directories: you will see this in the tuts/labs. The *chown* and *chgrp* commands can be used to change the owner or group associated with a file/directory. You have to have specific rights to do this (normally be the owner, or 'root' superuser).

Web servers

These permissions are important for most web servers, as they commonly run Linux/Unix. If you are accessing them from a Windows PC, they may be displayed thus->


Remote file permissions

Owner: ✓ Read, ✓ Write, ✓ Execute
Group: ✓ Read, ☐ Write, ✓ Execute
Other: ✓ Read, ☐ Write, ✓ Execute

These are usually only applicable on UNIX hosts.

Windows/NTFS Permissions

Like Linux, NTFS has a set of permissions that can be used to control who has access to files and folders. They are more complicated than the *nix permissions. Because some are very specialised, and only used on rare occasions, there are limited sets of standard permissions (also referred to as 'basic' permissions) that cover most situations.

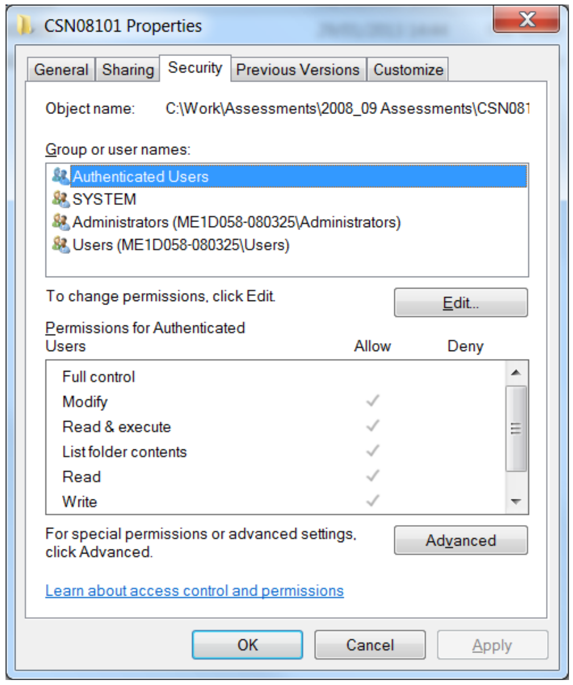
Some of the NTFS Permissions

Read Data	List Folder
Write Data	Create Folder
Execute File	Create File
Delete	Append Data
Change Permission	Read Attributes
Take ownership	Write Attributes

(you will not normally need to know these in detail)

Standard File Permissions (you should know these)

Read	Read file and permissions/ attributes/ ownership
Write	Modify file and attributes, view ownership & permissions,
Read & Execute	All of Read, plus execute the file.
Modify	All of Write, Read & Execute, plus delete the file.
Full Control	All of the other permissions plus change permissions and take ownership.



CSN08101 Properties

General | Sharing | Security | Previous Versions | Customize

Object name: C:\Work\Assessments\2008_09 Assessments\CSN081

Group or user names:
- Authenticated Users
- SYSTEM
- Administrators (ME1D058-080325\Administrators)
- Users (ME1D058-080325\Users)

To change permissions, click Edit. [Edit...]

Permissions for Authenticated Users	Allow	Deny
Full control		
Modify	✓	
Read & execute	✓	
List folder contents	✓	
Read	✓	
Write	✓	

For special permissions or advanced settings, click Advanced. [Advanced]

Learn about access control and permissions

OK | Cancel | Apply

Standard Folder Permissions (you should know these)

Read	See files, folders,permissions
Write	Create files/subfolders
List Folder contents	See file/subfolder names
Read & Execute	Read, move through folders
Modify	Delete the folder
Full control	Delete, take ownership

(some of these imply permissions on the files in the folder)

Groups and 'Deny'

In a network with a server, Windows has a very sophisticated system of groups. Again permissions are normally applied to groups. Users can get rights by being members of several groups. This gets complicated, so it is sometimes useful to 'Deny' access to a user or group. This overrides other permissions.

The security form

If permissions are selected that are a superset of other permission, the lesser permissions are selected as well. For instance, selecting the **Read & Execute** tick box means that the **List Folder contents** and **Read** tick boxes are selected as well.

Combining Folder & File Permissions

File permissions take precedence over folder permissions. If a user has read permission for a folder, and write permission for a specific file in the folder, they get write permission to the file. This applies even if the user has No Access to the folder. If they have, say, Read access to the file, they can get to the file, though they will have to type in the full path.

Inheritance

It is possible to set the permissions to be inherited. This means that a sub-folder created within a folder can inherit the permissions from the parent folder. This can be very useful if you want to set the same permissions for a whole set of folders and sub-folders. You may have to switch off inheritance if you want to specify that a file or folder has different permissions from its parent.

4 File System Implementation

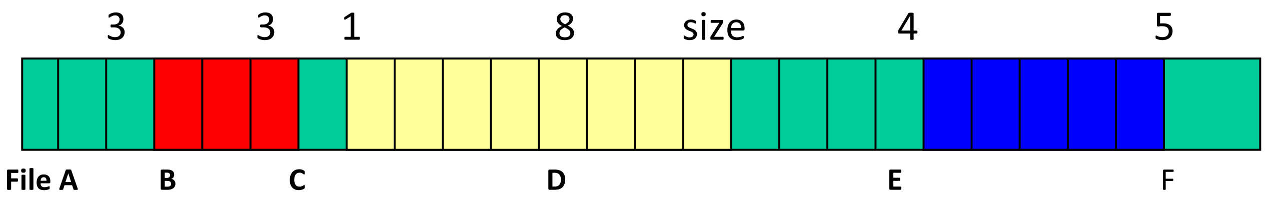
- Disk Layout
- Contiguous files
- Linked lists
- File Allocation Tables
- Indexing & inodes
- NTFS
- Distributed File Systems

4.1 Disk Layout

A typical disk drive is divided into partitions. If necessary, each partition can contain a completely different file system. For instance, a Windows system on one partition, and Linux on another. Usually the first sector (0) of the disk is reserved for the Master Boot Record: locating the code used to boot the computer. After that, the disk will be divided into blocks (typically of 1 - 4Kbytes). The block is the smallest unit of data that can be read or written.

4.2 Contiguous Allocation

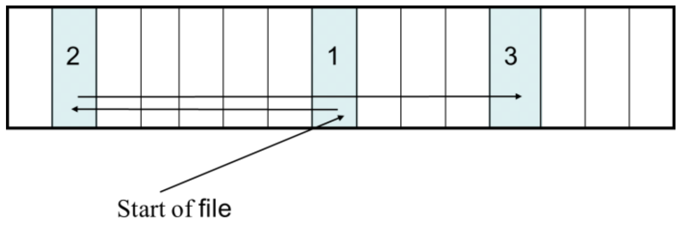
The simplest way to lay out files is in a sequence of contiguous blocks (i.e. blocks in consecutive locations). If the block size is 1K, and the file is 60K, then 60 blocks in a row are allocated.



Advantages & Disadvantages

This is a very simple scheme to operate. All that is needed is 2 numbers per file: the address where the file starts, and the number of blocks it contains. To get to any block is simple. Contiguous files can be read & written very quickly. Once the disk head is at the start of the first block, it just reads all the data directly. There is a big drawback: fragmentation. When files are erased, they leave different sized holes. Starting a new file means finding a suitable gap. The consequences of fragmentation are severe. Suppose you want to create a word-processed document. When you start it, the system will need to know how big it will be so that it can look for a suitable gap. This isn't always possible. So, contiguous file allocation is no longer used for magnetic disk drives. However, it is ideal for CD-ROMS. The layout of these is fixed when they are recorded. There is no problem with fragmentation, and the good response is ideal for CD-ROM drives that are usually slower than magnetic disks.

4.3 Linked Lists



If a linked list is used, the blocks that make up a file no longer need to be located together. This avoids the problem of fragmentation. Each block contains a pointer to the next. Locating any particular block is then a matter of navigating the links.

Advantages & Disadvantages

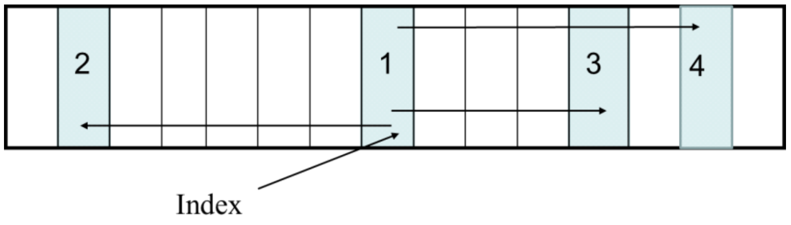
Linked lists are still simple to implement. As a file can consist of blocks anywhere on disk, there is no problem with fragmentation. Reading or writing the file serially is simple: just follow the links. A disadvantage is that some space is used up providing the links. However, the main disadvantage is that it is hard to randomly access the file. Suppose you want to get some data from the 151st block. All the blocks before that have to be read in order to find out the location of the one block of interest.

