

Aunur R. Mulyanto

SOFTWARE ENGINEERING

Book 1

Vocational Education



Directorate Technical and Vocational Education
Directorate General of Management Education Basic and
Middle
Department of National Education
Copy Right@Department of National Education
All right reserve

SOFTWARE ENGINEERING

Book 1

For Vocational Education

Author : Aunur R. Mulyanto
Cover Design : Team
Book Size : 17,6 x 25 cm

MUL MULYANTO, Aunur R.
Software Engineering Book1 For SMK / by Aunur

R. Mulyanto ---- Jakarta : Directorate Technical and Vocational
Education
Directorate General for Basic and Middle Education
Department of National Education, 2008.
xiv. 135 pages
Libraries : A1-A2
Glosarium : B1-B6
ISBN : 978-979-060-007-2
ISBN : 978-979-060-008-9

Published by:
Directorate Technical and Vocational Education
Directorate General Primary and Secondary Education
Department of National Education
2008

FOREWORD

We thank to Allah SWT, blessing and gift of His mercy, the Government, in this case, the Directorate of Development of Vocational High School of Directorate General of Primary and Secondary Education Management, Ministry of Education, has conducted the activities of the book trade as a form of purchase of copyright books text for students learning vocational SMK. Because textbooks vocational very difficult to get in on the market. Text book lesson this has been through the process of assessment by the National Education Standards as a text book lesson to SMK and have been declared eligible to be used in the feasibility of the learning process through the Ministry of National Education Regulation No. 45 Year 2008 on 15 August 2008.

We delivered the award at a high level to all the writers who have been well switch copyright works to the Ministry of National Education to be used widely by teachers and students SMK. Text book lessons that have been transferred the copyright to the Ministry of National Education is, can be downloaded , duplicated, printed, media transfer, or photocopy by the community. But for the multiplication of commercial sales price must meet conditions set by the Government. With the soft copy of this show will be easier for the public especially teachers and students of vocational school (SMK) in Indonesia as well as abroad to access and utilize as a reference.

We hope all parties can support this policy. To the students good luck in your study and hopefully can use this book as well as possible. We realize that this book still needs to be improved quality. Therefore, suggestions and criticisms are we expected.

Jakarta, 17 Agustus 2008

Director of Technical and Vocational Education

EDITORIAL

With all humility, we give thanks to Allah SWT. Because only with patronage, blessing and gift of His mercy , then this book can be completed.

Book entitled 'Software Engineering' is organized to meet the needs handbook for students Vocational Education. Especially on the program expertise Software Engineering. This book includes a description refers to the standard of competence and basic competencies for Software Engineering students from SMK class X, XI to class XII.

Each chapter contains a theory must be understood correctly by students and accompanied by examples of problems that are relevant to the theory. In addition there is also a matter that is based on the concept and discussed the theory test as a tool to measure the ability of students in the control of these materials.

In developing this book, the author seeks to be the material that is presented in accordance with the needs of competency to be achieved. Therefore, apart from the result of thought and experience as writers and practitioners of Software Engineering, which developed the material with other appropriate reference

In this moment I wish to thank to all parties that support this book can be published. Hopefully this book can be useful for students in developing capabilities. The author realized that this book still needs to be developed continuously, so that the suggestions from various parties, this book is useful.

The Author,

Table of Contents

FOREWORD.....	iv
EDITORIAL.....	vi
HOW TO USE THIS BOOK.....	xi
CHAPTER 1 INTRODUCTION.....	1
1.1 SOFTWARE ENGINEERING CONCEPT.....	2
1.2 SOFTWARE ENGINEERING OBJECTIVES.....	2
1.3 SCOPE.....	3
1.4 SOFTWARE ENGINEERING AND COMPUTER SCIENCE	3
1.5 SOFTWARE ENGINEERING AND OTHER DISCIPLINE.....	5
1.6 SOFTWARE ENGINEERING DEVELOPMENT.....	6
1.7 PROFESSION AND CERTIFICATION.....	7
1.8 SOFTWARE ENGINEERING AND PROBLEM SOLVING.....	7
1.8.1 The Problem and The Symptom.....	7
1.8.2. Type of Problem.....	9
1.8.3. Problem Solving.....	9
1.9.SUMMARY.....	11
1.10. EXERCISE.....	12
CHAPTER 2 SOFTWARE ENGINEERING METHOD.....	13
2.1 SOFTWARE ENGINEERING PROCESS MODELS.....	14
2.1.1.The Waterfall model.....	15
2.1.2 Prototyping model.....	16
2.1.3 Unified Process and Unified Modeling Language	17
2.2. STAGES IN SOFTWARE ENGINEERING.....	20
2.2.1. Analysis.....	20
2.2.2. Design.....	25
2.2.3. Construction.....	26
2.2.4. Testing.....	26
2.2.5. Maintenance and Configuration.....	26
2.3. SUMMARY.....	27
2.4 EXERCISE.....	27
CHAPTER 3 ELECTRONICS AND COMPUTER SYSTEMS	28
3.1. BASIC ELECTRONICS.....	28
3.1.1. Concept of Basic Electronics.....	28
3.1.2. Electronics Components.....	30
3. 2. DIGITAL ELECTRONICS.....	31
3.2.1. Concept of Digital Electronics.....	31
3.2.2. Logic Gate.....	31

3.2.3. Digital Circuit	33
3.3. COMPUTER SYSTEM	34
3.1.1 Hardware.....	34
3.1.2 Software.....	38
3.4. SUMMARY.....	41
3.5. EXERCISE	42
CHAPTER 4 OPERATING SYSTEM.....	43
4.1. Operating System Concept.....	44
4.2. TYPES OF OPERATING SYSTEM	50
4.2.1. DOS	50
4.2.2. UNIX	51
4.2.3. Microsoft Windows	54
4.2.4. Apple Mac OS	56
4.2.5. Linux	57
4.3. PREPARING AND OPERATING AN OPERATING SYSTEM	58
4.3.1. Installation.....	59
4.3.2. Booting	62
4.3.3. Text based commands.....	63
4.3.4. Work with GUI	76
4.4. WORK IN COMPUTER NETWORK	77
4.4.1. Preparation.....	77
4.4.2. Configuring network connection.	80
4.4.3. File, printer and resource sharing	81
4.5. SUMMARY.....	84
4.6. EXERCISE.....	84
CHAPTER 5 BASIC PROGRAMING ALGORITHM.....	86
5.1. VARIABLES, CONSTANTS, AND DATA TYPES.....	87
5.1.1. Variables.....	87
5.1.2. Constants.....	88
5.1.3. Data Type.....	88
5. PROGRAMING ALGORITHM STRUCTURE.....	94
5.2.1. Algorithm Concept.....	94
5.2.2. How to write Algorithm.....	95
5.2.3. Sequential Algorithm.	98
5.2.4. Branching Algorithm.....	100
5.2.5. Looping Algorithm	107
5.3. ARRAY MANAGEMENT.....	114
5.3.1. Array Concept.....	114

5.3.2. Looking for data in an Array	115
5.3.3. Ordering data in an array.	117
5.4. FILE OPERATION	119
5.4.1. Algorithm to write data to a file	119
5.4.2. Algorithm to read data file	120
5.5. SUMMARY.....	121
5.6. EXERCISE.....	121
REFERENCES.....	124
APPENDIX 1 GLOSARY.....	126
APPENDIX 2 SITE LIST.....	131
APPENDIX 3 FUNCTIONS BUILD-IN ON VISUAL BASE.....	133

HOW TO USE THIS BOOK

A. Global Description

This book is given the title "Software Engineering", together with one of the program expertise in the Vocational School (SMK). However, the actual contents of this book is not specifically discuss on Software Engineering. From the perspective of Computer Science field of five sub-areas covered in this book, the sub-field of Engineering Software, Operating System, Algorithm and Data Structure, Programming Languages and Database. This curriculum tailored to the level of expertise SMK Program for Software Engineering.

Main subject of Engineering Software in general, describe the basics of Software Engineering, and solving problems, and methods of software development. The discussion about the sub-field of Operating System contains a computer system, the system operating and working in the computer network. The scope of the material basic algorithms include algorithms and advance algorithms. Sub field Programming Languages take a big portion, including GUI programming with VB & VB.Net, Java programming, C + + programming, Programming of object-oriented and web-based. Sub-sector which is the last part of this book is a data base with the scope of the system databases, conceptual modeling, relational database, Microsoft Access and SQL.

B. Competency Map

In general, this book refers to the Standard and Competence Basic Competency (SKKD) for Vocational School (SMK) as follows :

1. Using a basic level of programming algorithm
2. Using advanced programming algorithm
3. Operate the database application
4. Create an application-based Microsoft Access
5. Mastering the basic techniques electronics
6. Mastering the digital electronics technology
7. Create file with HTML according to specifications
8. Applying the basics of creating a basic level of static web
9. Make the application using VB and VB.NET
10. Make the software application package
11. Doing programming data description (Structured-SQL Query Language) basic level
12. Operates programming data description language (SQL) advanced

13. Create a web page dynamic basic levels
14. Create a web page more dynamic level
15. Create a web application program using JSP Software Engineering
16. Make the application database using XML
17. Make a database program using Microsoft (SQL Server)
18. Make a database program using PL / SQL (Oracle)
19. Make the application using C + +
20. Explaining the system peripherals
21. Make the program in object-oriented programming language
22. Make the application using Java
23. Operate the computer operating system and text-based GUI

In this book, chapters not compiled based on SKKD, but the materials are developed based on the sequence of basic subject. So that in some chapters contain a mix of several standards of competence. Or a basic competency may not be on the competency standards groups such as SKKD on the list, but is on the other sub-chapters.

SKKD suitability and content of the chapter can be seen in the table below :

Competency Code	Competency	Relation
ELKA-MR.UM.001.A	Mastering the basic techniques electronics	3
ELKA.MR.UM.004.A	Mastering the digital electronics technology and Computer	3
TIK.PR02.001.01	Using a basic level of programming algorithm	5
TIK.PR02.002.01	Using advanced programming algorithm	6
HDW.OPR.103.(1).A	Operate the computer operating system and text-based	4
HDW.OPR.104.(1).A	Operate the computer operating system based GUI	4
TIK.PR02.020.01	Operate the database application	10 and 11
TIK.PR08.004.01	Create an application-based Microsoft Access	11
TIK.PR08.024.01	Create file with HTML according to specifications	13
TIK.PR08.027.01	Applying the basics of creating a basic level of static web	13
TIK.PR08.003.01	Make the application using VB and VB.NET	7
TIK.PR02.016.01	Make the software application package	7
TIK.PR03.001.01	Doing programming data description (Structured - SQL Query Language) basic level	12

TIK.PR03.002.01	Operates programming data description language (SQL) advanced	12
TIK.PR04.002.01	Create a web page dynamic basic levels	13
TIK.PR04.003.01	Create a web page dynamic advance levels	13

Competency Code	Competency	Relation
TIK.PR02.009.01	Operate the program in object-oriented programming language	8
TIK.PR08.012.01	Make application program using Java	8
TIK.PR08.001.01	Make application program using C++	9
TIK.PR06.003.01	Describe Peripheral system	3
TIK.PR08.005.01	Make database using PL/SQL	10 and 12
TIK.PR08.006.01	Make database program using SQL server	12
TIK.PR08.008.01	Make JSP based we application web program	14

C. How to use this book

This book is specifically aimed at students and teachers for the vocational school (SMK) expertise in RPL. However, this book is also open for general readers who are interested in the RPL, Algorithm and Programming, Database and the Internet. For students, this book can be a handbook, because this book is to provide learning materials that are complete enough for the subjects for three years at the school. Some parts of this book may need books to help enrich more insight and capacity building. Whereas for teachers, this book can be used as reference books to prepare teaching modules for the students.

This book is structured in such a way so that students can independently learn and encourage to try. Therefore, in this book, many will be found either in the form of illustration image, the scheme and program listings. This is so that students can easily understand the explanation or the application of a particular concept. The chapter will be ended with exercise of the subject.

1.1 SOFTWARE ENGINEERING CONCEPT

The Software Engineering term began to be popularized in 1968 at Software Engineering Conference that was held by NATO. Some people interpret Software Engineering to be limited on how to make a computer program. In fact, there are basic differences between software and computer program.

Software is all the command that can be used to process information. Software could take the form of the program or the procedure. The program was the collection ordered that was understood by the computer whereas the procedure was the order that was needed by the user in processing information (O'Brien, 1999). The software engineering is defined as follows:

A body of knowledge that discussed all the aspects of the production of software, starting from the early stage that is the analysis of the requirement for the user, determined the specification from the requirement for the user, the design, coding, the testing to the maintenance of the system after being used.

It is clear that Software Engineering is not only related to the production method of the computer program. The statement "all aspects of production" in the above definition, has the meaning of all the matters that are connected with the process of the production like project management, the determination of the personnel, the budget of the cost, the method, the schedule, the quality up to the training of the user was a part of Software Engineering.

1.2 SOFTWARE ENGINEERING OBJECTIVES

In general, software engineering objectives are fairly similar to other engineering fields. Let us examine Figure 1.2.

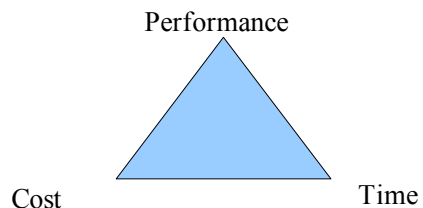


Figure 1.2. Software Engineering Objectives.

Figure 1.2 shows that an engineering field will always try to produce the highest performance output at the lowest possible cost at exact time. The software engineering aims for,

3 Software Engineering

- Lower software production cost.
- High performance and reliable software in time.
- Multi platform software.
- Low maintenance cost.

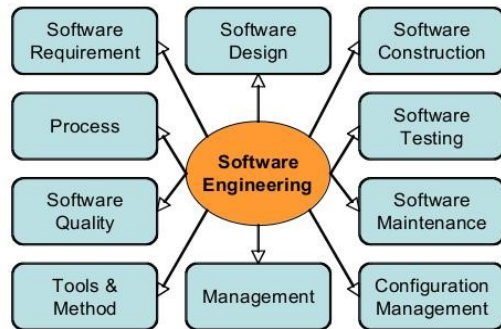


Figure 1.3. Scope of Software engineering (Abran et.al., 2004)

1.3 SCOPE

As defined above, the scope of software engineering is as follows,

- Software requirements related to the requirement specifications of the software.
- Software design includes software architecture determination, software components, interface, as well as other software characteristics.
- Software construction relates to the software development, including the algorithm, coding, testing, and debugging.
- Software testing covers software behavior evaluation and testing.
- Software maintenance Includes maintenance efforts as software is operated.
- Software configuration management is related to the configuration of software to satisfy a certain requirement.
- Software engineering management related to the management and the grating software engineering, including planning of the software project.
- Software engineering tools and methods include the theoretical study on aids and the software engineering method.
- Software engineering process is concerned with the definition, the implementation, the grating, the management, the change and the improvement of the process software engineering.
- Software quality is stressed on the quality and the software life-cycle.

1.4 SOFTWARE ENGINEERING AND COMPUTER SCIENCE

4 Software Engineering

Computer Science was born at the beginning of the 1940 's that was the integration from the theory of the algorithm, mathematical logic and the discovery of the storage method of the program electronically to the computer. Since then computer science had experienced continuous development and broadening.

The scope of knowledge in computer science often described as a systematic study in processes of the algorithm that explained and transformed information (Denning, 2000). It includes the theory, the analysis, the design, efficiency, the application and it application.

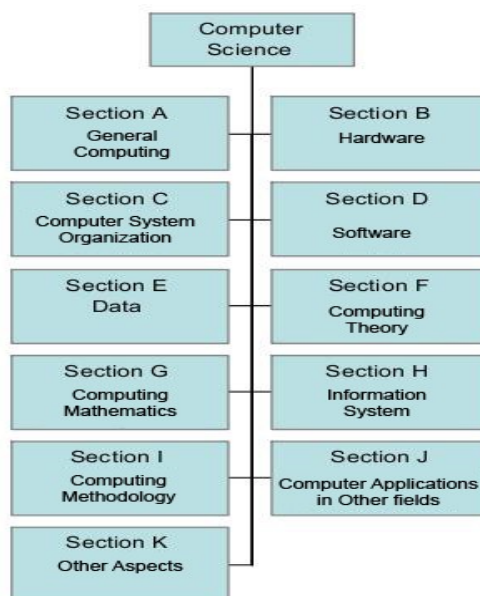


Figure 1.4. Computer Science Classification based on ACM (1998).

There are several branch of knowledge in the computer science discipline as seen in the Figure 1.4, 1.5 and 1.6.

Based on Denning's (2000) and Wikipedia's (2007), software engineering was the subsector of computer science that was equal to the other subsector. Whereas according to ACM (Association for Computing Machinery), software engineering is part of Section D (Software). Although being seen separated, in its application, the subsector software engineering always needed the support from the other subsector, especially algorithm and data structure, programming language, database, operating system and network, and information system.

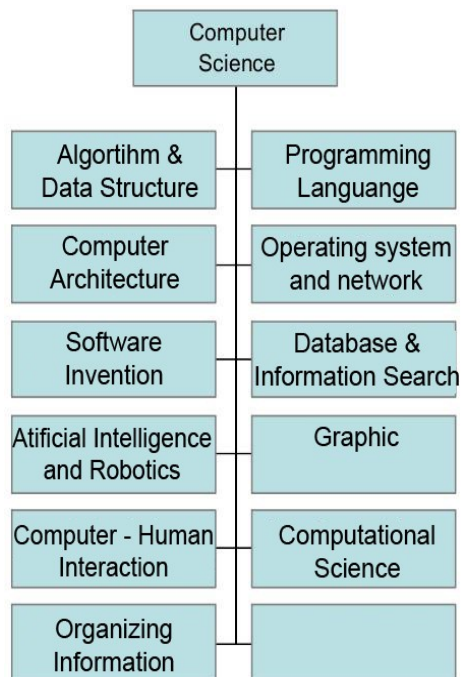


Figure 1.5. Computer Science Classification Discipline based on Denning (2000).

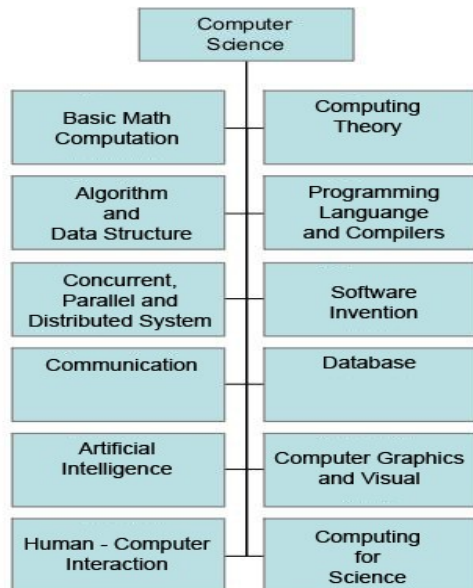
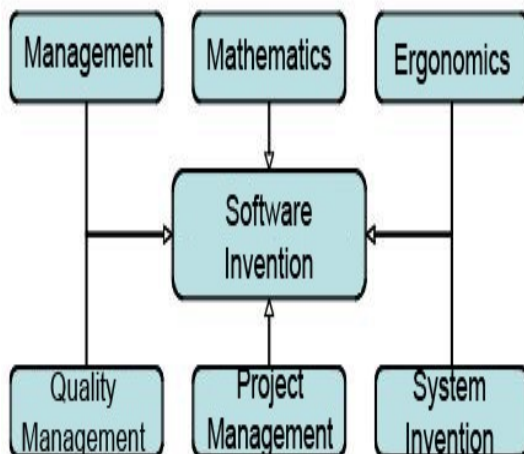


Figure 1.6. Computer Science discipline based on Wikipedia (2007).

1.5 SOFTWARE ENGINEERING AND OTHER DISCIPLINE

Since the scope of software engineering is quite wide, it relies heavily to other fields.

Not only with other sub sector in computer science but also other fields outside computer science. Software engineering relations with other fields can be seen in Figure 1,7.



Picture 1.7. Software Engineering relations with other fields

- Management field covers accounting, financial, marketing, operation management, economics, quantitative analysis, human resources management, policy and business strategy.
- mathematical field covers linear algebra, calculus, probabilistic, statistics, numerical analysis and discrete mathematics.
- Project management field covers project related matters, such as, project scope, budgeting, human resource, quality control, risk management, and project scheduling.
- Quality management field covers quality system development, risk and reliability management, quality improvement, and quantitative methods.
- Ergonomics field covers man and machine interaction.
- System engineering covers system theory, cost analysis, modeling, simulation, business process and operation.

1.6 SOFTWARE ENGINEERING DEVELOPMENT

Despite its early appearance in 1968, software engineering had the quite long history. Figure 1.8 presented the essence of the software engineering development. From the science side of discipline of knowledge, software engineering is relatively young and is continue to develop. Today, the direction of the development is as follows,

Agile Software Development, Experimental Software Development, Model-Driven Software Development and Software Product Lines.

Year	Event
1940	The first computer that allowed user to write the program code directly.
1950	Early generation of interpreter and macro language
	First generation of compilers
	Second generation of compilers
1960	Commercialization of mainframe computers
	The development customized software
	Software engineering concepts is being adopted
1970	Software development applications
	Commercialization of minicomputers
1980	Commercialization of personal computers
	Increase in software demand
	Object Oriented Programming (OOP)
1990	Agile process and extreme programming

	Drastic increase in memory capacity
	Increase in Internet users
2000	Modern interpreter platform (Java, .Net, PHP, etc).
	Outsourcing

Figure 1.8. Software Engineering Development

1.7 PROFESSION AND CERTIFICATION

Software Engineer profession is fairly new for Indonesians. Most Indonesians possibly more familiar with the term Information Technology expert, Information System Analyst, Operator or the other term. It is due to the confusion on the term software engineer as described in the early of the chapter. However in countries with mature information technology, Software Engineer term is more often used.

Software engineer certification is still debatable among expert and software vendor. Most certification in software industry is usually product specific. For example, the software company like Redhat Linux Inc., Adobe Inc., Oracle, or Microsoft, give certification to those who master their product.

ACM (Association for Computing Machinery) had run Software Engineer certification program in the 1980. Unfortunately, they have to discontinue due to lack of interest. IEEE (Institute of Electrical and Electronics Engineers) has Issued more than 500 certificates of software profession. Canada has issued a legal certificate for software engineer that was known as ISP (Information Systems Professional).

At this time, there is no software engineer certification in Indonesia. However, the National Competence for Computer Programmer has been defined. Although, it has not fully covered computer programmer field, it can be used for early approach to software engineer certification.

1.8 SOFTWARE ENGINEERING AND PROBLEM SOLVING

Software engineering had conceptually close to problem solving principles. Understanding on the problems, the strategy and the process as well as the system approach in the solution to the problem will help the software engineering processes.

1.8.1 The Problem and The Symptom

A problem can be seen as the difference between the condition that happened and the condition that it was hoped. It might also be interpreted as the difference between the condition now and the aimed condition. For example, a student hoped to receive 80 in a C++ Programming exam, but in fact he only received 60. The existence of this difference showed the existence of a problem.

Often it is difficult to distinguish between the symptom and the problem. The symptom is a sign of the occurrence of a problem. Pay attention to a person who is a medical doctor professional as shown in Figure 1.9. A doctor in treating the patient's illness will ask the symptom / the signs that were felt by the patient.



Figure 1.9. Medical Doctor Profession

What is the relationship between problem and symptom with Software Engineering? As being said earlier in the chapter, software that was results of Software Engineering to create an air to complete certain task or to solve certain problem. If we did not learn correctly the problem, it is impossible to determine how solve it. Thus, knowing well the problem as well as knowledge on the symptom of the problem would be very important.

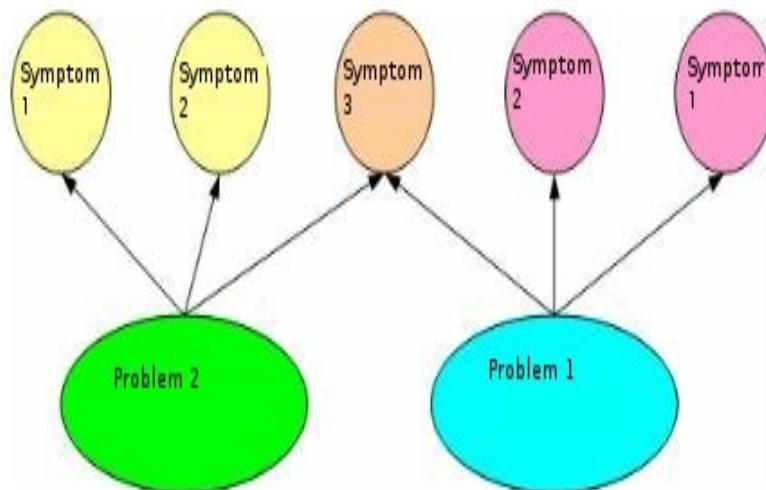


Figure 1.10. Symptom and Problem.

1.8.2. Type of Problem

Problem can be classified as shown in Figure 1.11.

- Problem in standard fulfillment. The problem in this group is related to the achievement of the standard that was determined in an organization. It usually relates to the long term objectives of the organization.

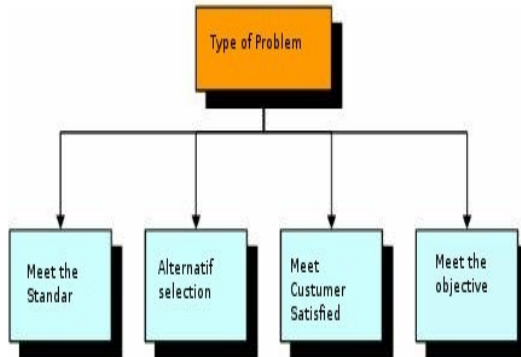


Figure 1.11.Type of Problem (Deek et.al., 2005)

- Problem in alternative selection. The problem in this group is in choosing the best solution from various alternatives based on certain criteria. This problem is often encountered in daily life, such as, choose the exact school, choose the residence location, choose field of work. The alternative and the criterion had the weight that was agreed to.
- Problem in customer's satisfaction fulfillment. In profit oriented organizations, problem in customer's satisfaction fulfillment is often emerged. The customers may have many wishes and one may be highly different than others. Fulfill all customer's wishes may not possible and very incriminating an organization. One must look for the solution that benefit both the customer and the organization.
- The problem in goal achievement. This type of problem resembles the first type of problem, problem in standard fulfillment. The difference, this type of problem is in achieving short term goals that might not be fixed can be altered in a short time.

1.8.3. Problem Solving

Problem solving is a process where a situation was observed and after the problem found, a solution is made by determining the problem, reducing or eliminating the problem or preventing the problem from happening. There were many steps of problem solving proposed by the experts, one of them can be seen in the Figure 1.12.

Figure 1.12 shows a series of different process stage that could be used in various stages, depended on the type and the characteristics of the problem. Different problem may need different method, even possibly the different stages. The critical stage in problem solving is in defining the problem. If the problem not clearly defined then the stages may be along and difficult to undertake. Moreover if it being forced, problem solution may not likely be found.

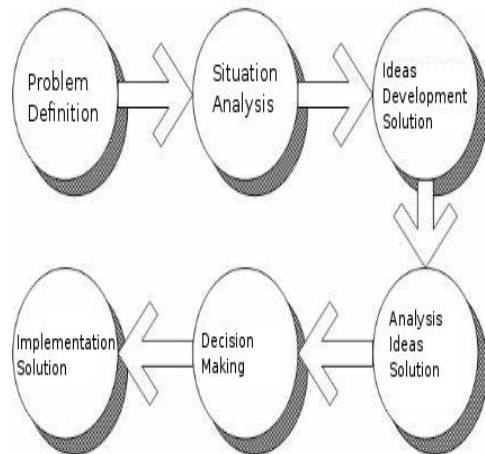


Figure 1.12. Problem Solving Processes (adopted from Deek et.al., 2005)

In general, problem solving may be carried out in four (4) main stages, namely,

- Understand and defining the problem. This stage is critical and the starting point of any problem solving processes. The aim of this stage is to get clear picture of the problem as well as to eliminate unimportant sections.
- Planning for problem solving. This stage is normally split into two (2) main section, namely,
 1. Looking for various ways to solve the problem.
 2. Planning to solve the problem.

There may be not one but several solutions to a problem. As an illustration, if we were in Surabaya and wanted to go to Jakarta, there are several options that can be used either by land, sea or air transportation. Over land transportation, train, bus or other transportation can be used. We can use the north, middle or south route. Thus, plenty of solutions can be used by us. Each has its own characteristics. However, we must choose one that meet certain conditions that would be best for our problem. After being chosen, then we could make the plan for problem solving and divide the problem into smaller parts. The rough plan for problem solving should only contain the main stages of problem solving.

- Draft and apply the plan to get the appropriate solution. In this stage, the rough plan is refined and clarified into details of more complete of problem solving plan.

- Check and provide results of the problem solving. The purpose of this stage is to check the resulting accuracy of the chosen method whether fulfill the objectives. Moreover, we can evaluate the chosen method usefulness.

1.9.SUMMARY

- **Software** is the command / tool that can be used to process information.
 1. **A program** is a collection of command that can be understood by computer.
 2. **A procedure** is the command needed by the users in information processing.
- Software Engineering is a body of knowledge that discuss all aspects of software production, starting from the early stage, namely, analysis of user requirement, determined the user requirement specification, design, coding, testing up to system maintenance after being used.
- The objective of Software Engineering is to produce high performance and high quality software, right on time, at low cost, and the multi-platform.
- Software Engineering is part of computer science that require support from other field of computer science as well as other field of knowledge.
- Software Engineering Certification is not readily available. However, other programmer certification be used.
- **A problem** can be seen as the difference between the condition that happened and the condition that it was hoped. **A Symptom** is the sign of problem.
- Type of problem:
 1. Problem in standard fulfillment.
 2. Problem in alternative selection.
 3. Problem in customer's satisfaction fulfillment.
 4. The problem in goal achievement.
- **Problem solving** is a process where a situation was observed and after the problem found, a solution is made by determining the problem, reducing or eliminating the problem or preventing the problem from happening.
- Main stages in problem solving:
 1. Understand and defining the problem.
 2. Planning for problem solving.
 3. Draft and apply the plan to get the appropriate solution.
 4. Check and provide results of the problem solving.

1.10. EXERCISE

1. Describe software and software engineering.
2. What are the difference between the computer program and the procedure?
3. Name five sub-field of computer science based based on Denning's.
4. Name five field that closely related to software engineering.
5. Describe symptom and problem?

CHAPTER 2 SOFTWARE ENGINEERING METHOD

As we worked with the computer shown in Figure 2.1., we need a series of stages and certain methods in order to be able to produce aim for our objectives. Likewise in the software engineering, it needs certain stages to be productive. A successful software engineering needs not only strong computation capacity like algorithm, programming, and database but also strong, but also determination in meeting the goal, resolution method identification, development method, goal identification, resources requirement identification, and other factors. Such matters are related to software engineering methods.



Figure 2.1. Work With Computer

The content of this chapter not within the standard Software Engineering competence. However, the writer think it would be beneficial to know how to carry out software engineering and methods that normally being used. Several part of this chapter will likely difficult to understand, teacher may be needed to explain. The summary of the chapter is written at the end of the chapter.
(Source: Microsoft Office 2007 Clip Art)

OBJECTIVES

After you learn this chapter, you should be able:

- Understand the general characteristics of the process model in software engineering.
- Name several models of the software engineering.
- Know principles from the method waterfall, prototyping, and unified process
- Understood the stages in the software engineering.

2.1 SOFTWARE ENGINEERING PROCESS MODELS

There are many models were developed in Software Engineering to help the process of software development these models generally refer to the system development process model that was known as System Development Life Cycle (SDLC) as shown in Figure 2.2.

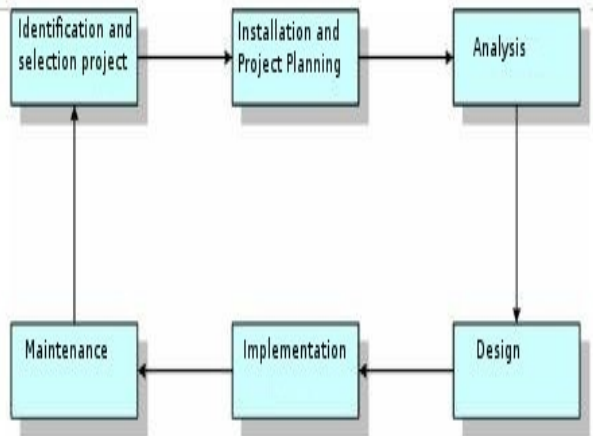


Figure 2.2. System Development Life Cycle (SDLC).

Each developed model has its own characteristics. However, in general, these models will fall into the following categories, namely:

- *Need for clear definition of the problem.* Main input in each software development model is in clear definition of the problem. The clearer the definition the better for problem solving processes. Thus, understanding the problem as explained in Chapter 1, is an important part of a software development model.
- *A structured development stages.* Although software development models may have different pattern, usually these models followed a general pattern of analysis – design – coding – testing - maintenance.
- *Stakeholder plays a very important role in the whole of the development stage.* Stakeholder in the software engineering may be the user, the owner, the developer, programmer and people who are involved in the software engineering.
- *Documentation is an important part of software development.* Each the stage in the model usually produces several articles, the diagram, the picture or other forms that must be documented and must not separated from the produced software.
- *The output of software development processes must have an economic value.* Although the value of software may be difficult to measure in terms of money. The effect of software usage give an added value to the organization. This could be in form of declining operating cost, efficient use of resources, increase in profit, improving organization image, etc.

There were many software development models, including The Waterfall Model, Joint Application Development (JAD), Information Engineering (IE), Rapid Application Development (the COUNCIL) including Prototyping, Unified Process (UP), Structural Analysis and Design (SAD) and Framework for the Application of System thinking (FAST). This book will discuss three (3) development models, namely, The Waterfall Model, Prototyping, and Unified Process (UP).

