EXAM PAPERS PRACTICE

Boost your performance and confidence with these topic-based exam questions

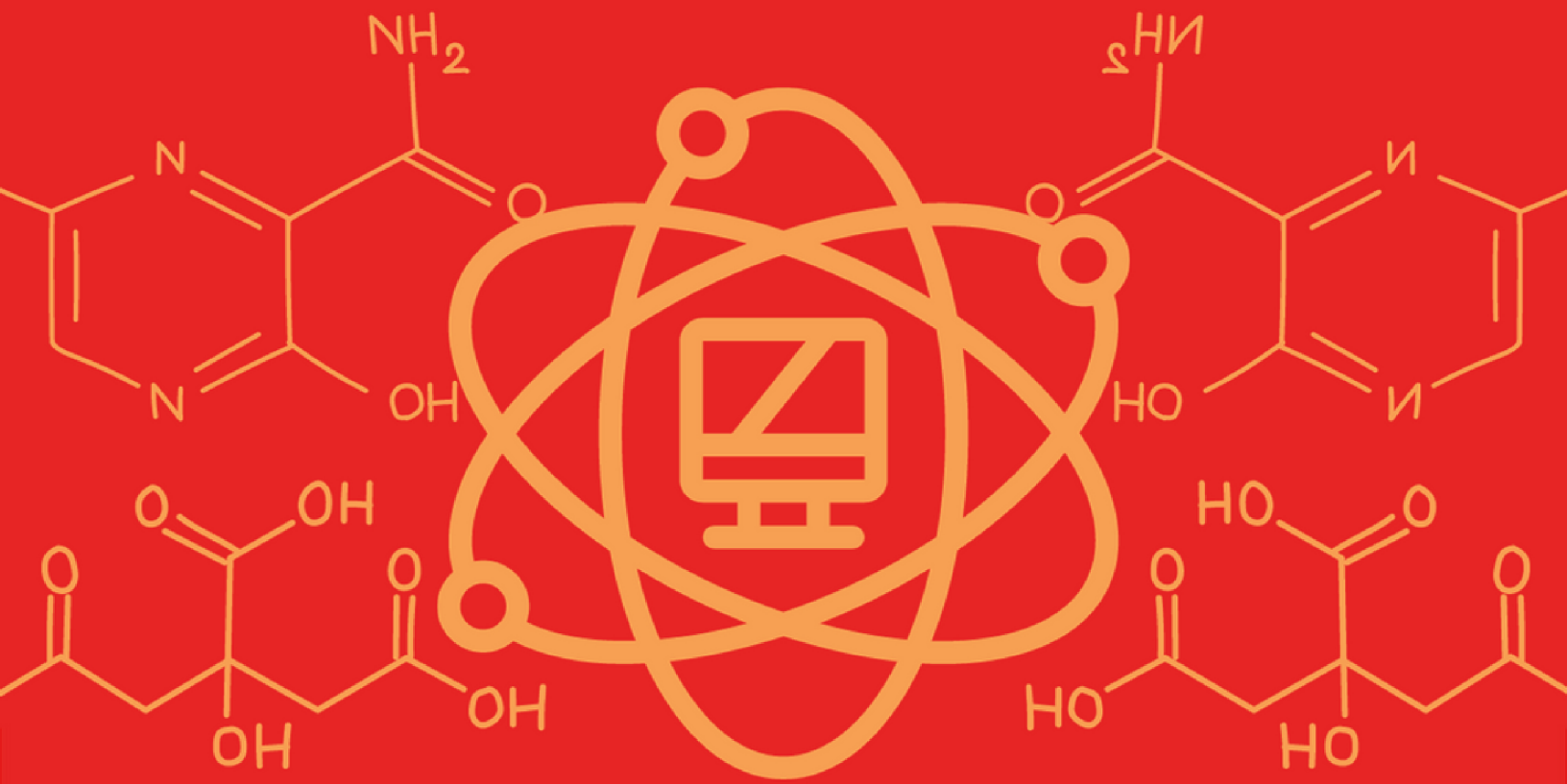
Practice questions created by actual examiners and assessment experts