4.4 File Allocation Table

The previously mentioned disadvantage can be removed by extracting the links from the actual blocks, and putting them in a separate table, usually known as a File Allocation Table (FAT). This is the strategy adopted by MS-DOS and Windows 95/98/ME. It works well until the disk size becomes very large. Then the table can become unmanageable (the table is normally held in Ram for rapid access). Its relative simplicity means that is has become the standard for file storage on small portable systems: USB sticks, digital cameras, mp3 players and so on.

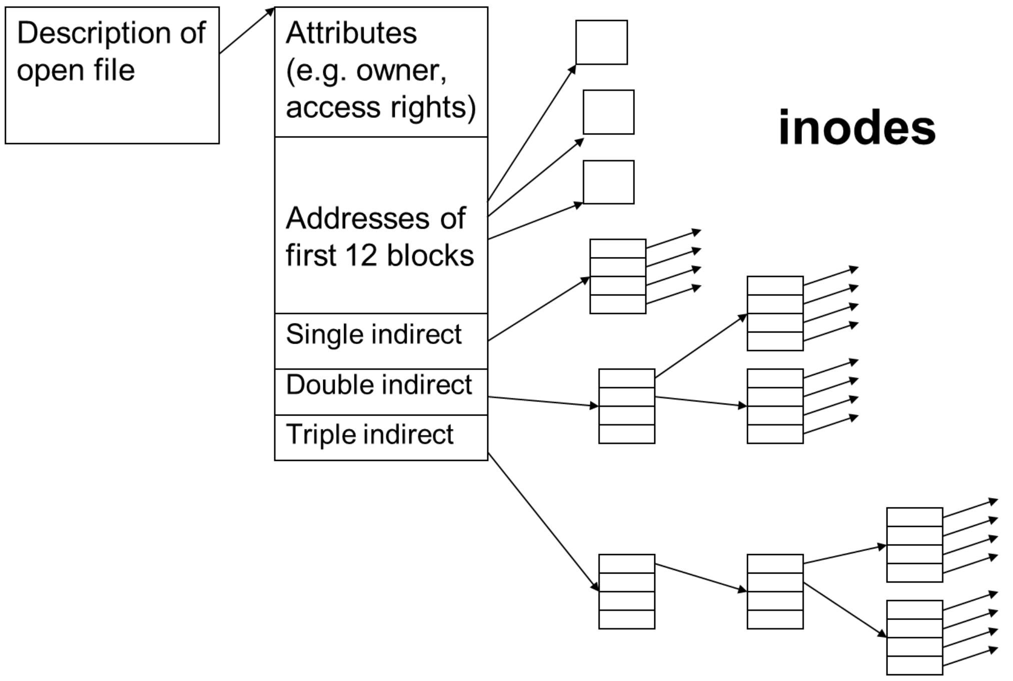
Indexing

Linked allocation is not very efficient when there is no FAT: the pointers must be found all over the disk to get the blocks in a file. Gathering all the pointers together and storing them in one index block means they can be retrieved easily:



4.5 INODES

The standard file system in Unix uses inodes (also written as I-nodes). This is short for index nodes. This has some similarities to the FAT system, but System Ram will only contain details of files that are currently open. This keeps the size of table in RAM to a more manageable size. Another feature of the inode system is that the inode will contain the addresses of some data blocks, but it will also contain indirect links to many more data blocks.

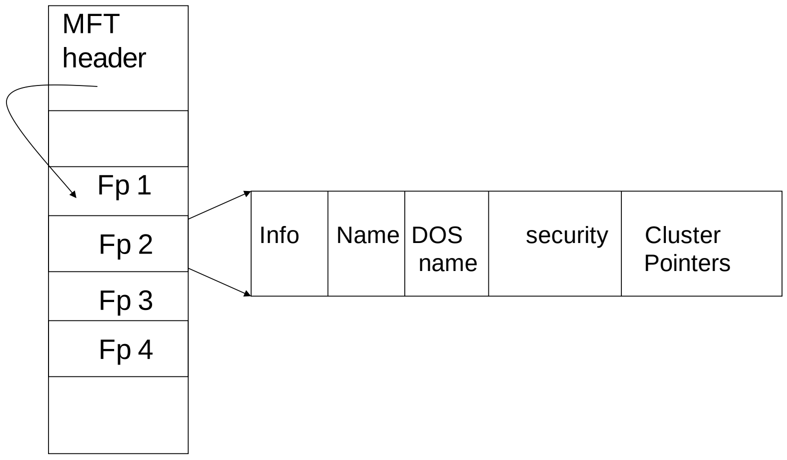


The Linux file system

Linux can use different file systems, for instance it may have to cope on a PC with its own files plus Windows files. The commonest Linux system is ext3fs, which uses the inode method. However, specific details, such as the size of block, are different from standard Unix implementations. Like Unix, Linux treats many things as if they were files, not just objects stored on disks. Files are basically objects that can handle streams of data. Although these could be from disk, they could be streams of data from the network or an I/O device (the screen or keyboard).

4.6 WINDOW'S FILE SYSTEM: NTFS

DOS & Windows grew up with FAT (in 16 bit, then 32 bit versions). So many people still use these that they will be supported for some time to come, but they do have severe limitations. NTFS is a much more sophisticated system that provides high levels of security, as well as disk compression & encryption. Files can be big (in theory up to 2^{64} bytes, in practice up to 16 TB). File names can be long (255 characters) and are stored in Unicode. This is a 16 bits-per-character code that allows other alphabets (e.g. Japanese, Indian).



The Master File Table

MFT DETAILS

Each file is identified by a unique 48 bit number. This is used as in index into the MFT. Every file and sub-directory has an entry in the MFT containing names, attributes and pointers to the actual data. The MFT is so important the there is a partial copy of it in case the MFT gets corrupted. The MFT also contains pointers to other specialised files, such as the boot file and the log file. This latter file contains details of transactions, so that errors can be recovered.

UNIX MOUNTING

Unix allows disk drives to be 'mounted'. This makes all the drives on a single computer appear as a single hierarchy. In the example: /usr/tmp could be on a USB drive, /lib might be one hard disk, and /kernel another.

NETWORK FILE SYSTEM (NFS)

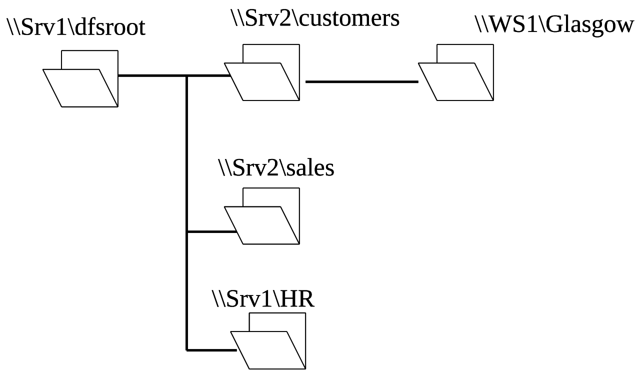
Sun Microsystems extended the idea of mounting to incorporate mounting of disks on remote computers. As Unix has always been intended for use on networked systems, this change was widely adopted. In the previous example, /bin might be a disk on a completely different computer. The user doesn't need to know where it is, they see a simple unified picture of the distributed storage system. NFS can be integrated with non-Unix systems. This is how the JKCC personal computers see files on Unix hosts.

WINDOWS DRIVE MAPPING

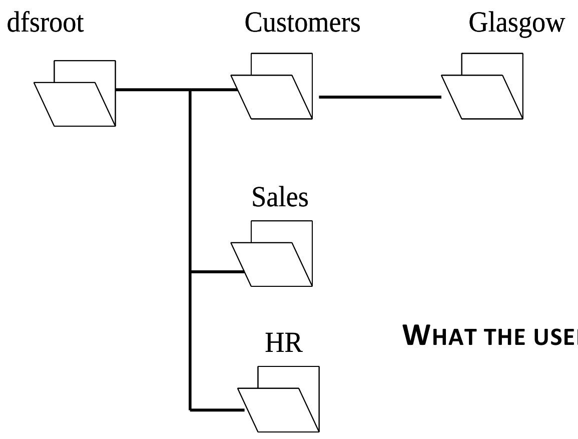
Windows provides an analogous feature: drive mapping. In this system, a network drive on a remote computer can be mapped as a virtual drive. For instance, a large network server may hold home folders that appear as the H: drive. This can be set up using My Computer or Windows Explorer…Tools...Map Network Drive

DISTRIBUTED FILE SYSTEMS

Many OS allow for a single hierarchical file system spanning multiple volumes and shares. (NFS on Unix, DFS on Windows). The user sees a coherent hierarchy, not just a collection of drive letters.