2.1.1. The Waterfall model

The life-cycle model is the main model and the foundation of many models. One of the models that is globally known in software engineering is The Waterfall Model. There are five (5) main stages in The Waterfall Model as shown in Figure 2.3. Known as waterfall since the process stage diagram resembled the stratified waterfall.

The brief stages in The Waterfall Model are as follows:

- Investigation stage is to determine whether a problem happening or is there an opportunity to develop an information system. In this stage the feasibility study must be carried out to determine whether the developed information system would be an appropriate solution.
- Analysis stage aim to look at the user and organization requirement as well as analyzing the available condition before the application of the new information system.
- Design stage determine the detail specification of the information system components, such as, human, hardware, software, network and data, and information products that in accordance with results of the analysis stage.
- Implementation stage is to get or to develop hardware and software (program coding), carried out testing, training and migration to the new system.

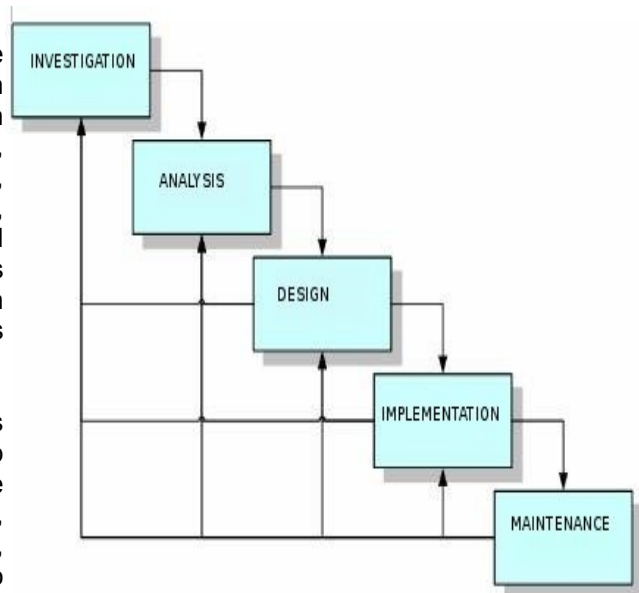


Figure 2.3. The Waterfall Model

- Maintenance stage carries out during the operation of the information system. In this stage, monitoring process, evaluation and improvement when being needed are carried out.

2.1.2 Prototyping model

Prototyping is one of software engineering approach that directly demonstrated how a software or software components will work in its environment before the actual construction stage is carried out (Howard, 1997).

Prototyping model could be classified to several types as shown Figure 2.4.

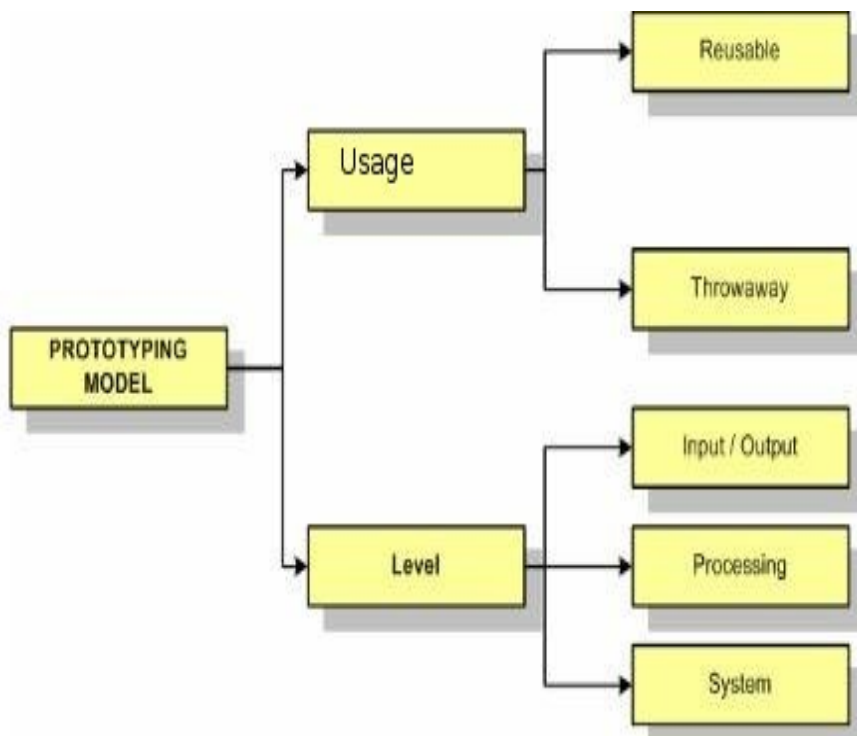


Figure 2.4. Classification of prototyping model (Harris, 2003)

- Reusable prototype: Prototype that will be transformed to the final product.
- Throwaway prototype: Prototype that will be thrown away as it completes its job.
- Input/output prototype: Prototype that was limited to the user interface.
- Processing prototype: Prototype that covered the maintenance file the

foundation and processes of the transaction.

- System prototype: Prototype that took the form of the complete model from software.

The stages in prototyping is normally an accelerate stages. The main strategy in prototyping is to build the easiest first and deliver to the user as soon as possible. Harris (2003) divided prototyping in six stages as being seen in the Figure 2.5.

The stages could be briefly explained as follows:

- Identification of the candidate prototyping. The candidate in this case covers user interface (menu, dialogue, input and output), main transaction file, and functions of basic processing.
- Engineering of prototype with help software like word processor, spreadsheet, database, graphics processor, and CASE (Computer-Aided System Engineering) software.
- Test prototype to confirm prototype able to undertake demonstration purposes.
- Prepare prototype USD (User's System Diagram) to identify parts of software that would be prototyped.
- Evaluate prototype with the user and carried out changes if being needed.
- Transform prototype into fully operated software and remove unnecessary codes, add needed programs, repeatedly improvement and test the software.

2.1.3 Unified Process and Unified Modeling Language

Unified Process (UP) or sometimes known as Unified Software Development Process (USDP) is the framework of development processes that use-case-driven, focus on software architecture, iterative and easy to grow (Alhir, 2005).

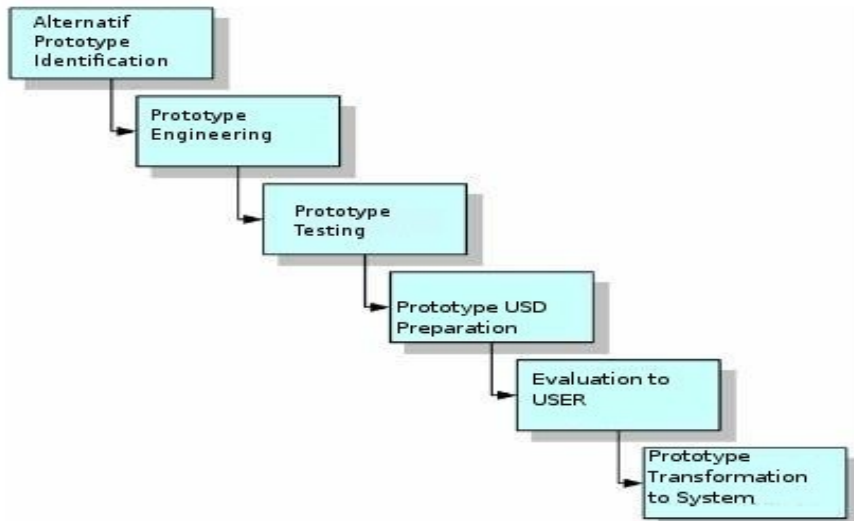


Figure 2.5. Prototyping model stages (Harris, 2003)

This framework is a relatively new in software development methodology. UP could be applied in various project scale, from small up to large scale.

The UP life-cycle generally will appear as shown in Figure 2.6. This chart is known as “hump chart”. Shown in the chart is the four (4) development stages, namely, inception, elaboration, construction and transition. Moreover, several activities must be carried out during the software development process, that is, business modeling, requirements, analysis and design, implementation, test, deployment, configuration & change mgmt, project management, environment. The stages and activities should be done in iterative manner (Ambler, 2005).

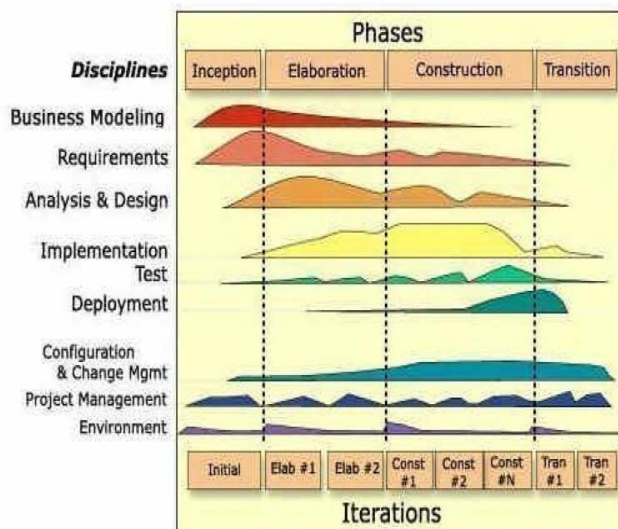


Figure 2.6. RUP Life Cycle (Ambler, 2005).

Short description of the four (4) stages in UP are as follows:

- Inception. This is the earliest stage that assess the carried out software project. It aims to get an agreement from stakeholders on its goal and project funding.
- Inception. This is the earliest stage that assess the carried out software project. It aims to get an agreement among the stakeholders on the objectives and project funding.
- Elaboration. The stage's objective is to get the picture on the general requirement, the condition and its main functions. This is important to know project risks, including software architecture risk, planning risk, and implementation risk. Despite its early stage, software engineering activities including business modeling, requirements, analysis and design are performed through iterative processes.
- Construction. This stage's objective is to construct the software to be used. The focus of this stage is to determine the level of priority in its requirement, its specification, in-depth analysis, the solution design to meet the requirement and the condition, coding and testing of the software. If possible, do beta testing to get early input from the user.
- Transition. The focus of this stage is to deliver the software to end user. Software would officially tested by both competent beta tester as well as the end users. Several activities, such as, data center migration, end user and staff training, should be performed in this stage.

In software development based on UP, **UML (Unified Modeling Language)** is normally used. Although UP requires the use of UML, UML can be use in various methods including other field out side information system field. UML is a standard modeling language and a collection of modeling techniques to specify, to visualized, to construct as well as to document the work during software development (Fowler, 2004). UML was born from in merging of many object oriented graphics based modeling languages developed in the end of the 80's and early 90's.

UML is simply used to draw the sketch of the system. The developer used UML to send several software aspects through graphic notation. UML defined notation and the semantics. Notation is a collection special forms that has certain meaning to depict various diagrams of software and the semantics defined how these forms could be combined. There are several kind of diagrams that are provided in UML, including:

- Use-case diagram. This diagram is used to show the use-software interaction.
- Activity diagram. This diagram is used to show the behavior of the software.
- Class diagram. This diagram is used to show class, feature, and relations. In

this diagram, object oriented approach plays an important role.

- Sequence diagram. This diagram is used to show the interaction among objects with the emphasis on process sequence or events.
- State machine diagram. This diagram is used to draw how an event change an object in its lifetime.
- Component diagram. This diagram is used to show the structure and the component connection.

2.2. STAGES IN SOFTWARE ENGINEERING

As described, despite different approaches, models have some similarities in using the following pattern, namely, stage analysis – design – coding (construction) – testing – maintenance.

2.2.1. Analysis

System analysis is a problem solving technique that break a system into smaller components to examine or to evaluate the component performance and its interaction to reach its objectives.

The analysis may be a crucial part of software engineering processes. The next processes will be highly depend on the analysis results. The brief stages in software engineering analysis is shown in Figure 2,7.

One of the most important part that usually carried out in the analysis stage is the business process modeling. **Process modeling** is a model that focused in all processes in the system that transform data to information (Harris, 2003). Process modeling shows data flow that enter and exiting in a process. This model is usually depicted the Data of Flow Diagram (DFD). DFD describe man, process and procedure interaction in transforming data into information.

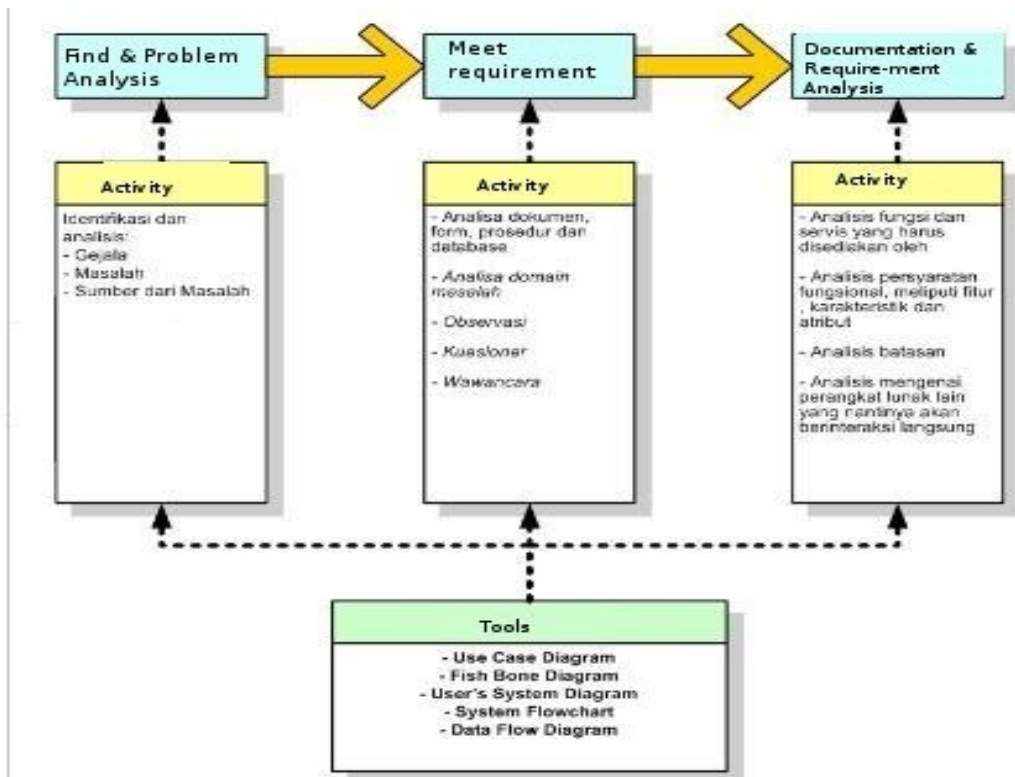


Figure 2.7. Stages and Activities in analysis.

There are generally four notations that often were used in DFD as shown in Figure 2.8.

External Entity symbolized the source of the data or the information recipient (the destiny from the data). External entity examples include the consumer who ordered a product, the manager who evaluated the report on the weekly sale, and so on.



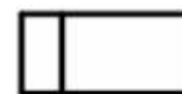
External Entity

Process is a series of stages to do data manipulation, such as, collection, sequencing, selection, reporting, and analysis etc.



Process

Data store is the place to store the data prior to use. The name used in the data store is the abstraction of the stored data. However, details, items, how to access and how it organized is not explained in this notation.



Data Store

Data flow shows the flow of data from one place to another. This data migration could be from external entity to a process, between processes, from a process to data store. In the drawing, each data flow must be given a label to identify the data.



Figure 2.8. Notation in DFD.

In creating DFD, there are several stages to be carried out in sequence. Figure 2.9. shows the sequencing of the stage.

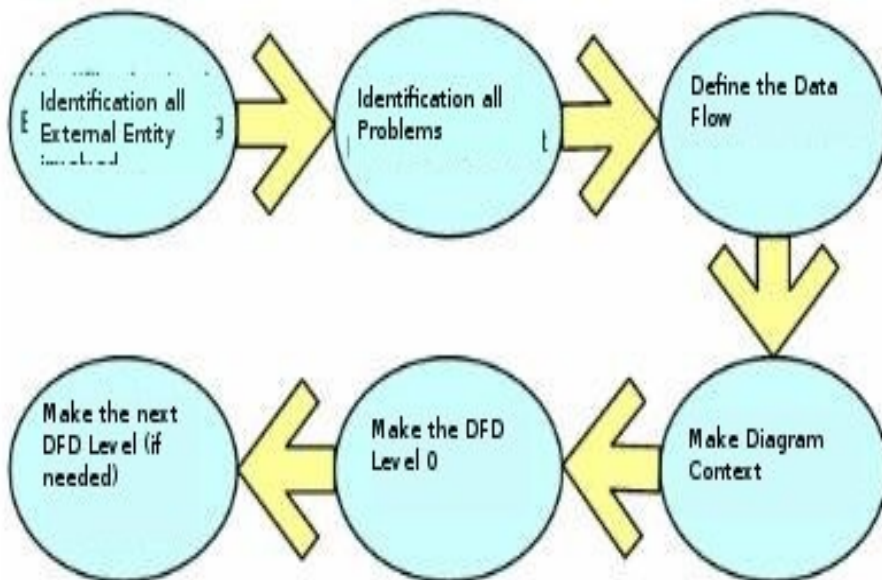


Figure 2.9. Stages in creating DFD.

Context the diagram is DFD scope of the system to show system boundaries, external entities that interact with the system and the main data flow between external entity and the system. Context diagram shows overall system as a single process. Figure 2.10 shows an example of context diagram.

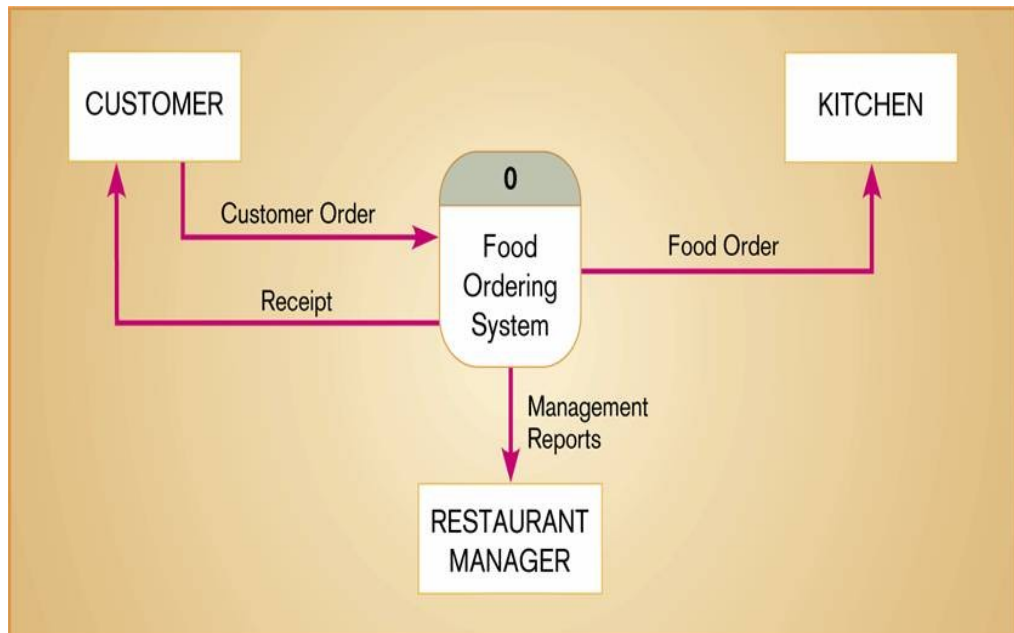


Figure 2.10. Context diagram of food ordering system (Hoffer et.al., 2002).

Context diagram in Figure 2.10 shows a single process to represent the modeled system. The process is given notation 0 to show this is the most abstract level of the system. Moreover, there are three (3) external entities, namely, customer, kitchen and restaurant manager. The three of them could play as data source (in the above example is customer) or as the information recipient (in the above example is customer, kitchen, and restaurant manager). Data flow is apparent in the picture shown to have one data flow enter the system and have three (3) data flows out of the system. Data flow is labeled to show what data is flowing, respectively.

After context diagram is correctly formed then detailing context diagram in DFD Level 0 would be the next step. DFD Level 0 is DFD that represents processes, data flow and main data storage in the system. DFD Level 0 this will be used as the foundation to build DFD below it (Level 1, 2, 3,... etc.) or normally known as decomposition DFD. Figure 2.10 is DFD level 0 from context diagram in Figure 2.11.

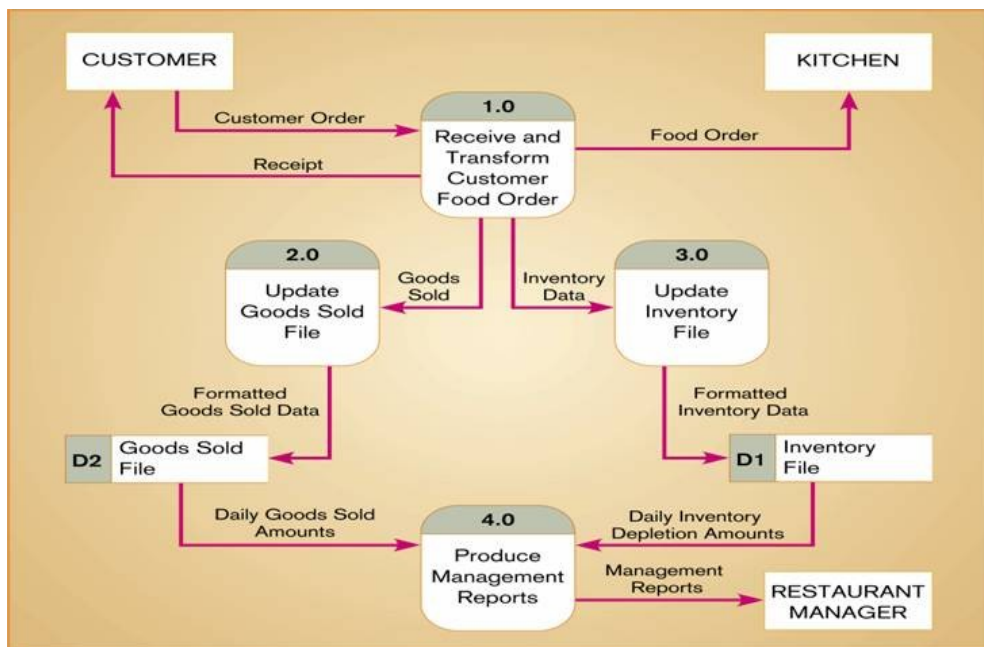


Figure 2.11. DFD Level 0.

Figure 2.11 shows the splitting of the process from one into four. Each process is numbered as 1.0, 2.0, 3.0 and 4.0, respectively. The total external entity must be fixed three (3) likewise input and output data flow in the system must be the same as in context diagram. Whereas data flow in the system (that flow between process and / or data storage) depend on the involved process and data storage. There are two data storage, namely, Goods Sold File and Inventory File. The two (2) data storage is used to keep the data from a process. The data will also be read / accessed by other processes. For example data storage Inventory File contained the data produced by process 3.0 (Update Inventory File). This data will be used by the process 4.0 (Produce Management Reports) to make the report that will be sent to the Restaurant Manager.

The next level in DFD, namely, level 1, 2, etc, is needed if current level is not detail enough. For example if DFD level 0 (Figure 14.12) is not detailed enough in showing the data flow, then it could be detailed in DFD level 1. Process is normally be the one that has to be detailed. Detail in the next level can be in all process or certain processes. DFD level 0 or the next level has similarities in the rules that shown in the following table.

Table 2.1. Rules in DFD

Group	Rules
General	<ul style="list-style-type: none"> ● inputs to a process will always vary with the outputs. ● objects (External Entity, Process, Data Storage, and Data Flow), which is in a DFD always have a unique name.
External Entity	<ul style="list-style-type: none"> ● Noun always used in the name of External Entity. ● Data can not flow directly from one External Entity to another External Entity.
Process	<ul style="list-style-type: none"> ● The name used in the process always use verb. ● There is no process that only produces output. ● There is no process that only accept input.
Data Storage	<ul style="list-style-type: none"> ● The name used in the Data Storage always use the noun. ● Data can not flow directly from the Data Storage Data Storage to one another. ● Data can not flow directly from the External Entity to the Data Storage also vice versa.
Data Flow	<ul style="list-style-type: none"> ● The name used in the data flow always use the noun. ● Data flow between the two notation have only one direction of flow. ● Branching (fork) shows the exact same data that flows from one place to two or more places. ● Merging (join) the data shows an exactly the same data as that flow to two or more places. ● Data Flow towards Data Storage means to update the data. ● Flow of data from the Data Storage means reading / retrieving data.

2.2.2. Design

Software design is the task, the stage or the activity that focused on the detail specification of a computer based solution (Whitten et.al, 2004).

Software design often refers as physical design. While system analysis stage stress on business aspect, software design focuses on technical and implementation side of a software (Whitten et.al., 2004).

The main output from software design stage is the design specification. The specification covers the general design specification for the system's stakeholders and detailed specification to be used in the implementation stage. The general design specification contains the general picture of the software for the stakeholders. USD diagram on the software is normally the important point in this stage. The detail design

specification or the detail architectural software design is needed to do system design so as to have a good construction, an exact and accurate data processing, valuable, user-friendly and has a good foundation for further development.

The architectural design consists of the database design, the process design, the user interface design including input design, output and report form, hardware design, software and the network. The process design is the continuation of process modeling that carried out in the analysis stage.

2.2.3. Construction

The construction is the stage that translate the logical and physical design into computer program codes. Construction techniques and methods will be described in more the detail in this book.

2.2.4. Testing

System testing involves all planned user groups. The level of acceptance is evaluated by all user groups based on the determined criteria.

2.2.5. Maintenance and Configuration

When a software is considered appropriate to be used, the next stage would be the software maintenance. There were several software maintenance types known as shown in Figure 2.12.

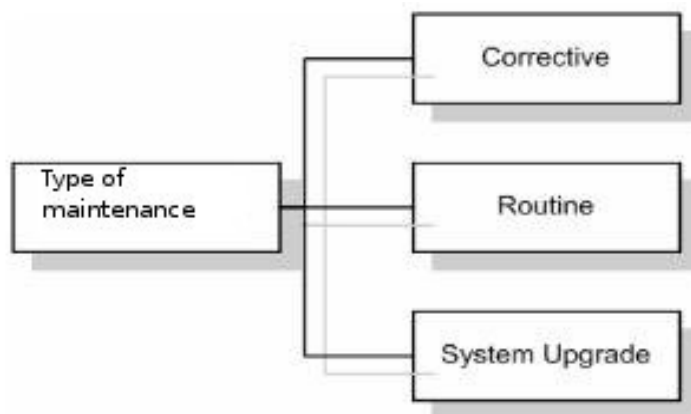


Figure 2.12. Maintenance Types.

Corrective maintenance is carried out to circumvent errors known as bugs. The maintenance is carried out by improving the code, increased the some parts or eliminated certain parts.

Routine maintenance is also known as preventive maintenance is carried out routinely to maintain the performance of the software whether there is any mistakes / errors or not.

System upgrade maintenance is carried out to change any components of the software. For example, platform or operating system changed, old to new version change require upgrade of the software.

2.3. SUMMARY

- Software engineering models generally refers to the system development process modeling that was known as System Development Life Cycle (SDLC).
- Software development model used is generally the Waterfall Model, Prototyping, and Unified Process (UP).
- The main stages of software engineering covers the analysis, the design, the construction, the testing and the maintenance.

2.4 EXERCISE

Name stages in System Development Life Cycle (SDLC).

1. Name the similarity of various software development models characteristics.
2. Name five software development models.
3. What is meant by construction stage in software engineering?
4. Show the notations in the Data Flow Diagram.

CHAPTER 3 ELECTRONICS AND COMPUTER SYSTEMS

If you open or look into the interior of an electronics equipment, you will see similar equipment as shown in Figure 3.1. There is a circuit board, cables that connect the components, and various electronics components. Electronics equipment like this that build a computer equipment. Thus, understanding electronics, digital electronics and computer systems is important for you to be active in the software engineering world.

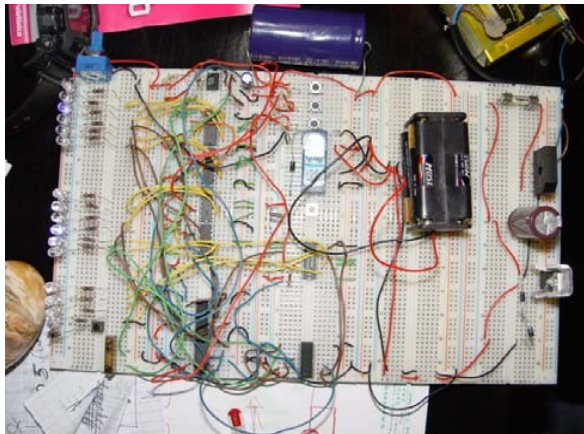


Figure 3.1. An electronics circuit

This chapter will discuss two standards of competence, those are basic electronics and digital electronics, especially that is used in a computer system. There were two basic competence in the standard of competence of basic electronics, namely, understanding principles of basic electronics and knowing electronics components. Whereas the standard of competence for the digital electronics consisted of competence in basic digital electronics and digital electronics in computers. In this book, each basic competence will be elaborated in the text. The summary will be given at the end of the chapter. Basic electronics and mathematics are the prerequisite of this topic.

THE OBJECTIVE

The chapter's objectives are:

- Mastering the concept of basic electronics.
- Knowing electronic components.
- Mastering the concept of digital electronics.
- Mastering the digital electronics and computer systems.

3.1. BASIC ELECTRONICS

3.1.1. Concept of Basic Electronics

Electronics is a knowledge that studied components / equipment that implement **weak flow of electric current** that was operated by means of controlling electron current or particle that carried an electric charge such as in computer, electronics equipment, thermocouple, semiconductor, etc. Electronics is a branch of physics, while knowledge in design and production of the electronics circuit is part of electrical engineering, computer engineering, electronics engineering and instrumentation.

Equipment that uses the electronics principles is known as electronic devices. Example of electronics devices are Cathode Ray Tube (RT), radio, television, cassette recorder, video cassette recorder (VCR), VCD player, DVD player, video camera, digital camera, desktop computer, laptop computer, Pocket Digital Assistance (PDA), robot, smart card, etc..

Electronics is based on the knowledge of electricity. **Electricity**, could be interpreted as follows:

- Electricity is state of sub-atomic particle such as, electron and proton, that caused the pulling and rejecting force among them.
- Electricity is the energy that flow through the cable. An electric current flow as electric charge flow from positive lead to negative lead.

There are two kind of electric charge: positive and negative. Through experiment, similar charge will refused each other and opposite charge is pulling each other. The amount of force between charges is defined by Coulomb law. The **Coulomb law** explain the force between two charges, that are separated by the certain distance. In the international system unit uses "C" to represent electric charge in Coulomb. The Q symbol is used to represent the quantity of electric charge. For example, "Q=0,5 C" mean the "quantity electricity charge is 0.5 coulomb".

When electricity flow through a special material, such as, wolfram and tungsten, the metal will glow light. Such materials are used in light bulb. As electricity flow through material that has resistance, it will release heat. The more the current flow, the hotter it will be. This characteristics is used in the element of iron and the electric stove.

The resistivity is equivalent to the voltage drop on the electronics component and the current pass through it. The resistivity can be formulated as follows:

$$R = V / I$$

or

$$R = \Delta V / I$$

Where V is the voltage and I is the current.

Voltage is the electric potential between two point in a circuit, and expressed in Volt. The potential energy in electric field creates electric current in a conductor. Thus, depends on the electric potential, a voltage can be categorized as extra low, low, high or extra high voltage.

Current is the amount of charge flow in certain time. Electric charge may flow through cable or other good conductor media.

$$I = \frac{Q}{t}$$

In the past, **conventionally electric flow** is define as positive charge current, although we currently know that electric current is produced by electron current that carried negative charge in an opposite directional flow. In International System (SI) unit, current is noted as **Ampere (A)**.

3.1.2. Electronics Components

- **Resistor**

Resistor is the basic electronics components used to limit the current flow into a circuit. As noted in the name, resistor is resistive and is generally made from the carbon material. The resistance unit of a resistor is Ohm and is normally designated by the symbol Ω (Omega). In general, a resistor has a tube shape with two copper tips at both end. In the resistor's body, several colored code rings are painted on its body to provide the resistance value without having to measure it using Ohmmeter.

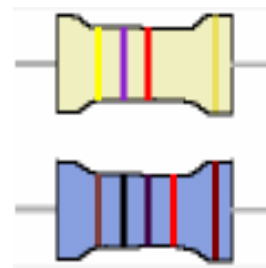


Figure 3.2 Resistor.

- **Capacitor**

A capacitor is an electronic component that keeps electric charge. A capacitor is made of two metal plates separated by a dielectric material. Some known dielectric materials are vacuum air, ceramics, glass etc. If voltage is applies to the two metal tips, one tip will be positively charged and the other negatively charged.

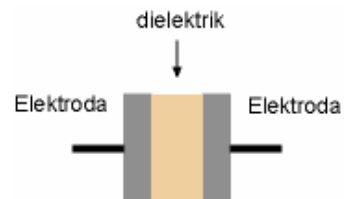


Figure 3.3. Capacitor.

- **Inductor**

An Inductor is a components that could store magnetic energy. This energy is represented by the existence of emf (electromotive force) if electricity pass through an inductor.

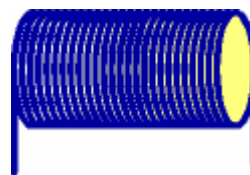


Figure 3.4 Inductor

An inductor in series to current flow will oppose fluctuation in the current pass through it. Usage in DC current will stabilize the DC voltage in fluctuate load. In AC circuit, to reduce unwanted current fluctuation. There are more inductor's application in filter, tuner circuitry etc.

3. 2. DIGITAL ELECTRONICS

3.2.1. Concept of Digital Electronics

Digital electronics is an electronics system that uses digital signal. Digital signal is a discreet signal. It is usually noted as 1 or 0. The notation 1 is a symbol of connection, while notation 0 is no connection. A simple example of this, 1 is when a switch is switched ON, and 0 is when it is OFF.

Digital electronics is an application of boolean algebra and is used in many areas such as computer, cellular telephone and other equipment. Digital electronics provides several advantages, such as, digital system can be easily controlled by computer and software, easier to store information as compared with analog information. Although, there are several disadvantages of digital system, namely, in several cases digital system needs more power, more expensive and fragile.

