# Chapter 5 System Software

## *5.1 Operating System*

**5 System Software**

**5.1 Operating System**

| Candidates should be able to: | Notes and guidance |
|---|---|
| Explain why a computer system requires an Operating System (OS) | |
| Explain the key management tasks carried out by the Operating System | Including memory management, file management, security management, hardware management (input / output / peripherals), process management |
| Show understanding of the need for typical utility software provided with an Operating System | Including disk formatter, virus checker, defragmentation software, disk contents analysis/disk repair software, file compression, back-up software |
| Show understanding of program libraries | Including: <br>• software under development is often constructed using existing code from program libraries <br>• the benefits to the developer of software constructed using library files, including Dynamic Link Library (DLL) files |

### Why does a pc need an operating system?

- The hardware is unusable without an OS
- Acts as an interface for the user to communicate with the hardware
- Provides an environment for the user to run other programs

## *Key Management tasks conducted by operating system*

### Process / Task Management

- Manages the scheduling of processes
- Allows multi-tasking
- Ensures fair access
- Handles priorities
- Manages the resources the processes need
- Enables processes to share information
- Prevents interference between processes

## File / Secondary Storage Management

- Storage space divided into file allocation units
- Space allocated to particular files
- Maintains/creates directory structures
- Specifies the logical method of file storage (e.g. FAT or NFTS)
- Provides file naming conventions
- Controls access // implements access rights // implements password protection
- Specifies tasks that can be performed on a file, close, minimize, delete, copy, duplicate, create, move etc.
- Creates files/folders
- Renames file/folders

## Peripheral / Hardware / Device / Input/Output Management

- Installation of device driver software
- Control of hardware usage by processes
- Device detection
- Power Management
- Keep track of device status
- Managing interrupts
- Sending control signals
- Control of buffers
- Management of queues

## Interrupt Handling

- Handles the signals sent when the attention of the processor is required elsewhere
- Halts the execution of the current process
- Stores the current values of the current process on the stack
- Loads and executes the appropriate ISR code
- Use of priorities for handling multiple interrupts
- Saves data on power outage

## Error Detection And Recovery Management

- Deals with interrupts
- Deal with run time errors generated by software
- Deal with hardware faults
- Error diagnostic messages
- Deadlock detection and recovery
- Safe-mode boot-up routines
- System shutdown
- Saves system restore points

## Security Management

- Provides user accounts and passwords
- Sets up user accounts
- Checks passwords, authentication
- Implements access rights
- Automatic backup
- System restore
- Ensures privacy of data

## Provision of user interface

- Allows a user to communicate with the hardware
- By making navigation around the system easier
- Provides facility for user inputting the data
- Provides facility for user outputting to the user
- e.g.  Command line / GUI / menu-driven

## Printer Management

- Installs printer driver
- Sends data to the printer buffer
- Sends commands to printer
- Receives and handles error messages/signals/interrupts from the printer

## Memory management

- Reads data from RAM, Writes data to RAM e.g. any current data
- RAM is assigned into blocks
- dynamic allocation of RAM to programs / processes
- Allocates virtual memory when there's insufficient space in RAM to run a program
- Allocates RAM to optimize performance
- Ensures fair usage of memory
- Organizes memory
- Keep processes separate
- To release memory when a process stops
- moves data from secondary storage when needed // manages paging, segmentation and virtual memory // Swaps data to and from the hard drive
- Memory protection, prevents the process to access memory not allocated to it, prevents two programs / processes occupying the same area of RAM at the same time

## *Utility programs*

### Disk Defragmenter

- Rearrange the data on a disk
- so that files are contiguous/together and all free space is collected together to improve efficiency
- Creates a larger area of contiguous free space
- Less time is taken to access files because each one is contiguous so there is less head movement

### Disk formatter

- Prepare a disk for initial use
- Makes existing data inaccessible
- Sets up the specified file system
- Partitions the disk into logical drives
- May check of errors on the disk

### File compression

Reduce the size of file

### Disk repair software

Examine the disk to find any bad sectors and marks bad sectors as unusable
Checks for any errors
Resolves any error on disk
Retrieves files from a damaged disk
reduces access times by optimizing storage

### Virus Checker

Scans files stored on a computer system
Scans files when they enter the system
Sets up schedules for virus-checking
Isolates/Quarantines/deletes viruses
Regularly updates the virus definitions
makes more RAM available for programs to run because it removes software that might be taking up memory / replicating

### Backup Software

Creates copy of the contents of disk
Allows user to decide what is backed up, all data, all files that have changed since last backup
Allows the user to set up an offsite backup
May encrypt backup files
Restores data if necessary

## *Program libraries*

### Library routine

- Pre written code
- To perform common / complex tasks
- Library is imported into the program, and the functions written in the library routine can be called in your program

| Advantages | Disadvantages |
|---|---|
| ✓ Code is already tested, so it is robust and more likely to work<br>✓ Saves programming time, no need to write from scratch<br>✓ The programmer can use functions that he/she may not be able to code<br>✓ In there is an improvement in the library routine, program is updated automatically | ✓ May not work with other code, compatibility issues<br>✓ Not guaranteed throughout testing, may have bugs<br>✓ May not meet exact needs, may need editing otherwise<br>✓ If library routine is changed, it might cause problems with the code or produce unexpected outcomes |

### DLL Files (dynamic link library files)

- A shared library file
- Code is saved separately from the main .EXE files
- Code is only loaded into main memory when required at run time
- The DDL file can be made available to several applications at the same time

