Information security in local area networks, especially in Waterloo Port

Coppens, J.A.W.M.

Award date:
1988

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering

MASTER'S THESIS

Information security in Local Area Networks, especially in Waterloo Port

J.A.W.M. Coppens

Supervisor	:	prof.ir. M.P.J. Stevens
Advisor	:	ing. P.H.A. van der Putten

August 1988

ABSTRACT

With the rise of interconnection of computer systems in networks, there is an increasing demand for security of the information on these networks. Especially the small local area networks are according to their security in their infancy. To restrict the range of this graduation work we concentrated on local area networks (LAN's) of workstations: computer systems especially meant for personal use. We will look at what security measures can be taken and what techniques are available or useful. In order to make the project some more practical we decided to investigate as well an existing local area network, named Waterloo Port. The project is a structuring of known structures and techniques, combined with some personal ideas and additions.

In this report you'll find a model to structure information security into a number of smaller, more compact security demands. A number of possible security techniques are proposed to realize security in a programmable way. Some techniques (selective access, monitoring, encryption and back-up and recovery) are described in some detail.

The Waterloo Port network is examined on it's security aspects. We distinguish access control, file protection, task protection and encryption as security techniques. Waterloo Port appears to be minimally secured and seems not to be resistent to serious hackers. Especially the authentication of users is poor and monitoring of network events is lacking.

Finally we made a start on implementing monitoring for security use in Waterloo Port. A specification of monitoring is formulated and relating aspects like the data structure of the information and the allocation of the monitoring process are described. A decomposition in a number of smaller, more simple processes is suggested and possible relations with existing Waterloo Port processes are mentioned. We lacked detailed information about Waterloo Port to finish this job.

CONTENTS

PREFACE

This report is the result of the graduate work to obtain a M.Sc.-degree in electrical engineering at the Eindhoven University of Technology. This graduate work has been carried out in the digital systems group (EB) of the department of electrical engineering in the period of september 1987 till august 1988.

The text of this thesis has been designed to be read sequentially. It consists of chapters of which each may consist of paragraphs of which each may consist of sections. Inside a chapter the paragraphs are preceded by an introduction. Equivalently inside a paragraph the sections are also preceded by an introduction. Equivalently inside a paragraph the sections also are preceded by an introduction. It should not hurt to browse through this thesis provided that one takes care not to skip over these introductions.

This thesis was first meant to be in dutch. After some time, when the influence of Waterloo Port in the project increased, in dialogue with the coach and the supplier of Waterloo Port (PC Robo Automation BV, Valkenswaard, the Netherlands) is decided to report in English. Because some parts had to be translated and the terminology had to be adapted, this may effect the clearness of this thesis.

In this place I would like to thank:

Prof.Ir. M.P.J. Stevens, of the department of electrical engineering, for the approval of the chosen subject and his supervisorship;

Ing. P.H.A. van der Putten, for his helpful suggestions, his critical remarks and his faith in me fixing the job;

PC Robo Automation BV, especially Leo van Bokhoven, for their interest, enthusiasm and fruitful discussions;

Mr. C. van de Watering, for helping me with the translation;

the members of the digital systems group, for making the graduation period a joyful time;

my parents and family, for their interest, encouragement, support and faith during my study.

1 INTRODUCTION

Everyone is talking about networks today, and special about computer networks. To explain this interest we must look at the advantages of coupling computer systems with each other (Tanenbaum, 1981)*. There are two tendency's in networking:

- coupling existing computer systems to one another;
- installing a number of small computers in a network, instead of installing one big computer.

The advantages of coupling existing computer systems to one another are:

- availability of data which is stored in other systems, data banks, and such;
- the possibilities to communicate in all kind of forms through the network, e.g. (world wide) organizational networks.

Some advantages of installing a number of small computers in a network, instead of installing a big one, are:

- the superior price/performance ratio of small computers over large ones;
- the possibilities to gradually innovate the system by replacing old computers in the network by innovated ones;
- the possibilities to gradually expand the system from very small to very large;
- a better reliability, for a single hardware or software failure in a well designed network will only bring down one processor, instead of wipe out all processing capability.