3.2.2. Logic Gate

Digital electronics or digital circuits build on logic gates. Logic gate execute logic operations on one or more input to provide a single output. Output is a result of a series of logic operation based on boolean algebra principles. In terms of electronics, there input or output is in the form of voltage or current depending the electronics' type.


Each logic gate needs power source with appropriate current and voltage. In the digital logic diagram, power is normally not included.


In its application, logic gate is build on blocks of electronics hardware.


These logic gates are normally made of transistors. The number of transistors highly depends on the gate's complexity. The basic form of the logic gate can be represented in truth table. There three (3) basic form of truth table, namely, AND, OR, and NOT. Shown is the truth table and the symbol of the gate.

Explanation of Figure 3.5 is as follows:

- AND, in two inputs A and B then output or signal is produced when A=1 and B=1.
- OR, in two inputs A and B then output or signal is produced when one of the input is 1.
- NOT, the output signal will be reverse of the input signal.


 AND gate	A B		F
	0	0	0
	0	1	0
	1	0	0
	1	1	1


 OR gate	A B		F
	0	0	0
	0	1	1
	1	0	1
	1	1	1


 NOT gate	A	F
	0	1
	1	0

Logic symbol Truth table

In addition to the above basic form, there several derivative form that important to know. Figure 3.6 shows the truth table and the symbol of NAND, NOR, and XOR gate. NAND is a result of NOT + AND operation, NOR is NOT + OR, whereas XOR is an exclusive OR. NAND and NOR is a logic gate largely used in many digital electronics equipments.

 NAND gate	A B		F
	0	0	1
	0	1	1
	1	0	1
	1	1	0

 NOR gate	A B		F
	0	0	1
	0	1	0
	1	0	0
	1	1	0

 XOR gate	A B		F
	0	0	0
	0	1	1
	1	0	1
	1	1	0

Logic symbol Truth table

Figure 3.6. NAND, NOR and XOR truth table and its logic gate symbol.

3.2.3. Digital Circuit

In the above section, we have studied the form of logic gate and its truth table. A digital circuit consists of one or more of these logic gates. Please see Figure 3.7, shown in the figure, the upper figure is four (4) NAND logic gate, one pin for 5V power supply, and one pin four ground. Whereas the lower figure is actual chip type 7400.

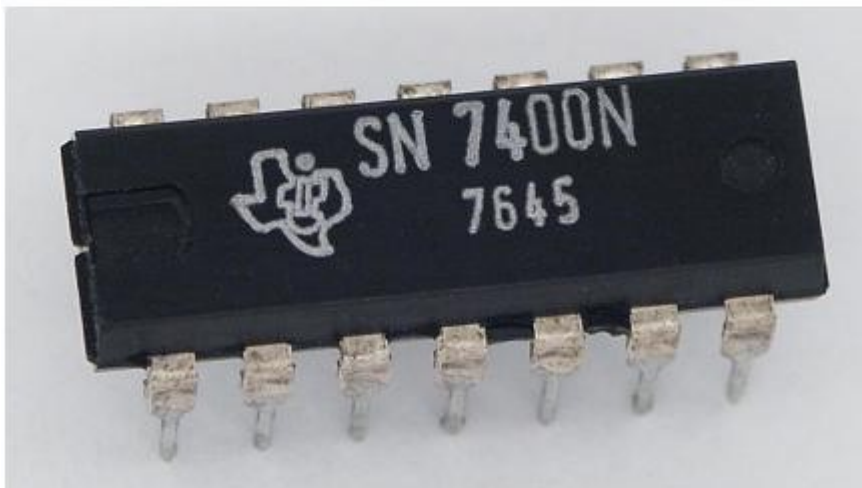
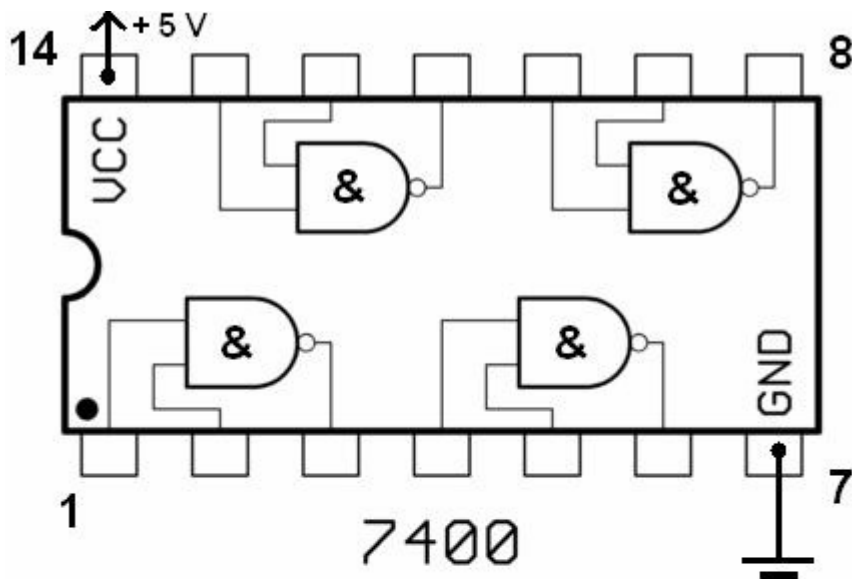


Figure 3.7. Digital circuit example and its hardware representation.

3.3. COMPUTER SYSTEM

The term computer came from Latin “Computare” means to count. Thus, any devices to do counting such as counting machine, calculator, or even abacus all technically called computer. However, in its development, computer had developed into different meaning.

Computer is an electronic data processing that worked and controlled by a set of instructions (the program) (Blissmer, 1985). A **computer system** consists of human, hardware, and software to get a useful information, to ease any job, faster process etc.

There are three (3) main component in computer system, namely, human as user, hardware and software (Figure 3.8) If one is not present then the computer will not properly perform. For example, if there is only human and hardware, computer system will not perform as no software to help human in running the hardware.

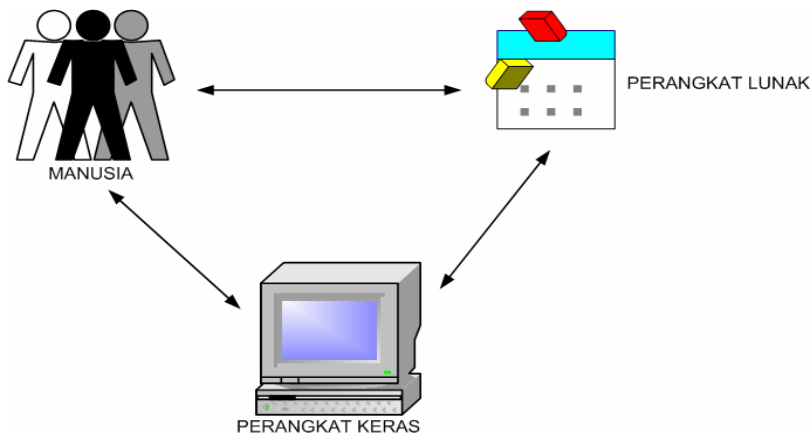


Figure 3.8. Computer System.

3.1.1 Hardware

Hardware is the physical part of computer. Hardware is distinguished from the data that is inside or that operated inside, and software that provided the instruction for the hardware to complete its task. Generally it has four (4) basic components to the computer that is mutually related (see Figure 3.9).

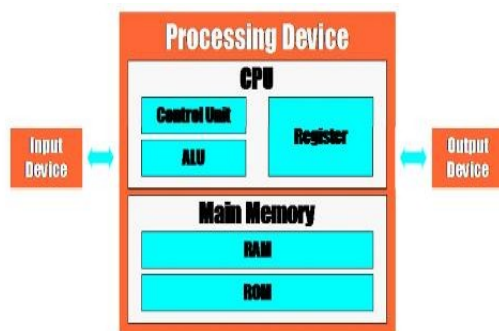


Figure 3.9. Basic Computer Components

- Input – a device that enables user to enter data or command into the computer. Input device examples including keyboard, mouse, joystick, and digitizer.
- Output – a device that provide the used on information produced by computer. The example of output devices are the monitor, printer, and plotter.
- Main memory – device that stores data, program and information produced by computer during processing. The main memory unit consisted of many cells, that each stores one information unit. There are two (2) types of main memory unit, namely, ROM (Read Only Memory) and RAM (Random Access Memory). ROM could only be written on just once and then read many times. RAM could be repeatedly write, remove and read. Data, Program, and Information that are processed is kept in RAM, and will be lost if the computer shuts down. That is the reason why data, program, and information is kept on secondary storage media such as hard the disk, diskette, CD, etc.
- Central Processing Unit – the brain of computer that process data, program, and information. There are two (2) important parts in CPU that is Arithmetic and Logical Unit (ALU) and Control Unit. Many people refer ALU as the brain of computer. ALU responsible for two (2) basic operations, namely, arithmetics operation and matching / comparisons. While Control Unit responsible for coordination activities of other unit, such as, how to recognize keyboard and work as input device.

The physical appearance of personal computer can be seen in Figure 3.10. A PC consists of many components,

1. Display
2. Motherboard
3. CPU
4. Main Memory
5. Expansion Cards
6. Power Supply
7. Optical Disk Drive.
8. Secondary Storage (Hard Disk)
9. Keyboard
10. Mouse

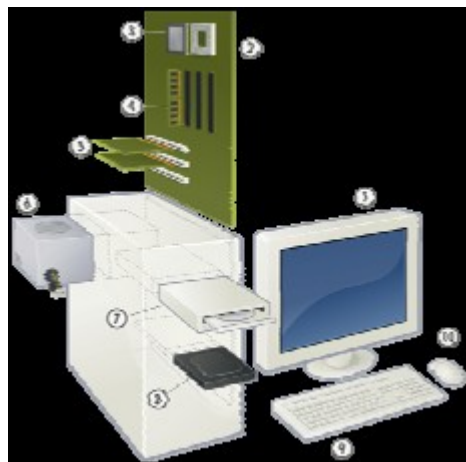


Figure 3.10. Computer Hardware.

The following is a short explanation of the computer's physical components:

1. *Display.* *Display* or *monitor* devices is the output unit of a computer. A cable connects the monitor to a video adapter installed in the *motherboard's* expansion slot. Computer sent signal to video adapter to put forward characters, picture or graphics. Video adapter will convert the signal into a collection instructions for the monitor to put forward text, or picture in its screen.



Figure 3.11. Display or monitor

2. *Motherboard.* Motherboard also as *mainboard*, *system board* or *logic board* (Apple Computer) and sometimes shortened as *mobo* is the main electronics system, such as in modern computer hardware. On this board we can put other components such as, main memory, processor, video adapter, voice adapter, etc, to form a complete computer system.



Figure 3.12. Motherboard of a computer.

3. CPU. *Central Processing Unit (CPU)*, often known as *Processor*, is a component in digital computer that interprets instructions and processes data as instructed by computer program. CPU provided the important part of digital system namely the ability to be programmed. CPU must be installed in every computer hardware.



AMD Athlon processor



Intel processor

Figure 3.13. Central Processing Unit (CPU).

4. Main Memory. Main Memory also known as Primary Storage, or Internal Memory, is the computer's memory with direct access to CPU without using any special input/output path. Main memory is used to keep the actively used data. Primary storage can be formed in several type of storage, such as, main storage, cache memory, and special registers.



Figure 3.14. Several types of main memory.

5. Expansion Cards. Expansion card is a printed circuit board (PCB) that can be inserted into motherboard's expansion slot to increase computer's functionality. The example of expansion card are video adapter, audio adapter card, network card etc.



Figure 3.15. Installation of expansion card

6. Power Supply. Power supply also known as Power Supply Unit (PSU) is the equipment to supply the required energy mainly electrical power in a computer.



Figure 3.16. Power Supply Unit.

7. Optical Disc Drive. Optical Disc is a secondary storage media with a disc like shape. Data kept in an optical disc may be accessed through the help of a laser beam onto the disc. There are two (2) major types of optical disc, namely, CD (Compact Disc) and DVD (Digital Versatile Disc). The device to read, write, or erase in an optical disc is known as Optical Disc Drive.



Figure 3.17. CD-RW Drive, an example of Optical Disc Drive.

8. **Secondary Storage (Hard Disk).** *Secondary Storage* is a device used to help the *Primary Storage (main memory)*, especially to keep data, program, or information that will be used again. Different from *primary storage*, data, program and information in *secondary storage* will not be lost even the computer is shutdown, except if it is deliberately removed. Secondary storage often used in a computer is *Hard Disk*. *Hard disk* keeps the data in a magnetic disc in certain patterns to represent the data.



Figure 3.18. Hard Disk

9. **Keyboard.** *Keyboard* is a device to type in text or character into a computer. This device also provides special control keys in computer. Figure 3.19 shows the layout of a *keyboard*.



Figure 3.19. General layout of a keyboard

10. **Mouse.** *Mouse* is usually a pointing device that can be used to detect relative movement in two dimensional plane to be display later on the screen. Mouse sometimes equip with other functions, such as, *wheel*. In addition, mouse play role in command execution through a its left / right button of either single click or double click.



Figure 3.20. Various types of mouse.

3.1.2 Software

Software is a computer program that act as a media to interact between the user and hardware, or can be view as the “interpreter” of the user command to pass it on or process by the hardware.

In a computer program, its content can be easily modified. In the computer, software is normally run on the RAM before being executed in the CPU. At the lowest level, software is in machine language specific to certain processor.

There were several software classifications. In general, software could be divided into three (3) groups, namely:

1. System Software

System Software is the software that help overlook the hardware and the computer system. The objective of **system software** is to remove the computer complexity as remote as possible from the application programmer. Thus, application programmers don't have to directly access memory and hardware in their program.



Figure 3.21. Windows XP. Operating System Picture

Example of system software are operation system, driver, diagnose software, windowing system, utilities etc. Among these softwares, operation system is the most important software. This software acts as interface between computer and the outside world. In hardware section, operation system will describe any available hardware connected to the computer. The operation system provides the driver of these hardware so as to be known and would work properly. Detailed explanation of an operating system can be seen in Chapter 4.

2. Programming Software

Programming Software provides aids or function to help programmer in creating computer program.

The software highly depends on the programming language used. The aids covers text editor, compiler, linkers, debugger etc. An Integrated Development Environment (IDE) or an integrated development environment combine these tools to ease programmer. We will focus on this in this book.

3. Application Software

Application Software is a software that used to help user in performing certain task that may not be related to computer.

There are several type of application software, such as, industrial automation software, business software, educational software, database, and computer game. Examples of these application can be seen in the following pictures.

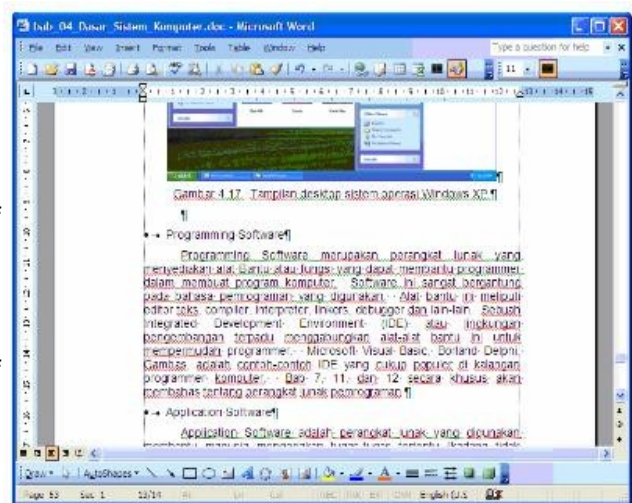


Figure 3.22. Microsoft Word (Word Processing Software)

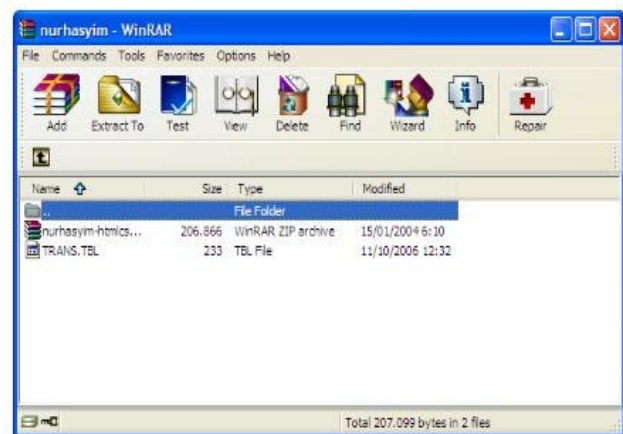


Figure 3.23. Winrar (File compression and extraction software)

XOR.

- Computer is an electronics data processing tool that work and controlled by a set of instructions (the program).
- Computer system consists of human, hardware, and software that interacts with each other to get useful information, ease in working, speedup processes and other objectives.
- There are three (3) main components in computer systems, namely, human, hardware and software.

3.5. EXERCISE

1. Explain the term electronics.
2. Explain the relationship between electric charge, resistance, voltage and current.
3. Name electronic components that you know.
4. Explain digital electronics.
5. Draw symbol of various logic gates.
6. Draw an example of a digital circuit.
7. Name components in computer systems.
8. What would be happen if a computer system lack one of its components?

CHAPTER 4 OPERATING SYSTEM

```

bash-2.05b$ cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pkgmetadata SYSTEM "http://www.gentoo.org/dtd/metadata.dtd">
<pkgmetadata>
<herd>base-system</herd>
</pkgmetadata>
bash-2.05b$ sudo /etc/init.d/bluetooth status
Password:
* status:  stopped
bash-2.05b$ ping -q -c1 en.wikipedia.org
PING rr.chtpa.wikimedia.org (207.142.131.247) 56(84) bytes of data.

--- rr.chtpa.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 112.076/112.076/112.076/0.000 ms
bash-2.05b$ grep -i /dev/sda /etc/fstab | cut --fields=-3
/dev/sda1                /mnt/usbkey
/dev/sda2                /mnt/ipod
bash-2.05b$ date
Wed May 25 11:36:56 PDT 2005
bash-2.05b$ lsmod
Module                  Size  Used by
joydev                  8256  0
ipu2200                175112  0
ieee80211               44228  1 ipu2200
ieee80211_crypt         4872  2 ipu2200,ieee80211
e1000                  84468  0
bash-2.05b$ █

```

Figure 4.1. text based operating system.

As we skimmed Figure 4.1, we may ask the meaning of the text in the figure. As we look more closely we may guess that some of the text are the instruction and others are the result as executed by the operating system. In many cases, we likely to ignore text based operating system although in reality it is very important.

This chapter will discuss the standard of competence in operating a text and GUI-based operating system. There are two (2) basic competences in this standard competence, namely, preparing PC operation, operate the PC in network environment and disconnecting network connection. In this book, each basic competence will be elaborated. The summary will be in end of the chapter. Before proceeding with the chapter, please review on the computer system in the previous chapter as well as supporting materials and mathematics.

OBJECTIVES

After reading this chapter, the reader should be able to:

- Explain on an operation system.
- Install and booting an operation system
- Operate a text or GUI operating system.
- Operate a PC in network environment.

4.1. Operating System Concept

As mentioned earlier, operating system is part of the *system software* group that takes care the computer hardware and overall computer system.

Operating system manages resource usage in computer and provides user interface to access those resources.

FUNCTION

General functions of operating system can be seen in Figure 4.2.

- **User Interface.**

User interface may be easily recognizable by the user . Through the user interface one may interact with the operating system, hardware as well as other software packages. The operation system is basically waiting for input or instruction from the user and translated these command into language that understood by the computer. User interface is the place where user write or submit these orders.

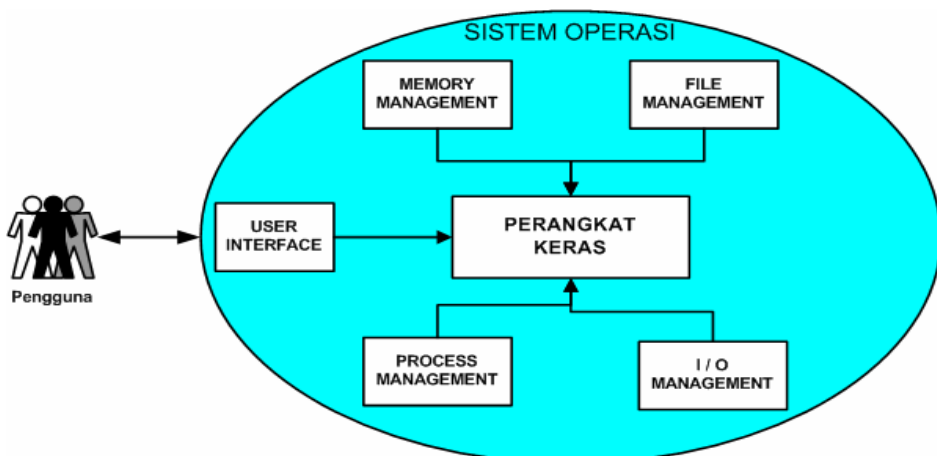


Figure 4.2. Operating system functions

In general there are two (2) interfaces, namely, *Command Line Interface* (CLI) and *Graphical User Interface* (GUI). CLI provides user facilities to submit command in text whereas GUI provides graphical based interface. Most modern operating systems provide GUI interface. Some GUI interfaces are integrated into the operating system kernel, such as, Microsoft Windows and the early version of Apple Mac OS. Others provide modular GUI interface, and not integrated directly into the operating system kernel, such as, Unix, Linux and Mac OS the X version and above.

● Memory management

The main memory also known as memory is a large *array* of *word* or *byte*, that reaches hundreds, thousands, or even millions. Every word or byte has its own unique address. The function of main memory is to store instruction / data that being actively used by the CPU and Input / Output devices. The main memory includes the storage of the volatile data that is not permanent and will lost as the computer shuts down.

The operation system is responsible for memory management activities such as:

- Keep track of the usage memory and who use it.
- Choose the program that would load into the memory.

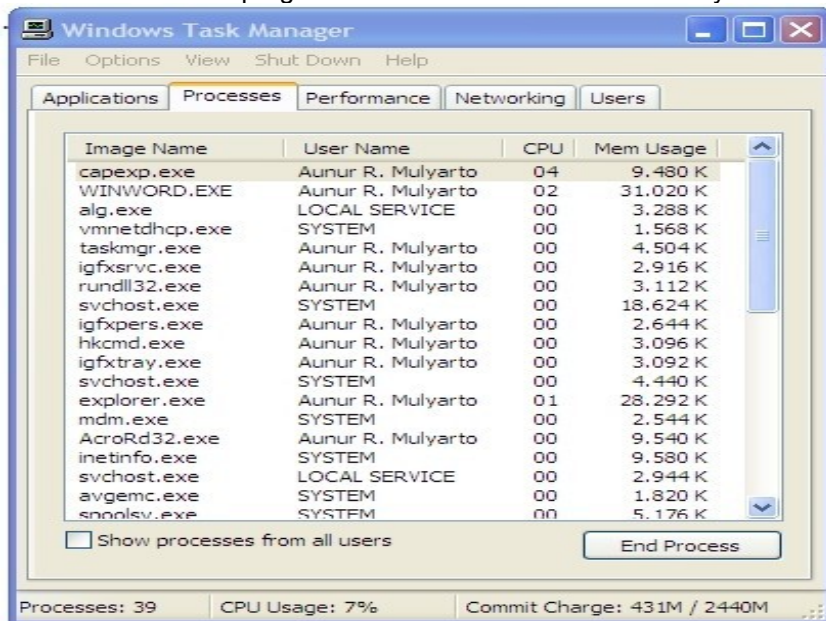


Figure 4.3. Memory Management in Microsoft Windows Operating System

● File management

File is a collection of information following the objectives of the one who make the file. File is generally representing program or data. File could have a hierarchical structure, such as, directory, volume, etc. The operating system implements abstract concept by locating / putting a file in a mass storage media, such as tape and disk.

The operation system is responsible for file management activities, such as:

- Creation and erasing the file.
- Creation and erasing a directory.
- Supporting file and directory manipulation.
- File mapping in *secondary-storage*.
- **File back-up into non-volatile media.**

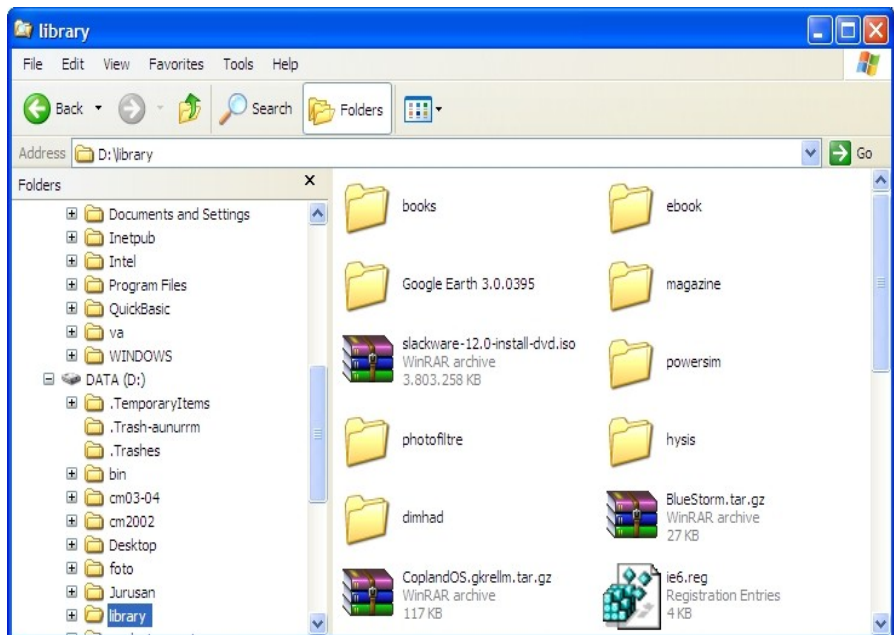


Figure 4.4. Windows Explorer as file management facility.

● Process Management

A process is an executed program. A process needs some resources to complete its task. The allocation of these resources was carried out by the operating system. For example, memory usage by the CPU, file opening, and Input/Output usage. Whenever a process stops, the operating system will reclaim all resources to be used again.

The operation system is responsible for process management activities such as:

- Create and remove user processes and system processes.
- Postponed or continue a process.
- Provided process synchronization mechanisms.
- Provided process communication mechanisms.
- Provided deadlock handling mechanisms.

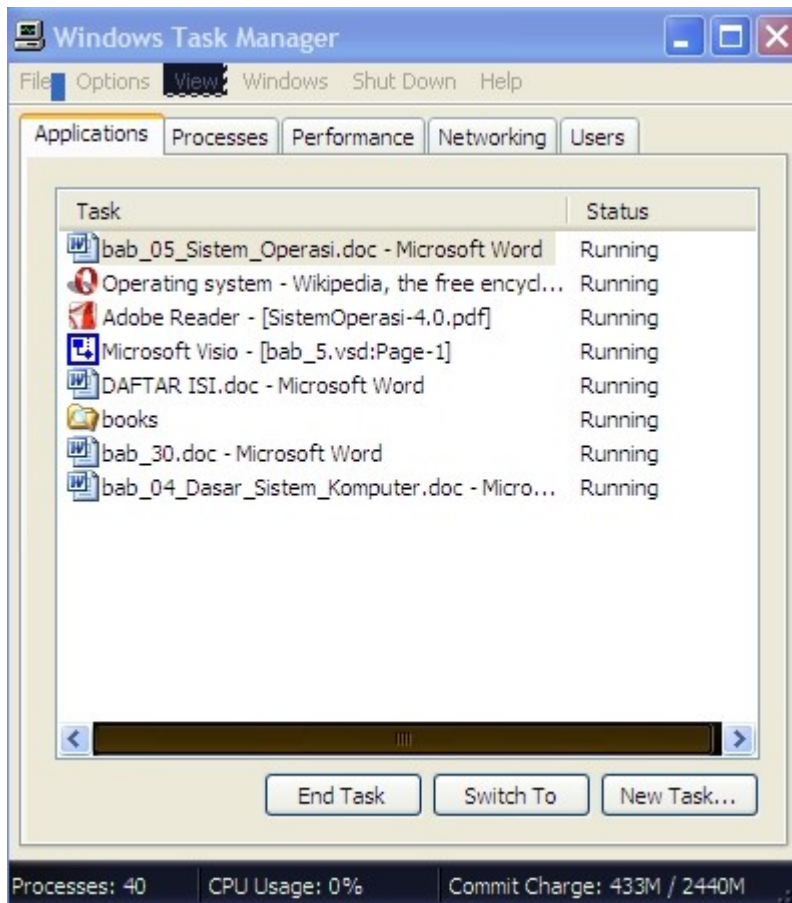


Figure 4.5. Process management in Microsoft Windows operating system.

- **Input / Output (I/O) management system**

Input / Output management system is also known as device manager. The system provides general device driver so that all have uniform input/output operation, in open, read, write, and close. For example: user may use same operation procedure to read a file on hardware, *CD-ROM* and *floppy disk*.

Operating system component for Input/Output system:

- Buffering: temporary cache data from / to Input / Output devices.
- Spooling: scheduling the usage of system Input / Output to make it more efficient through queuing etc.
- Provides driver: to enable specific operation for certain Input / Output hardware.

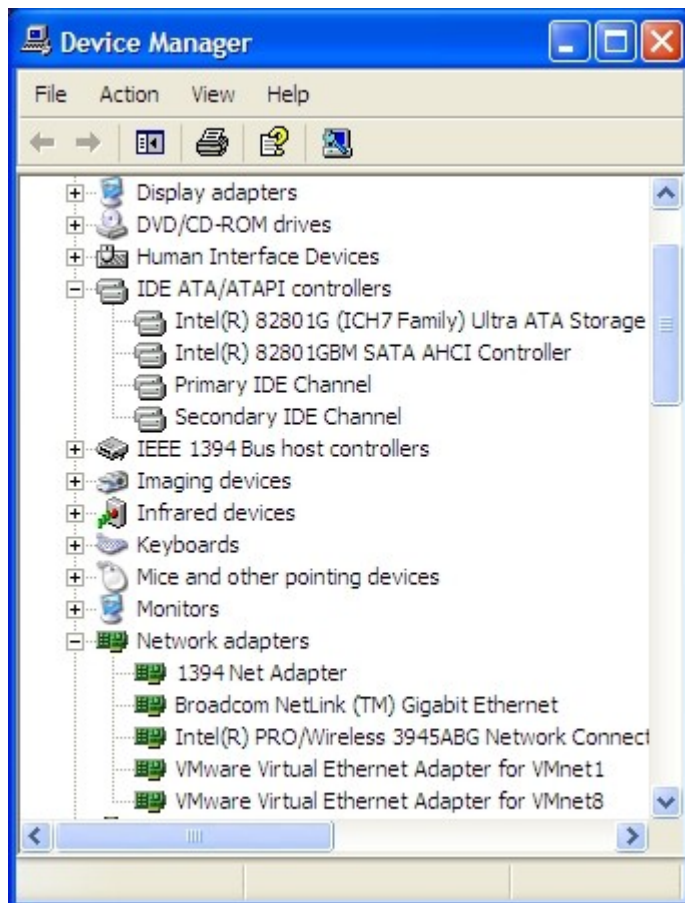


Figure 4.6. I/O Management in Microsoft Windows operating system.

- **BIOS**

BIOS is a short for *Basic Input / Output System*. BIOS is the program that will be executed in the early booting process of the computer. The main BIOS function is to identify computer hardware. BIOS is usually kept in ROM (*Read Only Memory*) on computer motherboard.

When the computer starts BIOS will try to identify the installed components / hardware in the computer, such as:

- *Clock generator.*
- *Processors and caches.*
- *Chipset (memory controller and I/O controller).*
- *System memory.*
- *All PCI cards.*
- *Primary graphics controller.*
- *Mass storage controllers, such as, SATA and the IDE controllers.*
- *Various I/O controllers, such as, keyboard / mouse and USB.*

After being recognized, BIOS will call the *boot loader* to boot the operating system.

BIOS can be configured through its built-in configuration facility. It can usually be accessed by pressing Del or F2 button (depends on motherboard type) when the computer just turned on. When we are able to enter the BIOS configuration facility, we will see something like Figure 4.7, we can do various configuration on computer hardware.

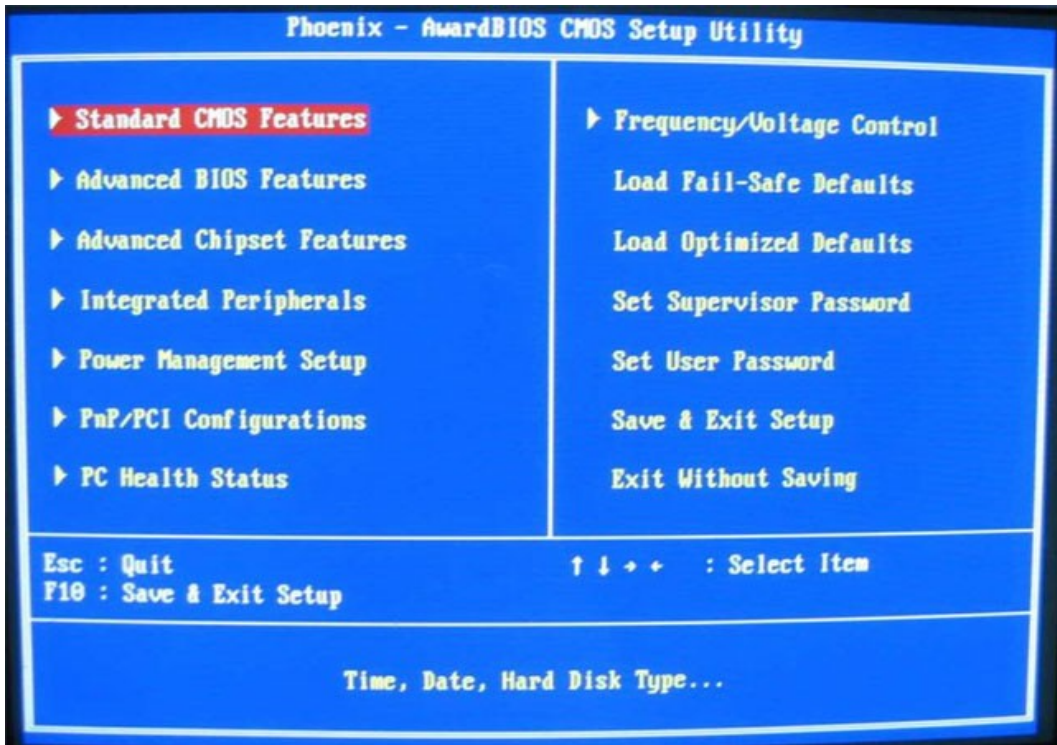


Figure 4.7. BIOS utility.

