Computer System
- Software (Part 3) -

Programming

PROGRAMMING is to describe (code) in a programming language a procedure (algorithm) defined by program design.

Each programming language has its own meanings of instructions and the syntax of uniting them, and so describes the algorithms in different ways. Therefore, a general standard (programming paradigm) is required with consideration given to the characteristics of each programming language.

Programming

Programming Style

(1) Clarity – understandability of a program. To increase the level of clarity, coding rules (standards) are generally established. Coding rules specify the rules to follow when doing the coding work: indenting, naming of variables and modules, comments, etc.

(2) Efficiency – ease of creating a program. To increase efficiency, redundant portions of a program must be removed as much as possible.

(3) Maintainability – ease of modifying a program. A person who does the work of programming usually does not modify a program. This is done by another person. To make the work of maintenance easy, it is important to create an easy-to-understand program.

To improve maintainability, a program itself should be highly structured to prevent a modification made in one part, from affecting other parts in a program.

Programming Language

- A programming language is a language for creating a program that instructs a computer to do work.
- Classification of Programming Languages:
  - **Assembly Language**. The closest language to machine languages. One assembly language instruction corresponds to one machine language instruction. The structural organization and grammar of an assembly language differs depending on the computer.
  - **Procedural Language**. Uses a procedure as the basic unit and describes process procedures (algorithms). Almost all procedural languages use a *compilation method* to translate a whole written program (source program) into a machine language program (object program) before it is executed.
    - **Fortran**. A programming language for scientific and technological calculation. It is the first high-level language that was widely used.
    - **COBOL**. A programming language for business processing. It is not suitable for the development of programs that use graphics.
    - **Pascal**. A programming language that was developed for programming education.

Programming Language

- **BASIC**. A conversational programming language for beginners. It also supports the *interpreter method*, which interprets and executes a program on a line-by-line basis.
- **C**. A programming language that was developed by AT&T Bell Laboratories to describe programs for the UNIX OS. It has high portability and is widely used for system development on a workstation or a PC.

❖ **Object-oriented Language**. Treat data and data processing (procedures or methods) as a unified unit called an **object**, and programs are created using objects. Object orientation is an approach that does not focus on processes but on the data that is subject to processing.

- **C++**. Includes the concepts and functions of object-orientation. It is used for more general purpose than C and has a high level of reusability.

Programming Language

- **Java**. Developed by Sun Microsystems, Inc. on the basis of C++. The programs (Java applications) developed with Java support multi-platform in which as program can be executed on different hardware or a different OS as long as in the environment in which JVM (Java Virtual Machine) is installed. A compilation method that uses a *runtime compiler* (JIT compiler) is prevalent.
  - **Java Servlet**. A Java program that is executed on a web server in response to a request from a client. A servlet is a program that is executed on the server side.
  - **Java Applet**. A Java program that is downloaded by a client from a web server and is executed on the client. An applet is a program that is downloaded to the client-side to be executed.
  - **JavaBeans**. A specification for handling a program that is developed with Java as an application component.
- **VB** (Visual Basic). Developed by Microsoft. It enables the creation of a program in a visual environment that uses windows and toolboxes. A macro language called VBA (VB for Application) which enables linking with other applications based on VB is supported by products, such as Excel, Word, and Access.

Programming Language

- **Script Language**. Simple programming languages that enable the end user to develop a simple program (called script). Script is an easily executable series of instructions. In script languages, the end user creates programs through various specifications in a GUI environment by using a mouse or other tools.
  - **JavaScript.** An object-oriented script language to write programs that run in a web server. It enables programming by directly writing instructions into the codes of a web page. ECMAScript is a specification for JavaScript that was standardized by ECMA (European Computer Manufacturers Association) International.
  - **VBScript.** An object-oriented script language to describe programs that run in a web browser. Some of the VB functions can be used on a web page.
  - **PostScript**. A page description language that was developed by Adobe Systems. It describes text data, graphics, and other elements in units of pages.
  - **Perl / PHP / Python / Ruby**. Languages that describe applications launched with CGI (Common Gateway Interface), a mechanism in which a web server launches an application program in response to processing requests from a browser.

Programming Language

- **Markup Language**. A type of programming language for printing and screen display. Text (or a character string) is enclosed in marks that are called *tags*, and attribute information is attached.
  - **SGML** (Standard Generalized Markup Language). A general purpose markup language that was standardized by ISO. It enables description of the logical structure, semantic structure, and attributes of a document so document management and data conversion can be performed easily. The structure of a document is defined with a DTD (Document Type Definition).
  - **HTML** (HyperText Markup Language). A hypertext description language that extends SGML to enable it to handle not just text but also graphics, images, movies, and so on. Hypertext interconnects content with other content (i.e. web page) through a link comprising character information. HTML encloses text with a start tag and an end tag to specify character size and color, image files to be included, web pages to be linked, and so on.
    - **CSS** (Cascading Style Sheets) can be used to define only the layout of a web page that is completely separated from HTML.