Networks appear in a variety of forms. In size they vary from small local area networks to large world wide networks. They also vary from coupling personal computers to coupling mainframe computers. To restrict the scope of this work, a restriction is made to local area networks of workstations.

A **Local Area Network (LAN)** is a collection of mutually connected computer systems and peripherals which can exchange data, with a limited geographical distribution.

A **workstation** is a computer system which is especially meant for personal use, though multi-user capabilities may be possible.

---

* Names and years in brackets refer to the references on page 72.

Reasons for choosing this part are:
- a relative new area with lots of possibilities for investigation;
- an immense fast rise of LAN's for workstations;
- smaller systems seem to be better for learning aspects.

Coupling computer systems to form a network has some implications. One of these implications is the demand for security of information in the system. Some facilities must be provided to make it possible to restrict the use of the network and that of the information in particular.

Besides treating information security in LAN's in general we look at one specific type of LAN: the Waterloo Port network from Waterloo Microsystems Inc., Ontario, Canada. The choice for this network is made on the availability of this network in the digital systems group (EB). We will investigate the available security facilities of Waterloo Port, and evaluate these measures on their pro's and con's.

The outline of this report is as follows:

First we will structure and shortly describe possible applications of local area networks for workstations in **chapter 2**.

In **chapter 3** an introduction to security in general is given, especially a kind of model for security and some security principles. **Chapter 4** treats programmable security measures.

Chapter 5 gives an introduction to the Waterloo Port network structure and its special user-interface. In **chapter 6** the available security facilities are described and evaluated.

Chapter 7 contains a possible outline of an implementation of monitoring events within Waterloo Port.

The report is concluded with some conclusions and recommendations.

2. APPLICATIONS OF LOCAL AREA NETWORKS

We can divide the applications of local area networks in some application groups. The division is made in such a way that relating applications which demand the same facilities, are put together. We distinguish applications for:

- sharing of information;
- sharing of peripheral devices;
- electronic communication;
- system management;
- operation-specific applications.

These application groups are subdivided and described in the following paragraphs.

2.1 Sharing of information

description:
By sharing information we mean the use of information which is not present in one's own workstation, but in some memory available in the local network. In practice this information will be in the memories which are at our disposal from file servers.

Applications:

1.1 the use of central software (multi-user packets);
1.2 the use of central files;
1.3 the realizing of one's own logical work spot,
independent of the physical work spot.

Application 1.1 The use of central software

To be able to use central software offers a number of advantages:

* Software packets need to be stored only in one location instead of in every workstation, and thus it is (disk)memory saving.

* New versions of software packets have to be installed only once, and so one will not be confused by using old and new versions simultaneously.

* Extensive packets consisting of several diskettes do not operate smoothly in workstations without a hard disk, because each time one has to exchange the diskettes. With the aid of this application the workstation is able to fetch the software from the network.

(We assume that when using a program a copy of the load-file is loaded into the memory of the workstation.)

This application first demands that software packages have been made suitable for **multi-user** use. If a program only has to be loaded into the memory, and only proper user files are used, there is no problem. The problems arise when the program is used by more workstations, and the program generates temporary files for every user, behind the scope of the user.

Application 1.2 The use of central files

One of the most important applications for local area networks is the ability to use central files. For instance when a database replaces a card file, it is important that there only is one central file, which can be browsed and changed by anybody. So, not each workstation or file server may have it's own copy, because in that case changes made by one user do not come through in the other files. But also for files which are not sensitive to changes, central use may be important in order to save memory, e.g. when using very large files (electronic telephone guide).

The most important demands which are imposed to the network by this application is the fact that files and preferably also parts of these files can be locked when they are in use. This is called **file locking** resp. **record locking**. If this is not done and more users are simultaneously making changes in the file, then the changes made by persons who have finished are overruled by the other files stored later on.

Application 1.3 The realization of one's own logical work spot.

By the realization of one's own work spot we understand the creation of a "home" directory whose exclusive rights are assigned to the user. In this "home" directory the user can put his own files and use them as his "work" directory.

This has the advantage that a user, independently of the workstation to be used, can reach his own files, without the need to work with diskettes every time. Mainly there where we do not have the availability of fixed workstations per (group of) user(s), but centralized workstation facilities, this application will prove it's value. Also when not all workstations have a hard disk memory, this application can be recommended.