4.2. TYPES OF OPERATING SYSTEM

The operation system has been developed for a long time. From its simple form in the past into its today modern form. Each has its strengths and weaknesses especially in its built-in functions. In this section, we will discuss several operating systems that being used by many computer users.

4.2.1. DOS

DOS stands from *Disk Operating System*. DOS refers operating system that being used in many computer to provide abstraction and management of secondary storage devices as well as its information. For example the usage of file system to management files in storage equipment. DOS may be run from one or two diskettes as storage media capacity during DOS era is fairly limited to about 1.4Mbyte maximum.

There were many DOS, such as, Apple DOS, Commodore DOS, Atari DOS etc. DOS is fairly dependent on the type of computer. The most famous DOS is the one that runs on IBM *Personal Computer* or its compatible.

To execute operating system command, DOS uses text based command or Command Line Interface (CLI). We must press ENTER after typing the command. DOS operation example can be seen in Figure 4.8.

```

C:\>dir/w
Volume in drive C has no label.
Volume Serial Number is 5CC3-1976

Directory of C:\

[AppServ]          AUTOEXEC.BAT          CONFIG.SYS
[Documents and Settings] [Inetpub] [Intel]
ISACER.id          [Program Files]      [QuickBasic]
[va]               [WINDOWS]
                   3 File(s)           7 bytes
                   8 Dir(s)           9.171.398.656 bytes free

C:\>cd QuickBasic
C:\QuickBasic>dir/w
Volume in drive C has no label.
Volume Serial Number is 5CC3-1976

Directory of C:\QuickBasic

[.]                [..]                [QBasic-7]    [QBASIC_4.5]
                   0 File(s)           0 bytes
                   4 Dir(s)           9.171.394.560 bytes free

C:\QuickBasic>

```

Figure 4.8. Example of DOS usage.

4.2.2. UNIX

UNIX is an operation system that was initially developed by a group at AT&T Bell Laboratories.. Unix is used in server as well as workstation. Unix environment and client-server model shows that Unix aims toward a strong computer network operating system rather than personal computer operating system.

UNIX is designed to be *portable*, *multi-tasking*, and *multi-user*. Unix uses plain text to keep data, system files, treated devices as file, and many small programs that executes through CLI combined with pipeline sign (|). In the above Figure 4.1, several Unix commands are combined with pipeline. Solid and stable concept of Unix often used as foundation of modern operation system. Figure 4.9. shows how Unix becoming the foundation of today's many modern operating systems.

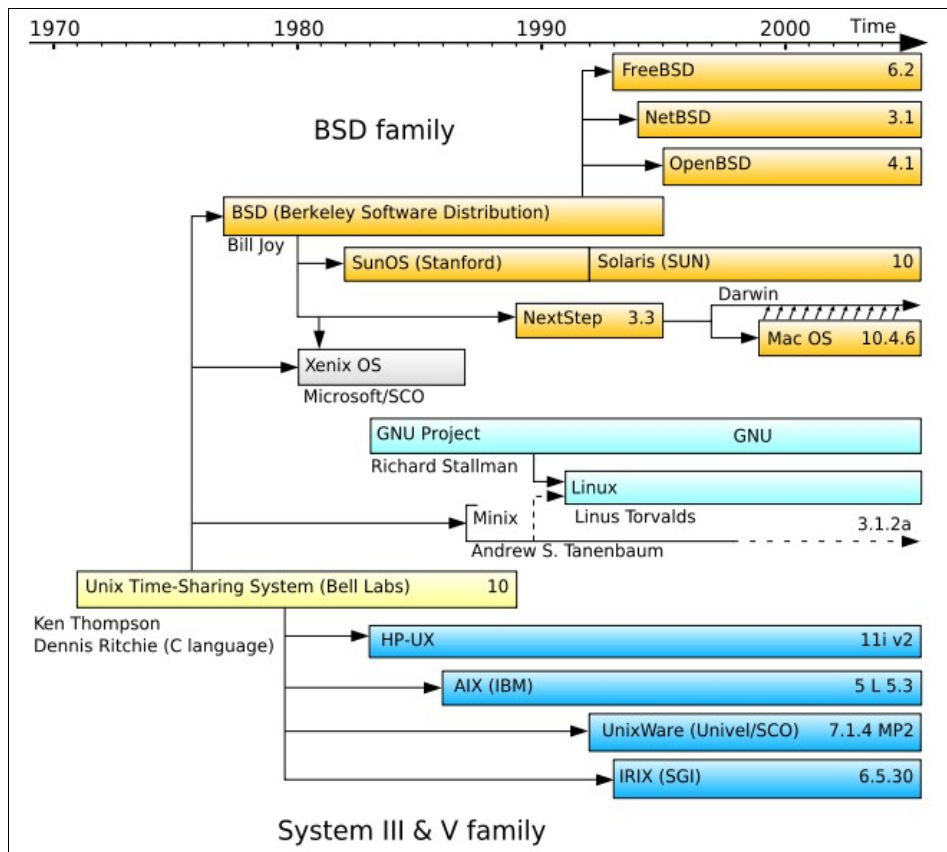


Figure 4.9. Unix and its derivative operating systems.

UNIX system consists of several components that usually bind together. These package is generally as follows:

- *Kernel* with sub the component such as:
 - *conf* — configuration file.
 - *dev* — driver hardware
 - *sys* — operating system kernel, memory management, process scheduling, system calls etc.
 - *h* — header files, defined the key structure in the system.



Figure 4.11. X windows system in UNIX.

4.2.3. Microsoft Windows

Microsoft Windows also known as Windows was initially only an add-on for MS-DOS since there was a high demand on GUI operating system. Early version of Windows run on top of MS-DOS. The early version of Windows shows several general function of operating system, such as, executable file type, its own hardware driver, etc.



Figure 4.12. Windows version 3.11.

Windows conceptually aims for personal computer. Windows did not initially support multi-tasking and multi-user concepts. Accommodation towards the network or functions client-server also was not as strong as UNIX families. Thus, security is a common problem in Windows operating system as connected to the network. However Windows advantage especially in the ease of use. In the latest version (Windows Vista), the multi-user and multi-tasking implementations are more mature. Moreover, the GUI is changed and uses three-dimensional effects.

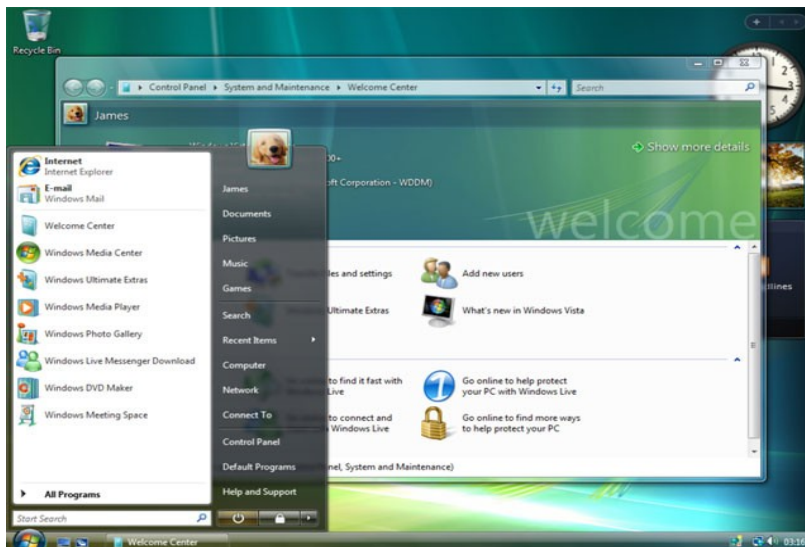


Figure 4.13. Windows Vista.

4.2.4. Apple Mac OS

As shown in Figure 5.10, Apple Mac OS is a descendant from UNIX through the BSD route (Berkeley Software Distribution). Thus, *multi-tasking*, *multi-user*, *networking* capabilities available to UNIX is also owned by Mac OS. Mac OS is a GUI based operating system. Apple is the pioneer in GUI based operating system. The use of icon, mouse and several GUI components were an extraordinary contribution towards the development of GUI based operating system.

The early version of Mac OS almost fully relies on its GUI capabilities and very limited CLI usage (Figure 5.15). Despite its ease there are several weaknesses, such as, multi-tasking is not perfectly working, limited memory management, and conflict in several programs. Improving the Mac OS is not an easy job.

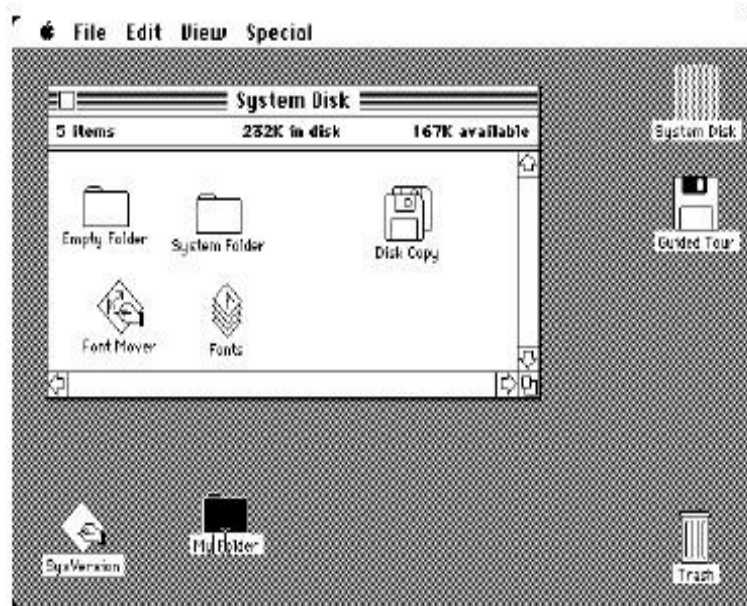


Figure 4.14. Early version of Mac OS.

In Mac OS X (the latest version), most of the weaknesses in the early version have been removed. Multi-tasking runs well and the memory management is far better. Moreover, Mac OS X GUI is known to be the best among existing operating systems.

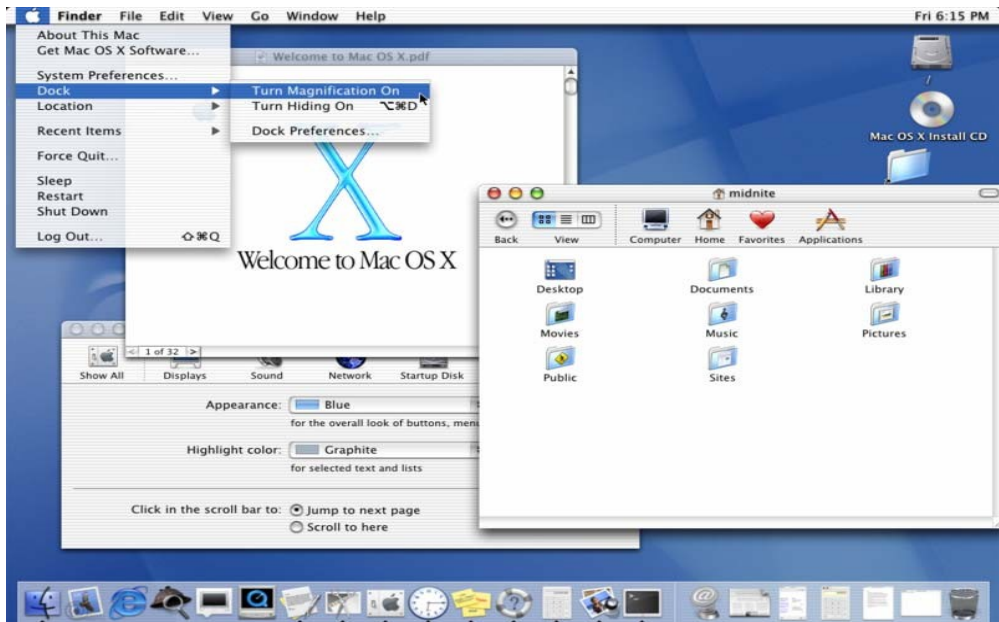


Figure 4.15. Mac OS X

4.2.5. Linux

Linux resembled UNIX systems, as main objective of Linux project is to be compatible with UNIX. Linux development was begun in 1991, when a Finnish student named Linus Torvalds wrote Linux, a *kernel* for 80386 processor, the first 32-bit Intel processor suitable for PC.

In many ways, Linux kernel is the core of Linux project, with additional components it forms a complete Linux operation system. Source code of Linux kernel is specially made for Linux project. However, its supporting softwares are not exclusively made for Linux,

but normally used in several UNIX like operating systems, such as, BSD operating system from Berkeley, *X Window System* from MIT, and the GNU project from *Free Software Foundation*.

Tool sharing works both directions. The main Linux library is originally started from GNU project. The library improvement is through community contributions, especially in addressing, inefficiency, and bugs. Other components, such as, GNU C Compiler, gcc, has a good enough quality to be directly used in Linux. The Linux's administrative network tool is using the code originally developed for 4.3BSD. Interestingly, the newer BSD, such as FreeBSD, borrow the code from Linux, for example Intel mathematics library for *floating-point-emulation*.

Today, Linux is one of the most rapidly developed operating system. The world wide developers groups are enhancing its features and help pushing the Linux operating system forward. Moreover, many developers are working to port applications into Linux.

The main problem for Linux users in the past is the text based interface. It prevents layman in using Linux, as It requires a thorough study of the not user-friendly commands prior to use it. However, today, the situation is changed with the presence of KDE and GNOME. Both have a good desktop GUI interface that change the perception of the world on Linux.

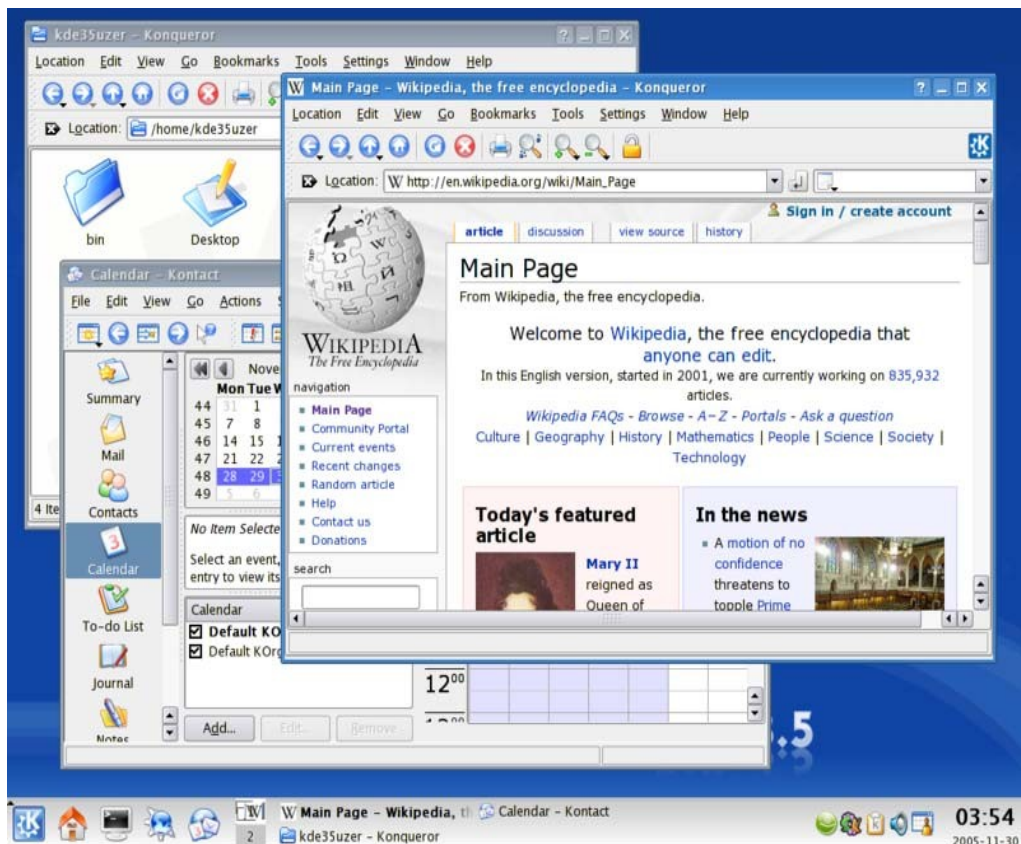


Figure 4.16. Linux with KDE desktop.

4.3. PREPARING AND OPERATING AN OPERATING SYSTEM

Knowing information system is not enough. Those who like to work in programming must understand thoroughly on how install, boot and operate the operating system, more deeply than normal users.

4.3.1. Installation

Installation is normally referred to installation of software package in a computer. Whereas operating system installation is installation of the operating system into the computer. An operating system must be firstly installed prior to other software. The operating system must be correctly installed before any application software can be run.

As previously explained, each operating system has its own characteristics. This includes the installation process of the operating system. Installation process is highly depend on the type of operating system. Based on its interface it can be categorized into two (2) main categories, namely, GUI based and CLI based. GUI based installation process can be found in Microsoft Windows operation system (full GUI in Vista version), Apple Mac OS X and above, several Linux versions, such as, Ubuntu and his descendants (Xubuntu, Kubuntu, Edubuntu, etc), Mandriva and his descendants (PC Linux OS), and newest version of Fedora. Whereas CLI based can be found in Linux Slackware version, Gentoo etc.

Installation process can also be categorized based on the installation source, namely, media sources, such as, CD, DVD or hard disk, or network sources. In general, CD or DVD is used as installation media. In this section, we will describe installation based on CD / DVD source.

Installation stages can be seen in Figure 4.16. These installation stages possibly varied between operating systems. However in general, the stages are not too much different between operating systems.

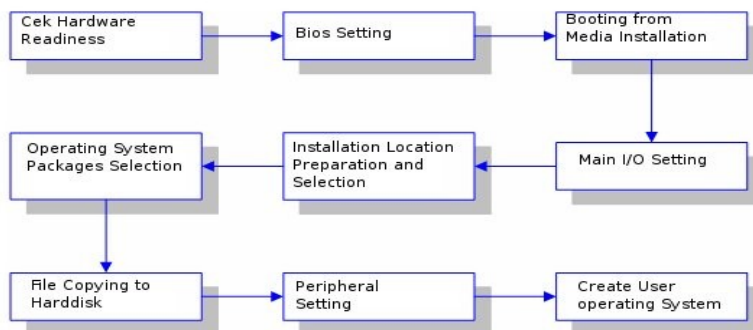


Figure 4.16. Operating system installation stages.

- Check hardware readiness. This stage confirms all hardware and its peripherals are correctly installed. Moreover, check the computer hardware specification whether it is supported by the operating system.
- Configure BIOS. This stage is to configure BIOS to give the first booting priority to the installation media, usually CD / DVD.
- Booting from the installation media. If BIOS configuration went well, then the computer should boot from the installation media. Figure 4.17 is the screenshot from the early stages of booting process.

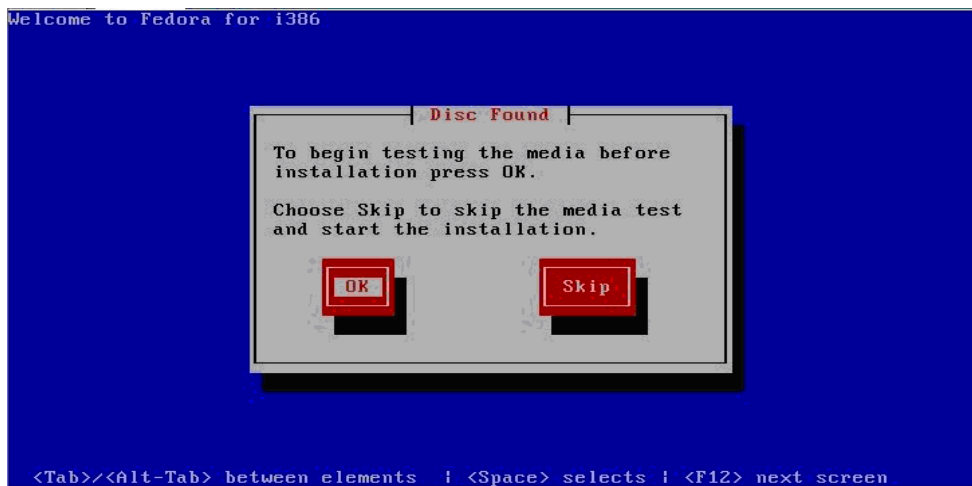


Figure 4.17. Testing of installation media.

- Configuration of main I/O. This stage is to arrange so that main Input / Output devices, namely, mouse, keyboard and the video, would correctly run and the installation process can be carried out.
- Preparation and determination of the location of the installation. Most installation is aimed for hard disk as its target. We must prepare the hard disk to be written. Preparation includes hard disk partitioning, including setting the volume size for each partition, and formatting of the partition as required by the operating system. Microsoft Windows may use either NTFS or FAT32 file system. While Linux may use ext2, ext3, ReiserFS, and XFS file system. Apple Mac OS X usually uses HFS+. Figure 4.18 shows the snap shot during setup of the installation location in Fedora Core 8 installation process.

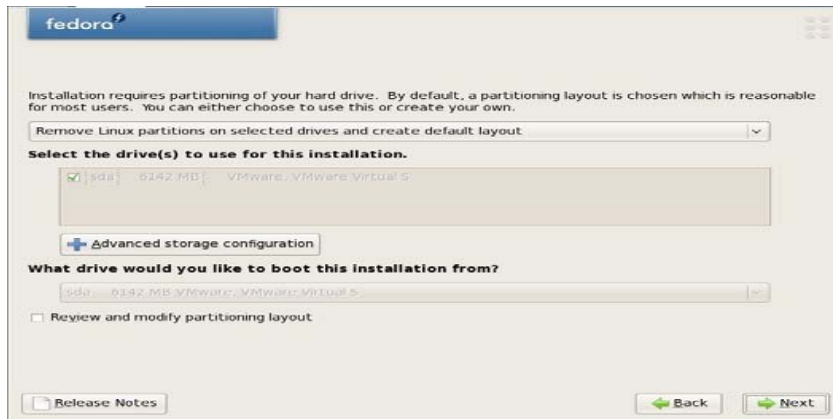


Figure 4.18. Process in determining the installation target.

- Configuration of application softwares to be installed. This stage may be skipped if *default* is chosen. For those who need to customized its operating system, it can be done by doing a *custom installation*. CD or DVD installation source has normally carried several applications to be selected during or after installation process.
- Write / copy to hard disk. After application selection process, application & operating system files will be written to hard disk. Figure 4.19 shows an example of screen shots during copy file process.

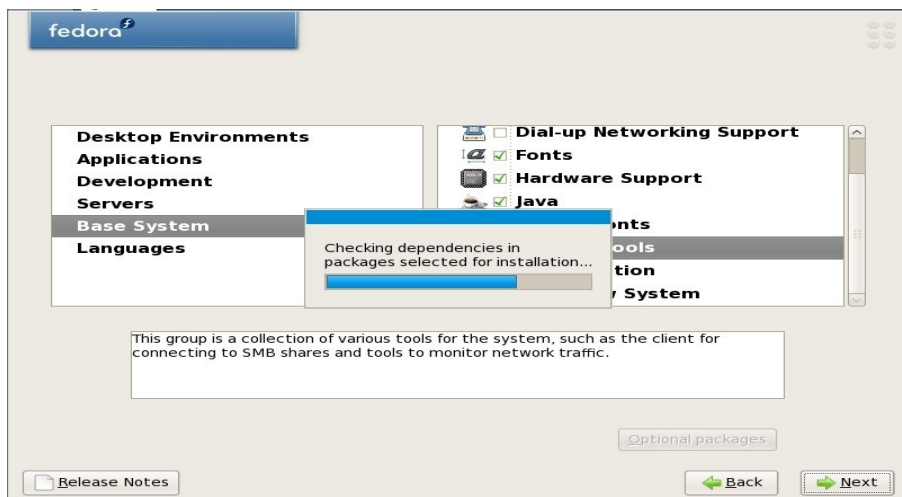


Figure 4.19. Copy file process in Fedora.

- Configuration of other peripherals. The objective of this stage is to install the needed driver for peripheral, such as, VGA card, sound card, *motherboard chip set*, etc to enable the computer in recognizing the peripheral. This stage is normally needed for Microsoft Windows operating system. It is not needed for those who use Linux as all drivers are normally built-in Linux kernel.
- Creating user. We need to create the user of the operating system. The user creation will provide username and password for each user. In general, there are two (2) level of user, namely, administrator and ordinary user. Administrator has the highest right to all section of operating system while ordinary user has a more limited right determined by the administrator.

4.3.2. Booting

Booting is the earliest process when the computer turns on. The booting process is shown in Figure 4.20. In the early booting process is executing a small program resides in computer's ROM with setup data as stored in CMOS. This stage is known as POST (*Power On Self Test*), after a successful completion of POST, BIOS code from ROM and BIOS of various adapters, such as, VGA adapter, will be loaded into the main memory (RAM). After loading all BIOS into the memory, computer starts reading the program starts up located in hard disk boot sector. At this stage, operating system will be loaded into RAM from hard disk.

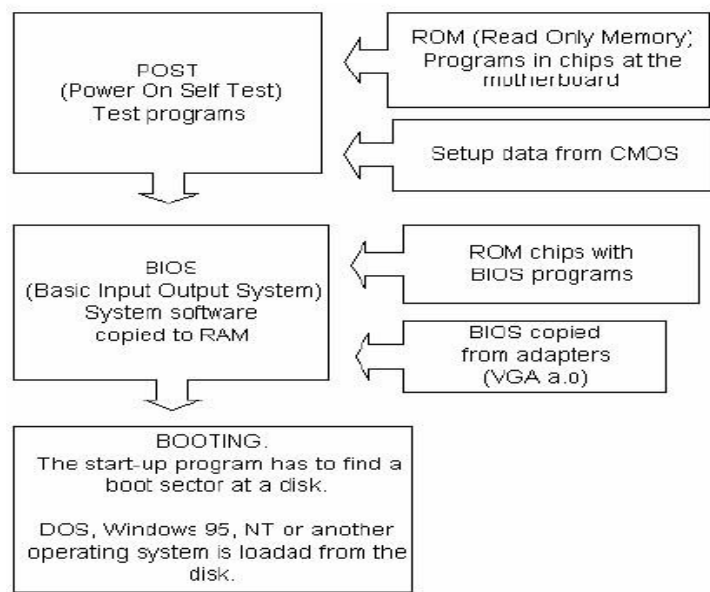
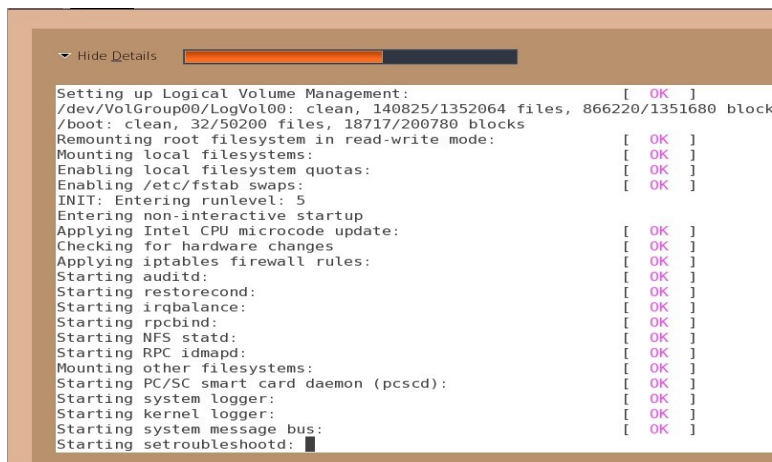


Figure 4.20. Early process of Booting.

In Microsoft Windows, a boot-splash usually Windows Logo is usually shown during boot process. No detail information to the users. In contrast, in Linux family, user may choose whether to show the detail information on booting process or put it behind a splash screen usually by using LILO or Grub. Figure 4.21 shows the booting process in Fedora Linux.



```

▼ Hide Details
Setting up Logical Volume Management: [ OK ]
/dev/VolGroup00/LogVol00: clean, 140825/1352064 files, 866220/1351680 block
/boot: clean, 32/50200 files, 18717/200780 blocks
Remounting root filesystem in read-write mode: [ OK ]
Mounting local filesystems: [ OK ]
Enabling local filesystem quotas: [ OK ]
Enabling /etc/fstab swaps: [ OK ]
INIT: Entering runlevel: 5
Entering non-interactive startup
Applying Intel CPU microcode update: [ OK ]
Checking for hardware changes [ OK ]
Applying iptables firewall rules: [ OK ]
Starting auditd: [ OK ]
Starting restorecond: [ OK ]
Starting irqbalance: [ OK ]
Starting rpcbind: [ OK ]
Starting NFS statd: [ OK ]
Starting RPC idmapd: [ OK ]
Mounting other filesystems: [ OK ]
Starting PC/SC smart card daemon (pcscd): [ OK ]
Starting system logger: [ OK ]
Starting kernel logger: [ OK ]
Starting system message bus: [ OK ]
Starting setroubleshootd: █

```

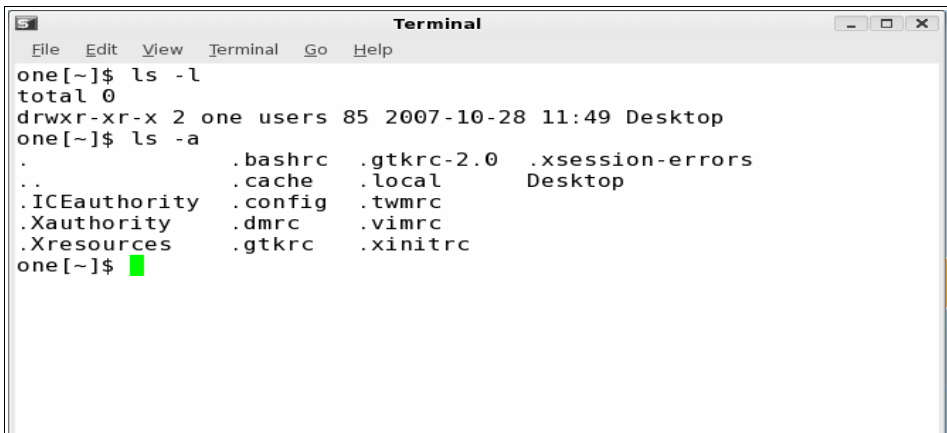
Figure 4.21 Booting process in Fedora Linux.

4.3.3. Text based commands

For some command line interface (CLI) is a nightmare, as one has to memorize all the command and not user friendly. There are several advantages with text based interface, such as:

- Faster execution.
- Less demanding on CPU and memory.
- No need for high performance hardware. No high resolution VGA card and monitor is needed.

CLI mode may be rarely used in GUI based operating systems, such as, Microsoft Windows or Apple Mac OS X. However, in Linux and Unix family, CLI mode is still being used especially for system and network administration. In this section, we will learn some of the frequently used CLI command that can be access through console, such as, Console, *xterm*, *aterm* etc in Linux as shown in Figure 4.22.



```

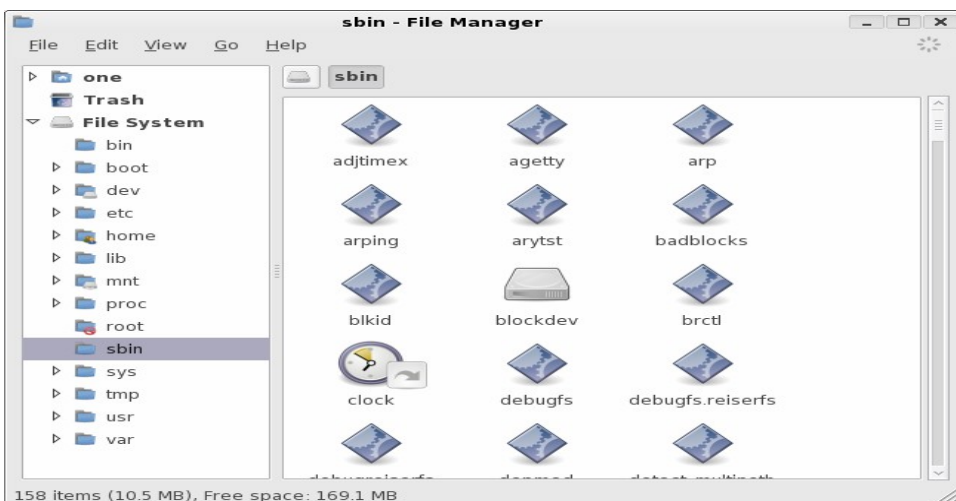
Terminal
File Edit View Terminal Go Help
one[~]$ ls -l
total 0
drwxr-xr-x 2 one users 85 2007-10-28 11:49 Desktop
one[~]$ ls -a
.          .bashrc    .gtkrc-2.0 .xsession-errors
..         .cache     .local      Desktop
.ICEauthority .config   .twmrc
.Xauthority .dmrc     .vimrc
.Xresources .gtkrc    .xinitrc
one[~]$

```

Figure 4.22. A terminal for CLI mode.

There are two (2) main groups of commands in CLI mode, namely:

- System administration commands. These commands can only be executed by those with administrator (*root*) privilege. These commands are normally stored in directory */sbin* (Figure 4.23) and */usr/sbin* (Figure 4.24).
- User commands. These commands may be accessed by ordinary users. These command is normally stored in directory */bin* (Figure 4.25) and */usr/bin* (Figure 4.26).

Figure 4.23. Commands in directory */sbin*.

