Types of Software:

Software is a set of programs, which is designed to perform a well-defined function. A program is a sequence of instructions written to solve a particular problem.

There are two types of software −

- System Software
- Application Software

System Software

The system software is a collection of programs designed to operate, control, and extend the processing capabilities of the computer itself. System software is generally prepared by the computer manufacturers. These software products comprise of programs written in low-level languages, which interact with the hardware at a very basic level. System software serves as the interface between the hardware and the end users.

Some examples of system software are Operating System, Compilers, Interpreter, Assemblers, etc.

Here is a list of some of the most prominent features of a system software −

- Close to the system
- Fast in speed
- Difficult to design
- Difficult to understand
- Less interactive
- Smaller in size
- Difficult to manipulate
- Generally written in low-level language

Application Software

Application software products are designed to satisfy a particular need of a particular environment. All software applications prepared in the computer lab can come under the category of Application software.

Application software may consist of a single program, such as Microsoft's notepad for writing and editing a simple text. It may also consist of a collection of programs, often called a software package, which work together to accomplish a task, such as a spreadsheet package.

Examples of Application software are the following −

- Payroll Software

- Student Record Software
- Inventory Management Software
- Income Tax Software
- Railways Reservation Software
- Microsoft Office Suite Software
- Microsoft Word
- Microsoft Excel
- Microsoft PowerPoint

Features of application software are as follows −

- Close to the user
- Easy to design
- More interactive
- Slow in speed
- Generally written in high-level language
- Easy to understand
- Easy to manipulate and use
- Bigger in size and requires large storage space

Programming Languages

Computer Program • A program is a set of instructions following the rules of the chosen language.

• Without programs, computers are useless.

• A program is like a recipe.

• It contains a list of ingredients (called variables) and a list of directions (called statements) that tell the computer what to do with the variables.

• A vocabulary and set of grammatical rules (syntax) for instructing a computer to perform specific tasks.

• Programming languages can be used to create computer programs.

• The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal.

• You eventually need to convert your program into machine language so that the computer can understand it. • There are two ways to do this: – Compile the program – Interpret the program

• Compile is to transform a program written in a highlevel programming language from source code into object code. • This can be done by using a tool called compiler.

• A compiler reads the whole source code and translates it into a complete machine code program to perform the required tasks which is output as a new file.

• Interpreter is a program that executes instructions written in a high-level language.

• An interpreter reads the source code one instruction or line at a time, converts this line into machine code and executes it.

Computer Programming

• Computer programming is the process of writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs.

• This source code is written in a programming language like C++, JAVA, and Perl etc.

Computer Programmer

• A programmer is someone who writes computer program.

• Computer programmers write, test, and maintain programs or software that tells the computer what to do.

What Skills are Required to Become a Programmer?

• Programming - Writing computer programs for various purposes.

• Writing - Communicating effectively with others in writing as indicated by the needs of the audience.

• Reading Comprehension - Understanding written sentences and paragraphs in work-related documents.

• Critical Thinking - Using logic and analysis to identify the strengths and weaknesses of different approaches

• Computers and Electronics - Knowledge of electric circuit boards, processors, chips, and computer hardware and software, including applications and programming.

• Mathematics - Knowledge of numbers, their operations, and interrelationships including arithmetic, algebra, geometry, calculus, statistics, and their applications.

• Oral Expression - The ability to communicate information and ideas in speaking so others will understand.

• Oral Comprehension - The ability to listen to and understand information and ideas presented through spoken words and sentences.

• Written Expression - The ability to communicate information and ideas in writing so others will understand.

• Written Comprehension - The ability to read and understand information and ideas presented in writing

• Deductive Reasoning - The ability to apply general rules to specific problems to come up with logical answers. It involves deciding if an answer makes sense.

• Information Organization - Finding ways to structure or classify multiple pieces of information.

Generations of Programming Language

• The first generation languages, or 1GL, are low-level languages that are machine language.

• The second generation languages, or 2GL, are also low-level languages that generally consist of assembly languages.

• The third generation languages, or 3GL, are high-level languages such as C.

• The fourth generation languages, or 4GL, are languages that consist of statements similar to statements in a human language. Fourth generation languages are commonly used in database programming and scripts.

• The fifth generation languages, or 5GL, are programming languages that contain visual tools to help develop a program. A good example of a fifth generation language is Visual Basic.