Besides, this stimulates the use of working in one's own directory, in which temporary files and such are stored. This will prevent the rest of the file system to be polluted by temporary files, which are quite often forgotten to remove.

It is preferable to direct a user immediately into his own directory when he identifies himself (e.g. during a log-in procedure). This because of convenience for the user and to prevent the just mentioned system pollution.

By replacing the diskettes by a permanent directory it is important to secure the recording of the files. Besides a recommendable recording by the user himself, for which facilities must be created, a central back-up activity yet is indispensable.

2.2 Sharing of peripheral devices

Description:
By sharing peripheral devices we understand the ability to use peripheral devices like printers and plotters, which do not need to be connected to one's own workstation.

Applications:
2.1 sharing of printers;
2.2 sharing of other peripheral devices.

Demands imposed by this application:
- the composition of queues and spooling.

application 2.1 Sharing of printers

One of the most required applications of local networks is the ability of sharing printers, which have been connected somewhere else to the network. Especially the use of a laser printer by all persons who are connected to the system, is a very obvious application of the network.

The printers connected to the network can be subdivided in 2 groups:
- local printers;
- network printers.

The local printers are only connected to a workstation, and are not available to the network. Network printers are at the disposal of each workstation in the whole network. These network printers preferably are connected to a file server, because then no communication through the network (from file server to printer) needs to take place.

There can be more than one network printer present and one or more local printers. So a user has the choice between a number of different printers. The user specifically selects one of the local or network printers to do his printing. This choice will be based on the required **print-quality, printer-speed** and **nearness** of the printer.

application 2.2 The use of other peripheral devices

Peripheral devices exist in many forms. Some examples are plotters, digitizers, modems and all other kind of communication systems. These peripherals are mostly very expensive and not available in large numbers, These systems are therefore especially suited to be included in a network.

The use of these devices is often dependent of the used software applications. Some departments use plotters intensive, while others may hardly ever use them. Connecting such a plotter on the network give the ability to use such a device for all parties.

2.3 Electronic communication

Description:
By electronic communication we mean communication with the aid of the workstation by means of the network. This communication can take place inside one's own network as well as communication towards other nets.

Applications:
The applications can be divided into different groups:
3.1 communication inside the network;
3.2 communication with other nets.

3.1 Applications of communication inside the local network:
3.1.1 electronic mail;
3.1.2 communication between workstations.

3.2 Applications of communication with other systems:
3.2.1 coupling to systems with equal protocols;
3.2.2 coupling to systems with different protocols:
- mainframe, mini computer, and such;
- standardized public nets: telephone, datanet, ISDN;
- private nets (standard protocols).

Demands imposed by this application are:
- a similarity as good as possible with complex OSI standards, which have not yet been crystallized-out completely in the higher layers;

- space on the screen for messages.

2.3.1 COMMUNICATION INSIDE THE NETWORK

By communication inside the network we mean the communication with users of the network. So we deal with messages which flow between people. This communication can occur in two manners.

If both persons, who want to communicate, have been connected to the network, and both work on a workstation, direct communication can be done between the workstations. Messages sent by one person, are directly displayed on the screen of the addressed person. This is called **communication between workstations.**

However, if at that moment the addressed person has not been logged-in to the network, then direct communication is not possible. However, then it is possible to leave a message behind in the system for the addressed person. Then we speak of **electronic mail.**

The essential difference between electronic mail and communication between workstations is: with electronic mail the message is stored in the network till it is removed by the receiver, while with direct communication storage is largely temporary.

application 3.1.1 Electronic mail

By electronic mail we mean the sending of electronic mail through the system, without the user actually being present in the network at that moment. This means that messages are written to an electronic postbox, that specifically belongs to one person. Everybody can write into this postbox, but only the addressed person is able to read and remove the messages.

This electronic mail application can be made as extensive as one wants. A very first hand function is a **standard message form**, in which receiver, sender, date, subject and a message field have been fixed. This gives a certain kind of uniformity to mail messages, and takes care these are always recognized as such. Also the creation of mail messages with such a standard is much easier.

A second handy function is an **acknowledgement of entered mail.** So a user doesn't have to look in his mailbox whether maybe new messages have come in. This acknowledgement falls apart in two parts:

- a message immediately after entering the system if messages have come in during absence;
- an acknowledgement as soon a message comes in during connection to the system.