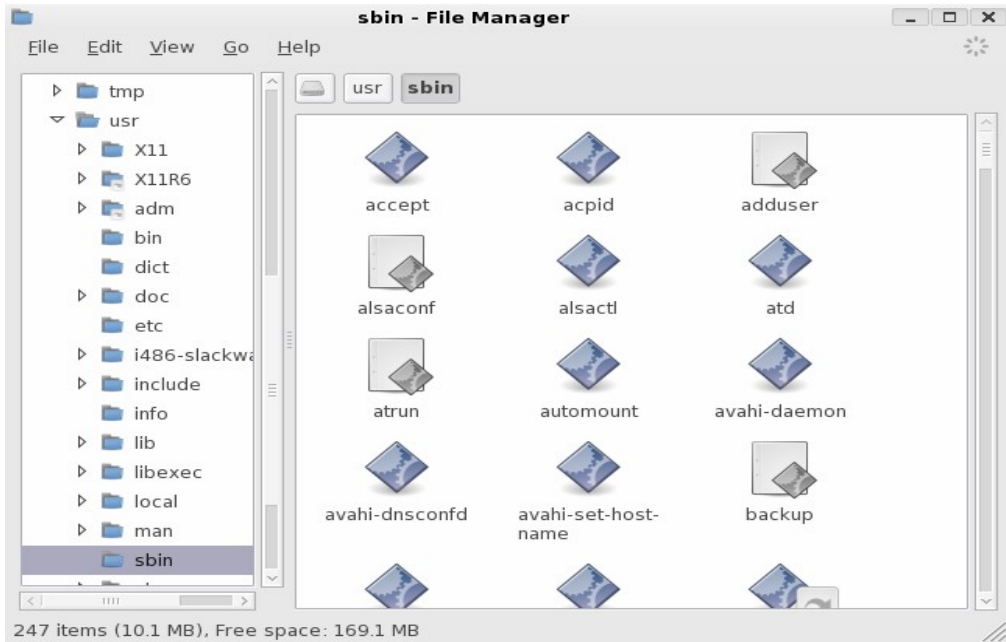


Figure 4.24. Commands in directory /usr/sbin.

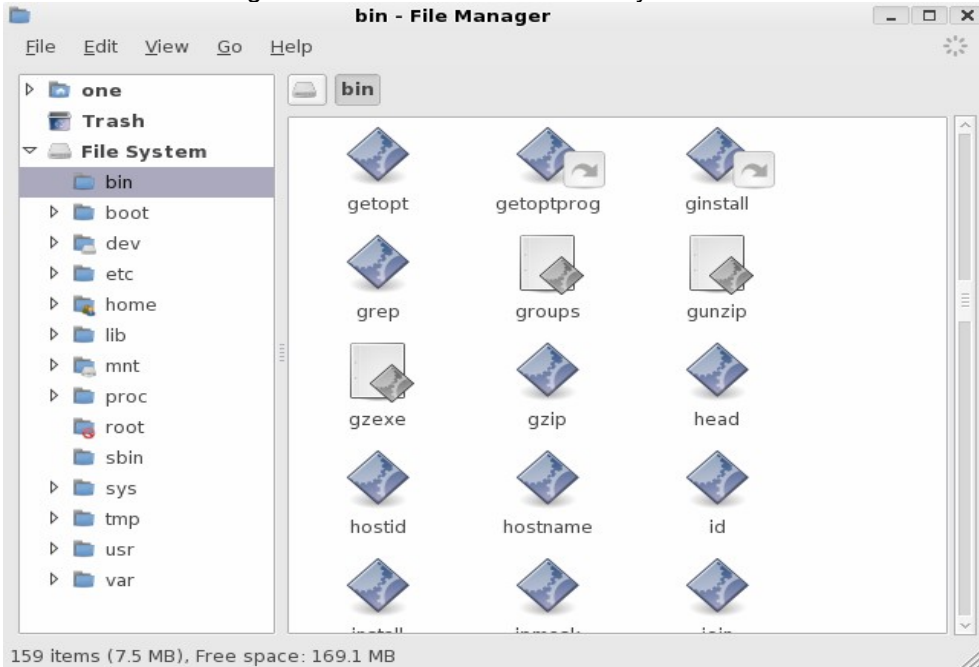


Figure 4.25. Commands in directory /bin.

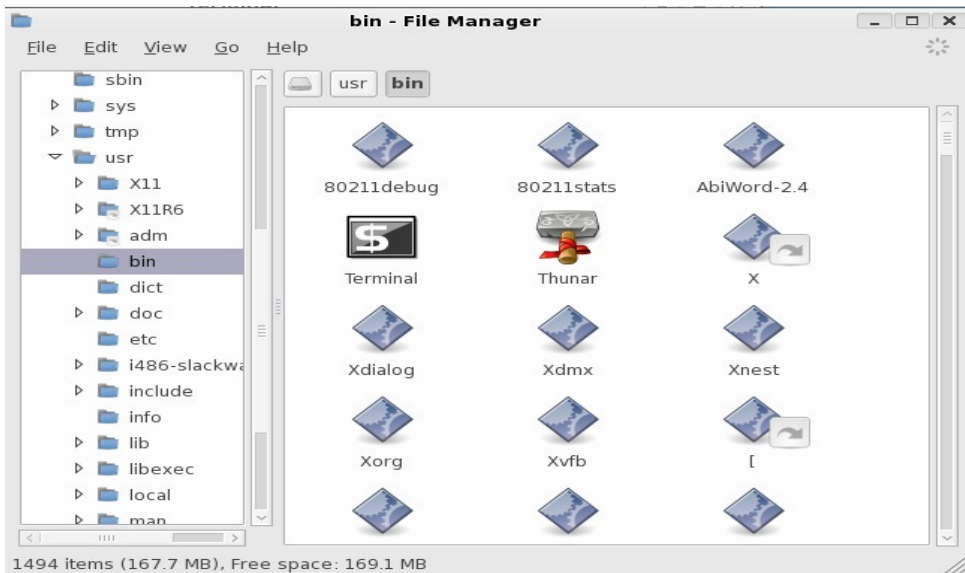


Figure 4.26. Commands in directory /usr/bin.

The followings are some of the important command in CLI mode:

- **Display the directory content**

To list the directory content, it can be done by using ls followed by other arguments as needed. Figure 4.27 shows how to use ls.

```

one[~]$ ls /usr/
X11      dict          include       local        spool
X11R6    doc           info         man         src
adm      etc          lib          sbin        tmp
bin      i486-slackware-linux  libexec      share

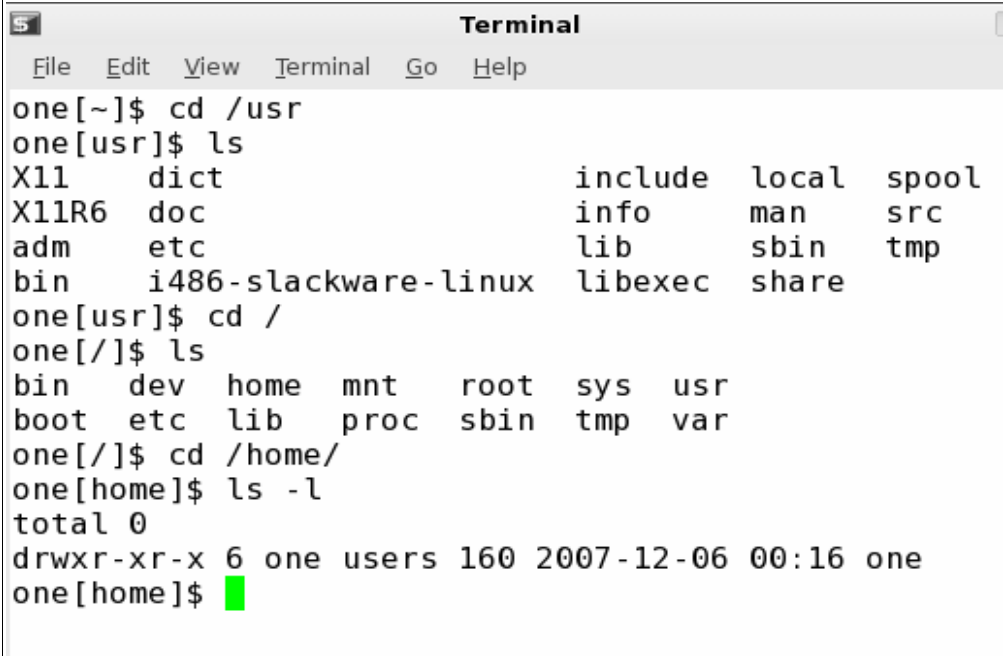
one[~]$ ls -l
total 0
drwxr-xr-x 2 one users 85 2007-10-28 11:49 Desktop
one[~]$ ls -a
.          .bashrc      .gtkrc-2.0   .xsession-errors
..         .cache       .local       Desktop
.ICEauthority .config     .twmrc
.Xauthority  .dmrc       .vimrc
.Xresources  .gtkrc      .xinitrc
one[~]$ ls --color /usr/
X11      dict          include       local        spool
X11R6    doc           info         man         src
adm      etc          lib          sbin        tmp
bin      i486-slackware-linux  libexec      share
one[~]$

```

Figure 4.27. Example of ls usage.

- **Change directory**

Change directory may be performed by using `cd` followed by the destination directory. Figure 4.28 shows the example of `cd` usage.

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal shows a user named "one" at a prompt. The user enters `cd /usr`, then `ls`, which lists the contents of /usr: `X11 dict include local spool`, `X11R6 doc info man src`, `adm etc lib sbin tmp`, and `bin i486-slackware-linux libexec share`. The user then enters `cd /`, and the prompt changes to `one[/]`. The user enters `ls`, which lists the root directory: `bin dev home mnt root sys usr`, `boot etc lib proc sbin tmp var`. The user then enters `cd /home/`, and the prompt changes to `one[home]`. The user enters `ls -l`, which shows the permissions and details for the /home directory: `total 0`, `drwxr-xr-x 6 one users 160 2007-12-06 00:16 one`. The prompt returns to `one[home]$` with a green cursor.

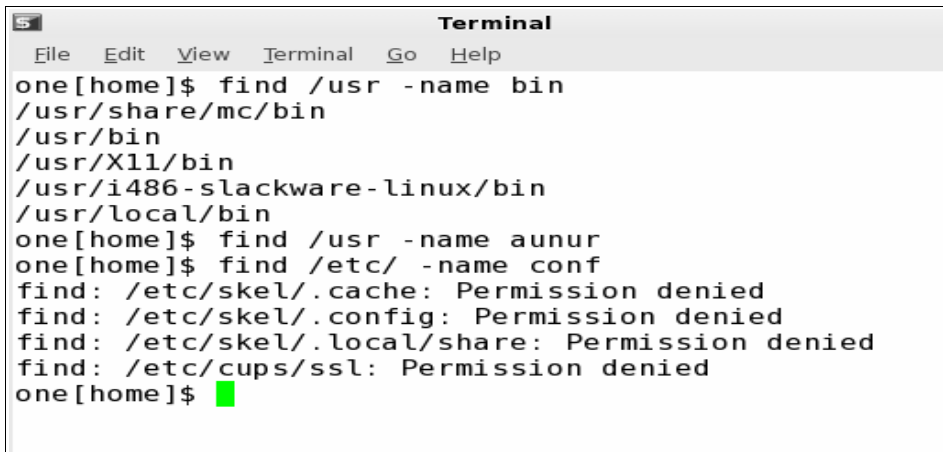
```
one[~]$ cd /usr
one[usr]$ ls
X11      dict      include  local    spool
X11R6    doc        info     man      src
adm      etc        lib      sbin     tmp
bin      i486-slackware-linux libexec  share

one[usr]$ cd /
one[/]$ ls
bin      dev      home     mnt      root     sys      usr
boot     etc      lib      proc     sbin     tmp      var
one[/]$ cd /home/
one[home]$ ls -l
total 0
drwxr-xr-x 6 one users 160 2007-12-06 00:16 one
one[home]$
```

Figure 4.28. Example of `cd` usage.

- **Locate a file**

To locate a file at certain location, the `find` command can be used. Figure 4.29 shows the example of `find` usage.

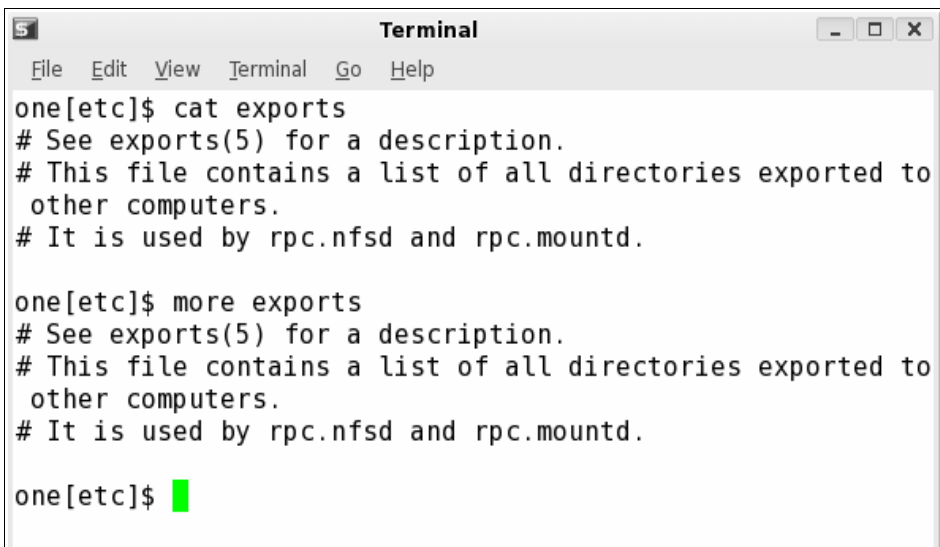
A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The prompt is "one[home]\$". The user enters "find /usr -name bin", and the output lists directories: "/usr/share/mc/bin", "/usr/bin", "/usr/X11/bin", "/usr/i486-slackware-linux/bin", and "/usr/local/bin". The user then enters "find /usr -name aunur", which returns no results. Next, the user enters "find /etc/ -name conf", and the output shows several files with "Permission denied" messages: "/etc/skel/.cache", "/etc/skel/.config", "/etc/skel/.local/share", and "/etc/cups/ssl". The prompt "one[home]\$ " is followed by a green cursor.

```
one[home]$ find /usr -name bin
/usr/share/mc/bin
/usr/bin
/usr/X11/bin
/usr/i486-slackware-linux/bin
/usr/local/bin
one[home]$ find /usr -name aunur
one[home]$ find /etc/ -name conf
find: /etc/skel/.cache: Permission denied
find: /etc/skel/.config: Permission denied
find: /etc/skel/.local/share: Permission denied
find: /etc/cups/ssl: Permission denied
one[home]$
```

Figure 4.29. Example of find usage.

- **Show file content**

To show the content of a file, we can use more, less or cat commands. Figure 4.30 shows some of the examples.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help) and window control buttons (-, square, X). The prompt is "one[etc]\$". The user enters "cat exports", and the output shows the contents of the file: "# See exports(5) for a description.", "# This file contains a list of all directories exported to other computers.", and "# It is used by rpc.nfsd and rpc.mountd.". The user then enters "more exports", and the output is identical. The prompt "one[etc]\$ " is followed by a green cursor.

```
one[etc]$ cat exports
# See exports(5) for a description.
# This file contains a list of all directories exported to
other computers.
# It is used by rpc.nfsd and rpc.mountd.

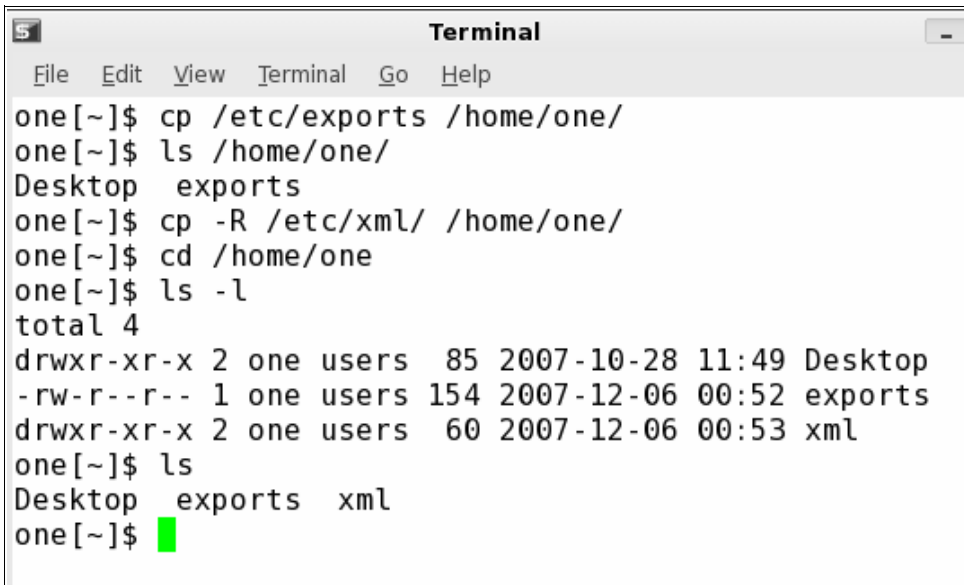
one[etc]$ more exports
# See exports(5) for a description.
# This file contains a list of all directories exported to
other computers.
# It is used by rpc.nfsd and rpc.mountd.

one[etc]$
```

Figure 4.30. Example of cat and more usage.

- **Copy file of directory**

cp command may be used to copy files of directory. Figure 4.31 shows the example of cp usage.

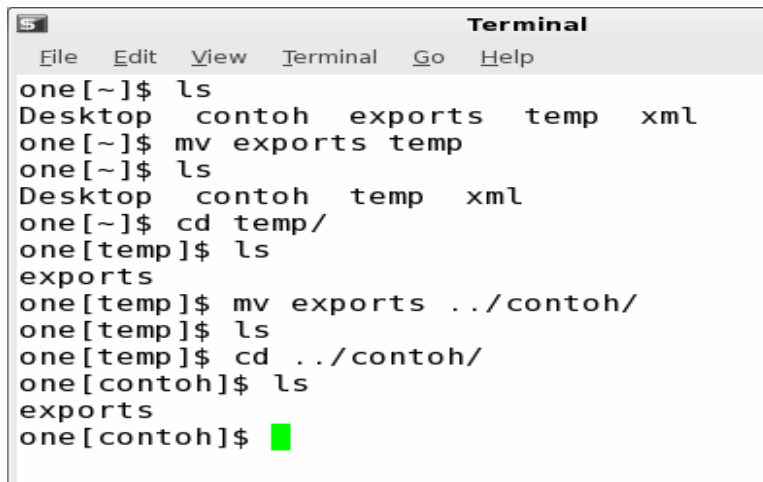
A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal shows a user named "one" at a prompt. The user enters the command "cp /etc/exports /home/one/". Then, they enter "ls /home/one/" which shows "Desktop" and "exports". Next, they enter "cp -R /etc/xml/ /home/one/". Then, they enter "cd /home/one". Finally, they enter "ls -l" which shows a detailed listing of four items: "Desktop" (a directory), "exports" (a file), and "xml" (a directory). The terminal ends with a prompt and a green cursor.

```
one[~]$ cp /etc/exports /home/one/
one[~]$ ls /home/one/
Desktop  exports
one[~]$ cp -R /etc/xml/ /home/one/
one[~]$ cd /home/one
one[~]$ ls -l
total 4
drwxr-xr-x 2 one users 85 2007-10-28 11:49 Desktop
-rw-r--r-- 1 one users 154 2007-12-06 00:52 exports
drwxr-xr-x 2 one users 60 2007-12-06 00:53 xml
one[~]$ ls
Desktop exports xml
one[~]$
```

Figure 4.31. Example of cp usage.

- **Move files.**

To move files, the mv command may be used. Figure 4.32 shows the example of mv usage.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The terminal shows a sequence of commands and their outputs. The user starts in the home directory (~) and lists files: Desktop, contoh, exports, temp, xml. Then they move 'exports' to 'temp'. They list files again: Desktop, contoh, temp, xml. Then they change directory to 'temp' and list files: exports. Then they move 'exports' to '../contoh/'. They list files again: exports. Then they change directory to '../contoh/' and list files: exports. The prompt returns to the home directory.

```
one[~]$ ls
Desktop  contoh  exports  temp  xml
one[~]$ mv exports temp
one[~]$ ls
Desktop  contoh  temp  xml
one[~]$ cd temp/
one[temp]$ ls
exports
one[temp]$ mv exports ../contoh/
one[temp]$ ls
exports
one[temp]$ cd ../contoh/
one[contoh]$ ls
exports
one[contoh]$
```

Figure 4.32. Example of mv usage to move files.

- **Rename a file**

The mv command may be used to rename a file. Figure 4.33. shows the examples of using mv command.

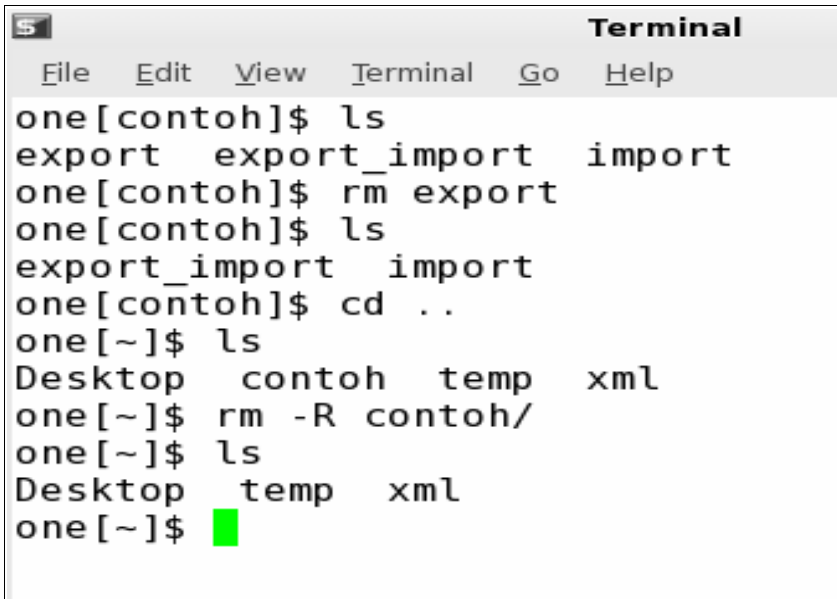
A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The terminal shows a sequence of commands and their outputs. The user is in the 'contoh' directory and lists files: exports. Then they rename 'exports' to 'import'. They list files again: import. Then they rename 'import' to 'export_import'. They list files again: export_import. The prompt returns to the 'contoh' directory.

```
one[contoh]$ ls
exports
one[contoh]$ mv exports import
one[contoh]$ ls
import
one[contoh]$ mv import export_import
one[contoh]$ ls
export_import
one[contoh]$
```

Figure 4.33. Example of mv usage to rename a file.

- **Remove files or directory**

To remove files or directory, rm command may be used. Figure 4.34 shows the example of rm usage.


A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The terminal shows a user named "one" in a directory "contoh". The user runs "ls" and sees "export", "export_import", and "import". Then they run "rm export". After another "ls", only "export_import" and "import" are visible. The user then runs "cd .." to go to the parent directory "~". A new "ls" shows "Desktop", "contoh", "temp", and "xml". Finally, the user runs "rm -R contoh/". A subsequent "ls" shows "Desktop", "temp", and "xml", with "contoh" removed. The prompt "one[~]\$ " is followed by a green cursor.

```
one[contoh]$ ls
export  export_import  import
one[contoh]$ rm export
one[contoh]$ ls
export_import  import
one[contoh]$ cd ..
one[~]$ ls
Desktop  contoh  temp  xml
one[~]$ rm -R contoh/
one[~]$ ls
Desktop  temp  xml
one[~]$
```

Figure 4.34. Example of rm usage to remove files or directory.

- **Make a directory**

To make a new directory, mkdir command may be used. Figure 4.35 shows the example of mkdir command.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The terminal shows a user named "one" in a directory "~". The user runs "ls" and sees "Desktop", "exports", and "xml". Then they run "mkdir contoh" and "mkdir temp". After another "ls", the terminal shows "Desktop", "contoh", "exports", "temp", and "xml". The prompt "one[~]\$ " is followed by a green cursor.

```
one[~]$ ls
Desktop  exports  xml
one[~]$ mkdir contoh
one[~]$ mkdir temp
one[~]$ ls
Desktop  contoh  exports  temp  xml
one[~]$
```

Figure 4.35. Example of mkdir usage.

- Understanding file and directory access right

In Windows operating system, there is no sufficient file protection for files and directories as it is only a limited attribute as shown in Figure 4.36.

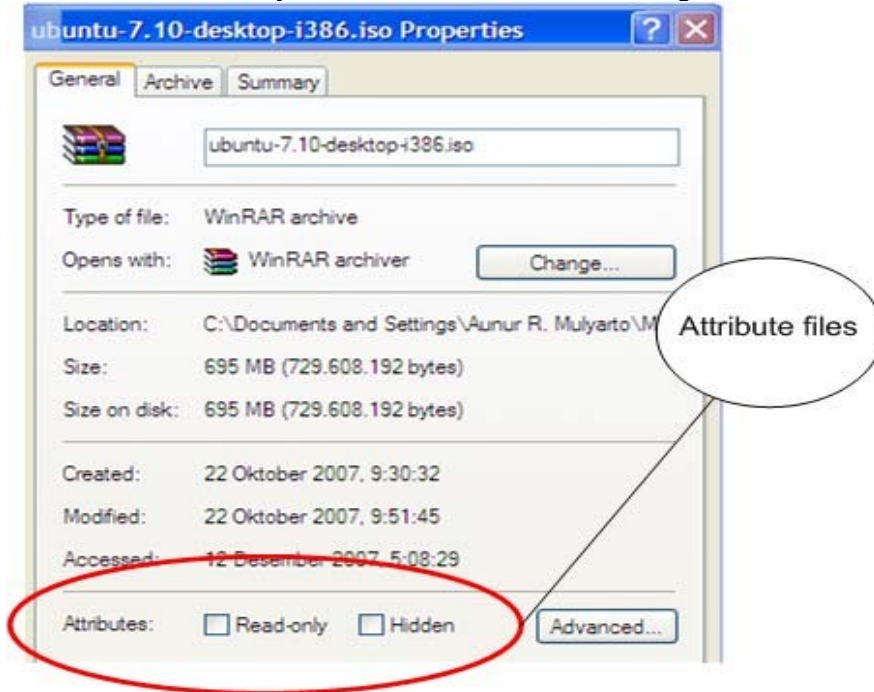


Figure 4.36. File / folder attribute in Microsoft Windows.

As shown in Figure 4.35, file / directory attribute are limited to Read-only and Hidden. If Read-only box is chosen, file can only be read. If Hidden box is chosen, file is hidden and will disappear from Windows Explorer.

In Unix family, including Linux, file / directory attribute is more tightly controlled. Thus, increase security and ease for users in managing their file and directories requirements.

There are four (4) important parameter in a file / directory, namely, attribute, user (owner) of the file / directory, group where the user belongs to, and the name of file / directory. The first three columns set the access right owner, the following three columns set the group's access right, and the last three columns set the other access right (other than owner and group member). The letter r indicates that the file / directory can be read, the letter w indicates it can be written, while the x letter shows that the file / directory may be executed. Please see carefully Figure 4.37.

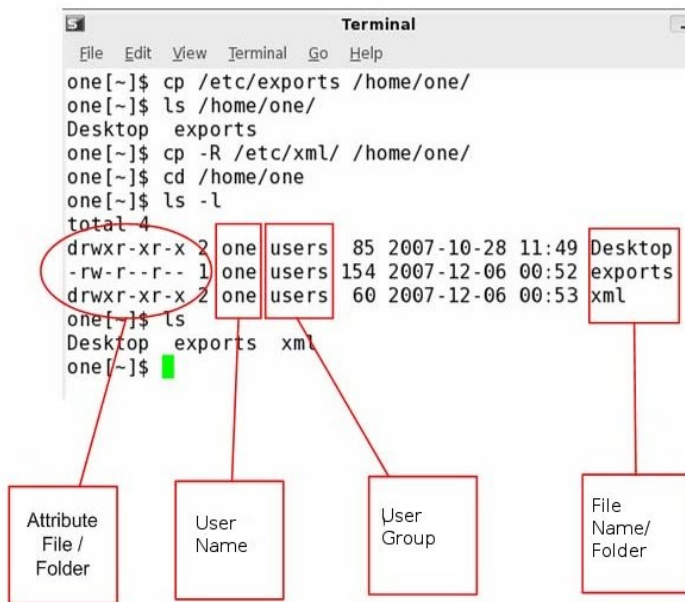


Figure 4.37. File / directory attribute in Unix family.

Figure 4.37 indicating the followings:

- **Desktop** and **xml** are directory as noted in mark **d**, whereas **exports** is a file noted by mark **-**.
- **Desktop** and **xml** have an attribute **drwxr-xr-x**, meaning the owner (**one**) has the right to read, write, and execute the directory. While the group (**users**) has the right only to read and execute. Other users may read and execute the directory.
- **Exports** has an attribute **-rw-r -- r--** meaning the owner (**one**) has the right to read and write into the file. While the group and others may only read the file.

To change attribute of files / directories, we can use the following commands:

Table 4.1. Commands related to file / directory management.

Commands	Function
chgroup [options] group file	Change the group ownership of files / directories.
chmod [options] attribute file	Change access right of files / directories.
chown [options] owner file	Change ownership of files / directories.

● Process control

Process control is an important part of Linux. Thus, all commands related to process control should be understood.

To see running process, we can use ps command. Figure 4.38 shows the example of ps usage.

```

one[~]$ ps
  PID TTY          TIME CMD
 3605 pts/0    00:00:00 bash
 3675 pts/0    00:00:00 ps
one[~]$ ps -f
UID          PID  PPID  C  STIME TTY          TIME CMD
one          3605 3603  0  06:40 pts/0    00:00:00 bash
one          3676 3605  0  07:01 pts/0    00:00:00 ps -f
one[~]$ ps -l
 F S   UID   PID  PPID  C PRI  NI ADDR SZ  WCHAN  TTY          TIME CMD
 0 S   1000  3605 3603  0  75   0  -   873 122738 pts/0    00:00:00 bash
 0 R   1000  3677 3605  0  77   0  -   574  -   pts/0    00:00:00 ps
one[~]$

```

Figure 4.38. Example of ps command usage.

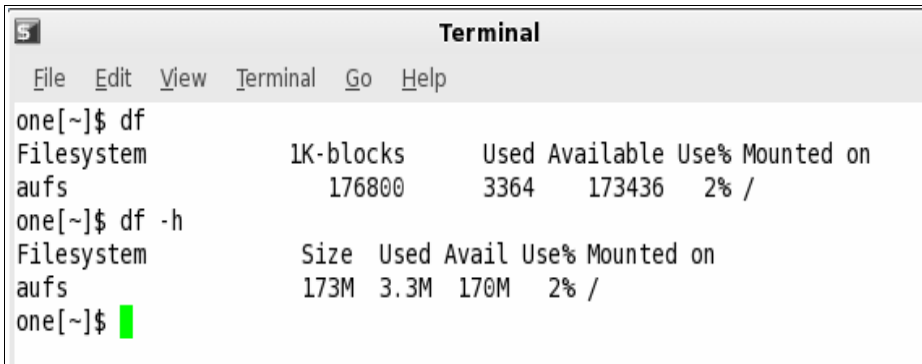
As shown in Figure 4.38, there are several options in ps command. The complete list of the options may be reviewed through command **man ps** in terminal. In the figure there are two (2) processes run by user **one**, namely, **bash with process ID (PID) 3605** and **ps -f with process ID (PID) 3676**.

To stop a running process, we can use **kill** command follows by the process

ID. For example, **kill 3605** will kill the bash process.

- **Checking disk space**

In many cases, we need to know the remaining space on our disk. Figure 4.39. shows the example of df usage to check on disk space.

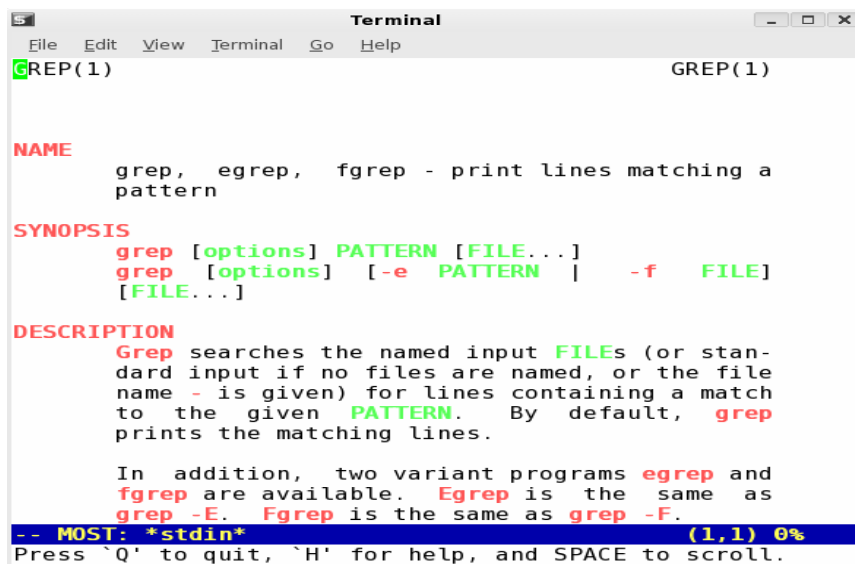


```

Terminal
File Edit View Terminal Go Help
one[~]$ df
Filesystem            1K-blocks      Used Available Use% Mounted on
aufs                  176800      3364    173436   2% /
one[~]$ df -h
Filesystem            Size  Used Avail Use% Mounted on
aufs                  173M  3.3M  170M   2% /
one[~]$
  
```

Figure 4.39. Example of df command usage.