Types of Programming Language

• There are three types of programming language: –

Machine language (Low-level language) – Assembly language (Low-level language) – High-level language

• Low-level languages are closer to the language used by a computer, while high-level languages are closer to human languages.

Machine Language

• Machine language is a collection of binary digits or bits that the computer reads and interprets.
• Machine languages are the only languages understood by computers.

• While easily understood by computers, machine languages are almost impossible for humans to use because they consist entirely of numbers.

Machine Language Machine Language 169 1 160 0 153 0 128 153 0 129 153 130 153 0 131 200 208 241 96 High level language 5 FOR I=1 TO 1000: PRINT "A";: NEXT I

Assembly Language

• A program written in assembly language consists of a series of instructions mnemonics that correspond to a stream of executable instructions, when translated by an assembler that can be loaded into memory and executed.

• Assembly languages use keywords and symbols, much like English, to form a programming language but at the same time introduce a new problem.

The problem is that the computer doesn't understand the assembly code, so we need a way to convert it to machine code, which the computer does understand.

• Assembly language programs are translated into machine language by a program called an assembler.

Example: – Machine language: 10110000 01100001 – Assembly language : mov a1, #061h – Meaning: Move the hexadecimal value 61 (97 decimal) into the processor register named "a1".

High Level Language

• High-level languages allow us to write computer code using instructions resembling everyday spoken language (for example: print, if, while) which are then translated into machine language to be executed.

• Programs written in a high-level language need to be translated into machine language before they can be executed.

• Some programming languages use a compiler to perform this translation and others use an interpreter.

• Examples of High-level Language:

• ADA • C • C++ • JAVA • BASIC • COBOL • PASCAL • PHYTON

Windows File System

- FAT Overview
- HPFS Overview
- NTFS Overview

NOTE: HPFS is only supported under Windows NT versions 3.1, 3.5, and 3.51. Windows NT 4.0 does not support and cannot access HPFS partitions. Also, the FAT32 file system is only supported in the Windows 98/95 and Windows 2000.

FAT OVERVIEW

FAT is by far the most simplistic of the file systems supported by Windows NT. The FAT file system is characterized by the file allocation table (FAT), which is really a table that resides at the very "top" of the volume. To protect the volume, two copies of the FAT are kept in case one becomes damaged. In addition, the FAT tables and the root directory must be stored in a fixed location so that the system's boot files can be correctly located.
A disk formatted with FAT is allocated in clusters, whose size are determined by the size of the volume. When a file is created, an entry is created in the directory and the first cluster number containing data is established. This entry in the FAT table either indicates that this is the last cluster of the file, or points to the next cluster.
Updating the FAT table is very important as well as time consuming. If the FAT table is not regularly updated, it can lead to data loss. It is time consuming because the disk read heads must be repositioned to the drive's logical track zero each time the FAT table is updated.
There is no organization to the FAT directory structure, and files are given the first open location on the drive. In addition, FAT supports only read-only, hidden, system, and archive file attributes.

Advantages of FAT

It is not possible to perform an undelete under Windows NT on any of the supported file systems. Undelete utilities try to directly access the hardware, which cannot be done under Windows NT. However, if the file was located on a FAT partition, and the system is restarted under MS-DOS, the file can be undeleted. The FAT file system is best for drives and/or partitions under approximately 200 MB, because FAT starts out with very little overhead. For further discussion of FAT advantages, see the following:

Microsoft Windows NT Server "Concepts and Planning Guide," Chapter 5, section titled "Choosing a File System"

- Microsoft Windows NT Workstation 4.0 Resource Kit, Chapter 18, "Choosing a File System"
- Microsoft Windows NT Server 4.0 Resource Kit "Resource Guide," Chapter 3, section titled "Which File System to Use on Which Volumes"

Disadvantages of FAT

Preferably, when using drives or partitions of over 200 MB the FAT file system should not be used. This is because as the size of the volume increases, performance with FAT will quickly decrease. It is not possible to set permissions on files that are FAT partitions.
FAT partitions are limited in size to a maximum of 4 Gigabytes (GB) under Windows NT and 2 GB in MS-DOS. For additional information on this limitation, please see the following article in the Microsoft Knowledge Base:
ARTICLE-ID: 118335
TITLE : Maximum Partition Size in MS-DOS