A third function is the creation of **address groups** by the user and the system manager. Certain users are grouped into groups, because they have certain common matters. By addressing such a group of users a message is sent to all members of the group at one time.

Furthermore we can ease the fact of **generating a reply** to an incoming message. The addressed person and the subject already are known, so only the message has to be entered. Also we can create the possibility of attaching the incoming message to this reply, such that query and reply stay together.
We can give messages a certain **status**, which indicates that a certain acknowledgement is expected. This acknowledgement may consist of a report as soon as the message is read, or an obligation to reply, as soon as the message is read.

application 3.1.2 Communication between workstations

Communication between workstations has to do with message-flow between users connected to the network. Messages sent by one person, directly are displayed on the screen of the addressed person.

This **addressing** can take place in several ways:
- addressing of persons;
- addressing of groups of persons;
- addressing of workstations.

These addressing possibilities should possibly exist simultaneously, and not instead of each other.

The message displayed on the screen, of course should not appear mixed up with other information. So a method has to be found in order to prevent this. For this sake there are a number of possibilities:
- reserving a special field on the screen for incoming messages;
- to make appear a special window on the screen, displayed over the existing text, without this being destroyed;
- temporarily storing a message with a direct acknowledgement in a special field on the screen. The only difference with electronic mail then is that the message is removed from the system, as soon as it is asked for.

Communication between workstations mainly is of importance for system messages. This is to be able to notify all users of the network about current matters like the system operation, e.g. drop-out of a server, maintenance of the system, and such. This function seems to be of less use, if the network only is located in one single room, so that direct verbal communication is possible as well, and probably deserves preference.

2.3.2 COMMUNICATION WITH OTHER NETS

One's own network is a first step towards the sharing of information. But is also opens the possibility to consult other shapes of information than those available in the own network. With this we can think of other networks as well as of larger computers which use the same protocols, as of a plurality of different nets following different protocols.

If the systems to be connected use the same protocol, we only need a traffic regulator, who leads the data-flow to the proper system. Here we mean by a network also a system! Such a traffic regulator between systems with an equal protocol is called a **bridge.**

However, if the systems to be connected use different protocols, we don't only need a traffic regulator, but also protocol conversion must take place. These 2 functions gathered into one system is called a **gateway**. For any different protocol a different conversion has to take place. Sometimes a gateway can perform more conversions, but other ones are only suited for the conversion of one single protocol. So we may need more gateways to connect a LAN to the various systems.

Of course we want the user to notice as less as possible where the information comes from. So we have to try to let protocol conversions and such take place in a transparent way. So here the system itself has to negotiate about this with the peer system.

application 3.2.1 coupling to systems with equal protocols

A LAN often has a restriction in the number of workstations to be connected. However, to keep the ability to connect an "unlimited" number of workstations communicate with each other, we want to be able to couple more LAN's to each other. Also for the preservation of an well-ordered structure of a LAN it may be useful to subdivide this into a number of smaller networks. But, of course, also here it must be possible to couple them to each other.

Sometimes a network and a mini- or mainframe computer use the same protocol. This mainly occurs on systems of one and the same manufacturer. Then a coupling seems to be quickly made.

application 3.2.2 coupling to systems with different protocols

Many systems will have a different protocol then the local network does. However, we want to be able to connect these as well.

We can subdivide the systems, which we want to connect in this application, into three groups:
- mini-, mainframe computer and such;
- private networks;
- public networks.

1: For some applications workstations do not have enough computing speed or memory space. In those cases a larger computer in the shape of a mini or mainframe is needed. Of course we want to be able to work with this larger computer from our workstation and preferably not only use it as a terminal.

2: Sometimes we want to couple a LAN to another private network. Examples of this are an already existing network in the same company, or on cooperation between two companies, where in both companies a network already exists. Many LAN's have a standard protocol for the lower levels, like: ETHER-net, ARC-net, token ring, and such. These standards then can be used as a start.

3: Finally we reach the public networks. This is the most extensive and least obvious area, because the expected possibilities are huge. For example we may connect to:
   - telephone net;
   - datanet;
   - (inter)national electronic mail systems;
   - telex and fax facilities;
   - databanks.