There are a plenty of CLI in Linux. If you like to know more on the detail options of certain command, please use **man** follow by the command as shown in Figure 4.40.



```

Terminal
File Edit View Terminal Go Help
GREP(1)                                     GREP(1)

NAME
grep, egrep, fgrep - print lines matching a pattern

SYNOPSIS
grep [options] PATTERN [FILE...]
grep [options] [-e PATTERN] | -f FILE
[FILE...]

DESCRIPTION
Grep searches the named input FILES (or standard input if no files are named, or the file name - is given) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.

In addition, two variant programs egrep and fgrep are available. Egrep is the same as grep -E. Fgrep is the same as grep -F.

-- MOST: *stdin* (1,1) 0%
Press `Q' to quit, `H' for help, and SPACE to scroll.
  
```

Figure 4.40 Example of man usage to see detailed options of a command.

4.3.4. Work with GUI

GUI in general will ease the users in using an operating system via a mouse to carry out several orders. There are several functions of a mouse, some of them are:

- One click – to pick an object / file prior to any operation.
- Double click – to execute a command / a file, such as, open a folder and execute a file.
- Right click – to open a menu context as shown in Figure 4.41.
- Drag and drop – to move a file or an object from one place to another place as shown in Figure 4.42.

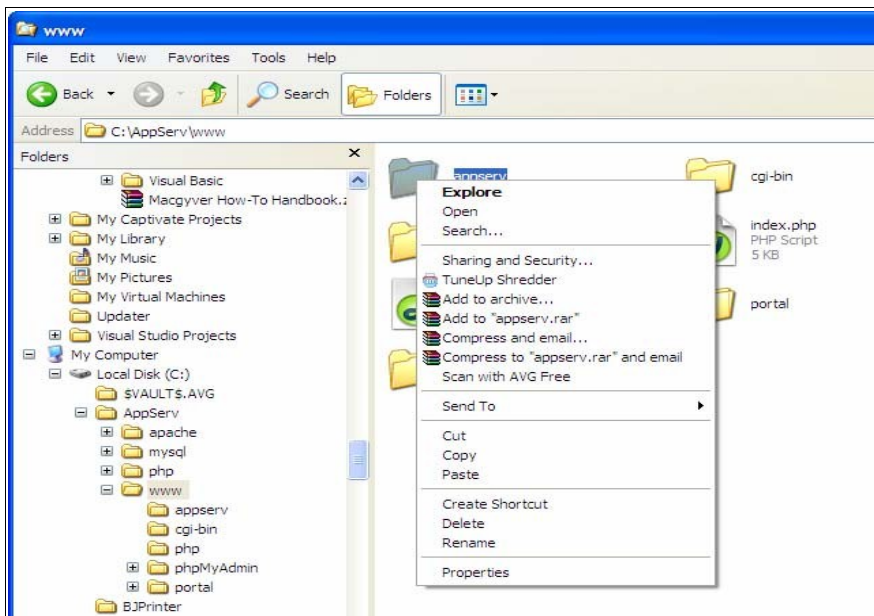


Figure 4.41. Right click to open a menu context.

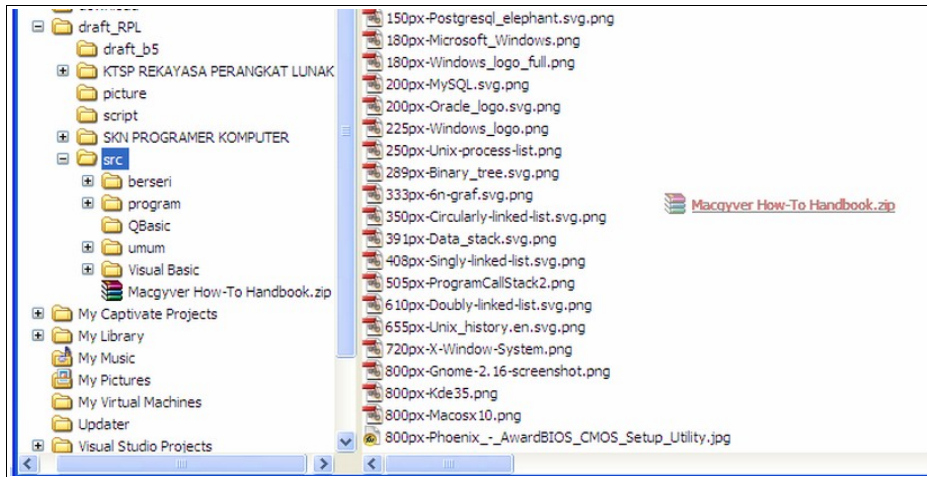


Figure 4.42. Drag and drop.

4.4. WORK IN COMPUTER NETWORK

Today, work in a networked computer is fairly common. Many sites with many computers will likely interconnect it into a network. Thus, basic knowledge in computer networking is very important.

4.4.1. Preparation

There are three (3) important things must be prepared prior to create a connection into computer network that is the hardware, the software and access to the network.

- **Hardware**

The required hardware would be depend on the type of network connection. To connect to a Local Area Network (LAN) a Network Interface Card (NIC) is needed. The driver software must be installed. To connect to a dial-up connection, a dial-up modem connected to a analog telephone cable is needed.



Figure 4.43. Network Interface Card

To see whether the required hardware, such as, NIC or phone modems, is correctly installed, it can be checked through the devices recognized by the operating system.

In Windows operating system, it can be done by right click on the My Computer icon in desktop and select Properties in the menu as shown in Figure 4.44. If we use System properties, select hardware tab and click on Device Manager (shown in Figure 4.44), so that Device Manager window is emerged as shown in Figure 4.45.

In Figure 4.45 may be seen a network adapter that has been correctly identified by the operating system. A question mark indicates that the hardware is not properly recognized as shown in Figure 4.45.

In Linux family, we check whether a device has been recognized or not by typing **lspci** as shown in Figure 4.46 and **ifconfig**.

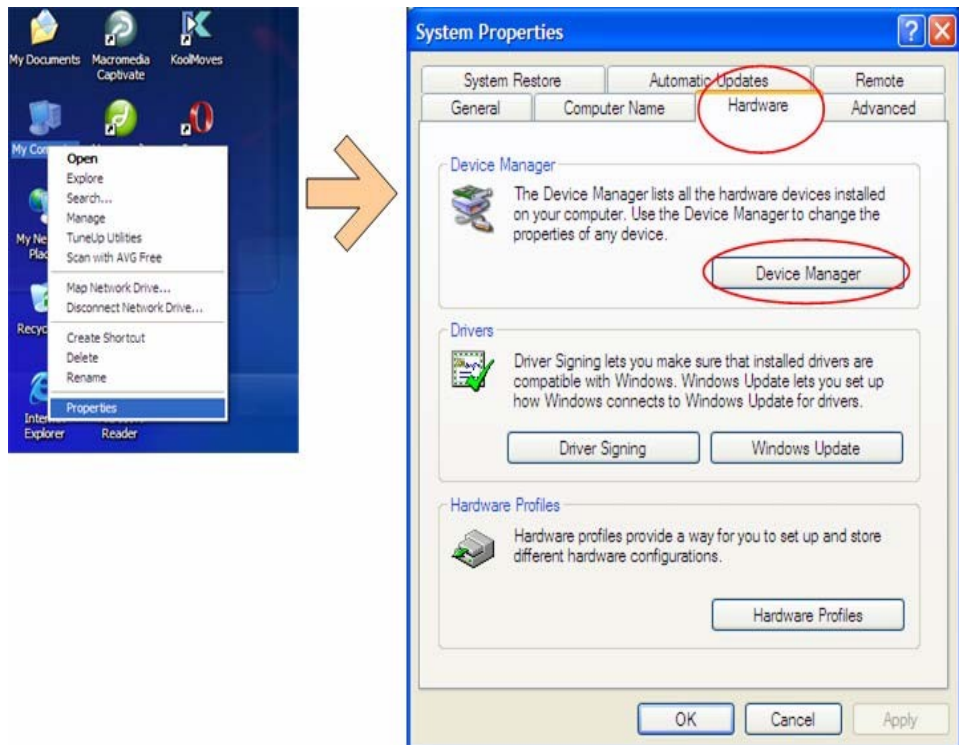


Figure 4.44. Open System Properties.

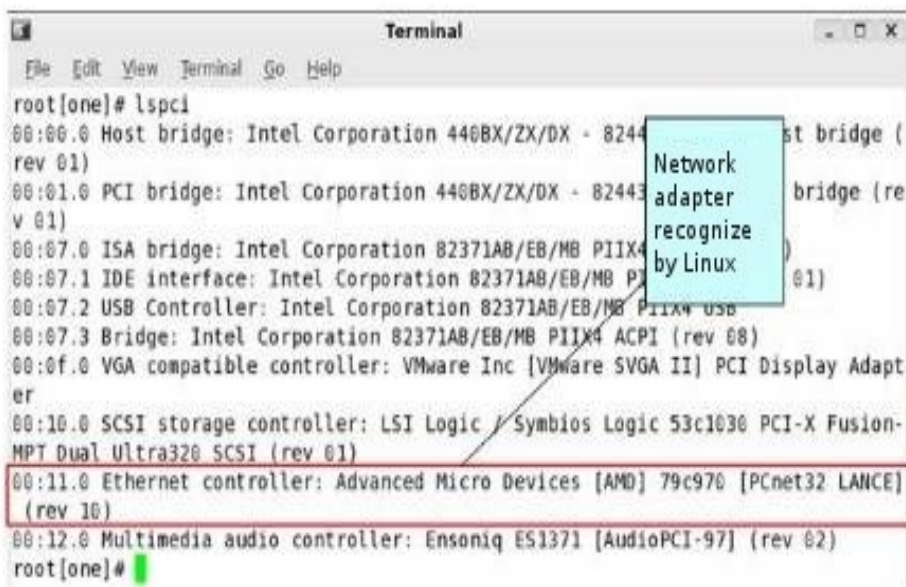


Figure 4.46. Output of lspci command to check on network adapter.

● Software

Apart from the operating system, the main software is the TCP/IP package needs to be correctly installed. In Windows operating system, it can be done

by double clicking on the connection type, in the properties windows, check if TCP/IP has been correctly installed & configured as shown in Figure 4.47.

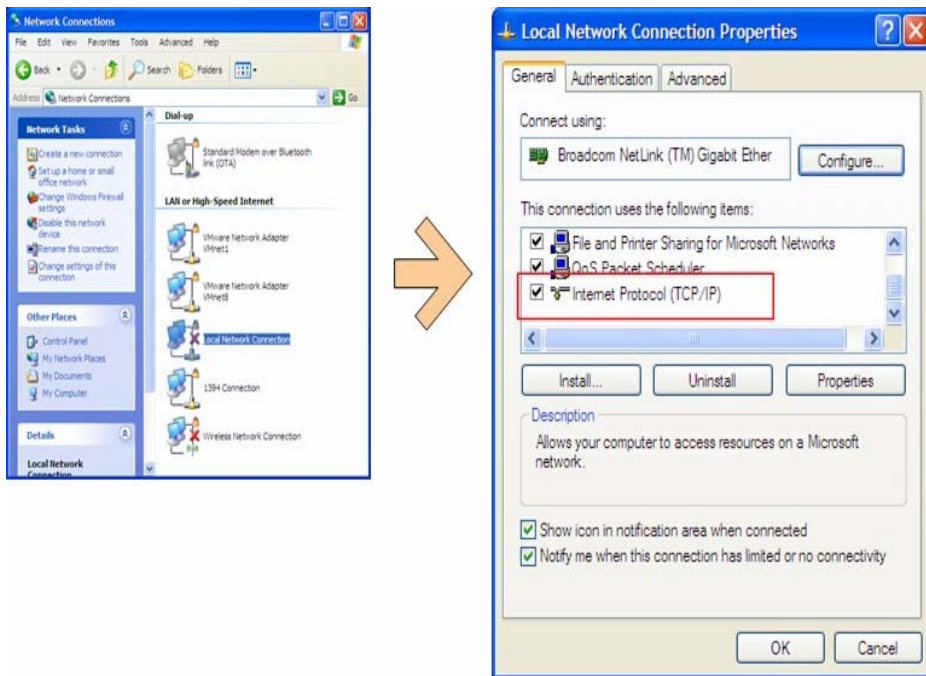


Figure 4.47. Check on TCP/IP protocol.

● Network Access

Access to network is related to the right to access the network, specially in getting an IP address and password to join the network. An automatic allocation of an IP address may be possible in a network with DHCP server. However, if there is no DHCP server, one needs to statically configure the IP address by consulting the network administrator.

4.4.2. Configuring network connection.

In today operating system, configuring a network connection is not a difficult task as most of it may be automatically performed by the operating systems. In a LAN with DHCP server, Windows computers (version 2000 or latest), and Linux workstations may be automatically connected to the network and automatically getting the IP address. In Windows operating system, network status may be checked in *systray* (*notification area*) located in the lower right hand side of the desktop as shown in

Figure 4.48.

```

Terminal
File Edit View Terminal Go Help
root[one]# lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443 (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443 (rev 01)
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 (rev 01)
00:07.2 USB Controller: Intel Corporation 82371AB/EB/MB PIIX4 USB
00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:0f.0 VGA compatible controller: VMware Inc [VMware SVGA II] PCI Display Adapter
00:10.0 SCSI storage controller: LSI Logic / Symbios Logic 53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI (rev 01)
00:11.0 Ethernet controller: Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10)
00:12.0 Multimedia audio controller: Ensoniq ES1371 [AudioPCI-97] (rev 02)
root[one]#

```

Figure 4.48. Network connection status.

4.4.3. File, printer and resource sharing

Main advantage of having a computer network is resource sharing, such as, file, printer, media (CD-RW or DVD-RW), scanner etc. In this section, file and printer sharing will be explained. Other resources may also be shared in fairly similar manner as file or printer.

- **File sharing.**

Directories and files in our computer may be shared in the network to be accessed by other computers vice versa.

To share a directory or a file in Microsoft Windows, the following stages may be used. Firstly, open the **Windows Explorer**, right hand side button click on the directory or file that we like to share and select **Sharing and Security**. In **Properties** windows select **Sharing** tab and in **Network sharing and security** check on **Share this folder on the network** and set the name of the shared directory as shown in Figure 4.49.

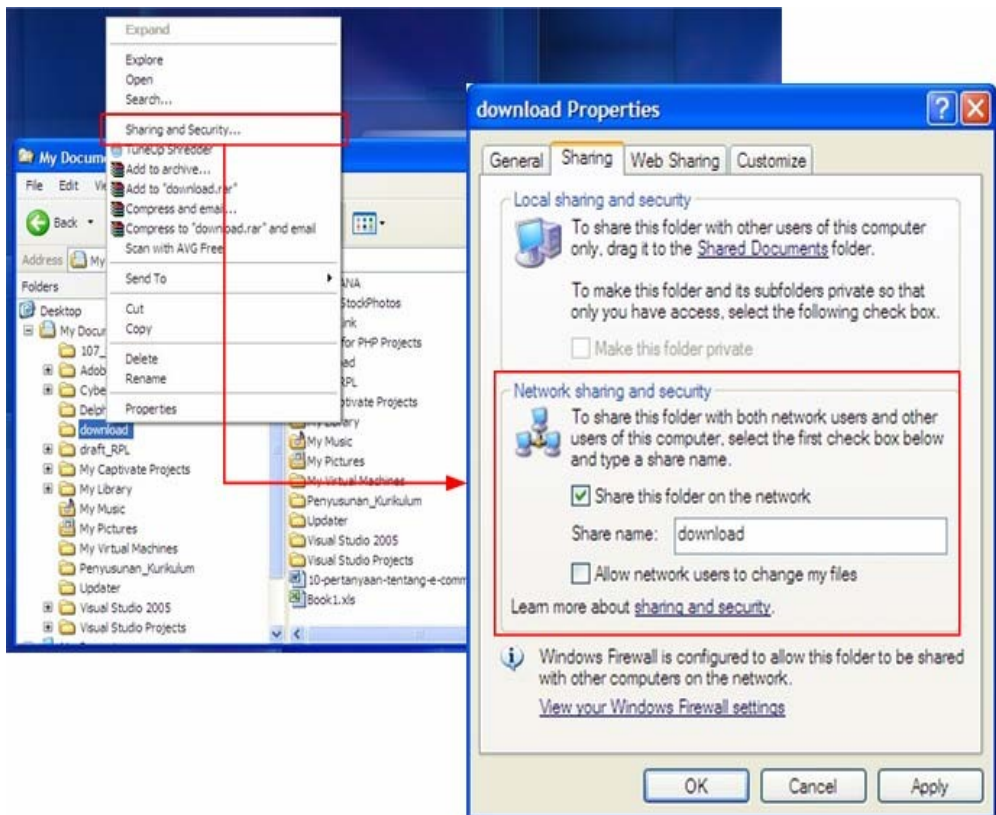


Figure 4.49. Managing File Sharing.

To be able to access a directory or files in other computer, we could use Windows Explorer, click on My Network Places -> Entire Network -> Microsoft Windows Network. We should find something like in Figure 4.50. From the Windows Explorer we can browse the shared resources on other computer on the network by clicking on the computer name.

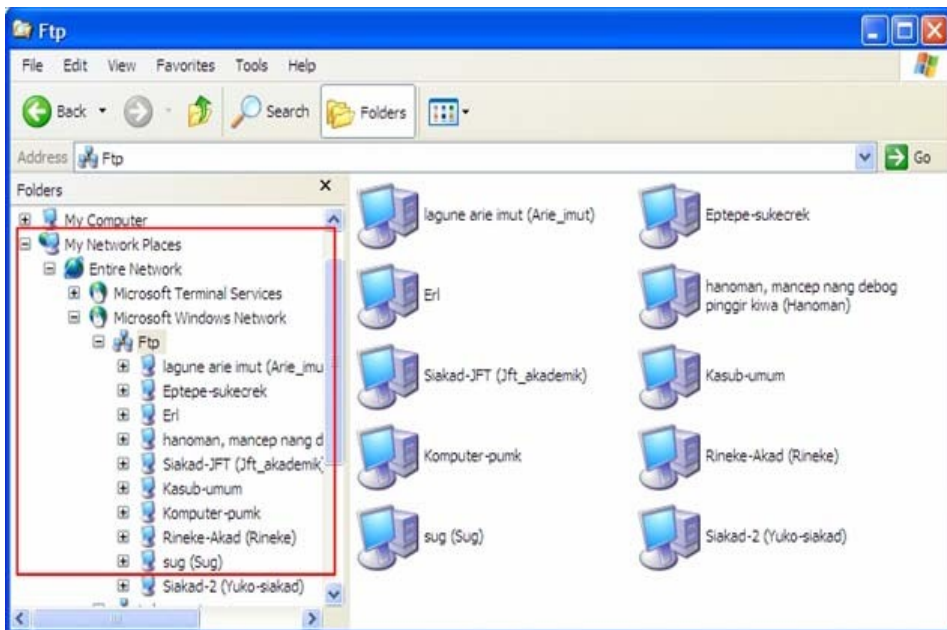


Figure 4.50. Browsing the computers on the network.

- **Printer sharing**

To share a printer on our computer, we need to do a fairly similar procedure as file sharing. Firstly, we need to open Printers and Faxes window through the following sequence Start -> Settings -> Printers and Faxes. After the window is opened, right click on the printer and select Sharing. After Properties windows is opened, select Sharing tab and click on Share this printer and provides the name of the shared printer as shown in Figure 4.51.

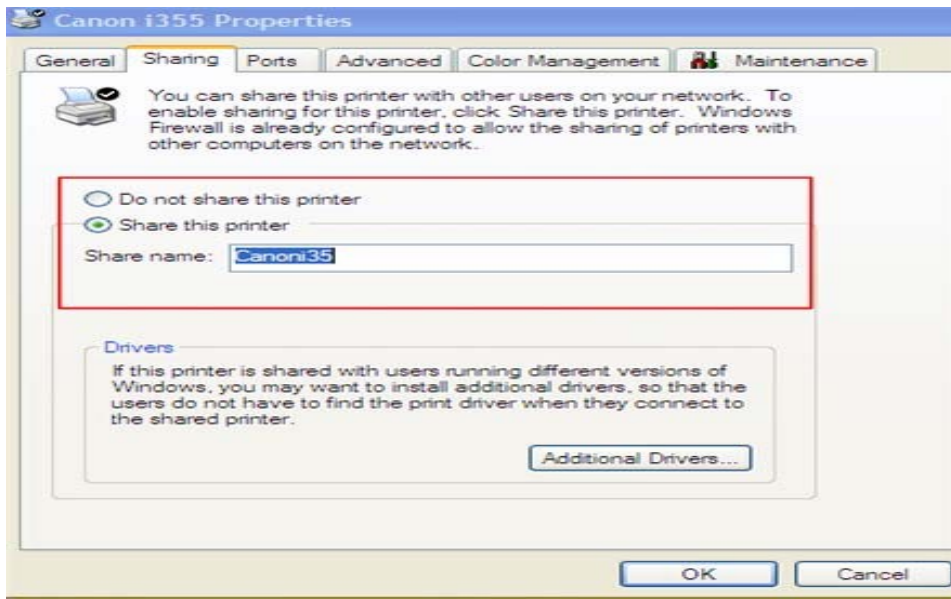


Figure 4.51. Printer sharing.

4.5. SUMMARY

- Operating system is the software that manages resource usage in a computer and provides user interface to access the resources.
- Operating system provides user interface, manages memory, files, processes and input / output.
- BIOS main function is to identify and recognize computer hardware and peripherals.
- There are several known operating system, namely, DOS, Windows, Mac OS, UNIX and Linux.
- Every operating systems must be installed prior to its usage.
- Operating system may use text based command or GUI depending on its configuration and facilities.

4.6. EXERCISE

1. Explain the term operating system.

2. Elaborate the functions of an operating system.
3. Describe the booting stages of a computer?
4. Please try to install one of Linux distro on a computer and observed the installation process. Compare the Linux installation process to Windows installation process. Describe the difference between the two installation processes?
5. Try to boot to Linux operating system, observes the booting process and compares to Windows booting process. Describe the difference?
6. Operate Linux operating system, open a terminal. Do a series of command using ls, cd, find, cat, cp, mv and mkdir. Take note what you find during the processes.
7. A file has the attribute `--rw-r -- r --` and owned by user rony. Describe the meaning of that particular attribute. How to configure so that other users have the right to read and write to that particular file?

CHAPTER 5 BASIC PROGRAMING ALGORITHM

Figure 5.1 shows a Russian stamp illustrating Muhammad ibn Mūsā al-Khwārizmī. For those who are active in the computer world should know him. He is an Islamic scientist many contribution in the field of mathematics, astronomy, astrology and geography and many becoming the foundation of modern science. From his name the term that will be studied in this chapter is emerged. From Al-Khawarizmi becoming **algorithm** in English and translated into **algoritma** in Indonesian.

Standard competence in basic programming is based on four (4) basic competences. In this book, each basic competence will be elaborated with exercises. The summary will be in the end of each chapter.
(source: www.wikipedia.org).



Figure 5.1. Muhammad ibn Mūsā al-Khwārizmī in a Russian stamp

The basic competence in this chapter is to explain the variable, the constant and type of data, create an algorithm / programming logic flow, applying array management, and file operation. Before proceeding please review the operation system, the principle of problem solving, and supporting materials such as mathematics.

At the end the chapter, some exercises will included from easy problems into difficult problems. This exercise is used to measure capacity in the basic competence. Thus, after studying the basic competence on your own or under teacher's guidance, you may measure your own capacity by doing the exercise.

OBJECTIVES

After studying this chapter, the reader should be able to:

- Explain the variables, the constants and the data types.
- Create algorithm / flow of programming logic.
- Apply management array.
- File operation.

5.1. VARIABLES, CONSTANTS, AND DATA TYPES.

Variables, constants and data types are three (3) parameters that always will be found during programming processes. Any programming language from the simplest to the most complex requires us to understand these three (3) matters.

5.1.1. Variables

A **variable** is a place that we could fill in a value, empty it, and recall it as needed. Every variable will have a **name (identifier)** and **value**. Please see the following example:

Example 5.1. variable name and value

```
username = "joni"
Name = "Al-Khawarizmi"
Price = 2500
Total Price = 34000
```

Shown in Figure 5.1, username, Name, Price and Total Price are the variable names. While, "joni", "Al-Khawarizmi", 2500 and 34000 are the value of each variable, respectively. These values will be kept in the name of their respective variable as long as no change is made.

In most programming languages, a variable must be initially declared to facilitate the *compiler* to work on it. If the variable is not declared then every time compiler met the new variable in the source code there will be a delay as the compiler create a new variable. This will slow down the compiler process. Moreover in several programming languages, compiler refuses to continue compiling process.

The variable name must follow the naming convention of the programming language. In general, there is a common convention among these programming languages. These conventions are:

- Variable name must be preceded with letter
- No space in the variable name. One may use underscore (`_`) to represent a space.
- Variable name should not contain special characters, such as: `.,+, -, *,/, <, >, &, (,)` etc.
- Variable name should not use key words of the programming language.

Example 5.2. Example of variable name.

Correct variable name	Wrong variable name
namasiswa	Nama siswa --- (space usage)
XY12	12X --- (starts with number)
harga_total	Harga.total --- (use . character as space)
JenisMotor	Jenis Motor --- (space usage)
alamatRumah	For --- (use programming language keyword)

5.1.2. Constants

A constant is a variable with fixed value and cannot be changed. Thus, a constant is also a variable but different in the value. A variable should be considered as a constant if no change in its value. In a source code, the value of a constant is normally declared in constant declaration. While a variable is usually determined by its name and data type without any value in it. Naming convention of variable as well as data type convention are also used for constants.

For example, if we write a program to do mathematical calculation that uses pi value (3,14159) that might be used in many places in the program, it would be much easier to use pi as a constant. The use of pi as a constant is much easier to write rather than repeatedly use 3,14159 in the source code.

5.1.3. Data Type

Data type is the type of data that may be used by the computer to satisfy in computer programming. Each variable or constant in source code should use a definite data type. A correct data type of variable or constant will save computer resources, especially memory. One of the important task of a programmer is to select the correct data type to produce an efficient and high performance program.

There are many available data types depending on the programming language. However, in general, data types may be grouped as shown in Figure 5.2.

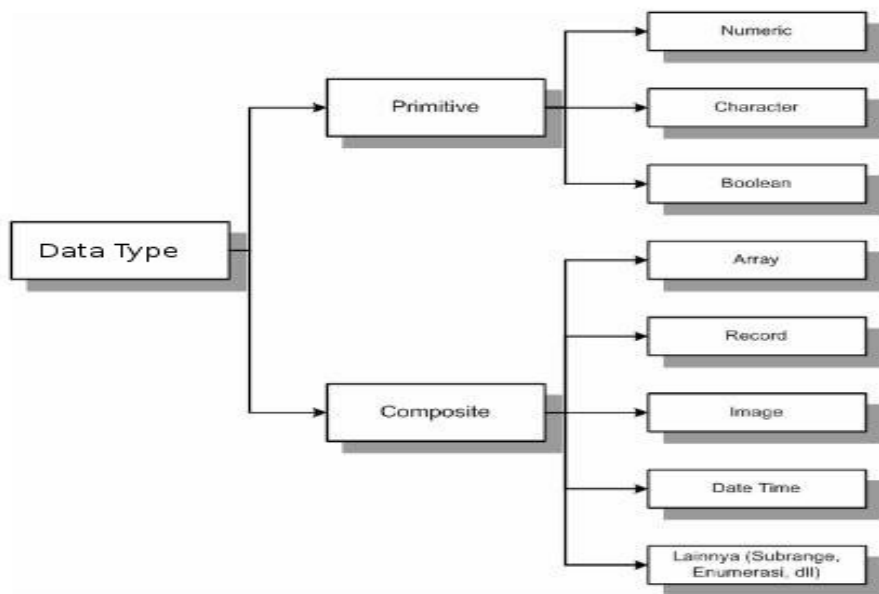


Figure 5.2. Data type grouping.

Data type *primitive* is the basic data type that directly available in the programming language. Whereas *composite* data type is data type that formed by two or three primitive data type.

Numeric data type

numeric data type is used in variable of constant to store a numerical value. All programming language should have a numerical data type, some differences in the numerical type it may accommodate.

Numeric data type is including integer (the integer) and float (fraction). In addition to type, some programming language adds precision into the data type, such as, single for limited precision of data type and double for more accurate data type. In the upcoming chapters discussing the application of programming language this section will be explained further.

The selection of data type for variable / constant should be carefully done. Manual or guide of respective programming language in data type section must be thoroughly read. Please see the following example.

Example 5.3. Numeric data type usage.

Program A

```
#include <iostream>
using namespace std;
int main() {
    int x, z;
    float y;
    x = 12;
    y = 2.15;
    z = x * y;
    cout << "X =" << x << endl;
    cout << "Y =" << y << endl;
    cout << "Z =" << z << endl;
    return 0; }
```

Result A

```
X =12
Y =2.15
Z =25
```

Program B

```
#include <iostream>
using namespace std;
int main() {
    int x;
    float y, z;
    x = 12.8;
    y = 2.15;
    z = x * y;
    cout << "X =" << x << endl;
    cout << "Y =" << y << endl;
    cout << "Z =" << z << endl;
    return 0; }
```

Result B

```
X =12
Y =2.15
Z =25.8
```

Program C

```
#include <iostream>
using namespace std;
int main() {
    int x;
    float y, z;
    x = 12;
    y = 2.15;
    z = x * y;
    cout << "X =" << x << endl;
    cout << "Y =" << y << endl;
    cout << "Z =" << z << endl;
    return 0; }
```

Result C

```
X =12
Y =2.15
Z =25.8
```

The above three (3) source codes (A, B and C) are written in C++. It is fairly similar, the only difference is in the data type. In program A, variable x and z declare as data int (integer) while y is floating point. The result is not correct. Z as X multiply by Y should be 25.8 (results of 12×2.15). However, since Z declares as data int, the result is 25.

Among the three (3) source code, only source code C provides the correct result. Why source code B is wrong? Please pay attention to the bold printed in the source code and pin point the error.

Character

Together with numeric data type, character data type is the frequently used data type. Character data type is often known as char or string. String data type is used to store text or anything between quotation marks ("...") or reaped single ('...'). Pay attention to the following example.

Example 5.4. Character data type usage.

Source code	Result
<code>#include <iostream></code>	<code>X = 5</code>
<code>using namespace std;</code>	<code>Isi variabel huruf</code>
<code>int main() {</code>	<code>= A</code>
<code> int x;</code>	<code>Isi variabel kata =</code>
<code> x = 5;</code>	<code>Java</code>
<code> char huruf = 'A';</code>	
<code> char* kata = "Java";</code>	
<code> cout << "X = " << x << endl;</code>	
<code> cout << "Isi variabel huruf = "</code>	
<code> << huruf << endl;</code>	
<code> cout << "Isi variabel kata = " <<</code>	
<code> kata << endl;</code>	
<code> return 0; }</code>	

In this example, we declare the variable *x* as int (integer), whereas the variable *huruf* and *kata* uses char data type (character). Note the execution result.

Boolean

Boolean data type is used to keep True / False value. Most language uses non-zero value as true and zero (0) as false. Boolean data type is often used to set a decision, such as on branching like IF... THEN or IF... THEN... ELSE.

Array

Array is a simple structured data. Array will store data with the same (homogeneous) data type in a variable. Each data location is indexed that used as the address of the data. More detailed explanation will be described in the chapter.

Record or Struct

As in Array, Record or Struct is a composite data structure. Record is used in Pascal / Delphi whereas Struct is used in C++. Different from array, record data type uses heterogeneous data type. For example, array may store a lot of single type of data, such as integer. Whereas in record, it can accommodate many data with different type of data in a single record, for example, one section may be integer, some section may be character, and other part may be boolean. Record usually accommodates data of an object. For example, a student may have name, address, age, birthplace, and date of birth. Name will use data string, address uses data string, age uses data type single (numeric), birthplace uses data string and date of birth uses data type date. The following is the example of record declaration in Delphi.

Example 5.5. Data type record declaration in Delphi.

```
Type TRecord_Siswa = Record
        Nama_Siswa   : String[30]
        Alamat       : String[50]
        Usia          : Real
    EndRecord
```

Image

Image or picture uses data type graphics. For example, graphics of the development of SMK students, family photo, trip video etc. In modern programming languages especially visual based programming language will likely have a good image support.

Date Time

Date and time is stored in specific format. Variable or constant declares with date data type may keep both date and time. This data type is categorized as composite data type as it forms by several data types. The following is an example of data type written in Visual Basic.

Example 5.6. Usage example of date time data type in Visual Basic.

```
Dim WaktuLahir As Date
WaktuLahir = "01/01/1997"
WaktuLahir = "13:03:05 AM"
WaktuLahir = "02/23/1998 13:13:40 AM"
WaktuLahir = #02/23/1998 13:13:40 AM#
```

Other data type

Subrange

Data type subrange has a range of value sets by the programmer. Such data type has a limited minimum and maximum value. Such data type is support by Delphi. The following is an example of data type subrange in Delphi.

Example 5.7. Data type subrange declaration in Delphi.

```
Type
    BatasIndeks = 1..20
    RentangTahun = 1950..2030
Var
    Indeks          : BatasIndeks
    Tahun           : RentangTahun
```

Enumeration

The following data type has sequential integer element. Each element is mapped to a name variable written in bracket. Such data type may be found in Delphi and declarative programming language such as SQL. The following is an example of data type enumeration in Delphi.

```
Example 5.8. Usage of data type enumeration.
Type
    Hari_dlm_Minggu = (NoI, Senin, Selasa, Rabu,
                       Kamis, Jumat, Sabtu,
                       Minggu)
    Nama_Bulan = (NoI, Januari, Pebruari, Maret,
                  April, Mei, Juni, Juli,
                  Agustus,
                  September, Oktober, Nopember,
                  Desember)
Var
    No_Hari          : Hari_dlm_Minggu
    No_Bulan         : Nama_Bulan
```

In the above example, data type Hari dlm Minggu includes the enumeration ranging from zero, Monday up to Sunday mapped to value 0, 1, to 7. Whereas data type Nama_Bulan includes the enumeration ranging from zero, January up to December mapped to value 0, 1, to 12.

Object

Data type object is used to store value related to objects provided by Visual Basic,

Delphi and other GUI based programming language. For example, we have a form with Command button named Command1, we may declare the variable as follows.