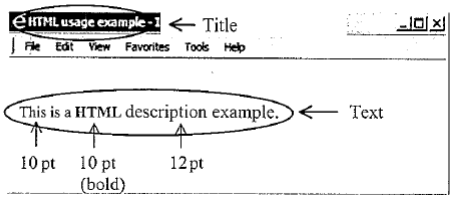
Programming Language

Example: How is the following HTML document displayed? The standard font size for the text is 10 pt.

<TITLE>HTML usage example</TITLE>
<BODY>This is a <B>HTML</B><FONT="+2">description example</FONT></BODY>

- <TITLE> is a tag that indicates the title of the web page
- <BODY> is a tag that indicates the text of a web page
- <B> is a tag that emphasize characters within the tags by using a bold typeface
- <FONT> is a tag that alters the font (size) of the characters within the tags

<Display example>



Programming Language

- **DHTML** (Dynamic HTML). This is an extended HTML specification. By incorporating a script written in JavaScript or VBScript into an HTML page, it enables the dynamic creation of a web page according to user operations, and the checking of user input data.

- **XML** (eXtensible Markup Language). This is a derivative of SGML and HTML, and is a hypertext description language that enables bidirectional linking. It is highly suitable for the Internet environment and is often used for corporate transactions via a network.
  - By using a DTD (Document Type Definition), users can define their own tags. Currently, a dedicated XML schema language called XML Schema is often used instead of DTD.
  - In HTML, the end tag can be omitted in some cases, but in XML there must be a start tag and an end tag even for a null element.
  - XSL (eXtensible Stylesheet Language) can be used for style sheets in XML.

Programming Language

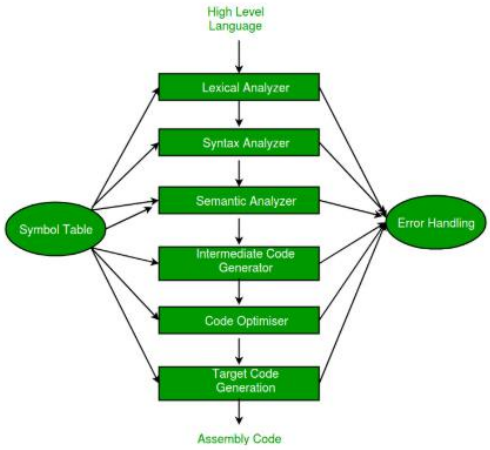
- **XML** (eXtensible Markup Language), technologies related:
  - **XML parser**. A software for the use of the structural elements of an XML document from an application. It allows use of APIs such as DOM (Document Object Model) which manages XML documents in a tree, and SAX (Simple API for XML) which reads and interprets an XML document in order from the top.
  - **SOAP** (Simple Object Access Protocol). A communication protocol for message exchange that calls data or services based on XML.
  - **SVG** (Scalable Vector Graphics). This is a format for two-dimensional vector graphics based on XML.
  - **SDL** (Service Description Language). This is an XML-based language that defines functions for web services. It has been absorbed by WSDL (Web SDL).
  - **Ajax** (Asynchronous JavaScript + XML). A mechanism that transmits XML documents without synchronization between a browser and a web server, and dynamically redraws the screen.
- **XHTML** (eXtensible HyperText Markup Language). A markup language that has redefined HTML (which extends SGML) by using XML. XHTML Basic is intended for use on cell phones and personal digital assistants and is a subset of combined modules defined on the basis of the specifications of the modularization of XHTML.

Language Processor

- A language processor is software that translates programs that are written in a programming language into a machine language or an intermediate language (a language between a machine language and a high-level language).
  - Even though the program before translation and the program after translation can process the same work, their content differs and they are distinguished as described below:
    - **Source program**. A program that is written in a programming language other than machine language,
    - **Object program**. A program that has been translated into machine language or an intermediate language.
  - Types of Language Processors:
    1. **Assembler** – a language processor that translates a source program written in an assembly language into an object program. Translating with an assembler is called “assembling”.
    2. **Compiler** – a language processor that translates a source program written in a high-level language into an object program. Translating with a compiler is called “compiling”. A high-level language that is translated by a compiler is called a compiler language.