   These are only some examples of possibilities which are already in prospect. This application can open up the whole world, and still new applications are added to it.

2.4 System management

Description:
By system management we mean those functions which arise from the use of a network. Because a larger system is created, there has to be a primary responsible person (the system manager), who makes and keeps the system optimally operating. This involves for instance the adaption of the network to specific requirements of the specific user-group, taking care of recording functions and "monitoring" the system. The system manager has several tasks, but here we describe only those which require utilities.

Applications:

4.1 utilities for back-up and recovery;
4.2 monitor functions for:
- performance analysis;
- peripherals availability;
- repair diagnostics.

4.3 installation of special facilities, e.g. user-friendly operation-shells and user-groups.

application 4.1 Utilities for back-up and recovery

Users are inclined to consider a network as a secure system, in which nothing can happen to their files. Of course this is not true. A network probably brings along greater risks concerning the possibility of loss of files. Besides the user himself other network users can reach those files. A good security of files however, can fight this risk to a large extent.

However, it is still necessary to archive the files in case they will be lost. For this sake the network has to offer some facilities. These facilities fall apart in two parts:
- back-up and recovery facilities for the users;
- archive facilities by the system manager.

1: For these back-up facilities for the user we think of an inter-active archive function. This should offer at least the following possibilities:
- back-up of directory, sub directory, selected files;
- back-up selection by means of creation date and time;
- back-up by means of extensions.

2: The system manager needs to have some facilities at his disposal to be able to archive (parts of) the system. Besides the possibilities which are also offered to the user, here more facilities must be created which guarantee a regular archivation. Besides a technical event this is an organizational problem. Generally we can pose that with a certain regularity the whole system has to be archivated, and somewhat more often parts of the system, e.g. files in which changes have occurred since a certain date (the last time an archivation was done).

application 4.2 Monitor functions concerning the network

For a good network management it is necessary that data are known about the use of the network in general, and those of certain applications in particular. This then contains as well statistical data as an actual view.

The statistical data serve for example to examine traffic flows and queuing, so that it is possible to tell problems and find solutions for this.

The current data represent for instance the connected users with their tasks. One can think of a multitude of monitoring functions, but the practical use of this can not be indicated in a simple general way.

application 4.3 Installation of special facilities

For a network often an instruction is needed to make users acquainted with (the possibilities of) the network. This is necessary for new users as well as for existing users, because the structure and applications of a network tend to change once in a while. This instruction can of course best take place via the network.That's why these facilities should be created here.

When using a PC many users make their own (software) tools, by which they can work faster and easier. A number of these tools are also suited for use by other persons. Then it may be useful to put a number of these tools available centrally, e.g. in the shape of a designer's shell. This can go very well together with the composition of user groups. The user groups are composed by means of common work areas, and for each work area a proper shell can be created. Beside these group-specific facilities also facilities can be made in which several groups participate, or the whole network. It is a task of the system manager to take care of these shells and for this sake tools have to be present in the network.

2.5 Operation-specific applications

Description:
The operation-aimed applications are all those applications which are extremely related to the use of other devices than the usual PC devices.

With this we think of application of the network for other tasks than general tasks. some examples are the processing of measuring results which are supplied to the network by special measuring systems, or the processing of video images for design purposes, or special CAD/CAM systems.

We will not deal here any further with this kind of applications.

3 INFORMATION SECURITY

In this chapter you find an introduction to the information security. First we will describe what information security is and why we need it. For a good view on information security we need some kind of model. We will describe one in paragraph 3.2. Because the model does not give a clause for building secure systems, a number of principles which can aid to design secure systems are given in paragraph 3.3. In the last paragraph of this chapter we will look at the relation between information security and user-friendliness.

3.1 Introduction

Security is protecting against evil. In our case the subject of protection is the information in a network. The evil what may happen to the information is unauthorized disclosure, unauthorized modification, destruction or loss. To protect the information in computers, the information security consists of three parts:

1) **Protection of information resources** against damage or loss. The physical media which contain the data must be protected against theft, fire, magnetic fields and other causes which makes them unusable or may damage them.
2) **Protection of information** against unauthorized disclosure, unauthorized modification or destruction. The information itself as data on the physical resources must be protected. Only authorized users must be able to read the data, change it or remove it.
3) **Reconstruction of information** in the case of modification, destruction or loss. Although sometimes forgotten, this point is an essential part of information security. When something has happened to the information, it must be possible to reconstruct it.