Detailed mark scheme

Suitable for all boards

Designed to test your ability and thoroughly prepare you

Operating System



CIE AS & A Level

Computer Science Revision Notes 9618

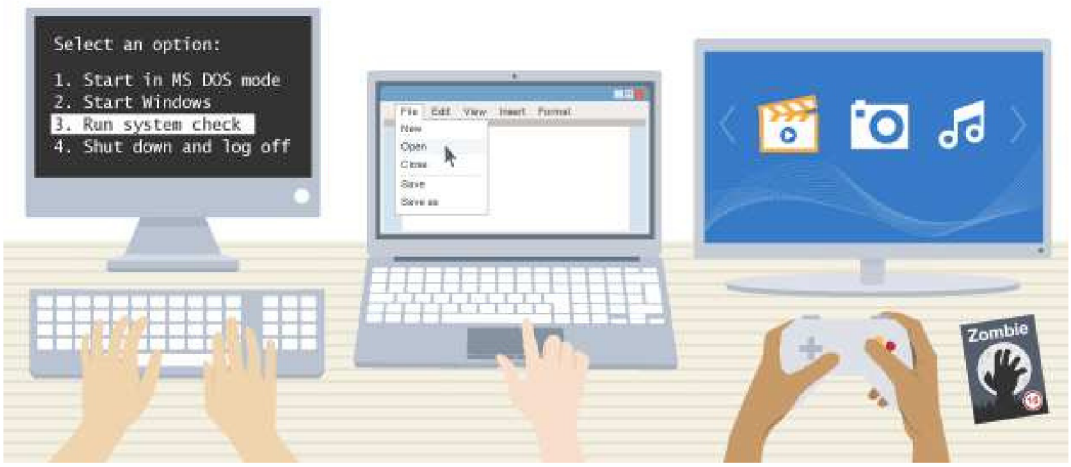
Syllabus Content:

5.1 Operating system

describe why a computer system requires an operating system
explain the key management tasks carried out by the operating system including
Notes and guidance
o Memory management
o File management
o Hardware management (input / output / peripherals)
o Processor management
show an understanding of the need for typical utility software used by Operating system:
Notes and guidance
o disk formatter
o virus checker
o defragmenter software
o disk contents analysis/disk repair software
o file compression
o backup software
Show understanding of program libraries:
Notes and guidance
o Software under development uses existing codes of program libraries
o Benefit to developer of software using library files, including Dynamic Link Library (DLL) files.

Operating systems

An operating system (or 'OS') controls the general operation of a computer, and provides an easy way for us to interact with computers and run applications.



On some computers it is possible to run a choice of operating systems. Games consoles have their own unique operating systems.
There are a few common operating systems available:
Mac OS X
Linux
Windows
Android (based on Linux)
iOS

Functions of the operating system

The operating system performs several key functions:

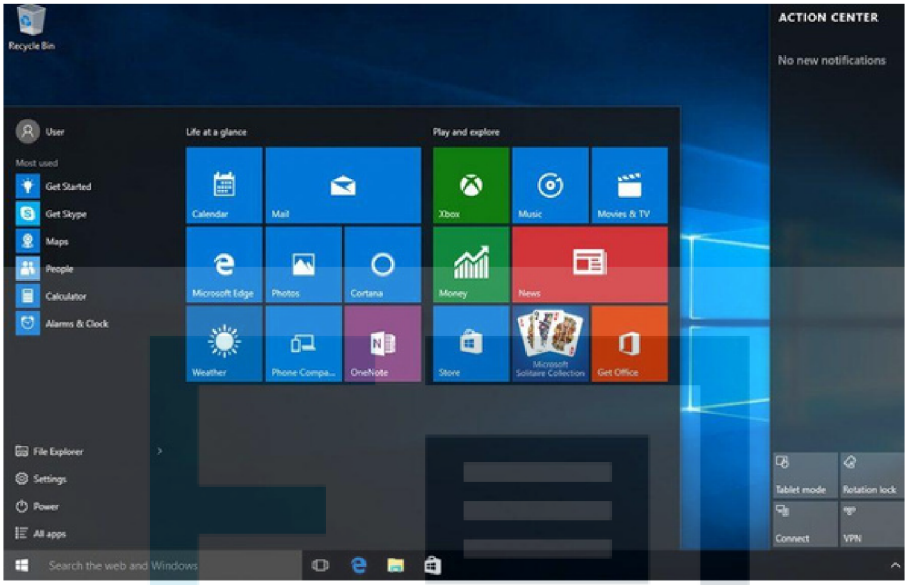
interface - provides a user interface so it is easy to interact with the computer
manages the CPU - runs applications and executes and cancels processes
multi-tasks - allows multiple applications to run at the same time
manages memory - transfers programs into and out of memory, allocates free space between programs, and keeps track of memory usage
manages peripherals - opens, closes and writes to peripheral devices such as storage attached to the computer
organises - creates a file system to organise files and directories
security - provides security through user accounts and passwords
utilities - provides tools for managing and organising hardware