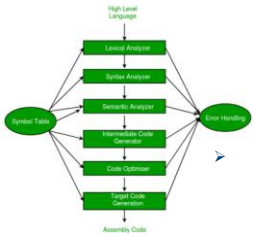
Language Processor (Compiler)

- Translation procedure of a compiler:



Language Processor (Compiler)

- Translation procedure of a compiler:



1) **Lexical analysis** – divides a source program, which is a sequence of character strings, in units of tokens.
   - A general programming language is defined with *formal language* based on *context-free grammar*. The program is divided into units of character strings (tokens) with *finite automaton* or other method which is used to interpret regular expressions.
   - **Formal languages**. These are languages, such as programming languages, for which a strict grammar is defined.
   - **Context-free grammar**. This grammar enables symbols to be replaced, without relying on the preceding or subsequent context. For example, with context-free grammar, if “A” is defined as “xy” or “xyz”, “xyz” can be replaced by either “Az” or “xA”.

Language Processor (Compiler)

- **Regular expression**. A notation that is used in the definition of a string format.

Example of notation in regular expression:

[A-Z]* [0-9]+

[A-Z]: Represents a single alphabetical character

[0-9]: Represents a single number

* : Represents the repetition of zero or more times of the immediately preceding regular expression

+ : Represents the repetition of one or more times of the immediately preceding regular expression

Note: In the strings defined in this example, after an alphabetical character is repeated zero or more times, a number is repeated one or more time. In other words, “AB1” and “123” are applicable, but “ABC” and “12A” are not applicable.

- **Finite automaton**. An automaton is a notation method that models the difference in processing of input depending on the internal state. A finite automaton is composed of a finite number of states and transitions, and is represented with a *state transition table*, a *state transition diagram* or such other method.

Language Processor (Compiler)

Regular expression: **[A-Z]* [0-9]+**

Finite automaton representation:

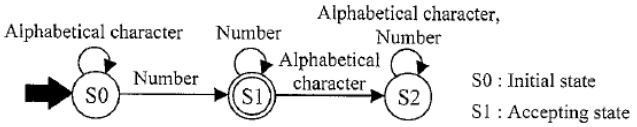
[Representation with a state transition table]

	S0	S1	S2
Alpabetical character (A - Z)	S0	S2	S2
Number (0 - 9)	S1	S1	S2

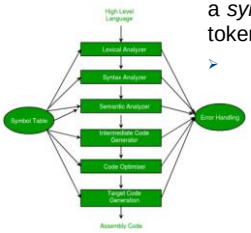
S0: Initial state

S1: Accepting state

[Representation with a state transition diagram]



Language Processor (Compiler)



2) **Syntax analysis** – interprets the token structure according to the grammar of the programming language, and generates a syntax tree. In context-free grammar, syntax is often defined with *BNF notation* or a *syntax diagram*, so this can be called an activity to confirm that the tokens in the program constitute the correct meaning.

- **BNF** (Backus Naur Form) notation. This is a metalanguage (i.e. language to define a language) that defines the grammar of a programming language or such other aspects. It is also used for the definition of XML.

[Example of BNF notation]

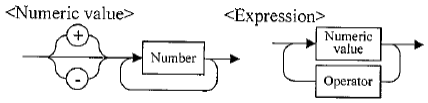
<expression> ::= <numeric value>|<expression><operator><numeric value>
<numeric value> ::= <number>|<sign><number>|<numeric value><number>
<operator> ::= <sign> | * | /
<sign> ::= + | −
<number> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Note: In this example, "+4−12*6" is interpreted as below.

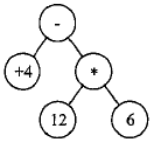
+4−12*6
→ <sign><number><sign><number><number><operator><number>
→ <numeric value><sign><numeric value><number><operator>
<numeric value>
→ <numeric value><sign><numeric value><operator><numeric value>
→ <numeric value><operator><numeric value><operator><numeric value>
→ <expression><operator><numeric value><operator><numeric value>
→ <expression><operator><numeric value>
→ <expression> … interpreted as correct <expression>

Language Processor (Compiler)

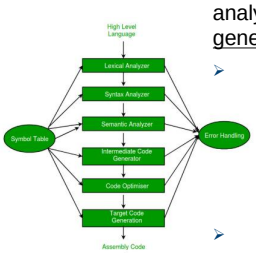
- **Syntax Diagram.** A diagram of a definition in BNF notation that is visually easy to understand.



- **Syntax tree**. This is a diagram of the results of syntactic analysis in a tree structure. An abstract tree structure especially refers to a tree structure that is composed of only the information that is required for semantic analysis.



Language Processor (Compiler)



3) **Semantic analysis** – analyzes a tree structure generated in syntax analysis for meaning in terms of a programming language, and generates an intermediate code.

- Semantic analysis includes interpretation of sections that cannot be analyzed with the grammar of the programming language alone (e.g. validity of variable names, calling of external functions). In the intermediate code, the *quadruple format* or *reverse Polish notation* is sometimes used to represent formulas.