For further discussion of other disadvantages of FAT, see the following:

Microsoft Windows NT Server "Concepts and Planning Guide," Chapter 5, section titled "Choosing a File System"

- Microsoft Windows NT Workstation 4.0 Resource Kit, Chapter 18, "Choosing a File System"
- Microsoft Windows NT Server 4.0 Resource Kit "Resource Guide," Chapter 3, section titled "Which File System to Use on Which Volumes"

HPFS OVERVIEW

The HPFS file system was first introduced with OS/2 1.2 to allow for greater access to the larger hard drives that were then appearing on the market. Additionally, it was necessary for a new file system to extend the naming system, organization, and security for the growing demands of the network server market. HPFS maintains the directory organization of FAT, but adds automatic sorting of the directory based on filenames. Filenames are extended to up to 254 double byte characters. HPFS also allows a file to be composed of "data" and special attributes to allow for increased flexibility in terms of supporting other naming conventions and security. In addition, the unit of allocation is changed from clusters to physical sectors (512 bytes), which reduces lost

disk space.
Under HPFS, directory entries hold more information than under FAT. As well as the attribute file, this includes information about the modification, creation, and access date and times. Instead of pointing to the first cluster of the file, the directory entries under HPFS point to the FNODE. The FNODE can contain the file's data, or pointers that may point to the file's data or to other structures that will eventually point to the file's data.
HPFS attempts to allocate as much of a file in contiguous sectors as possible. This is done in order to increase speed when doing sequential processing of a file.
HPFS organizes a drive into a series of 8 MB bands, and whenever possible a file is contained within one of these bands. Between each of these bands are 2K allocation bitmaps, which keep track of which sectors within a band have and have not been allocated. Banding increases performance because the drive head does not have to return to the logical top (typically cylinder 0) of the disk, but to the nearest band allocation bitmap to determine where a file is to be stored.

Additionally, HPFS includes a couple of unique special data objects:
Super Block

The Super Block is located in logical sector 16 and contains a pointer to the FNODE of the root directory. One of the biggest dangers of using HPFS is that if the Super Block is lost or corrupted due to a bad sector, so are the contents of the partition, even if the rest of the drive is fine. It would be possible to recover the data on the drive by copying everything to another drive with a good sector 16 and rebuilding the Super Block. However, this is a very complex task.
Spare Block

The Spare Block is located in logical sector 17 and contains a table of "hot fixes" and the Spare Directory Block. Under HPFS, when a bad sector is detected, the "hot fixes" entry is used to logically point to an existing good sector in place of the bad sector. This technique for handling write errors is known as hot fixing.
Hot fixing is a technique where if an error occurs because of a bad sector, the file system moves the information to a different sector and marks the original sector as bad. This is all done transparent to any applications that are performing disk I/O (that is, the application never knows that there were any problems with the hard drive). Using a file system that supports hot fixing will eliminate error messages such as the FAT "Abort, Retry, or Fail?" error message that occurs when a bad sector is encountered.
Note: The version of HPFS that is included with Windows NT does not support hot fixing.

Advantages of HPFS

HPFS is best for drives in the 200-400 MB range. For more discussion of the advantages of HPFS, see the following:

- Microsoft Windows NT Server "Concepts and Planning Guide," Chapter 5, section titled "Choosing a File System"
- Microsoft Windows NT Workstation 4.0 Resource Kit, Chapter 18, "Choosing a File System"
- Microsoft Windows NT Server 4.0 Resource Kit "Resource Guide," Chapter 3, section titled "Which File System to Use on Which Volumes"

Disadvantages of HPFS

Because of the overhead involved in HPFS, it is not a very efficient choice for a volume of under approximately 200 MB. In addition, with volumes larger than about 400 MB, there will be some performance degradation. You cannot set security on HPFS under Windows NT.
HPFS is only supported under Windows NT versions 3.1, 3.5, and 3.51. Windows NT 4.0 cannot access HPFS partitions.

For additional disadvantages of HPFS, see the following:

Microsoft Windows NT Server "Concepts and Planning Guide," Chapter 5, section titled "Choosing a File System"

- Microsoft Windows NT Workstation 4.0 Resource Kit, Chapter 18, "Choosing a File System"
- Microsoft Windows NT Server 4.0 Resource Kit "Resource Guide," Chapter 3, section titled "Which File System to Use on Which Volumes"

NTFS OVERVIEW

From a user's point of view, NTFS continues to organize files into directories, which, like HPFS, are sorted. However, unlike FAT or HPFS, there are no "special" objects on the disk and there is no dependence on the underlying hardware, such as 512 byte sectors. In addition, there are no special locations on the disk, such as FAT tables or HPFS Super Blocks.