User interface

The OS provides a user interface (UI), an environment for the user to interact with the machine. The UI is **either graphical user interface GUI** or text-based interface **(command line interface (CLI))**

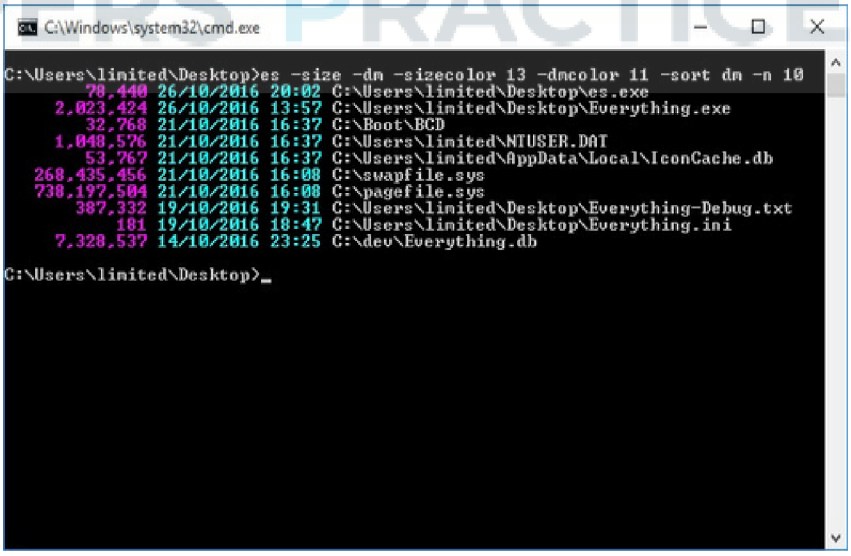
Graphical user interface (GUI)

The OS on most computers and smartphones provides an environment with tiles, icons and/or menus. This type of interface is called the graphical user interface (GUI) because the user interacts with images through a mouse, keyboard or touchscreen.



Command line interface (CLI)

An OS also provides a method of interaction that is non-graphical, called the command line interface (CLI). This is a text-only service with feedback from the OS appearing in text. Using a CLI requires knowledge of the commands available on a particular machine.



Advantages of using the command line include:

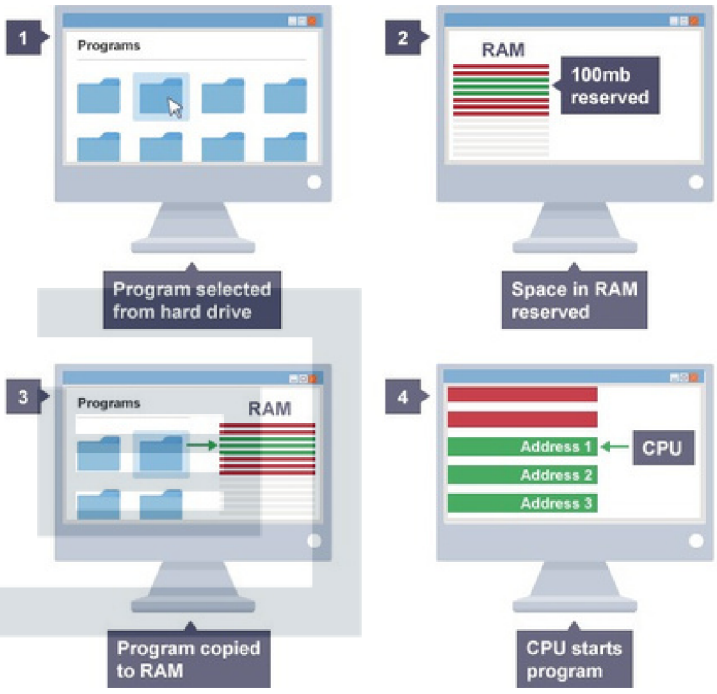
- a **faster** way to get tasks done it
- is **more flexible** than a GUI it uses **less** memory

Some games, such as Minecraft, also make use of a command line tool which allows the user to bypass the main interface and alter the game's mechanics or environment.