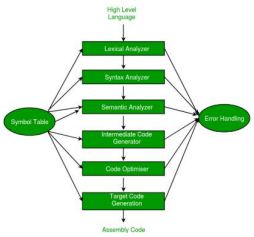
- **Quadruple format** (three-address statement). This is a method to represent an expression with four elements (i.e. operator, operand1, operand2, and result, including three variable addresses) in the form.

  For example, expression "s = 4 -12 * 6" is represented as:

  (*, 12, 6, T1)

  (-, 4, T1, s)

Language Processor (Compiler)



- **Reverse Polish notation.** This is a method for formulas in which an operator is written after variables (or expressions) or numeric values as the subject of the operation. Since the order in which the operators appear is the order of execution, it is used to represent formulas in a computer. There is also a notation method in which an operator is first written, this is called *Polish notation*.

  Example: s = 4 – 12 * 6, is expressed as

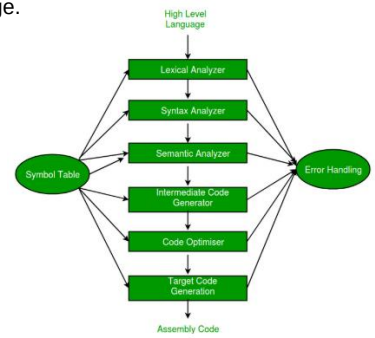
  Reverse Polish Notation: (s 4 12 6 * - =)

  Polish notation: (= s – 4 * 12 6)

4) **Optimization** – this analyzes intermediate code, modifies the program structure (e.g. expression simplification, composition modification), and optimizes the code so that it can be executed at a higher efficiency level.

Language Processor (Compiler)

5) **Code generation** – this converts the optimized intermediate code into the target object code. Basically, the intermediate code is converted into machine language. However, these days, it is temporarily converted to an intermediate code (e.g. byte code using Java byte code or CIL (Common Intermediate Language)), and then, upon execution, converted into an environment-dependent machine language.



Language Processor (Compiler)

- Types of compiler depending on the intended use:
  - **Precompiler** (preprocessor). This is a compiler that interprets additional (or supplemental) functions contained in a high-level language program, and then converts to a high-level language program before actual compilation.
  - **Optimization compiler**. This is a compiler that focuses on the optimization of processing, and reduces the size and execution time of the object program after compiling as much as possible.
  - **Cross compiler**. This is a compiler that generates an object program on one computer for another computer that has a different machine language. An assembler that is used for the same purpose is called a *cross assembler*.
  - **Runtime compiler** (JIT or Just-In-Time compiler). This is a compiler that translates a source program or intermediate code that is written in an intermediate language into a machine language just before its execution. It takes a somewhat long time at the start of execution, but it has the advantage of being able to execute a program without depending on the execution environment.

Language Processor

3. **Interpreter** – a language processor that translates and executes a source program written in a high-level language on an instruction-by-instruction basis.

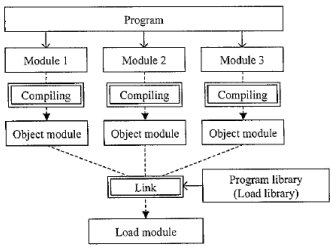
	Compiler	Interpreter
Object program	Generated	Not generated
Translation units	All statements at once (translation not possible if a program is not completed)	One instruction at a time (translation and execution possible even if the program is not completed)
Execution speed	Fast	Slow
Suitable application	Batch processing	Interactive processing

4. **Generator** – a language processor that generates object programs by providing parameters, such as input data/output data specifications and process conditions. Since it can generate an object program without the need to be aware of the processing procedure, it is sometimes referred to as a *non-procedural language*.
5. **Translator**. A software that converts a program into a different programming language. A compiler or an assembler can be called a translator that converts (or translates) a source program into an object program. This does not apply to interpreters.

Language Processor

- An object program created by a language processor cannot be executed as is. In order to execute an object program, various **service programs** (i.e. utility programs) are required.
  1) **Linker** (linkage editor) – a service program that assembles (links) multiple object programs (object modules) into a single executable program (load module). Linking performed by a linker is called *static linking* in order to distinguish it from *dynamic linking* which incorporates components (modules) from a DLL (Dynamic Link Library) by the OS just prior to execution.

In many cases, a source program that is translated into machine language by a compiler is originally a module into which a single program was divided. In such cases, the compiler processes other modules (e.g. external functions) as an unresolved address. The role of the linker is to resolve this unresolved address and integrate the divided and translated object program. When the linker does this, it also has a function to retrieve components (i.e. library modules) that are registered in the program library (i.e. load library) and incorporate them.