EXAMPLE DFS



WHAT THE USER SEES

DIRECTORY SERVICES

Server versions of Windows provide another feature: Active Directory, an example of a directory service. This provides a facility similar to NFS (plus some other features). Files, computers and printers across an enterprise (perhaps remotely connected via the Internet) appear as part of a single unified hierarchy. There is considerable administrative effort needed to set this up, but it makes things easier for the users. Novell's Zenworks handles multiple directory services across the University networks.

5 Memory Management & Virtual Memory

- Paging
- Swapping to disk
- Page Tables
- Protecting & sharing Memory
- Page Replacement

Introduction

The efficient use of memory in a computer involves a number of tasks.

- Resources, such as RAM, must be shared between a number of tasks. This implies protection, so that one task cannot interfere with data in memory belonging to another task.
- RAM can be made effectively larger using Virtual memory techniques.

5.1 Memory protection

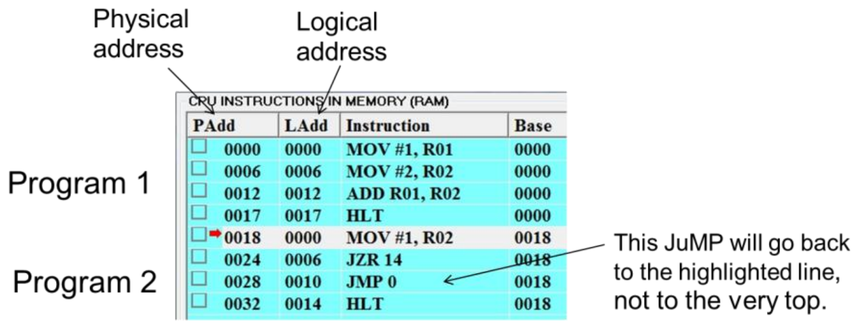
Each task must have some RAM to run in. Occasionally a task may accidentally or maliciously try and use RAM that belongs to another task or to the operating system. The operating system must protect the memory of each task.

Logical & physical addresses

When a task is compiled, it refers to a range of addresses (maybe it fits into the range 0-3,999). The operating system chooses where to load the software, eg starting at 10,000. The task would then access the physical RAM at locations 10,000 – 13,999. From the software's point of view, it is using logical addresses 0 – 3,999 and has no access to RAM belonging to other tasks.

Physical/ logical addresses on the CPU/OS simulator

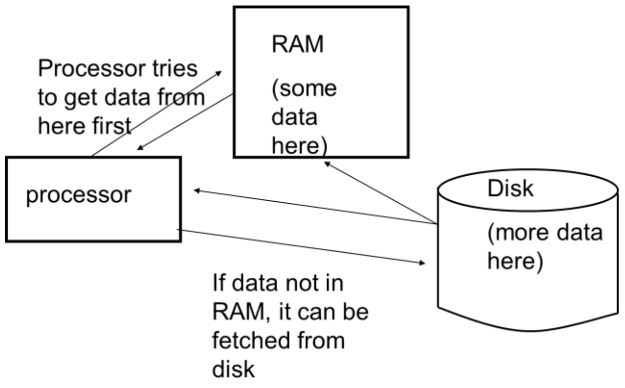
The second program is loaded with a base of 18. From the program's point of view, it looks like the instructions are at addresses 0, 6, 10… Actually they are at 18, 24, 28…


Physical address
Logical address
Program 1
Program 2
This JuMP will go back to the highlighted line, not to the very top.
CPU INSTRUCTIONS IN MEMORY (RAM)

PAdd	LAdd	Instruction	Base
0000	0000	MOV #1, R01	0000
0006	0006	MOV #2, R02	0000
0012	0012	ADD R01, R02	0000
0017	0017	HLT	0000
0018	0000	MOV #1, R02	0018
0024	0006	JZR 14	0018
0028	0010	JMP 0	0018
0032	0014	HLT	0018

5.2 Virtual Memory

Virtual memory is a technique for making relatively expensive RAM appear to be larger than it is by temporarily holding some information on disk, then transferring it to RAM when it is needed. The process of transferring data to and from disk is usually referred to as 'swapping'. Modern microprocessors may have facilities to support virtual memory, but it is the operating system that determines the strategy and implements the transfers between disk and RAM.


RAM
(some data here)
Processor tries to get data from here first
processor
Disk
(more data here)
If data not in RAM, it can be fetched from disk

DISK AS AN EXTENSION TO RAM

Main features of Virtual Memory

The size of "chunk" that is transferred between disk and RAM is important. The algorithm for choosing which chunk in RAM to get rid of to make room for a new chunk affects performance. All Virtual Memory systems will be slower than a purely RAM based system: they are generally most suitable for reasonably large systems, i.e. not embedded systems.

Questions: How much bigger than RAM is a hard disk (roughly)?

How much faster than a hard disk is RAM?

THE SIZE OF "CHUNK"

Some early systems swapped whole processes to and from disk. This gives poor performance due to the large size of processes, and is now not widely used. Some systems (OS-2, MULTICS) use segments. These are variable sized blocks with a logical purpose (data, code, stack). This can be quite difficult to implement due to the variable size of the chunks, but can give good protection of memory.

PAGES

The most common system by far is to use pages: equal sized chunks. Typical page sizes at present are about 4k-8k, corresponding to the block size transferred by disks. Using equal sized pages leads to many simplifications: any page will fit in any slot ("page frame"), no need to record the size of each page and so on. But it is harder to protect memory (e.g. 1 page may contain code and stack).

SWAPPING

When the processor needs to fetch a particular piece of information from memory (perhaps the next instruction, or some data), it produces the address of the information. At this stage, it is called a 'virtual address', as it does not yet correspond to the actual address of the information in memory. This address is translated, via tables, to the actual (physical) address of the information in RAM. There is also a bit that indicates whether the information is actually present in RAM. If not, it will be swapped in from disk

THE PRINCIPLE OF LOCALITY

One reason why paging works well is the 'Principle of Locality': after one memory access, the next is likely to be close by. This is because:

- Many accesses are to fetch code instructions. These tend to be stepped through in order, or often in small loops.
- Even data is usually accessed locally. For instance, if processing an image, the pixels are processed sequentially.

If a page has just been swapped into memory to satisfy a memory request, subsequent accesses are likely to be nearby, i.e. within the same page.

DEMAND PAGING

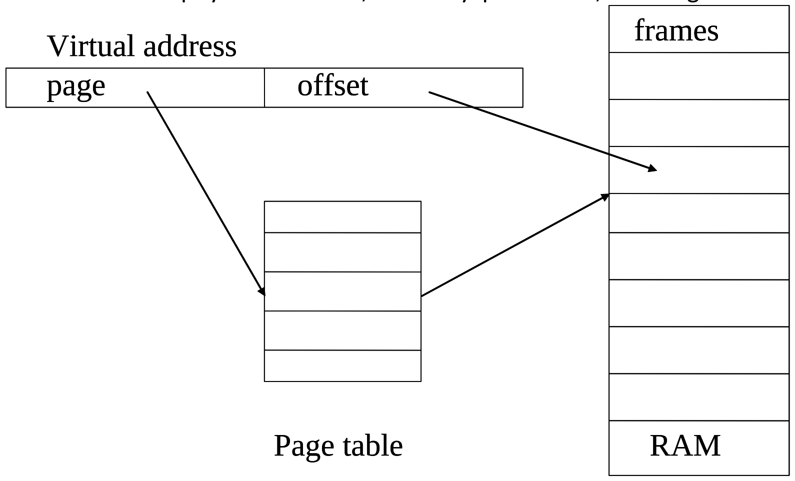
The commonest technique for fetching data from disk to RAM is to wait until the processor requests it: Demand Paging. Some Operating Systems try and predict ahead the pages that will be needed. As the bulk of the transfer doesn't need the processor's support it is usually possible to do this in the background. However, it does add to the traffic on the computer's buses, and so is not as widely used as the simpler Demand Paging.