Managing the CPU

The OS is used to run programs by clicking on an icon, selecting the program from a menu, or typing in an instruction at the command line.
When the OS runs a piece of software it has to find the program files on the storage drive, load them into main memory, and instruct the CPU to start executing the program from the beginning.
In each case, the OS performs the same sequence of steps:

1. the program code is found on the storage drive
2. a section of RAM is reserved for the program and space is allocated for the program's data
3. the program code is copied from storage into the reserved space in the memory
4. the CPU program counter is set to the memory location of the first instruction in the program, and execution begins



Multitasking

The OS makes it possible to run several programs at once. Several programs can be stored in RAM at the same time, however only one program at a time is processed by the CPU. Programs can be in one of three states:

running
waiting
runnable

Only one process can be running at any one time. CPUs are extremely fast, so if a program is processed for even a short time it can do quite a lot. The OS decides the best way to swap between running, runnable and waiting processes. It controls which process is being executed by the CPU at any point in time, and shares access to the CPU between processes. The job of working out when to swap processes is known as **scheduling**.

Swapping happens so fast that it appears that all processes are running at the same time. When there are too many processes, or some of them are making the CPU work especially hard, it can look as though some or all of them have stopped.

Managing memory

The OS manages how main memory is used. It decides:
how memory is shared between processes
what happens when there is not enough main memory to get the job done

Different processes running at the same time must not interfere with one another. This means they have to use different parts of the computer's memory.
The OS handles the transfer of data between processes. This is done by setting aside areas in memory where data values can be shared.

The OS uses buffering to set aside memory for the temporary storage of data. A process may output data and leave it in the buffer. This means that processes can each get on with their jobs at their own rate. It is only when the buffer is full that processes will need to wait.
Buffers are also used when you stream online content.

Peripheral devices

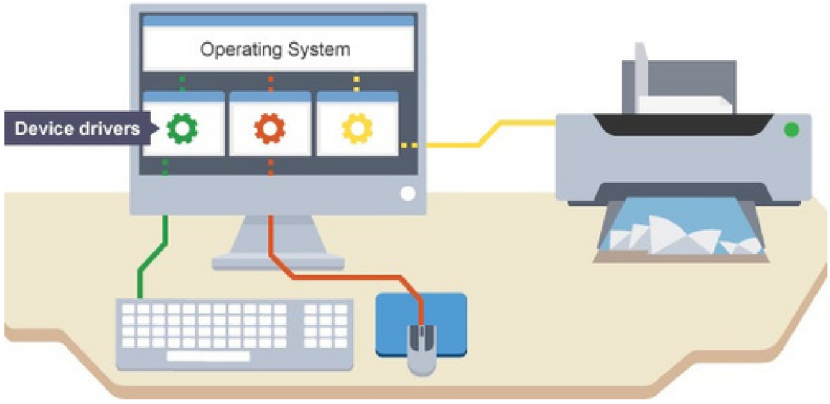
Protocols

Each peripheral is programmed with its own machine code. Each has its own rules that dictate how it transmits data values between the computer and the device. These rules make up a protocol for controlling and communicating with the device.
The protocol dictates the structure, speed and timing of the messages that are sent and received.

Drivers

The OS uses programs called **device drivers** to manage connections with peripherals. A device driver:

handles the translation of requests between a device and the computer
defines where a process must put outgoing data before it can be sent, and where incoming messages will be stored when they are received
wakes up the device when it is needed and put it back to sleep when it is not



An OS will have generic device drivers to enable it to connect to most common peripherals. Some peripherals, however, will have their own drivers that need to be installed before use.

Peripherals that use the same protocol may be controlled by the same driver. If a number of identical game controllers are plugged in, each device will store its data in a different place so they do not interfere with each other.

File systems

The OS organises files on the storage drive.
In order to retrieve data from a file, the computer needs to know:

- which storage device it is held on
- where it is stored on the device
- how files are organised on the device
- how much data is in it
- the protocol needed to communicate with it

It is the job of the OS to maintain this information for other programs, and it does this by providing a file system. The purpose of a file system is to provide programs with a uniform way of storing and retrieving data.

File management

The OS manages how data is organised into files. This makes it easier for the user to see files using programs like the WindowsFile Explorer or Mac OS X Finder. The OS organises where and how files are stored, deleted, read, found and repaired. It detects errors such as missing disks or incorrect file names, and informs the user that errors have occurred.
Each file has a unique name and the OS maintains a set of **look-up**tables that relate file names to locations on storage drives.