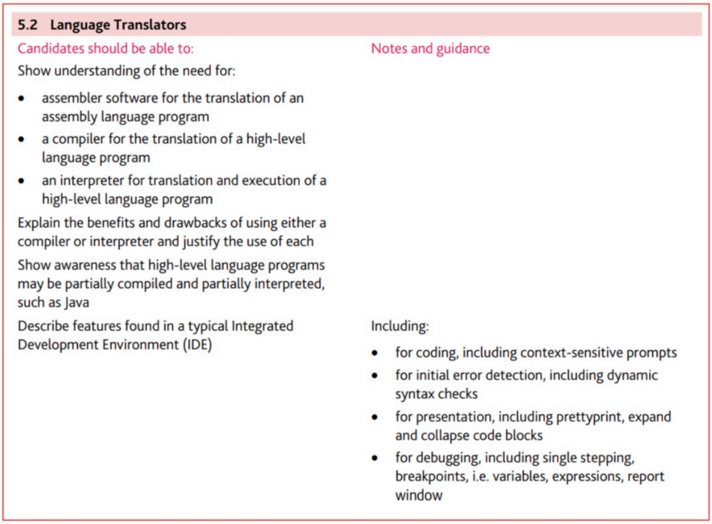
### Why should we use DLL files while developing

- DLL file is only loaded into memory when required
- So the executable file for the game is smaller because the executable does not contain all the library routines
- maintenance not needed to be done by the programmer because the DLL is separate from program
- Changes in the DLL file are independent of the main program
- no need to recompile the main program when changes are made to DLL because changes / improvements/ error correction to the DLL file code are done independently of the main program The program will get the benefit of the updates automatically
- The same DLL file can be used in several programs at the same time
- DLL routines are pre written saving the developer's time
- DLL routines are pre tested so should be reliable
- Developers can take advantage of other programmers expertise

## Why we shouldn’t DDL files while developing

- Application will not work if DLL is corrupted, could mean that the program stops working as expected
- Malicious changes to the DLL file could install a virus on the user’s computer
- An external change to the DLL could stop the application working or change the way it works
- The DDL file must be present at run time otherwise there’s an error
- Appropriate software must be available at run time to link the DDL file

## *5.2 Language Translators*



**5.2 Language Translators**

| Candidates should be able to: | Notes and guidance |
|---|---|
| Show understanding of the need for:<br>• assembler software for the translation of an assembly language program<br>• a compiler for the translation of a high-level language program<br>• an interpreter for translation and execution of a high-level language program | |
| Explain the benefits and drawbacks of using either a compiler or interpreter and justify the use of each | |
| Show awareness that high-level language programs may be partially compiled and partially interpreted, such as Java | |
| Describe features found in a typical Integrated Development Environment (IDE) | Including:<br>• for coding, including context-sensitive prompts<br>• for initial error detection, including dynamic syntax checks<br>• for presentation, including prettyprint, expand and collapse code blocks<br>• for debugging, including single stepping, breakpoints, i.e. variables, expressions, report window |

### *Language Translators*

#### Compiler
- Converts a high-level language into a different form
- Attempts to translate the whole source code
- Creates a separate error report at the end of the translation process
- If translation successful / no errors creates an executable file

#### Why would one use a compiler?
- To make an executable file
- So it can be distributed without the source code
- To test the program without re-compiling3
- Faster to run the executable file

## Interpreter

- Converts a high-level language into a different form
- Reads each line then translates it and executes it
- Needs the source code to be present when the user's program is run
- Stops when an error is encountered // displays errors where it finds them

## How JavaScript code in interpreted by an interpreter?

- The code is translated one line at a time
- And executed immediately
- Interpreter stops as soon as it finds any error

## Why would one use an interpreter?

- To run/test incomplete program, good for development
- To locate individual errors in the program since interpreter stops where error is occurred
- Programmer can correct errors in real time good for debugging
- To make changes and notice effects in real time

## Identify debugging tools that a typical IDE can provide

- **Breakpoints** - Stops the code executing at a set line
- **Single stepping** - Executes one line of the program and then stops
- **Report windows** - Outputs the contents of variables and data structures
- **Dynamic syntax check** - Underlines or highlights statements that do not meet the rules of the language

## Identify three other tools or features found in a typical IDE to support the writing of the program

- Color coding // pretty printing
- Auto complete
- Auto correct
- Context sensitive prompts - Displays predictions of the code being entered
- Expand and collapse code blocks

## Explain why high-level language programs might be partially compiled and partially interpreted.

- partially compiled programs can be used on different platforms as they are interpreted when run
- code is optimized for the CPU as machine code is generated at run time

## How is java compiled?

- Java uses a twostep translation process
- Java code is partially compiled and partially interpreted
- Code is first translated into intermediate code using java compiler
- Intermediate code is finally interpreted by the java virtual machine

**Benefit and drawback of using a language that's partially compiled and partially interpreted**

- Benefit: program may be interpreted on different platforms
- Drawback: intermediate code, program still needs to interpreted on user's computer which may run slowly
- Extra CPU resources may be required

**State drawbacks of using a compiler compared to an interpreter during program development.**

- larger amounts of source code take time to compile
- slower to produce the object code than an interpreter
- code cannot be changed without recompilation
- the program will not run if there are any errors
- errors cannot be corrected in real-time
- one error may result in other false errors being reported
- cannot easily test specific sections of the source code // cannot easily test unfinished source code