SWAPPING FROM DISK

If the processor refers to information that is not present in RAM, it has to be fetched from disk; this is referred to as 'page fault'. A whole page, will be swapped in. This will take time, so a typical multi-tasking system will find another task to run until the data is ready in RAM. The actual transfer (which takes milliseconds, much slower than the nanoseconds of RAM) will be handled by the DMA system (Direct Memory Access). This transfers data from the disk drive to memory (or vice versa) independently from the processor. **Question: Why is DMA used?**

MICROPROCESSOR SUPPORT

It is normal for a high-end microprocessor to provide support for virtual memory. This may be on chip, or on an external memory management unit. This will include translation of a virtual address to a physical address, memory protection, and registers for locating tables. At some stage, the microprocessor will have to call the operating system, usually by an 'exception'.



THE OPERATING SYSTEM

The O.S. will have to build and maintain tables showing how virtual addresses are translated into physical addresses. When the microprocessor signals a page is not present, the O.S. will have to fetch it from disk. The O.S. will need to pick a page in RAM to evict to make room for the new page (replacement algorithm).

Question: Why 4K for a page, not 4 bytes or 4 Mbytes?

USING THE PAGE TABLES

The virtual address produced by the heart of the processor (say the address of the next instruction) goes to the Memory Management Unit. The high order bits are used to select an entry in the tables. This gives details of the page, including whether it is present in memory, and if it is, which frame it is in. Having got the start address of the frame, the low order bits are then used as an offset to find the exact location within the page.

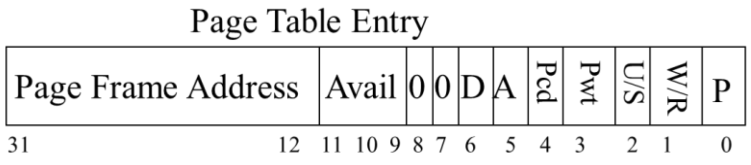
TABLES

The size of the page sets a lot of the details in the tables. For instance, the 32 bit Intel Pentium/Windows paging system uses 4 Kbyte pages. This means that the offset has to have 12 bits (2 raised to the power 12 gives 4096 locations). Many real systems use two (or more) levels of tables to make them more manageable. Remember that a PC with 8 Gbyte of memory using 4 Kb pages can have 2 M pages in memory, with more on disk...

TABLE ENTRIES

The following diagram shows the details of the information in a typical page system (the Pentium/Windows system). The details are not important, but you should note that there is a single bit (P, the Present bit) that shows whether data is in RAM. If it is, the address of the page frame is also in the table. There are other bits, such as the accessed bit, that show if the page has been used. These are used when deciding which page to replace.

P	0 = swapped to disk,	1 = Present in RAM
W/R	0 = read-only,	1 = writeable
U/S	0 = Supervisor	1 = User
A	0 = not yet accessed	1 = accessed
D	0 = not yet changed	1 = changed (Dirty)



TLB: A SHORTCUT

The page tables are set up by the operating system. Initially they are held in RAM. However, it would slow down memory operations if paging data had to be fetched from RAM every time before getting the actual data from RAM. The Translation Lookaside Buffer (TLB) holds some current page table entries in the processor. These can be accessed rapidly. This is an example of cache memory (see later).

PROTECTING & SHARING MEMORY

In a complex multi-tasking system, memory protection becomes important. Tasks may need to be protected from other tasks that have, accidentally or maliciously, gone wild. Also, the operating system has to be protected from user tasks. This affects the design of the operating system as a whole (remember the diagrams showing the split between user and kernel modes in the operating system).

PROTECTION & PAGES

The diagram showing the contents of the page table entries shows that there are some protection bits in these tables.

- Pages can be marked as being read-only or writable.
- There is a bit (U/S) that shows if the page is at User or Supervisor level.

Further protection may be provided by separating pages for different tasks. For instance, Windows keeps separate page directories & tables for each task, ensuring they can't interfere with each other.

MISSES (PAGE FAULTS)

Most of the previous discussion assumed that information was present in memory (a 'hit', as indicated by the Present bit). If data is not Present (a 'miss'), it has to be fetched from disk. One or more files are set aside on the disk to hold the pages that don't currently fit into RAM. In Windows, the file is called Pagefile.sys. Its size can be adjusted using **Control Panel..System..Performance..Virtual Memory.**

PAGE REPLACEMENT

If RAM is not completely full, a page that has been fetched from disk will easily slot into RAM. Normally, the RAM will already be full, and a page will have to be removed to make room. The algorithm for choosing which page to replace is important for the system performance. Ideally, you would want to get rid of a page that is never going to be needed again. Removing a page that was about to be used again would reduce performance considerably.

REPLACEMENT ALGORITHMS

Of course it is not possible to predict exactly what will happen. The best that can be done is to get rid of a page that is unlikely to be used again soon. There are a number of these algorithms (different ones are used by different operating systems). They are mostly based on taking note of whether a page has been used recently. The Principle of Locality (supported by a lot of measurements!) shows that pages that have recently been used are likely to be used again in the near future.

6 TUTORIALS

6.1 TUTORIAL 1: TYPES OF OPERATING SYSTEMS

1. Which of the following items are parts of the Operating System?

- Command line interface
- Word processor
- Network card
- Memory management & virtual memory system
- Device drivers
- Task scheduling system
- Disk drives

2. Linux is a free operating system, with similar capabilities to Windows. As Windows has to be paid for, even if it comes with a new PC, why doesn't everybody use Linux?

3. Briefly describe the function of two of the main components of an Operating System.

4. i) List one feature that a Multi-User Operating System would have that a Single-User Operating System would not have.

ii) List one feature that a Network Operating System would have that a Single-User Operating System would not have.

5. Microsoft would like everybody to upgrade to Windows 8 (more money for them). Can you suggest two reasons why it might be a good idea to upgrade, and two reasons why it might not?

6. The example diagrams in the notes illustrating the structures of Unix and Windows show that users can only get at the operating system services via a restricted range of calls. Why aren't users allowed to directly access the facilities provided by the operating system?

7. A large organisation has a lot of users on its computer network. Explain why it is important to be able to organise users into groups

8. What is Windows Server 2012, and why does it cost a lot more than Windows 7/8? (you can use a web browser to check typical costs, for instance by searching for Windows Server at www.amazon.co.uk).

9. Name three mobile device operating systems with a significant market share.

10. How are the operating systems of the iPhone, iPod Touch and Apple Mac related?

11. What other operating system is used as the basis for the Android operating System?

6.2 TUTORIAL 2: TASKS/PROCESSES & SCHEDULING

1. Describe how an operating system designed to let multiple users share a single computer would ensure that each user gets a fair-share of the processor's time to execute their code.
2. Briefly describe what is meant by task scheduling in a multi-tasking operating system, such as Windows or Linux.
3. Briefly explain what is meant by multitasking.
4. Why does the way in which tasks are scheduled affect the performance of a multitasking system?
5. In Windows, the user can set the initial priority of a task within a limited range. Administrators can give tasks a much higher priority, within the high-priority range normally reserved for real-time tasks. Why is this right not granted to everybody?
6. In Unix, the user has little control over a task's priority. The command 'nice' can be used to run a task at a low priority. This feature is not widely used, but why would it be used at all?
7. The diagram of task states in the notes showed that a task could move from 'running' to 'waiting'. Give an example of why this might happen, and what would cause the task to stop waiting.
8. How many tasks can be running at any one time?
9. What is a DLL?
10. In Windows, a user task starts with a fairly low base priority, but this priority can grow. What is the advantage of having this sort of system?
11. A selling point of many microprocessors intended for the embedded mark is the speed with which they can do a context switch. Why would this be important?

6.3 Tutorial 3: File systems

1. A Windows Long File Name will be automatically abbreviated to 8.3 format if necessary. What is the scheme for doing this?
2. Why do some systems require you to open a file before using it? Why should it be closed when it is finished with?
3. In NTFS, the file records contain space for a file name and a DOS name. Why is this?
4. The file system used by some versions of Windows is based on the File Allocation Table. Describe briefly one advantage and one disadvantage of such a system when compared with a more sophisticated system such as the Unix i-node system, or NTFS.
5. What information is held in a File Allocation Table? FAT is quite an old system; give two examples of where it is still widely used.
6. If files are allocated contiguously, how would you find a given block within the file?
7. Data on a hard disk drive is usually stored in fixed size blocks (typically a few Kbytes). Describe one way of keeping track of these blocks.