Example 5.9. Data type object usage example.

```
Dim A As Command Button
Set A = Command1
A.Caption = "HEY!!!"
A.FontBold = True
```

In this example, variable A is declared as data type object as Command Button. Next, we may set variable A as control Command button on the form (Command1). Thus, we can access all *property*, *method* and *event* of Command1 object using variable A.

Variant

This type of data type is only in Visual Basic. This is the most flexible data type as it may accommodate all type of data type.

5. PROGRAMING ALGORITHM STRUCTURE

5.2.1. Algorithm Concept

Algorithm is a sequential logical stages to systematically solve a problem. Problem may be anything, but noted that all problems must have an initial condition prior to work on its algorithm. Algorithm concept may be similar to recipe. A recipe usually has a list of needed materials or spices, and sequence and how to build or to cook it. If the material is not available or not calibrated then the recipe may not be built / cook. Likewise if cooking stages is not sequential, a correct result will not be received.

Different algorithms may be applied to a problem with same condition. Algorithm complexity level may be measured by the amount of computation to solve a problem. In general, algorithm with short execution time to solve a problem would have low complexity, while algorithm with long execution time may have much higher complexity. Please note pn the following simple algorithm.

Example 5.10. Algorithm to calculate area of a triangle.

1. Start
2. Read data on base length and height.
3. Area is base multiply by half of the height.
4. Show the area.

5. Stop

The above algorithm is a simple algorithm with only five steps. In this algorithm, there is no repetition or selection processes. All the steps are carried out by only one pass.

For a glance, the algorithm looks right. However, as we look closer the algorithm contains a big fundamental mistake, namely, no limit on value of base and height. What would happen if value of base length and height is zero (0) or negative? The result would not be correct. To make sure all input data conforming the requirement, if input of base length and height less than 0 then the program will be terminated. Thus, the following algorithm will be used.

Example 5.11 Refinement of the algorithm to calculate the area of a triangle.

1. Start
2. Read data on base length and height.
3. Check base length and height, if value of base length and height more than zero (0) then continue to step 4 else stop.
4. Area is base multiply by half of the height
5. Show the area
6. Stop

From the above explanation, the main summary of an algorithm are, firstly, **an algorithm must be correct**. Secondly, **an algorithm must stop**, and after stop, **an algorithm must provide the correct result**.

5.2.2. How to write Algorithm

There are three (3) methods to write an algorithm, namely,

- **Structured English (SE)**

SE is a good enough tool to depict an algorithm. SE is basically an English, but we can modify it using Indonesian and call it Structured Indonesian (SI). The algorithm such as in Example 5.10 and 5.11 are written in SE. Since it uses daily language, SE is suitable to be used in communicating algorithm to software users.

- **Pseudocode**

Pseudocode resembles SE. Due to its resemblance, SE and Pseudocode are considered to be the same. *Pseudo* means imitation or resembled, whereas *code* refers to program code. Thus, *pseudocode* means code that resembles the instruction of the program code. Pseudocode is based on the actual programming language, such as, BASIC, FORTRAN or PASCAL. PASCAL

based *pseudocode* is often used. Sometimes, *pseudocode* refers as *PASCAL-LIKE* algorithm. If Example 5.10 is written in pseudocode based on BASIC, it appears as follows.

Example 5.12. Pseudocode.

1. Start
2. READ alas, tinggi
3. $\text{Luas} = 0.5 * \text{alas} * \text{tinggi}$
4. PRINT Luas
5. Stop

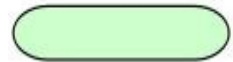
In Example 5.12, the algorithm resembles BASIC. READ and PRINT statements are the *keywords* in BASIC. By using pseudocode, translation process from algorithm to source code can be made easier.

● **Flowchart**

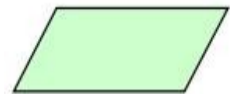
Flowchart is scheme / *chart* that shows the program *flow* in logical manner. Flowchart is a tool to show algorithm using certain notations / symbols. The following section will discuss it in a more detail.

There are several important symbols used to show an algorithm as shown in Figure 5.3.

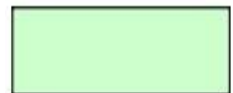
This notation is known as Terminator to show the beginning and the end of an algorithm



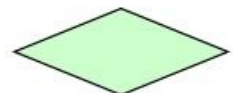
This notation is known as Data to represent the data input or output or to represent data entry operation or printing the result.



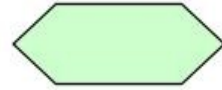
This notation is known as Process to represent a process.



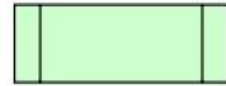
This notation is known as Decision to represent condition selection in a program



This notation is known as Preparation to set starting value, end value, the increase / decrease value of the counter.



This notation is known as Predefined Process to show the process is done by other subprocess, such as, procedure, sub-procedure, function.



This notation is known as Connector to show continuation of flowchart from one page to another page.



This notation is known as Arrow to show the data flow from one process to another process.



Figure 5.3. Symbols used in flowchart.

There are two (2) types of flowchart, namely, program logic flowchart and detailed computer program flowchart. The program logic flowchart is used to show each step of the computer program in a logical manner and usually is prepared by an system analyst. Whereas the detailed computer program flowchart is used to show the detail instruction- of the computer program and usually prepared by a programmer. If the Example 5.10 is made into program flowchart, it is shown in Figure 5.4.

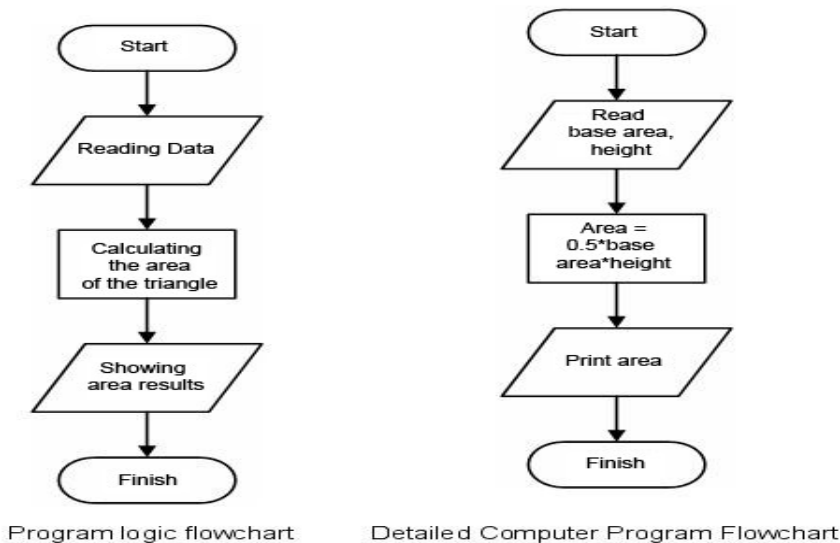


Figure 5.4. Program flowchart.

5.2.3. Sequential Algorithm.

There are three (3) foundation in algorithm structure, namely, sequencing, branching and looping. An algorithm will usually uses the three (3) structures to solve a problem.

In this section, we will firstly discuss sequential algorithm structure. The sequential structure is similar to a car run in a straight road and no turn or intersection as shown in Figure 5.5. The car will pass kilometers of road until it reaches its destination.



Figure 5.5. Car travels in straight road.

The sequential structure consists of one or more instruction. Each instruction is sequentially executed based on its sequence in source code, and executed after one instruction is completed. Instruction sequence sets the final stage of the algorithm. If the sequence is changed, the final result may be changed. According to Goldshlager and Lister (1988) the sequential structure follows the following rules:

- Each instruction is executed one by one.
- Each instruction is carried out very precise, none is repeated.
- Instruction sequence in the process is the same is action sequence in the algorithm.
- At the end of the final instruction is the end of the algorithm.

Example 5.13. Flowchart to calculate an area.

Create a flowchart to calculate:

1. Volume of a block.
2. Area of a circle.

Solution:

This exercise is a problem for sequential algorithm as no branching and looping is necessary. To calculate the volume of a block, we need to decide on the needed input and output variable. To calculate the volume of a block we need to know the length, width, and height of the block. Volume of a block will be the output. On area of a circle, radius is the input and area will be the output. To calculate the area of a circle, we will need the phi constant. Flowchart of these two (2) problem is shown in Figure 5.6.

Example 5.14. Flowchart to convert temperature.

Create a flowchart to calculate the temperature conversion from Fahrenheit to Celsius using the following formula $^{\circ}\text{C} = 5/9 \times (^{\circ}\text{F} - 32)$.

Solution:

This exercise uses sequential algorithm. The input variable is F and the output variable is C. Flowchart of this exercise may be seen in Figure 5.7.

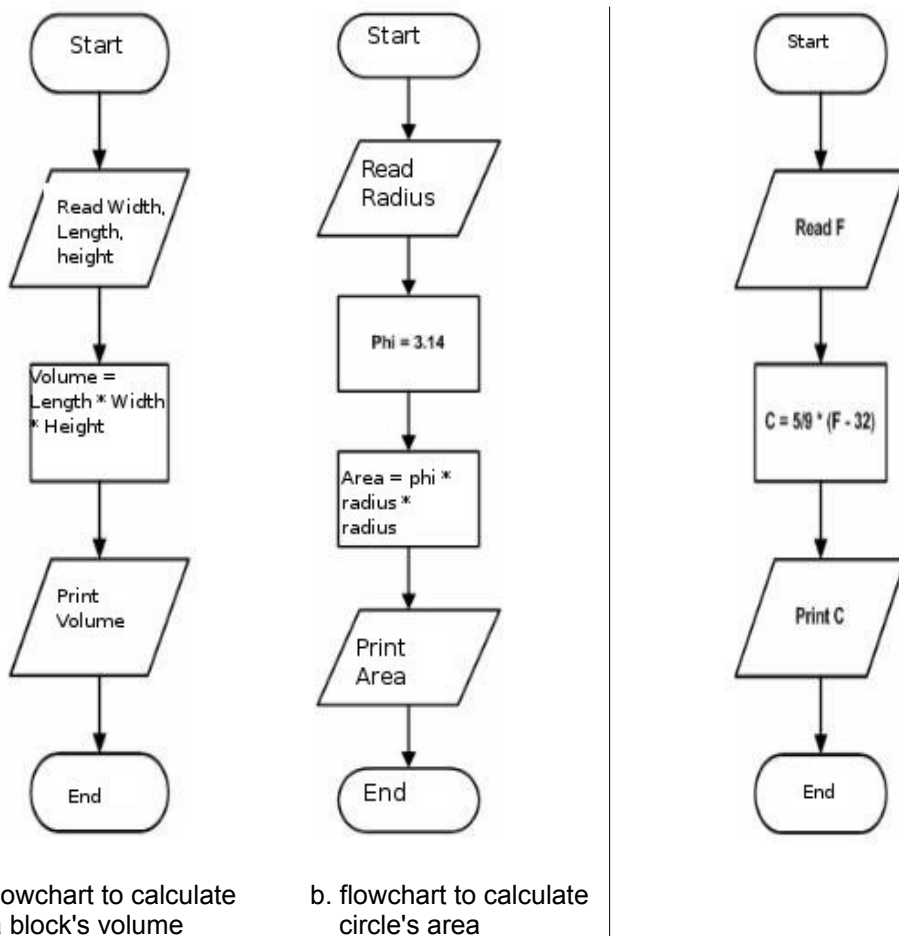


Figure 5.6. Flowchart to calculate block's volume and circle's area

Figure 5.7 Flowchart to calculate temperature conversion

5.2.4. Branching Algorithm

A program will not always follow a sequential structure, sometimes we must change program sequence and jump to certain line in the program. This is known as branching or decision process. This occurs when the car reaches an intersection as shown in Figure 5.7. The driver must decide to turn left or right.

In the branching structure, a program will be branched as the required condition is met. In such process, decision symbol in the flowchart must be used. Decision symbol is meant to test a condition. The result will determine which branch is used.

Example 5.15 Branching structure for age limitation problem.



Figure 5.8. The car at an intersection

A rule to watch a movie is as follows, if the age of the spectator is more than 17 years then the spectator is permitted and if less than 17 years then the spectator is not permitted to watch. Create a flowchart for the problem.

Solution:

The above problem is a typical problem for branching structure. The problem uses the statement *if then*

The flowchart to solve the above problem is shown in Figure 5.9. In Figure 5.9, it uses the Decision symbol. At Decision symbol the required condition is inspected, whether the age is more than 17 years or not. If the answer yes then the program will produce a text output of "Silahkan Menonton", whereas if input the age less than 17 years then the program will produce text output "Anda Tidak Boleh Menonton".

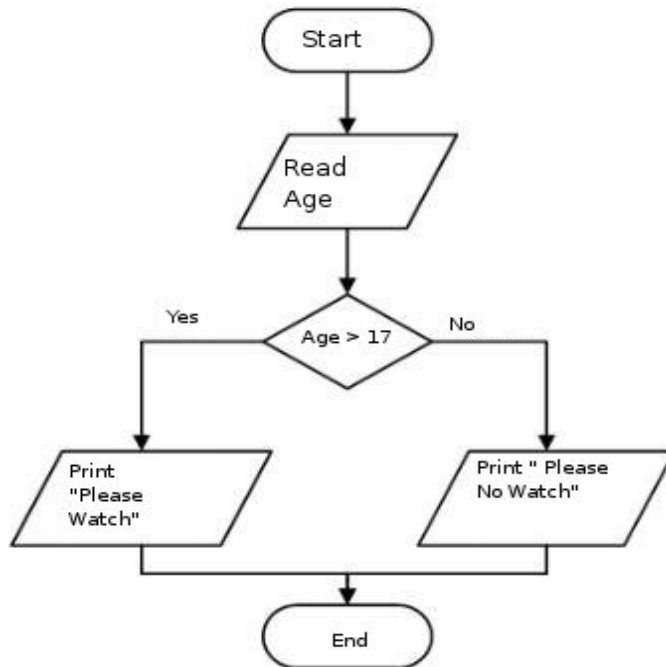


Figure 5.9. Flowchart to solve movie spectator problem.

Example 5.16. Branching structure for the calculation of two numbers.

In a calculation, $P = X + Y$. If P positive, then $Q = X * Y$, whereas if P negative then $Q = X/Y$. Create a flowchart to calculate P and Q

The solution:

In this example the needed input is X and Y , at the condition inspection is carried out on P value whether positive (including 0) or negative. Please see the resulting flowchart in Figure 5.10.

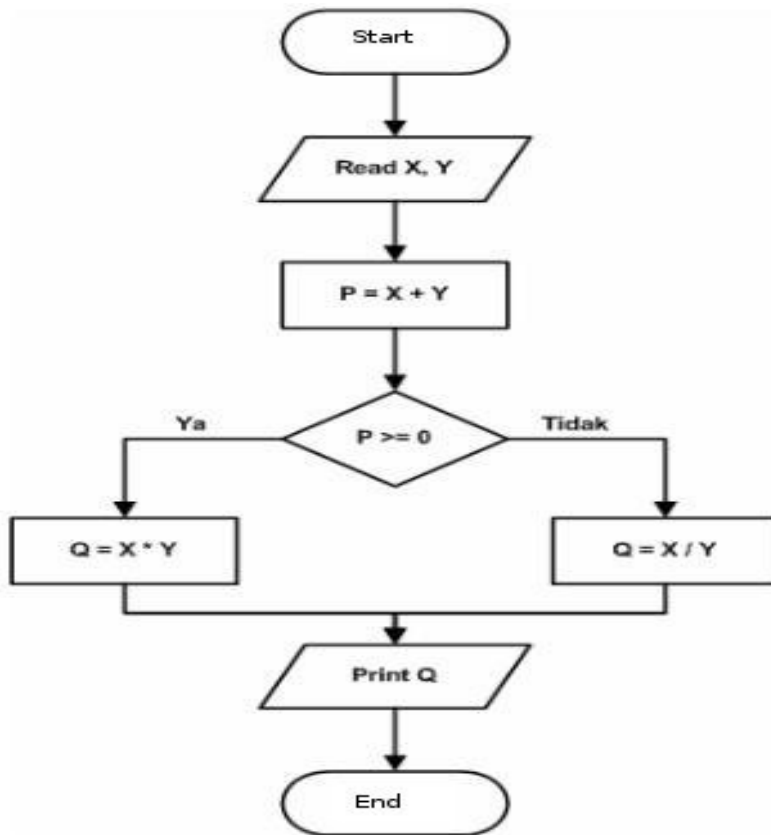


Figure 5.10. Flowchart to solve the calculation of two numbers.

The two examples (5.15 and 5.16) show a simple branching involving one branching. In more complicated problems, we may encounter more branching. We may also encounter a branching structure within a branching structure, or normally called as nested branching. Please see the following example.

Example 5.17. Nested branching for a photocopy problem.

A copy service uses the following rule:

- Subscriber price would be Rp. 75,- / page.
- Non-subscriber would be Rp. 100,- / page if less than 100 pages. If more than 100 pages it would be Rp. 85,- / page.

Create a flowchart to calculate the total cost for someone to copy X number of pages.

Solution:

This example looks complicated. There are two (2) branches. The first branch will look at whether he / she is a subscribed customer or not. The second branch, if he / she is not a subscriber then check if the number of pages more than 100 pages or not.

This is a nested branching problem. Please look carefully at the second requirement.

If he / she is not a subscribed customer, **then if** he / she make less than 100 sheets **then** the price is Rp. 100 / page.

The second if is located within the first if.

Input needed in this problem would be the status of the customer and the number of sheet / page to be copied. Thus, the input variable would be:

- Status – the status of the customer.
- JLF – the number of sheets / pages to be copied.

In addition, there are several other variable namely HPP to store the price per page and TH to store the total cost. Please note that Status using data type character, and this the data should uses “”.

Flowchart to solve the above problem is shown in Figure 5.11.

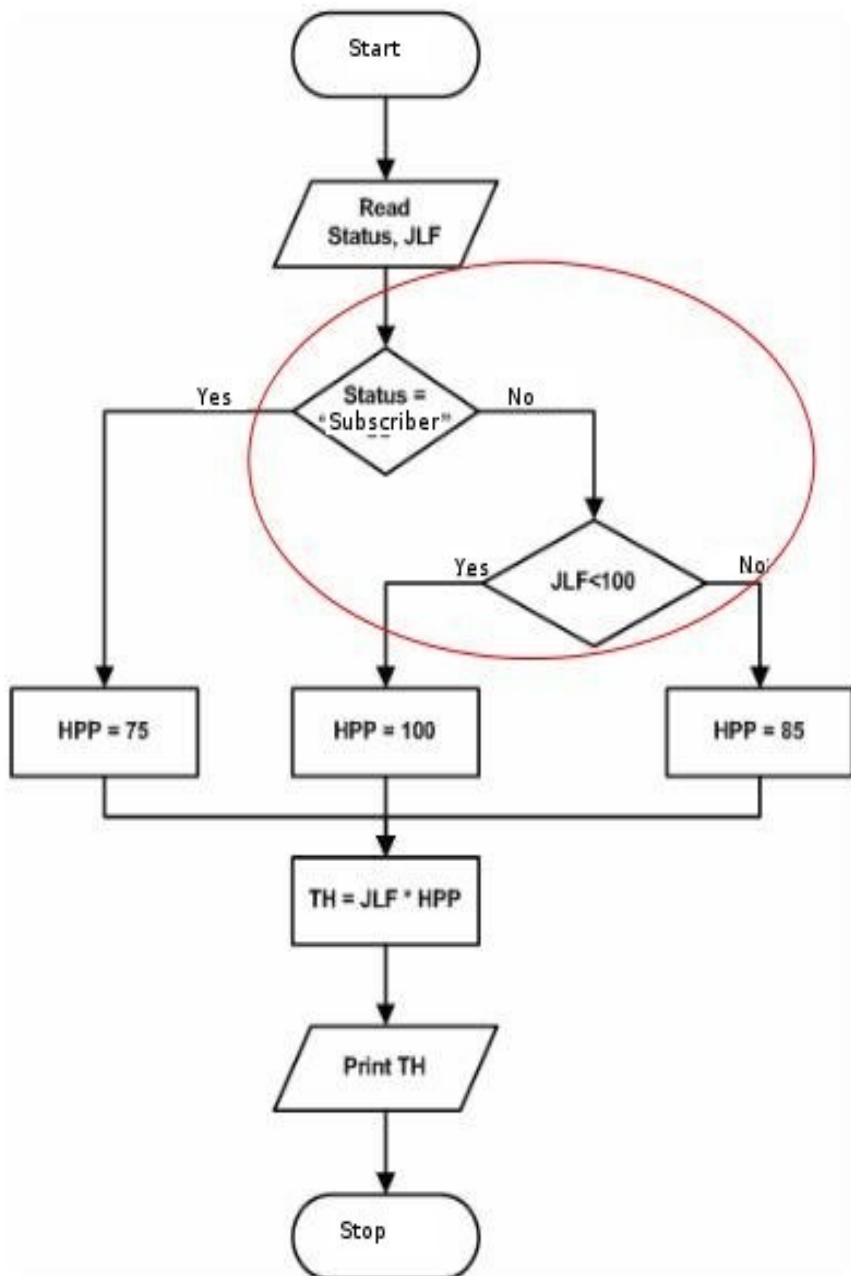


Figure 5.11. Flowchart to solve photocopy pricing problem.

Example 5.18. Nested branching in student's passing grade.

The passing rule of student who take web programming course is as follows.

- If mid test mark is more than 70 then student is final mark would be the same as mid test mark.
- If mid test mark is less than or equal to 70 then student will only be passed if the final mark is equal or more than 60 where the final mark = (mid term mark x 40%) + (final mark x 60%).

Create a flowchart to solve the problem with output student number, student name, and passing status.

Solution:

In this example, there are two branching. The first branch is to check whether the mid test mark is more than 70. If the mid test mark is less than 70, then check whether the final mark is more than 60.

The needed variable input is the student number (NIM), student name, mid test (NUTS) and final exam mark (NUAS). Whereas the variable output consisted of NA that was used to keep the value and the Status of the end to keep the status of the passing. The output variable is NA to store the final mark and Status to store the passing status.

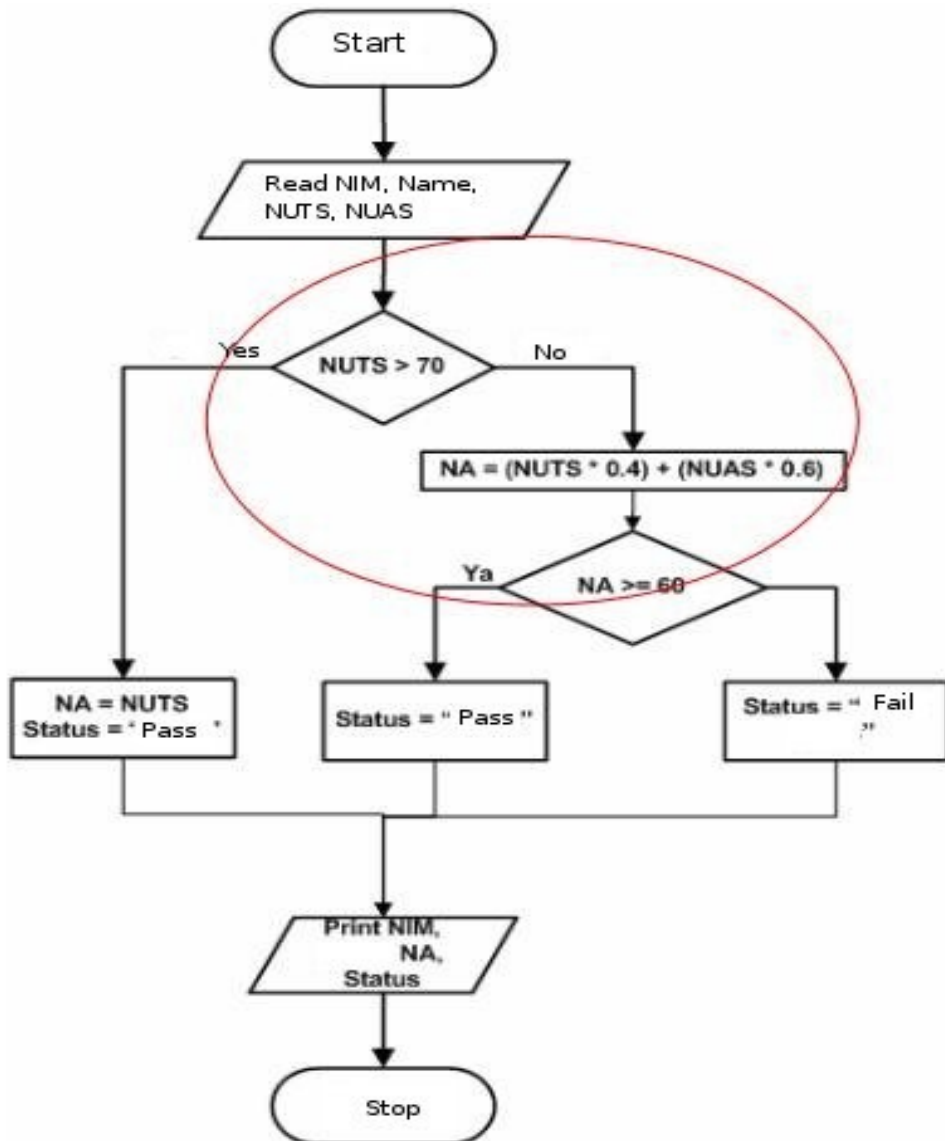


Figure 5.12. Flowchart to solve student passing grade problem.

5.2.5. Looping Algorithm

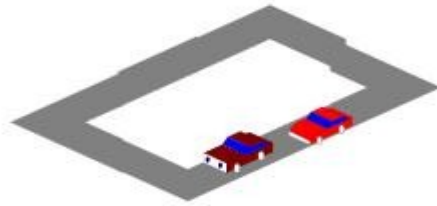


Figure 5.13. Car racing in a circuit.

In many cases, we encounter a job that must be repeatedly done. One of the simplest examples is a car racing as shown in Figure 5.13. In car racing, participants must follow the circuit tracks as set by the race rule. Whoever is the fastest in reaching the finish line, he wins.

In creating a computer program, we sometimes must repeat one or a group of instructions to achieve the required result. In a computer, a repetition process is easy to carry out. This is due to the advantage of a computer as compared to a human to do a task or repeated instruction without being tired, bored, or lazy. Compared to the race car driver, at one time he must be feeling tired driving his race car in circles.

The looping algorithm consists of two parts, namely,

1. **Repetition condition**, the requirement to be met to carry out the repetition. This condition is usually stated in a Boolean expression that must be tested whether being true or false.
2. **The repetition body (the loop body)**, that is one or more instructions that will be repeated.

The repetition structure is usually started by an initialization section and a termination section. **Initialization** is the instructions that are carried out prior to repetition. Initialization is usually used to provide an initial value to the variables. Whereas **termination** instruction is carried out after the completion of the repetition.

There are several forms of the repetition, each with its own requirements and characteristics. Several forms may be used for the same case, but some of the forms may be suitable for a certain case only. The selection of the form may influence the correctness of the algorithm. The correct repetition form depends on the problem.

- **Repetition structure using For.**

Repetition using *For* is the oldest form of repetition in programming language. Almost all programming languages provide this method, despite they may be different in syntax. In *For* structure, we need to know how many times the loop bodies will be looped. In

this structure we normally use a variable, namely, loop counter, to take note the number of loop has been done. The value of loop counter may be increasing or decreasing during repetition process. General *flowchart* using *For* structure is shown in Figure 5.14. Please note the usage of symbol *preparation* in the *flowchart*.

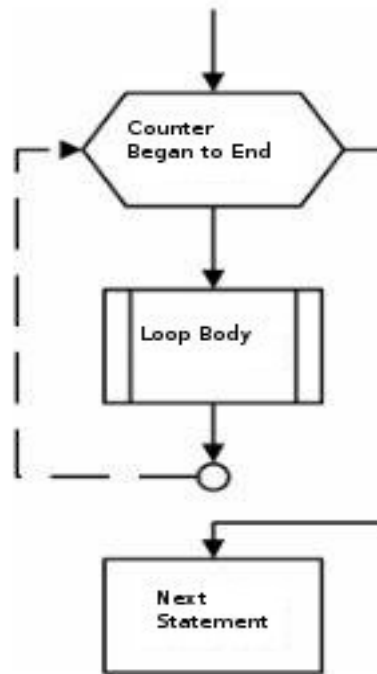


Figure 5.14. Repetition algorithm using For.

In execution of repetition using for, the following steps must be followed:

1. Initialized counter value.
2. Checked whether counter is larger than its maximum / final value. If true then exit the repetition process. If the counter is decreasing, check whether the counter is lower than its minimal / final value. If true then exit the repetition process.
3. Execute the statements / instructions in the loop body.
4. Increase / decrease the *counter* in accordance with the *increment* value. If no increment value, use 1 as default.
5. Repeat step no 2.

It is import to initialized *counter* at the beginning of the loop. If we try to change final value in the *loop* body, it would give not much impact into how many repetition to be carried out.

Example 5.19 Algorithm to print a statement 100 times.

One upon a time, during our elementary school days done an unforgiving mistake, the teacher told us to write something 100 times as a punishment. An example of the statement would be “I will will not repeat such behavior again”. How can an algorithm help in this case?

The solution:

In this example, we need the variable counter, for example, *I*. Initial value of *I* is 1 and the final value is 100. Whereas the increment at each repetition is 1. Instruction to print will be repeated until *counter* reaches 100. Flowchart to do the task may be seen in Figure 5.15.

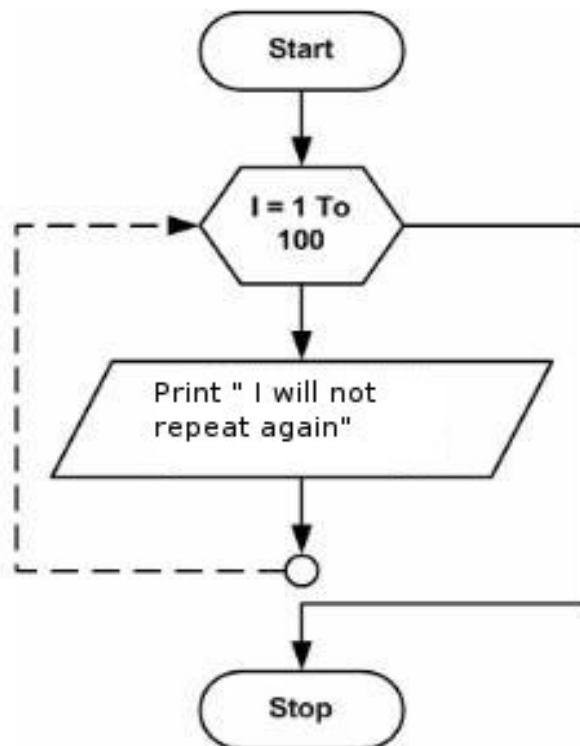


Figure 5.15. Flowchart to write a statement 100 times.

Please note how easy to carry out the repetition. In Figure 5.15 the *increment* is not included, as mentioned earlier if increment is not included then *increment* default to one.

Example 5.20. Flowchart to print the member of an association.

An A association with member 1, 3, 5,..., 19. Create the flowchart to print the member of this association.

The solution:

In this example, we need variable counter, say A (similar to the association name). Its initial value is 1 and the final value is 19. From the pattern it is clear the increment is 2 (1 to 3, 3 to 5, etc). Thus, in every repetition, A increased by 2 The flowchart to solve the problem is shown in Figure 5.16.

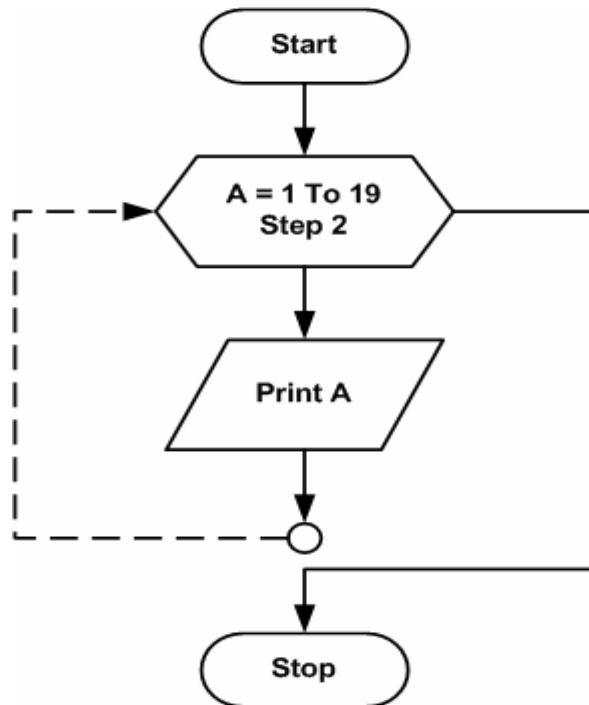


Figure 5.16. Flowchart to print the member of association A.

In Figure 5.16, note the symbol preparation. There is an additional statement, namely, **step 2**. This is the *increment*. For each repetition, *counter* A increases by 2 so that it will be printed as 1, 3, 5,..., 19.

Example 5.21. Determine the results from a repetition *flowchart*

Look to *flowchart* in Figure 5.17. Determine the result of the *flowchart*.

The solution:

In this example, we tried to determine result of a *flowchart*. What do you think? Let's analyze the *flowchart*.

In the *flowchart*, after start, a process contains a statement $A = 1$. This is known as **initialization**. It provides the initial value for $A = 1$. The variable *counter* X is initialized to 1 and end value is 10, without *increment* (or uses the *default increment* to 1). Entering the loop body for the first time, variable A will be immediately printed. Value of variable A is 1. Next process is the statement $A = A + 2$. This is possibly a rather strange statement, it is needed in a programming. It means substitute the old A value with the new one that has been increased by 2. So A will be 3. Afterwards carry out the second repetition. At this stage, A value is 3, will be printed as 3. After that, A is replaced by $A + 2$. The new value is 5. And so on. Thus, the output of this flowchart is 1, 3, 5, 7, ..., 19.