Linker (linkage editor) operation

Language Processor

- **Service programs**,
  2) **Loader** – a program that stores load modules in main memory and executes them. There is also a program called a *linking loader* which makes an object module into an executable load module and stores it in main memory.

  3) **Editor** – a software that performs an auxiliary role concerning program input, modification, and other programming activities. This refers to a *text editor* that is used for character input, but there are also other editors such as *graphical editors* that are used for inserting diagrams or images and *structure editors* that increase the input efficiency according to the structure of a specific programming language.

  4) **Emulator** – a microprogram that creates a pseudo environment including a different OS and other programs, and executes the programs of another type of machine. **ICE** (In-Circuit Emulator) is an emulator that imitates the functions of a given microprocessor by using hardware in the development of a microprogram or such other activity. It is used as a testing tool (debugging tool) in development.

Language Processor

- **Service programs**,
  5) **Simulator** – a program or system that performs simulation concerning the complex phenomena of natural science and such other fields. In the case of a language processor, it is sometimes used as a program development tool to simulate the behavior of a program.

  6) **Sort / merge programs** – a program that performs sort processing on data. A *merge program* is a program that performs merge processing to combine multiple sorted data into one. Since there is a close relationship between sorting and merging, they are often provided as a common service program.

Program Attributes

1) **Relocatable program**
   - A program that can be read into memory by a loader and executed at any address. It is sometimes called a *relocation program*. The program is made executable by actions including correction of the address information corresponding to the load position. Programs for which the address can be changed even during execution are called *dynamic relocatable programs*.

2) **Reusable program**
   - A program that can be used any number of times. Normally, a program must be reloaded into main memory for each execution. However, a reusable program does not require any reloads and can be used repeatedly once it is loaded. When the execution of a reusable program starts, actions including the initialization of variables are performed to maintain process consistency. Therefore, it cannot be used for multiple tasks at the same time.

3) **Reentrant program**
   - Allows the correct results to be obtained even if its used in multiple tasks simultaneously. A reentrant program is divided into a variable part and a procedure part, and implements concurrent processing by allocating a variable part to each task while only the procedure part is shared.

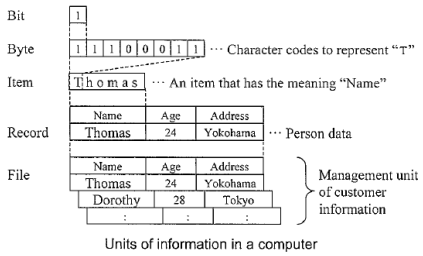
Program Attributes

4) **Recursive program**
   - A program that can call itself during program execution. It can return to the state immediately prior to execution by recording the states during processing with the LIFO (Last-In First-Out) method. However, even if the program is able to call itself, there is no guarantee that it can also be called from another task, so it is not necessarily a reentrant program.

Files

- A file is a unit that a computer uses to manage information. The data management function of an OS can be described as a function to manage files.
- Files and Records

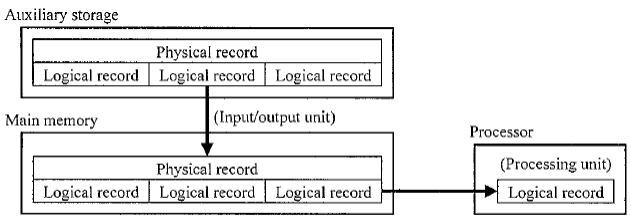
- Units of information in a computer:
  - **Bit** – the smallest unit of memory in a computer. It records either 0 or 1.
  - **Byte** – this is a collection of multiple bits (eight bits). Since a character code is usually represented in 8-bit form, it is a unit that represents a single character.
  - **Item** – this is a unit that is composed of several characters and has a single meaning
  - **Record** – this is a unit that includes multiple related items and is processed as data
  - **File** – this is a unit that includes multiple records of the same format and manages data.



Units of information in a computer

Files and Records

- **Classification of Records**
  1. **Physical records and logical records**
     - To handle records that are actually stored on an auxiliary device, it is not efficient to perform input and output on a record basis, so this is performed with a unit that groups multiple records called a *block*. This is referred to as a **physical record** as it is a record that is handled physically. A record that forms a processing unit is called a **logical record.**



Physical records and logical records

Files and Records

2. **Record formats Classification**
   - **Fixed length record**. A record format in which the size (i.e. number of bytes) of all records is fixed. These records can be made into blocks, and are also easy to handle.
   - **Variable length record**. A record format in which the length of each record is different. Even though the record size varies, these records can be made into blocks because the record size information is stored at the beginning of each record.
   - **Undefined length record**. A record format in which the length of each record is different like variable length records. Undefined length records, however, cannot be made into blocks, because they contain no information concerning the size of each record.