6.4 Tutorial 4: File commands

1. Describe the purpose of the chmod command in Unix/Linux.
2. A file on a Linux system has the permission 711. What does this mean?
3. The Unix/Linux command chmod can be used to set the permissions of a file. What number would follow chmod in order to give the user read, write & execute permission; the group read and write; and all others no access?
4. Explain how access rights to resources (such as files) can be controlled for a large number of users in a multi-user operating system, such as Windows 7 or Unix/Linux
5. For either Windows, or Linux, give an example of a command that will:
   I. Delete a file
   II. Copy a file.
   III. List the files in a directory.
   IV. Make a new directory.
   V. Change the Working Directory to the Parent Directory.
6. Which of the following are standard file permissions for NTFS files in Windows? Circle the number of any that are file permissions.
   I. Read.
   II. Modify Owner.
   III. Partial Control.
   IV. Read + Execute.
   V. Write.
   VI. Full Control.
   VII. Execute
7. Newer versions of Windows allow long file names to be used. Are these case sensitive?
8. The permissions for a Unix file can be given in rwx format (with a – representing no permission). These are always written user, then group, then others. What do the following represent?

   rwxrw-r--　　　　rwxr-----　　　　r-xr-xr-x

9. The Unix file permissions, are also quoted as numbers, with a 1 corresponding to permission, and 0 to no permission. The resultant binary number may then be written in octal. What do the following represent?

   110100000　　　　764　　　　644

10. You want to give a user access to run a program, but not to make changes to the program. What NTFS permission would you use?
11. What file system does the University network use?
12. A web page is held in directories on a Linux system. What permissions do you need to assign to the directories and files to make the pages visible to the outside world?

6.5 Tutorial 5: Virtual Memory & Memory Management

1. In a Virtual Memory system, explain how the virtual address is converted into the actual physical address.
2. Which of the following statements are true about paged virtual memory systems? (Circle any that are TRUE.)
    - The top part of an address is used to locate an entry in the page tables.
    - Pages are variable sized chunks that can be moved between disk and RAM.
    - The bottom part of the address is used to locate an entry in the page tables.
    - The TLB (Translation Lookaside Buffer) holds a copy of all the data in the current page.
    - The bottom part of an address is used to locate a specific address within a page.
    - There is a single bit (the Present bit) that shows whether a page is present in RAM.
    - In the Pentium processor family, there is one level of tables in the page table system.
3. Windows and Linux both swap pages to the hard disk. Briefly explain why this is done.
4. Paging relies on the 'Principle of Locality' for its effectiveness. Explain what this Principle is.
5. Briefly describe the role of the TLB (Translation Lookaside Buffer) in a virtual memory system.
6. With Virtual Memory, part of the disk is used as an extension to RAM. Explain why the performance of such a system can be improved by adding more RAM.
7. Windows tries always to have some free pages available in RAM. What is the advantage of this?
8. The Android OS for mobile phones is one of the most widely used.
    a. Is it a single or multi user operating system?
    b. Why does it not use virtual memory?
    c. Programs, mp3 files and so on can be stored using a file system. How would this be implemented in a phone?
    d. What processors does Android run on?

7 Labs

7.1 Lab 1: Multitasking

Introduction

The older Windows families (95, 88, ME) provided multitasking facilities, but in a relatively unsophisticated way. The more 'professional' line of Windows (NT, 2000, XP, 2003, Vista, 7, 8) use a different system that is a lot more effective and robust. In particular, all of the latter versions of windows have a utility, called Task manager, which allows a lot of control over tasks, and also provides a lot of information about the tasks that are currently running.

Task Vs Process

These two terms are normally used interchangeably. For instance, Task Manager manages tasks, but when you start it running, it refers to them as processes. In principle, each task/process is a separate piece of code that can run independently of others. When you start up a PC, the Operating system itself consists of a number of processes. If you start up applications, such as word processors and spreadsheets, that will add more process to the ones that are already running.

A complication is that a task/process may be split into smaller units, called threads. So, a word processor may have one thread that can handle printing, while another handles the text that is being typed in at the keyboard. Threads are not processes though. A process may be broken down into a number of threads that share common information: they don't exist in isolation from each other. This reduces the overheads involved in switching between threads.

Task Manager

You should start up **Task Manager** (right-click on the task bar to get it running, or go to the Ctrl-Alt-Del menu). This has a number of tabs. The first shows you all the applications that are running (i.e. software that is not part of the operating system). If you start up some applications such as the Calculator, Wordpad etc., you will see them here. You can select a task here and end it if has got stuck (crashed). If you go to the **Processes** tab, you will see something quite different. This shows you all the processes (tasks) that are currently running. If you have an application such as the Calculator running, you should see that as **Calc.exe**. However there are a lot more tasks running. These are mostly the tasks that make up the Operating System (i.e. they are Windows itself). There should be **Winlogon.exe**. This is the task that, among other things, checks to see if Ctrl-Alt-Del has been pressed. You can sort the display by any column simply by clicking at on the column heading, so you can sort them by name, for instance. As well as the operating system, there will be some other processes, such as virus checkers, that start automatically, and are not shown as applications. If you want to find out what a process does, Googling the name often gives further information. For instance, I wanted to know what ZenRem32.exe did, and Googling this took me to 'Bleeping Computer' (a good source for this sort of information). It said:

Filename:	**ZenRem32.exe**
Description:	Part of the Novell Windows client. It has a service name of Remote Management Agent and is found in the C:\Program Files\Novell\ZENworks\RemoteManagement\RMAgent\ folder.

Priorities

You can change what is displayed by going to **View..Select** columns. Select **Base Priority**, and you will see a column showing the priority of each task/process (this is a measure of how important they are). Most tasks will have normal priority, but some will have a high priority, for instance, task manager itself.

What priority has **Winlogon** got? (This is the process that checks for you pressing the Ctrl-Alt-Del combination).

Why does this process have this priority?

There is a process called System Idle process. This is the process that has to be there so that the operating system has something to run if there is nothing else to do.

What priority has **System Idle Process** got, and why?

If you sort the processes by priority, you can find out which tasks have high priority. There shouldn't be many. Have a look at them, and see if you can work out what they do.

THREADS

If Task manager does not already show the number of threads, select this by going to the View…Select Columns option. This will show you the number of threads associated with each process. Some processes may consist of a number of threads, some may be simpler. Start up **Calculator** (you should find it under accessories).

- What is the largest number of threads in a single process?
- How many threads does calculator consist of?
- What happens if you start up a second copy of **Calculator**?
- How does the operating system keep track of the different instances of **Calculator**?

If you have not used this facility already, you can try killing a process in Task Manager by selecting it, then going to **End Task** (if you are in the **Applications** tab), or **End Process** if you are in the **processes** tab. This another example of the confusing use of two terms, task and process, for the same thing.

- Can Task manager kill itself?

The next tab, **Performance** shows you graphically how much of the memory is being used, including the virtual memory. This will be described in a lecture soon, and I don't want to go into too much detail just now, but have a look at the performance tab and see what happens when you start up, or close, a large application such as Internet Explorer.

PROCESS EXPLORER

This is a tool that is provided by (a subsidiary of) Microsoft; it does all that Task manager does, plus more. You can download it from the Microsoft site if you want to try it at home. Copy the folder to the desktop of your machine if it has not already been copied (your tutor will explain where from). Double click on the procexp.exe icon to start the program. The display shows the system processes at the top with the user processes below. Putting the cursor over a process will provide more information on it: have a go and see what some of the processes do.

You will see that processes are colour-coded: services (often parts of the operating system) are pink, and user processes are pale blue. A new object is shown briefly in green, and a deleted object is shown in red. You will only see these if you start or stop a new process (such as calculator). If you go to **View...Select Columns**, you can select the columns that are displayed, just as you can with Task Manager. However, Process Explorer can show much more. If you go to the Process Performance tab, you can select Base Priority and Context Switches. A context switch is the name for switching from one task to another.

- What are some typical values of the Base Priority?
- What are typical values for the number of context switches?
- Why would interrupts lead to context switches?

A few other columns to investigate are:

'Process Performance: CPU History'	Shows mini graphs of processor usage for each task
'Process I/O: Delta Reads'	Shows recent I/O reads (wiggle the mouse rapidly to create some I/O events)
'Process Network: Delta Receive Bytes'	To see how many bytes are received from the network every second

In old systems code was linked together with library files at compile time, then the whole bundle was loaded when the code was run. Now, a more flexible system is used, where the bits of code needed from libraries are loaded dynamically as they are needed.