As we can see, Figure 5.16 and 5.17 provide the same output. Between both flowchart, Figure 5.17 is the recommended flowchart. It is longer but more structured. In addition, not all programming languages provides increment facility.

Look carefully at flowchart in Figure 5.17, especially the position Print A statement. Please try to change its position so that it is placed after $A = A + 2$ statement. How is the result?

Like branching structure, we'll find nested repetition structure. Thus, a repetition inside another repetition structure. Please see the following Example 5.22

Example 5.22. Determined results of a repetition flowchart.

Please see flowchart in Figure 5.18. Determine results of the flowchart.

The solution:

In this example, we try to determine the result of nested repetition *flowchart*. What do you think? Let's analyze the *flowchart*.

In Figure 5.18, there is two (2) preparation symbols. The first is to initialize variable *counter* X , and the second is to initialize variable *counter* Y . Variable *counter* Y is located after variable *counter* X . Thus, it means

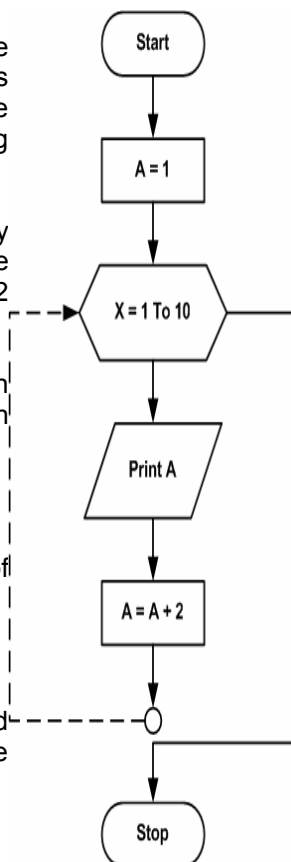


Figure 5.17 Flowchart to print certain number

repetition using variable counter *Y* within repetition using variable counter *X*. It is known as nested repetition.

In this kind of repetition, the flow of program execution is as follows.

- Variable *X* will be initialized with counter value of 1.
- Variable *Y* will be initialized with counter value of 0.
- Calculate *Z* as *X* multiply by *Y*. Since *X* = 1 and *Y* = 0, thus, *Z* = 0.
- Print *X*, *Y*, and *Z* on screen.
- Repeat the flow, increase *Y* by 1 while *X* is 1, as *X* is not repeated yet.
- Now *Z* = 1 as *X* = 1 and *Y* = 1.
- Print *X*, *Y*, and *Z* on screen again.
- Repeat the flow, increase *Y* by 1 so that *Y* = 2, thus, *Z* = 2.
- Exit *Y* rotation as final counter of *Y* = 2 is reached.
- Increase *X* by 1 so that *X* = 2, and do the same process with *Y* again.
- Repeat the process again to reach the counter *X* = 3. Thus, the final result of the flowchart is as follows.

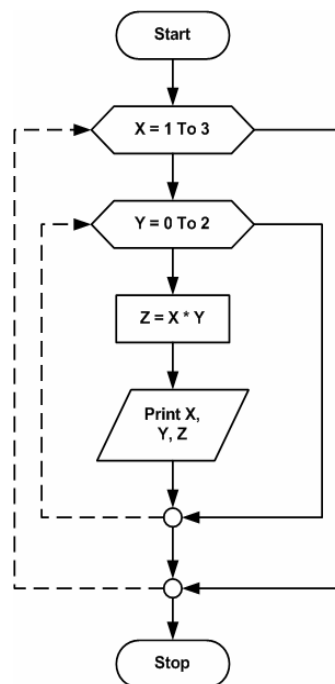


Figure 5.18. Flowchart with nested repetition.

X	Y	Z
1	0	0
1	1	1
1	2	2
2	0	0
2	1	2
2	2	4
3	0	0
3	1	3
3	2	6

From Example 5.22, there are rules that must be satisfied in nested repetition, namely.

- Each repetition (the loop body) must have respective variable counter.
- Repetition must not overlap.
- Repetition structure using *While*.

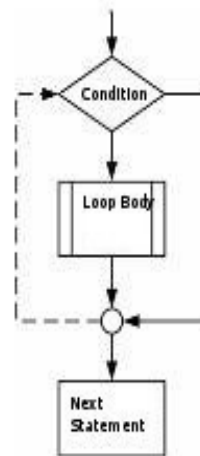
In repetition with *For*, the number of repetition is known as we set the start and stop value in the beginning of the repetition. What if we don't know how many repetition should be done? Repetition using *While* is the answer to such problem. Similar to *For*,

repetition using *While* structure is supported by most programming language with different syntaxes.

While structure will repeat the loop body as long as the *While* condition is hold true. Thus, there is no need to know the number of repetition. General flowchart for *While* structure is shown in Figure 5.19.

In Figure 5.19, preparation symbol is no longer used. However, decision symbol is used to control the repetition.

In *While* repetition, we normally need to initialize variables.



Example 5.22 Repetition with *While* to print the variable.

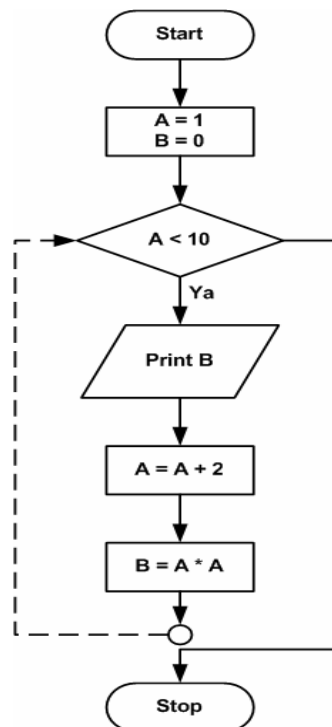
Picture 5.19. Flowchartumum *While*.

Examine *flowchart* in Figure 5.20. How is the *output* of the *flowchart*?

The solution:

Examine Figure 5.20. Can you describe the output of the flowchart? Take a closer look at the flowchart execution stages as follows.

1. In the flowchart, there are two variables A and B. Initialized the variables so that A = 1 and B = 0 before doing the loop. Variable A is the *counter*.
2. In decision symbol, the A value is checked whether met the condition (< 10). If True then the next statements is executed, if not then the loop will be stopped. At the beginning of the execution this condition is met as A = 1.
3. Do Print B.
4. A value is increase by 2. Thus, the new A will be 3. Whereas the B = 9 as a result of $A * A$ with A = 3.



5. The loop proceeds and check whether A is less than 10. At this stage, $A = 3$, thus, the condition is met. Steps repeated to step no. 3, until variable A is no longer met the requirement (< 10). Thus, the output of this flowchart is 0, 9, 25, 49, 81.

As in repetition using For, repetition with While may be used for nested repetition. The rules and methods is the same as repetition with For.

Figure 5.20 Flowchart repetition with while to print certain numbers.

Several programming languages provides repetition by means of *Do... Loop* and *Repeat... Until*. These two (2) methods resembles While, the difference is located in the condition. In While structure, the condition is check before the loop body. Whereas Do... Loop and Repeat... Until, the condition inspection is carried out after the loop body.

5.3. ARRAY MANAGEMENT

Variable array has been briefly touched. In this section, the array will be described in a more detail.

5.3.1. Array Concept

Thus far, the used variables are common variable with a name and point to a certain value in numeric or string. New value may be given to the variable by removing the old value. Would it be possible to store several values into the variable and keep them all? The solution is using index in the name of the variable. Such method is called array.

Array is a data structure that stores a collection of elements with same data type. Each elements may be accessed through its index. Index of array must be a sequential data type, such as, integer or string. Array may be pictured as locker with series of storage with sequential number as shown in Figure 5.21. To store and to retrieve from a certain box, we need to know the number of the box.

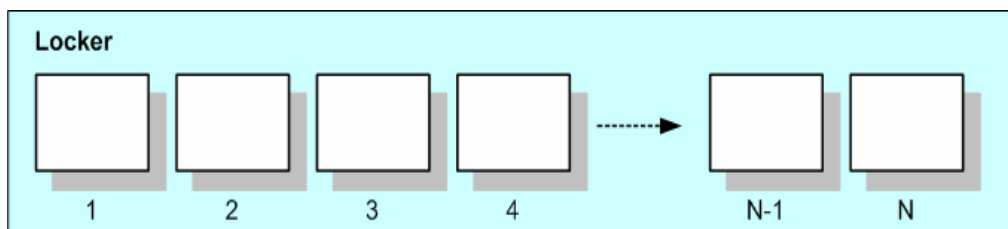


Figure 5.21. Locker with many indexed storage box.

In variable array, we are not only set the data type, but also the number of elements in the array or the upper index limit of the array. In many programming languages, such as, C++, Visual Basic, and several others, the index in an array starts at 0 not 1. Method to write variable array may be different from one programming language to another. However, in variable array, the data type and index limit must be set. To fill the data into the array, we need to specify the index number of the array to which the data will be stored or retrieved.

The declaration example, store and retrieve data in an array as follows.

Example 5.23. Using an array in C++ and Visual Basic.

Array pada C++	Array pada Visual Basic
<pre>#include <iostream> using namespace std; int main() { // Mendeklarasikan array A dengan 3 buah elemen bertipe int int A[3]; // Mengisikan nilai elemen array A[0] = 5; A[1] = 10; A[2] = 20; // Menampilkan nilai elemen array cout<<"Nilai elemen ke-1 = "<<A[0]; cout<<"Nilai elemen ke-2 = "<<A[1]; cout<<"Nilai elemen ke-3 = "<<A[2]; return 0; }</pre>	<pre>'Mendeklarasikan array A dengan 3 buah elemen bertipe integer Dim A (2) as Integer 'Mengisikan nilai elemen array A(0) = 5; A(1) = 10; A(2) = 20; 'Menampilkan nilai elemen array Print A(0); Print A(1); Print A(2);</pre>

Please examine the two (2) above codes. In the variable declaration the maximum index number of the array is 2 but the number of elements is 3 as index starts from 0 nor 1.

5.3.2. Looking for data in an Array

One of the common problem in an array is how to look for certain element in an array. For example, in the above locker in Figure 5.21 available 100 box. Then, we need to find locker belongs to one of the student named "Rudy". Another example, in a school with lots of students, we are asked to find data of a student based on certain name or certain address. Please examine the following example.

Example 5.24. Searching on an array.

$A[0] = 23$
 $A[1] = 22$
 $A[2] = 45$
 $A[3] = 12$
 $A[4] = 10$
 $A[5] = 34$



Which
Element
Contain 12

In this example, we are asked to find an element containing the number 12 from an array. There are six (6) element in the array. How do you think to solve the above algorithm?

The general method and the easiest is by using linear search. In this past, this method is consider inefficient as it requires significant amount of time. However, as computer becoming more powerful, execution time of linear search method is no longer a problem. Linear search compares the element of the array with the value we are looking for, One by one starting from the first element.

Applying the method in Example 5.24, program flow will take the following steps.

- Set the number we are looking for, namely 12.
- Retrieve the first element $A[0]$, compare its content (in this case 23) with the one we are looking for. If it is the same then stop.
- If false then continue with the next element $A[1]$, compare the content with the one we are looking for. If it is the same then stop.
- If false then continue to the next element until we find the element.

Example 5.24 will give the result that element $A[3]$ is 12. Describe the flow in flowchart is shown in Figure 5.22.

- I is the variable counter and index.
- N is the index upper limit.
- $Bil[I]$ is the name variable array containing numbers.
- A is the variable that we are looking for.

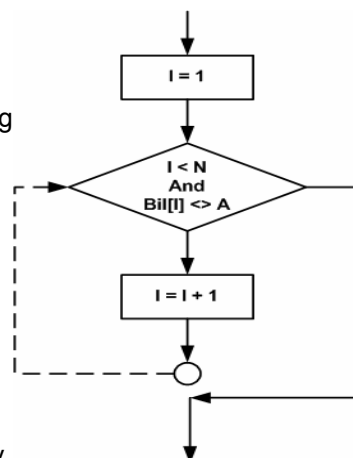


Figure 5.22. Flowchart to search on number in an array.

Flowchart in Figure 5.22 uses While structure to do the repetition. Two (2) conditions must be met, namely, $I < N$ and $Bil[I] \neq A$. Meaning that I must be less than the upper limit of the index N and the content of $Bil[I]$ is not the same of the one we are looking for, namely A , and, thus, the searching process proceeds to the next index. As long as the condition is met, the searching continues. Note that we use “and” that means both condition must be met. Searching will be terminated if one of the conditions or both conditions no longer met. Thus, if $Bil[i]$ is the same as A , searching process will be terminated as While condition is no longer hold true.

5.3.3. Ordering data in an array.

another common problem in an array is to order or to sort the data in an array. Examine the example 5.24. In the example, it is clear that the element is not sorted. How can we sort the element from the large number to smaller ones or the reverse?

There are several algorithm can be used to sort a collection of number, such as, bubble sort, selection sort, shell sort, quick sort, etc. In this book, we will discuss bubble sort. Although the performance may not as good as other, it is easy to understand and commonly used. Examine the following example.

Example 5.24. Sorting using *bubble sort*.

For example a variable array Bil with 5 elements, i.e., "5 1 4 2 8". Sort from the smallest value to the highest value.

The solution:

We will use bubble sort to sort an array. Bubble sort is carried out by comparing two number in consecutive location. If the order is correct, then continue to the next two numbers. If wrong order, then switch the place of the two numbers.

Let's apply the algorithm. Examine the following table array. The initial condition of $J = 0$. First, compare $Bil[0]$ and $Bil[1]$. $Bil[0] = 5$ whereas $Bil[1] = 1$. Use bubble sort rule, $Bil[0]$ is not in a correct location as it is bigger than $Bil[1]$. So we must switch the contents of these two element in the array so that $Bil[0] = 1$ and $Bil[1] = 5$ (examine the line of $J = 1$). Next step is to compare $Bil[1]$ with $Bil[2]$. $Bil[1] = 5$ and $Bil[2] = 4$, so we must switch the contents from this element (examine the line of $J = 2$). Continues until we compare $Bil[3]$ with $Bil[4]$.

J	Bil[0]	Bil[1]	Bil[2]	Bil[3]	Bil[4]
0	5	1	4	2	8
1	1	5	4	2	8
2	1	4	5	2	8
3	1	4	2	5	8
4	1	4	2	5	8

As shown, the last position of the table is not fully sorted. As only one pass is performed starting with Bil [0]. We will carry out the comparison again, but starts at Bil [1] since Bil [0] should be the smallest number now. So that our table will be as follows.

J	Bil[0]	Bil[1]	Bil[2]	Bil[3]	Bil[4]
1	1	4	2	5	8
2	1	2	4	5	8
3	1	2	4	5	8
4	1	2	4	5	8

In the table above, the sequence is correct. However, the algorithm is not completed as it must check the next rounds. It needs to check two (2) more round to compare Bil [2] and Bil [3]. The two (2) tables are as follows.

J	Bil[0]	Bil[1]	Bil[2]	Bil[3]	Bil[4]
2	1	2	4	5	8
3	1	2	4	5	8
4	1	2	4	5	8

J	Bil[0]	Bil[1]	Bil[2]	Bil[3]	Bil[4]
3	1	2	4	5	8
4	1	2	4	5	8

Shown in Figure 5.23 is the flowchart.

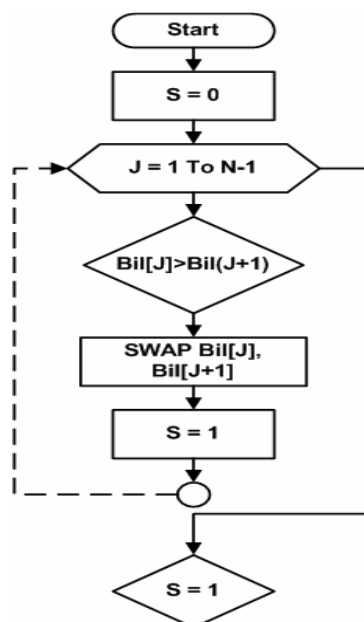


Figure 5.23. Flowchart to sort numbers.

5.4. FILE OPERATION

File is often used to store data to be more permanent. Normally, data and outputs will be lost after the program terminated, so file is used to keep the data. There are two (2) kind of files, namely, program file and data file. Program file contains program code, whereas data file contains data. There two (2) kinds of data file, namely, sequential data file and random access data file. The main difference between the two is shown in the following table.

Sequential data file	Random access data file
Record / data must be sequentially read from the first record.	Record may be random.
Field length of each record may be different.	Field length of each record must be the same.
Modify and adding a record is more difficult.	Modify and add a record is easy.

5.4.1. Algorithm to write data to a file

The algorithm to write data for sequential data file and random access data file is the similar, the only difference is the mode used. The followings is the algorithm to write

data in Structured English (SE).

```

Open "mode", <buffer number>, "file name"
Write <record number>, field 1, field 2, .. field n
Close buffer number

```

Mode O tells that the file is opened for writing.

Example 5.25. Example to implement algorithm to write data.

For example, we have a data file named "siswa.dat" with field name of the student, the address, the telephone number. To write the data would be as follows.

```

Open "O", #1, "siswa.dat"
Write #1, <nama>, <alamat>, <no.telepon>
Close #1

```

Notation #1 shows that siswa.dat is placed in buffer no 1. This notation must be the same and used throughout the program. It means that we placed a file with buffer #1 then when open, write, read and close must use this notation. Likewise when we placed it in buffer no #2.

5.4.2. Algorithm to read data file

The algorithm to read data is similar to write data, but uses mode I instead of O. I means input so that data file is open to be read. The followings is the algorithm to write data in Structured English (SE).

```

Open "modus", <buffer number>, "nama file data"
While not EOF:
    Input <record number>, field 1, field 2, .. field n
    Print field 1, field 2, .. field n
End while
Close buffer number

```

The While Not EOF statement is used to check whether it reaches the last line of data or end of file (EOF). If false it will read all data and print until it reaches last line. Input statement is used to read data from file into the program. While statement print is used to print onto screen.

The example 5.26. Example of algorithm to print data.

Data file named "siswa.dat" similar to example 5.25 with field student's name, the address, the telephone number. To read the data is as follows.

```

Open "I", #2, "siswa.dat"

```

```
While not EOF:
    Input #2, <nama>, <alamat>, <no.telepon>
    Print <nama>, <alamat>, <no.telepon>
End while
Close buffer number
```

5.5. SUMMARY

- Variable is the location to store or to remove value and to retrieve it as needed. Every variable has a name / identifier and value.
- Constant is a variable with fixed value and cannot be changed.
- Data type is the type of data that can be process by the computer that satisfy the requirement in computer programming.
- Data type could be grouped into primitive data type and composite data type. The primitive data type includes numeric, character, and boolean. Whereas the composite data type includes array, record / struct, image, date time, subrange, enumeration, object and variant.
- An algorithm is logical sequence to solve a problem that systematically build. An algorithm must be correct and stop when completed. After completion, an algorithm should provide a correct result.
- An algorithm could be written in Structured English, Pseudocode And Flowchart.
- Sequential structure consists of one or more instruction. Every instruction is executed based on the sequence.
- In branching structure, program will change its sequence if the required condition is met.
- Repetition structure consists of repetition condition and loop body and can be done using For and While.
- Array is data structure to store a collection of elements with the same type, each element may be directly accessed through its index. Searching method in an array may uses linear search whereas ordering it may use bubble sort.
- Data file may be sequential or random access. Method in reading and writing is distinguished by the mode.

5.6. EXERCISE.

1. Determine whether the variable name is right or wrong. If it is wrong, provides the reasons.
 - a. nama.guru
 - b. NamaGuru
 - c. 2x
 - d. harga/buku
 - e. hargaPerBuku

2. Determine match data type for the following variable (not variable name) and provide the reasons.
 - a. total student
 - b. Body weight
 - c. Body height
 - d. Student name
 - e. Birthplace
 - f. Birth date
3. Examine the Example 5.3.
 - a. If in program A all variables (x, y and z) are declared as data type int, how much is z?
 - b. If in program A all variables (x, y and z) are declared as data type float, how much is z?
 - c. If in program B, variable x is declared as float, how much is z?
4. There are two glasses A and B, glass A contains red color solution, glass B contains yellow color solution. Mixed the content of these two glasses, so that glass A contains yellow solution and glass B contains red solution. Create the algorithm in Structured English (SE).
5. Variable A = 6, Variable B = 10. Create a flowchart to switch the value of A and B so that variable of A = 10 and variable B = 6.
6. PT. Sandang Nyaman plan to use computer to calculate weekly payment of its employees. The needed data is the employee name and the working hours of a week. The hourly payment rate is Rp. 4500,-. Create the flowchart for this problem with the output the employee name, their working hours and the received payment.
7. Same as problem no. 6, but if the working hours exceeds 25 hours per the week then it is considered overtime. The hourly overtime payment is one and a half of the hourly payment in normal condition. How is the flowchart?
8. A seller of the primary school textbook is trying to attract buyers with the following provisions:
 - If the total book is less then or equal to 100 copies, then the buyer does not receive any discount.
 - If the total book more than 100 copies but less or equal to 200 copies, then the first 100 copies received 5% discount, and the rest receive 15% discount,
 - If the total book more than 200 copies, then the first 100 copies receive 7% discount, for the second 100 copies receive 17% discount and the rest receive 27% discount.

If the price Rp 5000 / book, create the flowchart to calculate the rule with the output total book and price to be paid. (Determine all variable, constants and its respective data type).

9. Examine Example 5.16. Create flowchart to produce the same output as Example 5.16 using While repetition.
10. Two (2) associations, $A = \{1, 3, 5, \dots, 19\}$ and $B = \{2, 5, 8, \dots, 29\}$. Create flowchart to look for same member between association A and B.
11. Create flowchart to print the following multiplication table:

:

1 x 1 = 1	2 x 1 = 2	3 x 1 = 3	4 x 1 = 4
1 x 2 = 2	2 x 2 = 4	3 x 2 = 6	4 x 2 = 8
1 x 3 = 3	2 x 3 = 6	3 x 3 = 9	4 x 3 = 12
1 x 4 = 4	2 x 4 = 8	3 x 4 = 12	4 x 4 = 16
1 x 5 = 5	2 x 5 = 10	3 x 5 = 15	4 x 5 = 20

12. Sort the following numbers using bubble sort algorithm.

- a. 5 - 3 - 12 - 2 - 52
- b. 4 - 33 - 4 - 3 - 12 - 2
- c. 5 - 3 - 12 - 2 - 50 - 52 - 10

REFERENCES

- Anonymous.** 2004. Guide to the Software Engineering Body of Knowledge (SWEBOK). The Institute of Electrical and Electronics Engineers, Inc.
- Balter, A.** 2006. Sams Teach Yourself Microsoft® SQL Server™ 2005 Express in 24 Hours. Sams.
- Bass, L.,** P. Clements, and R. Kazman. 2003. Software Architecture in Practice. 2nd Edition. Addison-Wesley.
- Cormen, T.H.** 2001. Introduction to Algorithm: Second Edition. The MIT Press.
- Deek, FP.,** J.A.M. McHugh, and O.M. Eljabiri. 2005. Strategic software engineering : An Interdisciplinary Approach. Auerbach Publications.
- den Haan, P.,** L. Lavandowska, S.N. Panduranga, and K. Perrumal. 2004. Beginning JSP 2: From Novice to Professional. Apress.
- Dobson, R.** 1999. Programming Microsoft Access 2000: The Developer's Guide to Harnessing the Power of Access. Microsoft Press.
- Felleisen, M,** R.B. Findler, M. Flatt, and S. Krishnamurthi. 2001. How to Design Programs; An Introduction to Computing and Programming. The MIT Press.
- Kak, A.C.** 2003. Programming With Objects: A Comparative Presentation of Object Oriented Programming with C++ and Java. John Wiley & Sons, Inc.
- Kaisler, S.H.** 2005. Software Paradigm. John Wiley & Sons, Inc.
- Kennedy, B.** and C. Musciano. 2006. HTML & XHTML: The Definitive Guide, 6th Edition. O'Reilly.
- Lafore, R.** 1998. Data Structures & Algorithm in Java. Waite Group Press. 2nd Edition.
- Laurie, B** and **P. Laurie.** 2001. Apache: The Definition Guide. O'Reilly and Associates, Inc.
- Leffingwell, D.** and **D. Widrig.** 2003. Managing Software Requirements: A Use Case Approach. 2nd Edition. Addison-Wesley.
- Lischner, R.** 2000. Delphi in a Nutshell. O'Reilly and Associates, Inc. Rekayasa Perangkat Lunak A1

- Luckey, T.** and J. Phillips. 2006. *Software Project Management for Dummies*. Wiley Publishing, Inc.
- McConnel, S.** 2003. *Professional Software Development: Shorter Schedules, Higher Quality Products, More Successful Projects, Enhanced Careers*. Addison-Wesley.
- Meyer, B.** 2000. *Object Oriented Software Construction*. 2nd Edition. ISE, Inc.
- Musciano, C.** and B. Kennedy. 2002. *HTML and XHTML: The Definition Guide*. 4th Edition. O'Reilly and Associates, Inc.
- Navarro, A.** 2001. *Effective Web Design*. 2nd Edition. SYBEX, Inc.
- Powell, G.** 2006. *Beginning Database Design*. Wiley Publishing, Inc.
- Riordan, R.M.** 2005. *Designing Effective Database Systems*. Addison Wesley Professional.
- Robbins, J. N.** 2006. *Web Design in a Nutshell*, 3rd Edition. O'Reilly.
- Suehring, S.** 2002. *MySQL Bible*. Wiley Publishing, Inc.
- Taylor, D.A.** 1998. *Object Technology: A Manager's Guide*. Addison-Wesley.
- Van Roy, P** and S. Haridi. 2004. *Concepts, Techniques, and Models of Computer Programming*. The MIT Press.

APPENDIX 1 GLOSSARY

Abstraction

Is simply cause from something complex by modeling class depends on problems.

Algorithm

Step sort logical of solve problems in sort systematically.

Array

Data structure save some same elements.

Attribute

Characteristic to difference one entity to other entity.

Authentication

Process check legality someone as user in a system (such as DBMS)

Basic Input/Output System (BIOS)

Codes program run first time when computer on (booting)

Database

Data collection in connect each else other, store in hardware computer and use software to manipulation.

Command Line Interface (CLI)

Between face user with command text model.

Compiler

Translation language program in top level to machine language for all code program. Process called compilation.

Component Object Model (COM)

Infrastructure available by Visual Basic to access objects or other controls as long has between face.

Constraint

Restriction/limitation problem

Control

Activities monitoring and evaluating and feedback to make sure whether system in good work or not.

Counter

Variable number that use in algorithm structure for repetition.

Database Management System (DBMS)

Special software use for management database.

Disk Operating System (DOS)

One of old operation system in CLI base.

Electronic

That science about low-tension electric current with operation by electric current control or particle ion electric in an equipment.

Entity

Individual for representative extension that make different from other.

Extensible Hypertext Markup Language (XHTML)

HTML lat version (4.01) that rewrite by rule strict and tight with term XML

Extensible Markup Language (XML)

Rules to make markup language.

Feedback

Data about system **kinerja**

Flowchart

Scheme/chart which show flow in a logical program.

Indication

Signal for a problem.

Logical gate

Blocks formation from electronic hardware.

Graphical User Interface (GUI)

Between user face by graphics model.

Identifier

Name of variable or constant.

Computer science

A systematic study in algorithm process, explain and transfer information.

Inheritance

Principle from characteristic parents to children or generation which apply in classes.

Initialization

First instruction to variable or expression program.

Input

Elements into system.

Integrated Development Environment (IDE)

Software to help easily make computer application.

Interpreter

Translate language program at top level to machine language one by one lines, read and direct translatability.

Cardinality

Total max entity in an association entity and can relation with others entity association.

Constanta

Variable with constant value data and can not change.

Loop

Repletion process command.

Problem

Difference between actual situation and hopeful situation or difference between present condition with target or aim wish.

Model

Make a simple system or Imitation a simple system.

Multi-tasking

Task operation system to run duties/application with the same time.

Multi-user

Task operation system to run by different user with the same time.

Output

Move elements result from change process to aim wish.

Solve Problem

A process where situation analyze then make solutions if has problem by define, less or remove or cut problem.

Object Oriented Programming – OOP

Paradigm program where use object and integrate to make application and computer program.

Web program

A way to make web page by language web program (script)

Software

All instruction where use to process information.

Permissions

A proses to determine what user can do to a system.

Pointer

Variable that keep address in memory computer.

Polymorphism

Ability objects to has more than one form.

Programmer

A person who work to make computer program.

Procedure

- Instruction that user need to process information.
- Commands that a part of big program and has function to certain job.

Process

Change or transformation input to output

Prototyping

Approach software development in directly to demonstrate how software or components software work in environment before construction actual steps do.

Pseudocode

The way write of algorithm with codes similar language program.

Query

Request or search certain data in database.

Record

Data line from table.

Software Engineering

A discipline science that explain all aspect software production, start from first step that analyse user need, to make sure specification user need, design, codes, test until maintain system after use.

Relationship

Connect in between entities.

System

Elements which interaction to get certain goals.

Database System

Elements such as database, software, hardware, and user which interaction each other to get certain goals that is data organization.

Solution

End part or output from solve problem process.

Stored procedure

Cut of Code program which can receive parameter input and has result one or more parameter output and use for operation database.

Structured Query Language (SQL)

Query language structure to manage database.

Strategic solve problem

Method or approach which use by someone when get problem.

Structure algorithm

The way or steps to make an algorithm.

Data type

Data type which can manage by computer to facility need in computer program.

Trigger

Specific type from stored procedure which will execute when get problem.

Variable

Place where we can fill-in or make empty value and re-call if need in a program.

View

Table virtual which contain base on query and do in database.

Web browser

Software that function to translate HTML codes become view as we want.

Web dynamism

Web pages which contain and information change as want by user.

Web server

Software which do to serve question client to certain web pages.

Web statics

Web pages which contain and information not change.

APPENDIX 2 SITE LIST

Following is the list of important Internet sites used as references for the book.

Address	Comments
http://www.apache.org	Official site of Apache Web server. This site provides the source code of Apache for various operating systems as well as a complete documentation.
http://www.borland.com	Official site of Borland. Borland is a software house that produces Borland Delphi, Borland Jbuilder, Turbo Pascal, Turbo Delphi, Borland C++ etc.
http://www.debian.org	Official site of Debian Linux.
http://www.eclipse.org	Official site of Eclipse Project, an integrated development software that supports many programming language.
http://www.google.com	Google search engine.
http://www.ilmukomputer.com	Indonesian site the provides documentation on various section of computer science.
http://www.javasoft.com	Official documentation and online information on Java programming supported by Sun Microsystem.
http://www.kambing.vlsm.org	Indonesian site that provides iso of various linux distribution to be freely downloaded. In addition, this site is mirroring various linux distribution and application runs on linux.
http://www.linuxdoc.org	This site provides Linux documentations that can be freely downloaded.
http://www.microsoft.com	Official site of Microsoft. A company that produce Windows operating systems, IDE Microsoft Visual Studio, Microsoft Office, Microsoft SQL Server, etc.
http://www.mysql.com	Official site of MySQL Database Software. This site provides the installation file for MySQL for various platform. In addition, it provides a complete documentation of MySQL.
http://www.netbeans.org	Official site of IDE NetBeans, a Java development software.
http://www.php.net	Official site for PHP programming language and interpreter. This site provides the source code and installtion file of PHP for various operating systems. In

	addition, it provides a complete documentation of PHP.
http://www.w3.org	Official site of The World Wide Web Consortium (W3C). W3C is a consortium that sets the standard and Web related technology, especially, on HTML, XML, CSS, XHTML etc. The documentations on the technology may be downloaded from this site.

APPENDIX 3 FUNCTIONS BUILD-IN ON VISUAL BASE

IsNumeric(expression)

This function use to test whether an expression get numeric result or not. The value return is Boolean.

IsEmpty(expression)

Function to check whether an expression already fill-in value or not. The value return is Boolean.

IsNull (expression)

Function to check whether an expression has not valid data, usually use to check contain field recordset.

IsArray (varname)

Function to check whether a variable is an array.

IsDate (expression)

Function to check whether an expression can conversion to date.

IsError(expression)

Function to check whether an expression is error value.

IsObject(expression)

Function to check whether an expression use to an OLE Automation object..

IsMissing(argname)

Function to check whether an argument optional to procedure, it pass or not.

CBool(expression)

Conversion an expression to Boolean.

CByte(expression)

Conversion an expression) to Byte.

CCur(expression)

Conversion an expression to Currency.

CDate(date)

Conversion an expression to date.

CDbl(expression)

Conversion an expression to Double.

CInt(expression)

Conversion an expression to Integer.

CLng(expression)

Conversion an expression to Long.

CSng(expression)

Conversion an expression) to single.

CStr(expression)

Conversion an expression to string.

CVar(expression)

Conversion an expression to Variant.

Asc(string)

Function to show code character from first letter to string.

Chr(charcode)

Function to show character from one code to character code.

Format(expression)[, format[, firstDayFromSunday[,sundayFirstFromYear]]])

Formating an expression base on expression format/

Hex(number) and Oct(number)

Show string which represent Octal or Hexa from number.

Str(number)

Show string which represent number.

Val(string)

Show number from string.

Now

Return a Variant (Date) which show date and time base on computer system.

Time

Return time the present system.

Timer

Return number which show total second since midnight.

Date

Return date the present system.

Time = Time and Date = Date

Arrange time or date system:

Run system by Microsoft Windows 95, date need from 1 Jan 1998 until 31 Dec 2099.

Run system by Microsoft Windows NT, date need from 1 Jan 1980 until 31 Dec 2079.

Hour(time), Minute(time) and Second(time)

Return a Variant (Integer) such as number 0 s/d 23 for jam, 0 s/d 59 for minutes, and 0 s/d 59 for second.

Day(date), Month(date), and Year(date)

Return a Variant (Integer) such as number 1 s/d 31 for month, 1 up to 12 for month, and year. Let us see Picture 1.1. For most users, the above picture is fairly familiar. In the picture, the desktop of Microsoft Windows operating system is shown. Several icons are shown in the picture. Double clicking on the icon will open a certain software application that can be used to do a certain task.