Files and Records

- **Classification of Files**
  1. Classification based on purpose of usage
     - **Master file**. This is a file that records the information that forms the core of business (e.g. product information, customer information).
     - **Transaction file.** This is a file that records information, such as updates for a master file.
     - **Historical file**. This is a file that records the history of a process. It is also called a *journal file* or a *log file*, and is mainly used as a measure against faults.
     - **Backup file**. This is a file that is an exact duplicate of the content of an important file. It is mainly used as a measure against faults.
     - **Working file**. This is an intermediate file that is created temporarily during the implementation of a process.

Files and Records

2. Classification by storage (or usage) period
   - **Archived file (saved file)**. This is a file that is to be continually used over a long period, and is also called a *permanent file*. This classification includes master files.
   - **Temporary file**. This is a file that is used temporarily as needed and is generally erased after it is used. This classification includes files such as transaction files and working files.

3. Classification based on users
   - **System file**. This is a file for system operations and management, such as programs that constitute an OS.
   - **User file**. This is a file that records programs or data used by a user.

File Access Methods

1) **Sequential access** – a method for accessing records in order from the start of a file. It can only read records from a file from the beginning of the file in order. Therefore, it is a good method for reading and processing all records in order from the top of the file, but it is not suitable for processing from a specific record in the middle of a file.
   - When it writes records to a file, it appends the records after the last record in order (if there are no records already present, it writes in order from the top), so there is no wasted area. This means it is efficient for recording, but to insert a record in the middle of a file, it is necessary to remake the file.

2) **Direct access** (random access) – a method that directly accesses a specific record independent of the order of the records in a file. It is suitable for processing (reading or writing) on a specific record, but is not suitable for the processing of all records.

3) **Dynamic access** – an access method that combines direct access and sequential access. Example is the creation of a list of names beginning with “s”. The method for reading and writing records will be to first perform direct access to records with names beginning with “s”, and then perform sequential access for the subsequent records.

File Organization Formats

- A file organization format defines the order of records to be recorded in a file and the structure. The user must consider specifically how the file is to be used and consider which organization format is appropriate.

1. **Sequential organization file** – has a dedicated organization format for sequential access, which writes records sequentially on storage media. This organization can be implemented on all storage media including DASD (Direct Access Storage Devices). On magnetic tape, only sequential organization can be implemented.

2. **Relative organization file** – has a format that writes records in order as in a sequential organization file, and then assigns a relative number (relative address) to each record from the top of records. It uses storage media efficiently, and direct access with the relative number of a record is also possible.

File Organization Formats

3. **Indexed Sequential organization file** (or indexed sequential file) – is composed of two areas: an area for data, and an area in which information (i.e. index) for direct access to a specific records is stored. Its usage of storage media is efficient, and it can create large files.

   - **Prime area**. This is an area in which records are stored. Records in this area are stored in logically ascending order, and sequential access is possible.

   - **Index area**. This is an area in which index records that match record keys with record addresses are stored. Type of indexes include a master index, a cylinder index, and a track index.

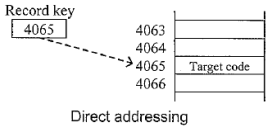
   - **Overflow area**. This is an area in which records ejected from the prime area are stored. If many records are stored in this area, the efficiency of access decreases and reorganization is required. Type of overflow areas include a cylinder overflow area and an independent overflow area.

File Organization Formats

4. **Partitioned Organization file** – a file is divided into multiple units called *members* to store it. The information from each member is managed with a *directory*.
   - Members are subfiles into which a sequential organization is divided, and update management can be performed in units of members.
   - Deletion of members only removes members from the directory and the corresponding members in the memory area remain as is. As a result, the number of waste areas in which unused members are stored increases, and storage efficiency decreases, so it is necessary to periodically reorganize memory areas.
   - Generally used as a program library.

File Organization Formats

5. **Direct Organization file** – has a dedicated format for direct access that enables access to a specific record by the specification of an address. The location of records is not sequential, so it does not use storage media efficiently. It can only be implemented on DASD (Direct Access Storage Devices) such as hard disks.
   - **Direct addressing.** Uses the value of a record key as the record address. No processing is required to calculate the address, but depending on the record key, there may be many variations of addresses, so it is not practical.



Direct addressing

   - **Indirect addressing**. Changes the value of a record key in accordance with a rule and calculates the record address. This conversion is called **hashing**, and the used rule (function) is called a **hash function**. Therefore, a direct organization file that uses indirect addressing is sometimes called a **hash organization file**.