The library code is stored in Dynamic Link Libraries (DLLs). You can see what DLLs are currently in use. If the display window does not already show a lower pane, go to **View** in the main menu and click on **Show Lower Pane**. Now, in the View menu, go to **Lower Pane View**, and select DLLs. Now, when you have one of the user processes highlighted, you will be able to see the DLLs it uses. One thing you should see is KERNEL32.DLL. This is the base of the Windows kernel. The heart of the operating system (whether you are running 2000, 2003, XP) has not changed significantly since it was written for NT in the mid-nineties. If you click on one of the small graphs at the top, you will get a moving graph showing things like CPU usage similar to the display you get with Task Manager, except you can see what causes the peaks by hovering the cursor over them. Also, if you are using a PC with dual (or quad) cores, you will get a separate graph for each core CPU.

EXAMINING ONE TASK IN DETAIL

Process Explorer can provide a lot of information about an individual task. With Process Explorer running, start up a web browser (if you have not already done so). Find the browser in the list of processes, right click on it and select 'properties'. You should get a pop-up window that lets you investigate what this task is doing. Selecting 'Performance Graph' will give you a graphical display of how much processor time and memory (private bytes) this process is using. You can also select a tab to see how much of the Graphics Processor Unit (GPU) time this task uses.

Select the TCP/IP tab. Now go to an external web site and watch what happens in the TCP/IP window. Can you explain what the difference is between having the 'Resolve addresses' box ticked or not?

Investigate the other tabs related to this task. You should be able to see details of all the threads that are running, the disk and I/O and the many security settings that control which groups can do what.

PERMISSIONS

The last part of the lab looks briefly at the permissions that determine what you are allowed to do with folders and files. The permissions are actually the NTFS (NT File System) permissions, but they are referred to as 'Security'. Start My computer or Explorer and right-click on a folder on your PC. Go to 'Properties' and select the 'Security' tab. This should show you a list of who has access to this folder, and what rights they have. What you actually see will depend on the folder you select. Make a note of the 6 folder permissions:

1.

2.

3.

4.

5.

6.

To make it easy to assign permissions in an organization, windows allows you to set up and use groups, so you may see that some individuals and groups have access. Now try doing the same thing with a file. Are the permissions the same as with a folder? If not, what permissions are different?

Now try creating a file or folder (it doesn't matter what is in it). Can you change the permissions on the new file/folder? Can you give permissions to other users?

7.2 Lab 2: Process States

Objectives

This lab introduces the Operating Systems simulator. It demonstrates the basic states used in the operating system of the simulator, and shows how processes move between different states. It also introduces the Process Control Block (PCB).

Programs used

ALWAYSBUSY.sas; WAITFORINPUT.sas

Introduction

Processes in the CPU-OS simulator can have one of three states: Running, Ready and Waiting. Some example programs will help to make it clear how the operating system moves processes from one state to another. The example programs have been written in the simulator's high level language, but the details of the programs are not very important. So, for simplicity, we will load the assembly language versions. This is just to save the compilation step. I will include the programs in this sheet so you can see what the program does.

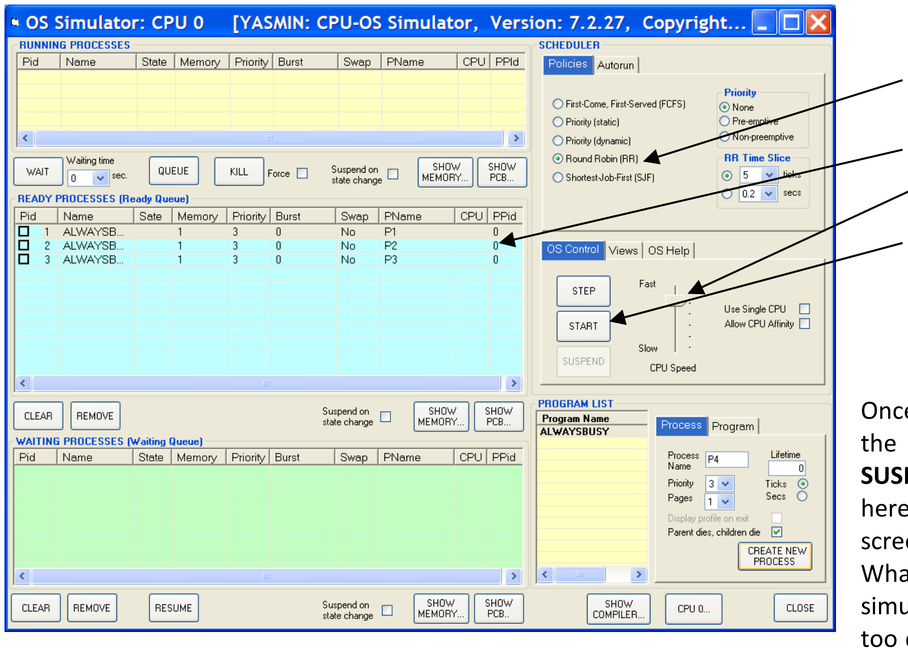
Moving between states

The first program is very simple. It just goes round an infinite loop, keeping the processor busy.

```
program AlwaysBusy
      while true  %Do a forever loop
      wend              %that keeps the CPU busy
end
```

Start up the CPU simulator, and load the assembly language version of the program: ALWAYSBUSY.sas

Click on the **OS O…** button to get the Operating System simulator running.   Now we need to get some processes set up. Click three times on the **CREATE NEW PROCESS** button. That will create three identical processes with the same code in each one. To begin with, you should see them in the queue of ready processes. Each will have a different Process ID (PID) that the operating system uses to keep track of it. Before the processes are run, you need to change the Scheduler policy to Round Robin (if you don't, the default FCFS will run one process forever).



READY PROCESSES (Ready Queue)

Pid	Name	Sate	Memory	Priority	Burst	Swap	PName	CPU	PPid
1	ALWAYSB...		1	3	0	No	P1		0
2	ALWAYSB...		1	3	0	No	P2		0
3	ALWAYSB...		1	3	0	No	P3		0

WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid

Scheduler policy: First-Come, First-Served (FCFS) / Priority (static) / Priority (dynamic) / [✓ Round Robin (RR)] / Shortest-Job-First (SJF)

Priority: [✓ None] / Pre-emptive / Non-preemptive

RR Time Slice: [✓ 5 ticks] / 0.2 secs

Program Name: ALWAYSBUSY

Process Name: P4   Priority: 3   Pages: 1   Lifetime: 0 [✓ Ticks] / Secs
Display profile on exit
Parent dies, children die ✓

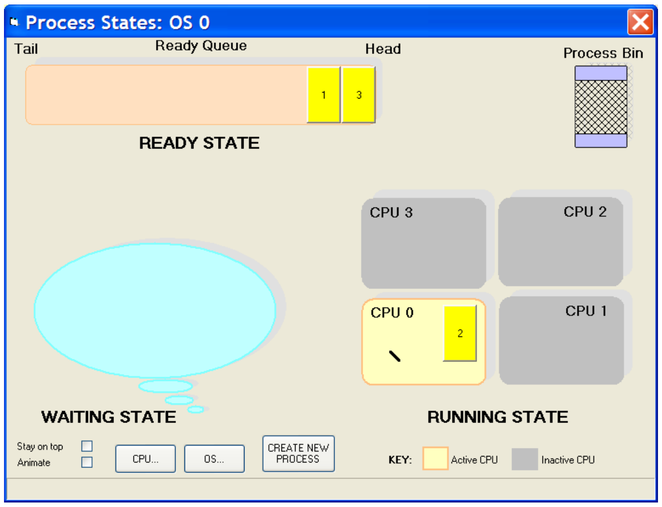
Round Robin scheduling has been selected.

There are three processes ready to run.

The speed has been set to about 40.

The START button will start the simulation.

Once you have round robin selected, you can start the OS simulation with the **START** button (the **SUSPEND** button pauses the simulation). Be careful here: this is the start button on the OS simulator screen, not the run button on the CPU simulator. What you see depends on the CPU speed within the simulation. At the fastest setting changes are a bit too quick to follow, but at a speed of about 30 or 40 you should be able to see the three processes being scheduled one after the other. With three processes and a single CPU, there are obviously going to be several on the ready queue at any time. To explore a feature that will be needed later, click on the **Suspend on state change** button below the **RUNNING PROCESSES** window.   This will cause the processing to stop whenever a process starts to run. This can be handy to see in detail things like the process ID of the currently running process. Having halted the simulation, you will need to click on the resume button to restart the simulator.