The goals of NTFS are to provide:

Reliability, which is especially desirable for high end systems and file servers

- A platform for added functionality
- Support POSIX requirements
- Removal of the limitations of the FAT and HPFS file systems

Reliability

To ensure reliability of NTFS, three major areas were addressed: recoverability, removal of fatal single sector failures, and hot fixing.

NTFS is a recoverable file system because it keeps track of transactions against the file system. When a CHKDSK is performed on FAT or HPFS, the consistency of pointers within the directory, allocation, and file tables is being checked. Under NTFS, a log of transactions against these components is maintained so that CHKDSK need only roll back transactions to the last commit point in order to recover consistency within the file system.

Under FAT or HPFS, if a sector that is the location of one of the file system's special objects fails, then a single sector failure will occur. NTFS avoids this in two ways: first, by not using special objects on the disk and tracking and protecting all objects that are on the disk. Secondly, under NTFS, multiple copies (the number depends on the volume size) of the Master File Table are kept.
Similar to OS/2 versions of HPFS, NTFS supports hot fixing.

Added Functionality

One of the major design goals of Windows NT at every level is to provide a platform that can be added to and built upon, and NTFS is no exception. NTFS provides a rich and flexible platform for other file systems to be able to use. In addition, NTFS fully supports the Windows NT security model and supports multiple data streams. No longer is a data file a single stream of data. Finally, under NTFS, a user can add his or her own user-defined attributes to a file.

Advantages of NTFS

NTFS is best for use on volumes of about 400 MB or more. This is because performance does not degrade under NTFS, as it does under FAT, with larger volume sizes.
The recoverability designed into NTFS is such that a user should never have to run any sort of

disk repair utility on an NTFS partition. For additional advantages of NTFS, see the following:

- Microsoft Windows NT Server "Concepts and Planning Guide," Chapter 5, section titled "Choosing a File System"
- Microsoft Windows NT Workstation 4.0 Resource Kit, Chapter 18, "Choosing a File System"
- Microsoft Windows NT Server 4.0 Resource Kit "Resource Guide," Chapter 3, section titled "Which File System to Use on Which Volumes"

Disadvantages of NTFS

It is not recommended to use NTFS on a volume that is smaller than approximately 400 MB, because of the amount of space overhead involved in NTFS. This space overhead is in the form of NTFS system files that typically use at least 4 MB of drive space on a 100 MB partition.

Currently, there is no file encryption built into NTFS. Therefore, someone can boot under MS-DOS, or another operating system, and use a low-level disk editing utility to view data stored on an NTFS volume.

It is not possible to format a floppy disk with the NTFS file system; Windows NT formats all floppy disks with the FAT file system because the overhead involved in NTFS will not fit onto a floppy disk.

For further discussion of NTFS disadvantages, see the following:
Microsoft Windows NT Server "Concepts and Planning Guide," Chapter 5, section titled "Choosing a File System"

- Microsoft Windows NT Workstation 4.0 Resource Kit, Chapter 18, "Choosing a File System"
- Microsoft Windows NT Server 4.0 Resource Kit "Resource Guide," Chapter 3, section titled "Which File System to Use on Which Volumes"

Graphical Applications

A GUI (**graphical** user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. ... GUI objects include icons, cursors, and buttons.

t consists of picture-like items (icons and arrows for **example**). ... The main pieces of a **GUI** are a pointer, icons, windows, menus, scroll bars, and an intuitive input device. Some

common **GUIs** are the ones associated with Microsoft Windows, Mac OSX, Chrome OS, GNOME, KDE, and Android.

A major **advantage of GUIs** is that they make computer operation more intuitive, and thus easier to learn and use. **GUIs** generally provide users with immediate, visual feedback about the effect of each action. **GUI** allows multiple programs and/or instances to be displayed simultaneously.

UI is "graphical user interface" and **UI** is just "user interface." **GUI** is a subset of **UI**. **UI** can include non-graphical interfaces such as screen readers or command line interfaces which aren't considered **GUI**.