Hierarchies

File systems work in a similar way to the way that libraries organise books. Folders and directories correspond to different sections of the library. Inside each folder can be other folders (sub-sections within a subject) and files (the books themselves). If you need to access a specific file you just need to know how to look for it in the index which describes where each file is located.

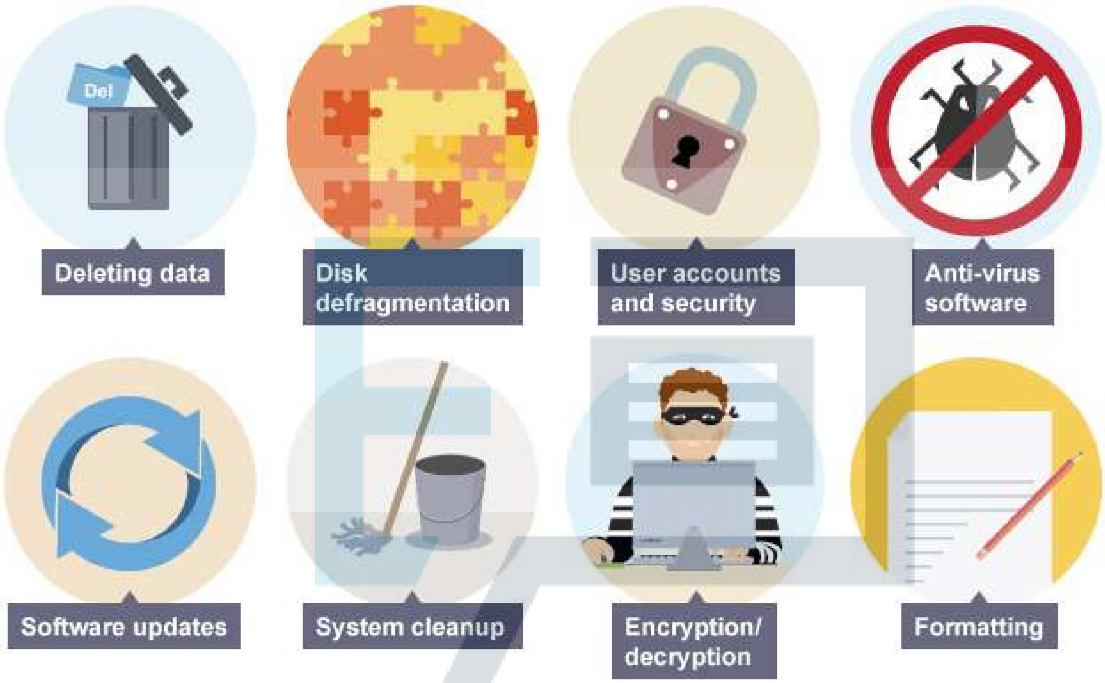
Most file systems are hierarchical and contain directories that contain lists of other files.

Hierarchical file systems usually have a special directory at the root. It can be imagined to be similar to a tree - the branch points are directories that lead to other files. Data files are at the ends of branches and these include files containing program code.

File systems can become corrupt if a computer is turned off before a program is copied to a new location.

Utilities

The OS uses applications called **utilities** which allow the user to manage the computer. There are many different utility programs and they may vary across operating systems. They are often accessed via a special menu or control panel in the OS.



Maintenance utilities

These include:

Backup: This allows the user to restore the system to a previous state which is saved as a backup. This is only usually used if a system malfunctions.

Disk cleaner: The storage drive is divided into a number of clusters. The table of contents serves as an address book, keeping a record of each file and the clusters used to store that file. When a file is deleted, the address to the location on the disk is removed.

Hard Disk formatter & checker:

Storage drives need to be formatted to be compatible with an OS. The OS usually formats storage media when it is connected to the computer. It is often the case that a storage drive cannot be compatible with both Windows and Mac OS X.

A disk formatter will typically carry out the following tasks:

removing existing data from a disk that has been used previously
setting up the file system on the disk, based on a table of contents that allows a file recognised by the operating system to be associated with a specific physical part of the disk
Partitioning the disk into logical drives if this is required.

Another utility program, which might be a component of a disk formatter, performs disk contents analysis and, if possible, disk repair when needed.