There is a graphical representation of what is going on. It doesn't add much to what can be seen on the OS Simulator window, but it might be helpful to look at it. In the **Views** tab (middle right in the OS Simulator), you will find a button: **VIEW PROCESS STATES**. This shows the states as the animation runs (after getting the view, you need to select the OS Control tab to get back to the **START** and other controls). Try running this, and experiment with the CPU Speed to get an animation you can see. In the diagram, Process 2 is Running, 1 & 3 are Ready to run

TIDYING UP

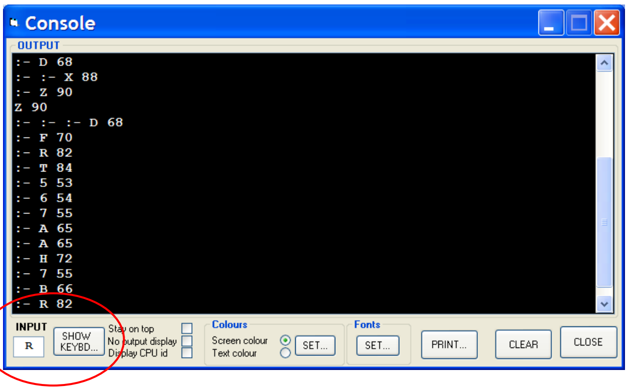
Before the next program can be loaded, there is a bit of clearing out to do. The program memory on the CPU must be cleared out, but as you may have noticed, many of the buttons on the CPU simulator are greyed out once the OS simulator is running. As far as I can tell, all the processes have to be killed off on the OS simulator before control is handed back to the CPU simulator. Do this by **REMOVE**ing any processes in the **READY PROCESSES** or **WAITING PROCESSES** queues, then killing the running process. The tricky bit here is that the **REMOVE** only works when the OS simulator is suspended, the **KILL** only works when the simulator is running. Having done that, the **REMOVE ALL PROGRAMS** button on the CPU simulator becomes enabled so that you can remove the old program.

WAITING QUEUES

Having cleared the old code from the CPU simulator, load the assembly language program: WAITFORINPUT.sas This is the assembly language version of the following program:

```
program WaitForInput
var KeyVal integer

while true
      write(":- ") %Display this prompt
      read(KeyVal) %Wait for keyboard input
      writeln(KeyVal) %Display input character
      if KeyVal = 90 then %Test if uppercase letter Z
            break * %Break from program if a Z character
      end if
      for n = 1 to 20 %processing to keep CPU busy
      next
wend
end
```

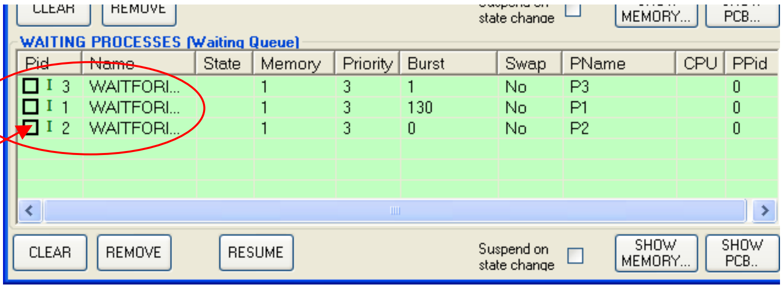


```
:- D 68
:- :- X 88
:- Z 90
Z 90
:- :- :- D 68
:- F 70
:- R 82
:- T 84
:- 5 53
:- 6 54
:- 7 55
:- A 65
:- A 65
:- H 72
:- 7 55
:- B 66
:- R 82
```

Run it directly from the CPU simulator to see what it does. You will need to get the console window displayed using the INPUT OUTPUT button on the CPU simulator window. The program spends most of the time waiting for a key to be pressed. When the key is pressed, (use the INPUT box bottom left on the console, or the SHOW KEYBD…) the read statement echoes the key that was pressed and the ASCII value representing that key is stored in a variable. The variable is then printed out, showing in decimal the ASCII equivalent of the key. If the key that was pressed was a capital Z, then the program terminates. Having checked this out for yourself, go back to the OS simulator and click the CREATE NEW PROCESS button to create one process based on the program. When you run this you should see that the process runs for a bit then sits in the waiting queue, waiting for you to press a key. When you do press a key, it goes back to the running state (actually it goes through ready to running, but if you have the speed turned up, this may be difficult to spot).

Process Control Blocks

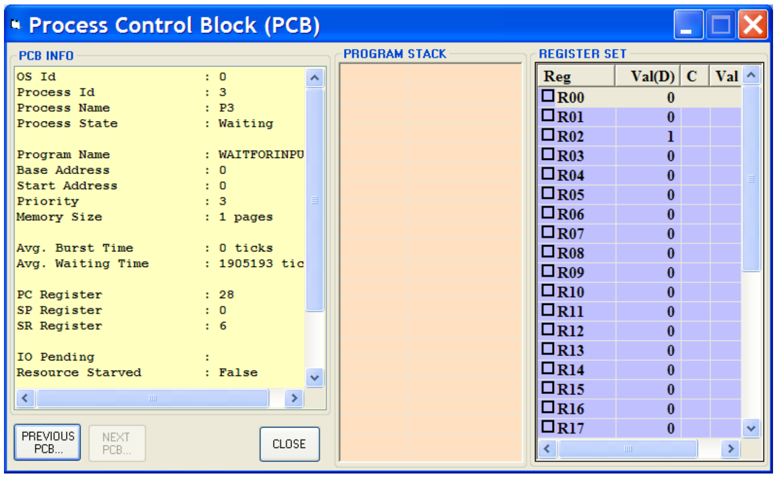
Assuming you are still at the OS simulator with one process based on the WAITFORINPUT code, use the CREATE NEW PROCESS button to create two more processes. Make sure the speed is set near to the fast end of the scale (about 8 works well) and start the processes running. You should find that the processes fairly quickly end up in the waiting state if no keys are pressed. Pressing a key (either on the virtual keyboard, or by entering a character to the INPUT box on the console window) will cause one process to move from the waiting queue, via the ready queue to the running state. Pressing several keys will unblock several processes; due to the speed of the simulation, key presses may not be processed for some time.



WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
I 3	WAITFORI...		1	3	1	No	P3		0
I 1	WAITFORI...		1	3	130	No	P1		0
I 2	WAITFORI...		1	3	0	No	P2		0

If you start from a stage where three processes are waiting and enter two key presses, you should end up with one process running, one ready to run and one waiting. SUSPEND the simulator at this point. Now go to the waiting process section at the bottom of the window and click on SHOW PCB.. to display the Process Control Block. You should see something like this->



```
OS Id               : 0
Process Id          : 3
Process Name        : P3
Process State       : Waiting

Program Name        : WAITFORINPU
Base Address        : 0
Start Address       : 0
Priority            : 3
Memory Size         : 1 pages

Avg. Burst Time     : 0 ticks
Avg. Waiting Time   : 1905193 tic

PC Register         : 28
SP Register         : 0
SR Register         : 6

IO Pending          :
Resource Starved    : False
```

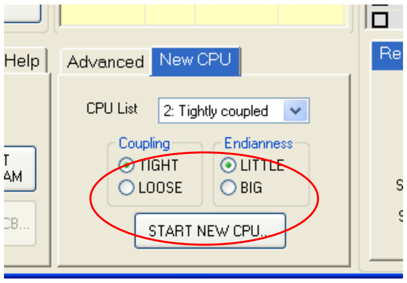
Reg	Val(D)	C	Val
R00	0		
R01	0		
R02	1		
R03	0		
R04	0		
R05	0		
R06	0		
R07	0		
R08	0		
R09	0		
R10	0		
R11	0		
R12	0		
R13	0		
R14	0		
R15	0		
R16	0		
R17	0		

The left section shows the process ID and the state. You will see that PC (the Program Counter) has been saved so that execution can resume when the process is next scheduled. The rest of the window shows the context that has been saved, including the stack and registers.