Files Organization Formats

- Hash Methods for calculating a record address from a record key with indirect addressing:
  1) **Division Method**. This is a method that divides a record key by a given value (e.g. the closest prime number to the total number of records) and uses the remainder as the address.

Example: When the record key is 123456 and the number of records is 100,

123456 ÷97 (closest prime number to 100) = 1272 r. 72

→ 72 is used as the record address

  2) **Superposition (Registration)**. This splits a record key into parts in accordance with a rule, and uses the sum of these parts as the address.

Example: When the record key is 123456, it is split into parts comprising the three digits in the first half and the three digits in the second half

123 + 456 = 579

→ 579 is used as the record address

File Organization Formats

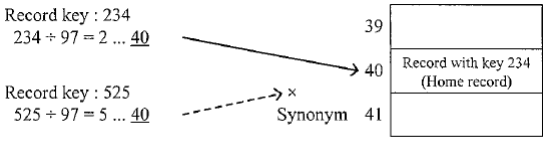
  3) **Radix conversion**. Record keys are normally handled as decimal values, but this uses the result of conversion to a radix other than decimal (e.g. ternary) as the address.

Example: When the record key is 123456, ternary radix conversion is performed

$1 \times 3^5 + 2 \times 3^4 + 3 \times 3^3 + 4 \times 3^2 + 5 \times 3^1 + 6 \times 3^0 = 543$

→ 543 is used as the record address

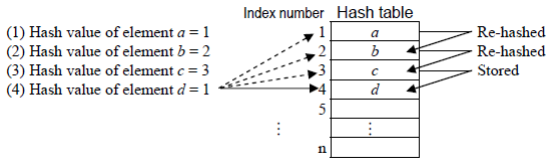
- With indirect addressing, different keys may be converted to the same record address and this phenomenon is called **synonym**. When this happens, the record that can be used with the converted address is called the *home record*, and the one that cannot be used is called the *synonym record*.



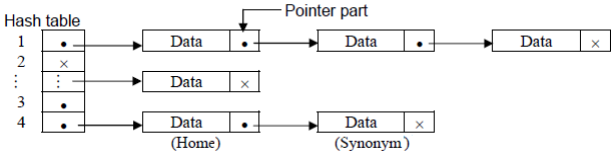
Occurrence of a synonym (with division by 97)

File Organization Formats

- There are two methods to deal with a synonym in indirect addressing:
  1) **Sequential method**. This stores the synonym record in an available area that is close to the calculated record address. Synonyms are prone to occurring in some cases.



  2) **Chain method**. This secures another memory area, and stores the synonym record in it. In this case, a pointer is required to indicate the address in which the synonym record is stored.

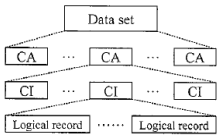


File Organization Formats

6. **VSAM Organization file** – can be used with an OS that uses a virtual memory system. This can also be described as a way of implementing conventional file organization format virtually. A file is called a data set.

Data set name	Corresponding file organization
ESDS (Entry Sequence Data Set)	Sequential organization file
KSDS (Key Sequence Data Set)	Indexed sequential organization file
RRDS (Relative Record Data Set)	Direct organization file / relative organization file

- Data set structure
  - CA (Control Area). This is a logical control area that manages partitioned memory areas.
  - CI (Control Interval). This is a logical input / output unit that groups multiple logical records.



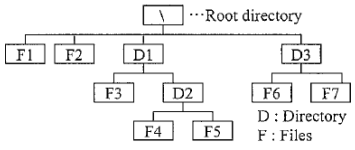
File Management in Small Computers

- **File System** – a mechanism that manages files that are stored on auxiliary storage media such as hard disk drive. A different file system is provided by each OS, and a typical example is the FAT (File Allocation Tables) file system that is used in Windows-family OSs.
  - In a FAT file system, a hard disk is called a *volume*, and a volume is divided into units called *clusters* (group of sectors) for management. File I/O is performed as a batch in units of clusters.
  - FAT32 is a typical file system, which uses a cluster identifier 32 bits to divide a volume into 2^{32} clusters.
  - Currently, NTFS (NT File System) is widely used because the size of HDD has become large and the maximum volume size of NTFS is designed to be large.
  - Other typical file systems except for the Windows OS:

File system name	Supported OSs	Notes
UFS (Unix File System)	UNIX-family	Divided into partitions
HFS (Hierarchical File System)	Mac OS	HFS+ is also used on iPods.

File Management in Small Computers

- **Directory Management**. The OS of a small computer manages files hierarchically in a tree structure. In UNIX, the whole file system is managed as a single tree. In this system, a *directory* is used to group multiple files in higher hierarchy.