The program first checks for errors on the disk.

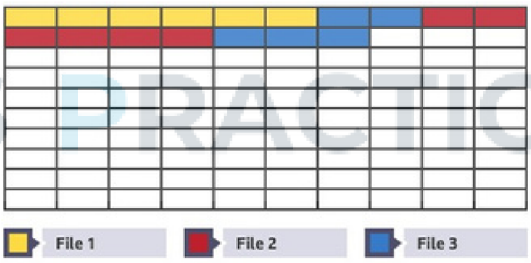
Some errors arise from a physical defect resulting in what is called a **'bad sector'.**

There are a number of possible causes of bad sectors. However, they usually arise either during manufacture or from mishandling of the system.
An example is moving the computer without ensuring that the disk heads are secured away from the disk surface.
Other errors arise from some abnormal event such as a loss of power or an error causing sudden system shutdown.

As a result some of the files stored on the disk might no longer be in an identifiable state.
A disk repair utility program can mark bad sectors as such and ensure that the file system no longer tries to use them.
When the integrity of files has been affected, the utility might be able to recover some of the data but otherwise it has to delete the files from the file system.

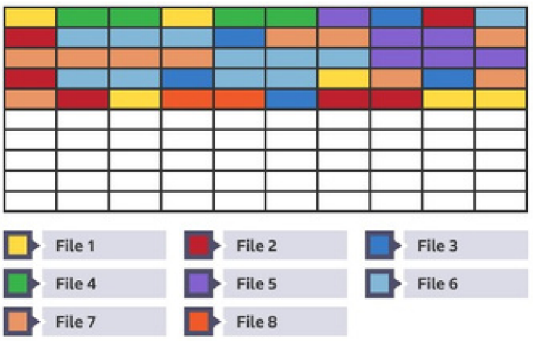
Disk defragmentation:

When a file is stored on a hard disk it is actually stored not as a whole file, but as a series of segments. Sometimes the segments run together in sequence (see File 1) and sometimes the segments are split up over a disk (see File 3). This is known as fragmentation.



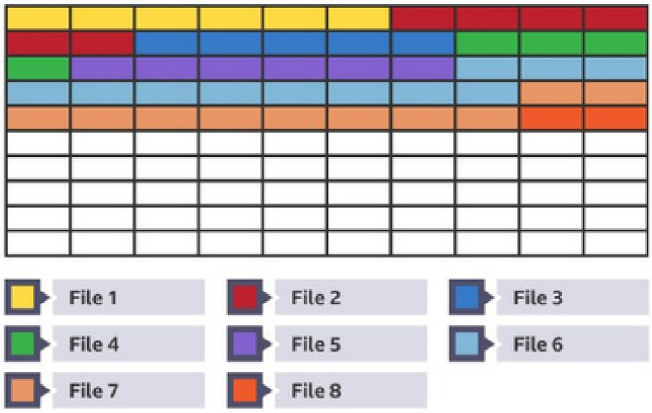
Over time, more and more files become fragmented, as do individual files. A fragmented disk takes longer to read from and write to, making a computer run slower.

When files are deleted, unused clusters become available for reuse.
These can end up being distributed across a drive, especially if the original files were small.
If a large file is then written to a drive, its data could be spread across different clusters leading to file fragmentation.



Defragmentation involves rearranging the information on a disk so that files appear in continuous sequences of clusters.

This will improve file access times. Most modern operating systems run this process automatically.



Defragmentation software takes the fragmented files and rearranges the segments so that they run contiguously. This decreases read/write time, thereby speeding up computer performance.

Backup software: This allows the user to restore the system to a previous state which is saved as a backup. This is only usually used if a system malfunctions.

It is quite likely that you perform a manual backup every now and then using a flash memory stick. However, a safer and more reliable approach is to have a backup utility program do this for you. You can still use the memory stick to store the backed-up data but the utility program will control the process. In particular it can do two things:

establish a schedule for backups

only create a new backup file when there has been a change.

Disk cleaner: The storage drive is divided into a number of clusters. The table of contents serves as an address book, keeping a record of each file and the clusters used to store that file. When a file is deleted, the address to the location on the disk is removed

Security utilities

These include:

user accounts - allow the user to allocate specific users and protects personal files and programs from unauthorised access.
encryption - can encrypt data when it is stored, or whenever it is transmitted over a network.
anti-virus software - detects and blocks viruses.
firewall - can be used to filter between trusted and untrusted networks and prevent programs from communicating through the use of ports.