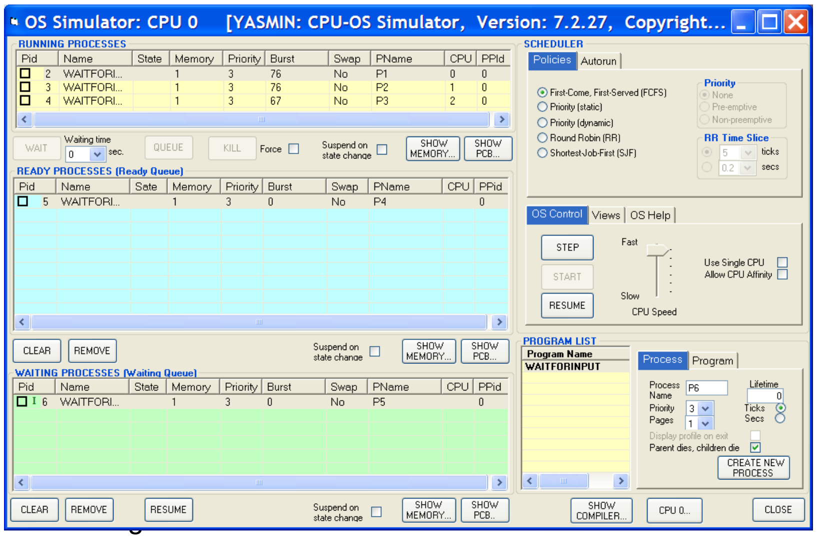
If you look at the PCBs for the Ready task, you will see that the contents are very similar, except the state is Ready rather than Waiting. However, if you look at the PCB of the running task, you will notice a distinct difference: as the task is Running, there is no need to preserve the context. So there is no copy of the registers, the stack or the PC as they are all currently in use.

Multiple Processors

You can start multiple CPUs running. This reflects the normal situation in a desktop PC nowadays where there will be multiple cores, each running a separate process. The START NEW CPU.. button is at the bottom of the CPU simulator window, however it will be greyed out if there are any processes running. You need to Kill any running processes (entering capital Z is one way of terminating the process as that is the character that the WAITFORINPUT program uses to break.) Here I have 3 CPUs, each running a process. There is also a process Ready to Run, and one waiting. The 2 screens show alternate views of this. Try it out for yourself with several CPUs.



CPU List: 2: Tightly coupled
Coupling: [✓ TIGHT] LOOSE
Endianness: [✓ LITTLE] BIG



RUNNING PROCESSES

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
2	WAITFORI...		1	3	76	No	P1	0	0
3	WAITFORI...		1	3	76	No	P2	1	0
4	WAITFORI...		1	3	67	No	P3	2	0

Waiting time 0 sec. Suspend on state change ☐

READY PROCESSES (Ready Queue)

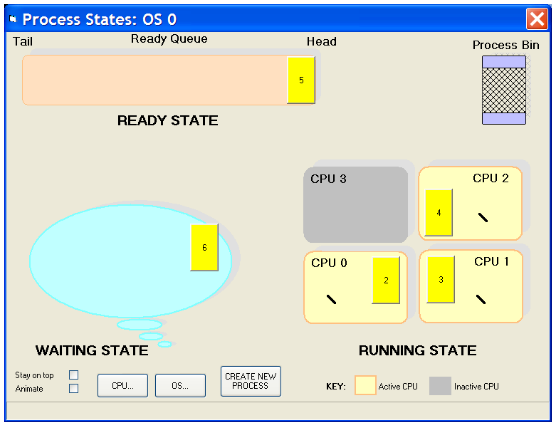
Pid	Name	Sate	Memory	Priority	Burst	Swap	PName	CPU	PPid
5	WAITFORI...		1	3	0	No	P4		0

WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
I 6	WAITFORI...		1	3	0	No	P5		0

[✓ First-Come, First-Served (FCFS)] Priority (static) Priority (dynamic) Round Robin (RR) Shortest-Job-First (SJF)
Priority: [✓ None] Pre-emptive Non-preemptive
RR Time Slice: [✓ 5 ticks] 0.2 secs
Use Single CPU ☐ Allow CPU Affinity ☐

Program Name: WAITFORINPUT
Process Name P6, Priority 3, Pages 1, Lifetime 0, [✓ Ticks] Secs
Display profile on exit ☐
Parent dies, children die ✓



7.3 Reference: Unix/Linux & Windows commands

Introduction

This document is intended to be a quick reference guide summarising some commands (particularly associated with files) in the Linux and Windows systems. In the lab time, we may only have time to look at Linux commands. You can investigate the Windows versions by going to **Run..** in the Start menu, and typing CMD to get a command window. In recent versions of Windows, there is a more powerful system: the **Windows PowerShell** that is usually found under **Accessories**.

Permissions

Unix

Unix uses nine bits: r,w & x for the Owner, the Group and all Others ('Everyone').

Read Permission: For a file, this allows you see the contents of the file. For a directory, this allows you to list the directory contents.

Write Permission: For a file, this allows you to change the contents of the file. For a directory, this allows you to change the contents of the directory (add or delete files).

Execute Permission: For a file, this allows you to run an executable file (or a script containing commands). For a directory, this allows you to change to this directory and make it your working directory.

Windows

File permissions can only be used if the file system is NTFS. This means that 95, 98 & ME can't use permissions. If the file system is NTFS, then there are similar (but not identical) permissions to Unix. They include Read, Write, Execute and Delete. As one of the groups is the built in group 'Everyone', there is the same possibility as with Unix that different permissions could be applied to the Owner, Group and Everyone. People may acquire rights from being in different groups.

Setting Permissions

With Windows & NTFS, you can use Explorer or My computer, then right-click on a file or folder, then go to the **Properties…Security** to change the permissions.

With Unix, you use chmod. The command works in several ways (use **man** for more details). You can use the **rwx** notation with **ugo** for user, group and others:

chmod u=rwx document1	(Set the user permision to read and write and execute)
chmod go=-- document1	(set the group and others permission to read, but not write or execute)

Help

Unix help uses the **man** command (manual). For instance, to get help on the **ls** command, type **man ls**.

The help for Windows command-line commands is not always so obvious. For instance, some versions will support **help,** e.g. type **help dir** to get help on the dir command. The method that should work with all versions of Windows is to type the name of the command you want help on, followed by **/?**

For example, type **chdir /?** to get Help on the **chdir** command.

Note: To display Help one screen at a time, type the command followed by **| more**.

For example, type **dir /? | more** for Help on the **dir** command. This should work in Unix or Windows.

Moving around

Both Unix and Windows Command use cd (short for change directory). To change up a level to the parent directory, both use

cd.. (**..** is the short hand for the parent directory. Unix also uses **~** for your home directory). To go to a specific directory:

cd /usr/tmp	(Unix, note this is case sensitive)

cd \windows\profiles (Windows, not case sensitive)

Unix uses a single file hierarchy containing all the drives, so you don't need to specify the drive, only the directory. In Windows (unless using Dfs/Active Directory) you have to specify the drive if you want to move to a different one. E.g., to go to a directory on H:

cd h:\home\spreadsheets

Getting Information

If you want to know where you are (the Present Working Directory), type **cd** in Windows, and **pwd** in Unix. Depending on the set up, the same information may be displayed in the command prompt.

To display the contents of a directory use **ls** (Unix) or **dir** (Windows). Both have switches (i.e. options):

ls -l	(Unix: view the current directory and show the permissions)
ls -la /etc	(Unix: view the /etc directory, showing permissions, and all files)
dir a:	(Windows: show the contents of the floppy disk)
dir /os	(Windows: show the contents or the current directory ordered by size)

Creating and deleting directories

This is similar in Windows and Unix. Both use **mkdir** to make a new directory, and **rmdir** to remove a directory. Windows allows these to be abbreviated to **md** and **rd** respectively, Unix needs them in full:

mkdir newdir	(make a sub-directory called newdir in the current directory)
rd c:\temp\olddata	(remove the Windows 'olddata' sub-directory from c:\temp)

Manipulating files

Copying a file uses slightly different commands in Unix and Windows: **cp** and **copy** respectively. In both cases you have to specify the source file to be copied and the destination. If you want to give the copy a different name, you can specify this. Otherwise the copy will have the same name:

cp ~/.profile ~/profBak	(copies the .profile file to profBak, both in the home folder).
copy \windows\office\temp.dat \windows	(copies temp.dat without changing the name).

Moving a file can be done with the move commands (**mv** in Unix, **move** in Windows):

move \windows\temp.dat .. (move the temp.dat file up one level)

You can change the name of a file by moving it (in the same directory) to a file with a different name:

mv profBak profOld (change the name of the file from profBak to profOld)

Note that Windows allows you to do the same thing using the rename command:

rename profBak profOld (the **rename** command can be abbreviated to **ren**)

To delete a file in Unix, use **rm** (remove). In Windows use **del** or **erase**

rm surplus.file del report.doc