Hierarchical structure diagram for files

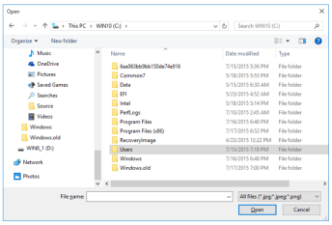
  - Files are referenced with the two methods below that specify the path (route):
    - Absolute path: specifies the route from the root directory
    - Relative path: specifies the route from the current directory

Example: In the figure above, what is the absolute path and relative path to specify F3 when the current directory is D2? Here, ".." indicates the parent directory. "\" indicates the root directory when it is at the top of a path and a separator when it is in the middle.

Absolute path: \D1\F3 (root directory → D1 → F3)
Relative path: ..\F3 (current directory D2 → parent directory D1 → F3)

File Management in Small Computers

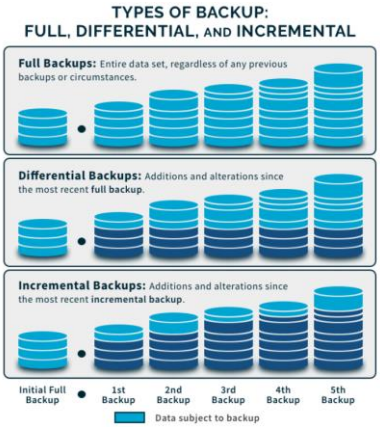
- **File Sharing** – a mechanism to enable multiple users to use a single file. In NAS (Network Attached Storage), file systems that support multiple protocols are embedded, so file sharing between different OSs or between different types of servers can easily be performed.
- **Symbolic Link** – a function that assigns a different name to a file or directory in the file system of a Unix-family OS. It allows a user or application to handle it in the same way as the actual file.
- **Folders**. In a Windows-family OS, file manipulation is also performed in a GUI environment. A file to be used is not specified as a path but is specified by selecting an icon. In this case, the name *folder* is used instead of directory. By selecting folders to switch windows, the user changes the current directory. Folders can be shared in the same way as files.



Backup

- Backup refers to saving data, such as files, on an auxiliary storage device in case of a fault or such other situation. Since data recorded in a file is very critical, it is necessary to make efforts to improve reliability and availability through backups.

- Types of backups:
  - **Full backup**. Copies all data. The time taken for the backup is long, but a restore can be performed only with the latest full backup data.
  - **Differential backup**. Copies all data that has been updated since the previous full backup. The restore for this uses only the latest full backup data and the latest differential backup, so it involves less work than an incremental backup.
  - **Incremental backup**. Copies only the data that has been updated since the previous backup (full backup and incremental backup). The time taken for the backup is shorter than a differential backup, but the restore uses the latest full backup and all the incremental backup data since the last full backup.



Backup

[Points for attention in backup operations]

- Backups should be performed periodically after a schedule is created. Especially in cases where there are multiple people involved, a plan should be created so that backup can be implemented outside of working hours.
- In order to prevent operational errors or neglecting to perform backup, backup processing should be automated as much as possible, which also makes it less laborious.
- Storage media that has plenty of capacity should be selected. However, if files are backed up onto the same HDD they cannot be read if a fault occurs, so a separate storage media should be used.
- Thorough consideration should be given to the importance of the files in terms of the storage location of the backup files. If the backup files are stored in the same room as the originals, they may also be lost if a fire or other disaster occurs. If it is a room for which entry / exit control is not performed, then important information may be obtained by a third party.
- In order to protect data from disasters, accidents, or such other situation, a *multiple backup* that copies twice, the main and the secondary, should be take and stored in a location that is geographically separate.
- Backup files should have generation information (e.g., version numbers), and management should be performed for as many generations as possible.

References

- Internet Source:
  - https://www.pcmag.com/encyclopedia/term/windows-ce
  - https://en.wikipedia.org/wiki/Fragmentation_(computing)
  - https://en.wikiversity.org/wiki/File:External_Fragmentation.svg
  - https://www.differencebetween.com/difference-between-internal-and-vs-external-fragmentation/
  - https://commons.wikimedia.org/wiki/File:File_system_fragmentation.svg
  - https://www.geeksforgeeks.org/phases-of-a-compiler/
  - https://docs.microsoft.com/en-us/windows/uwp/files/quickstart-using-file-and-folder-pickers
  - https://securityboulevard.com/2020/03/types-of-backup-understanding-full-differential-and-incremental-backup/
- Books:
  - Textbook for Fundamental Information Technology Engineers (Introduction to Computer Systems) 3rd Edition
  - FE Exam Preparation Book Volume 1, First Edition: January, 2008
  - New FE Textbook Vol.1 IT Fundamentals, First Edition: April, 2015