Program libraries

Program libraries are used when software is under development and the programmer can utilise pre-written subroutines in their own programs, thus saving considerable development time

When software routines are written (such as a sort routine), they are frequently saved in a program library for future use by other programmers. A program stored in a program library is known as a **library program**.

We also have the term **library routines** to describe subroutines which could be used in another piece of software under development.

The 'programs' in a program library are usually subroutines created to carry out particular tasks. A programmer can use these within their own programs.

All newly developed programs are likely to contain errors, which only become apparent as the programs are tested or used. It saves a programmer a lot of time and trouble to be able to include already tried and tested subroutines taken from a program library.

The most obvious examples of library routines are the built-in functions available for use when programming in a particular language.

We will discuss the methods available for translation of source code. For now, we simply need an overview of what happens. The source code is written in a programming language of choice. If a compiler is used for the translation and no errors are found, the compiler produces object code (machine code). This code cannot be executed by itself. Instead it has to be linked with the code for any subroutines used by it. It is possible to carry out the linking before loading the full code into memory and running it.

The 'programs' in a program library are usually subroutines created to carry out particular tasks. A programmer can use these within their own programs.

All newly developed programs are likely to contain errors, which only become apparent as the programs are tested or used. It saves a programmer a lot of time and trouble to be able to include already tried and tested subroutines taken from a program library.

The most obvious examples of library routines are the built-in functions available for use when programming in a particular language.

We will discuss the methods available for translation of source code. For now, we simply need an overview of what happens. The source code is written in a programming language of choice. If a compiler is used for the translation and no errors are found, the compiler produces object code (machine code). This code cannot be executed by itself. Instead it has to be linked with the code for any subroutines used by it. It is possible to carry out the linking before loading the full code into memory and running it.

Advantages of DLL (dynamic link library):

There is a major disadvantage in linking library routines into the executable code, because every program using a routine has to have its own copy. This increases the storage space requirement for the executable file and memory usage when more than one process uses the routine.

The alternative is to use a routine from a dynamic linked library (DLL). When a DLL routine is available the executable code just requires a small piece of code to be included.

This allows it to link to the routine, which is stored separately in memory, when execution of the program needs it.

Many processes can be linked to the same routine. This has the advantage that the executable files for all programs need less storage space.

Memory requirement is also minimised.

Another advantage is that if a new version of the routine becomes available it can be loaded into memory so that any program using it is automatically upgraded.

Disadvantage of DLL (dynamic link library):

The main disadvantage of using a DLL is that the program is relying on the routine beingavailable and performing the expected function.

If for some reason the DLL becomes corrupted or a new version has bugs not yet discovered the program will fail or produce an erroneous result.

The user running the program will find it difficult to establish what needs to be done to get the program to run without error.

Language translators:

The use of an assembler for translating a program written in assembly language has been discussed in Chapter 6 (Sections 6.02, 6.03 & 6.04). This chapter will introduce the translators that are used to translate a program written in a high-level procedural language.

Compilers and interpreters

The starting point for using either a compiler or an interpreter is a file containing source code, which is a program written in a high-level language.

Interpreters:

For an interpreter the following steps apply.

1. The interpreter program, the source code file and the data to be used by the source code program are all made available.
2. The interpreter program begins execution.
3. The first line of the source code is read.
4. The line is analysed.
5. If an error is found, this is reported and the interpreter program halts execution.
6. If no error is found, the line of source code is converted to an intermediate code.
7. The interpreter program uses this intermediate code to execute the required action.
8. The next line of source code is read and Steps 4–8 are repeated.

Compilers:

For a compiler the following steps apply.

1. The compiler program and the source code file are made available but no data is needed.
2. The compiler program begins execution.
3. The first line of the source code is read.
4. The line is analysed.
5. If an error is found this is recorded.
6. If no error is found the line of source code is converted to an intermediate code.
7. The next line of source code is read and Steps 4–7 are repeated.
8. When the whole of the source code has been dealt with one of the following happens.
9. If no error is found in the whole source code the complete intermediate code is converted into object code.
10. If any errors are found a list of these is output and no object code is produced.

Execution of the program can only begin when the compilation has shown no errors. This can take place automatically under the control of the compiler program if data for the program is available. Alternatively, the object code is stored and the program is executed later with no involvement of the compiler.