The need for information security in networks comes from the fact that more and more information is stored on the network. When the information is stored on removable media like floppy disks, the information can be locked in a locker, like we would with certain paper information. But there is an increasing need for save network systems, on which you can leave your information on non-removable media, and that information must be as save as in a bank vault.

3.2 Model for security

Protection is a shelter from unwanted matters. These unwanted matters are here breaches of the information contained in the network. The breaches can be divided into three categories according to their consequences:

1) **breaches of the confidentiality,** by which an unallowed access to and use of information is made possible (unwanted publication or use);
2) **breaches of the reliability,** by which the correctness and the completeness of information is not guaranteed any more (unwanted changes or additions);
3) **breaches of the continuity,** by which no information processing is possible anymore (destruction by fire, theft, vandalism).

The information should be protected against these breaches. For the fight against these breaches three sorts of security measures can be taken:

1) **organizational security measures**
   Protection is to a very great extent an organizational problem. All protective measures to be taken should be supported by organizational measures. These measures have the shape of guide-lines and procedures (Aalders, 1985).
2) **physical security measures**
   The physical protection involves all measures that are taken to protect network equipment and information sources against unwanted use, theft, fire and other calamities. This already starts with the protection of the surroundings of the building in which the network is present up to the actual using of the network or the locking of the information sources.
3) **programmable security measures**
   The programmable security protects by means of programs the information contained in the network. This protection can selectively give access to the network and to parts of the network,including information.

These three sorts of security measures together should guarantee a sufficient protection. The weaknesses of one measure should be compensated by the other measures. this is represented graphically in figure 3.1.

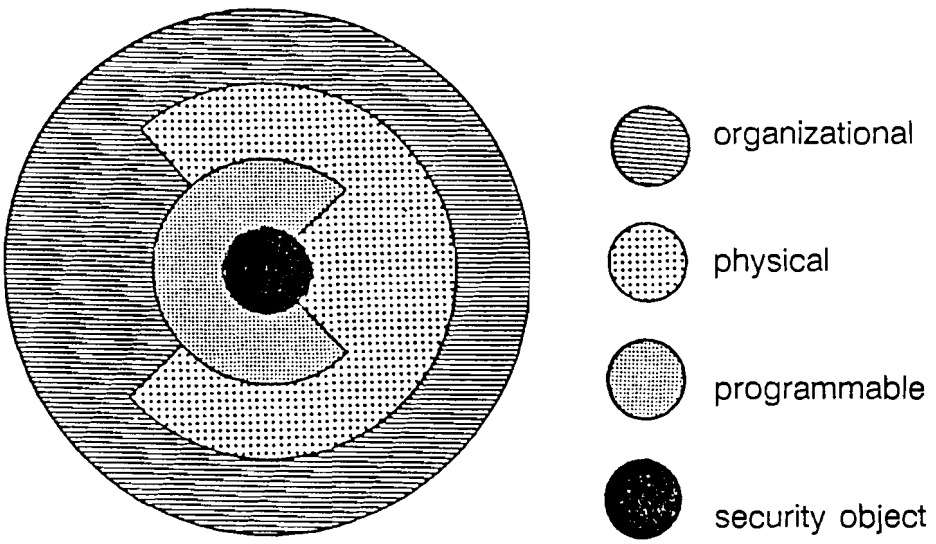

Figure 3.1 : *relation between the sorts of security measures*

The organizational security measures must support the physical and programmable security measures.

The security measures should deal with the unwanted matters in three ways:

1) **preventive measures:**
   minimize the chance that the breach can take place;
2) **repressive measures:**
   minimize the damage in case the breach takes place;
3) **corrective measures:**
   plan the repair of the damage as good as possible in case it takes place.

The various sorts of security measures and the nature of approach should have their influence on all three categories of breaches. We can denote this graphically by a cube, shown in figure 3.2. Each of the 27 minicubes is characterized by the category breach, the sort of security measure and the nature of the security measure (Bautz, 1987).

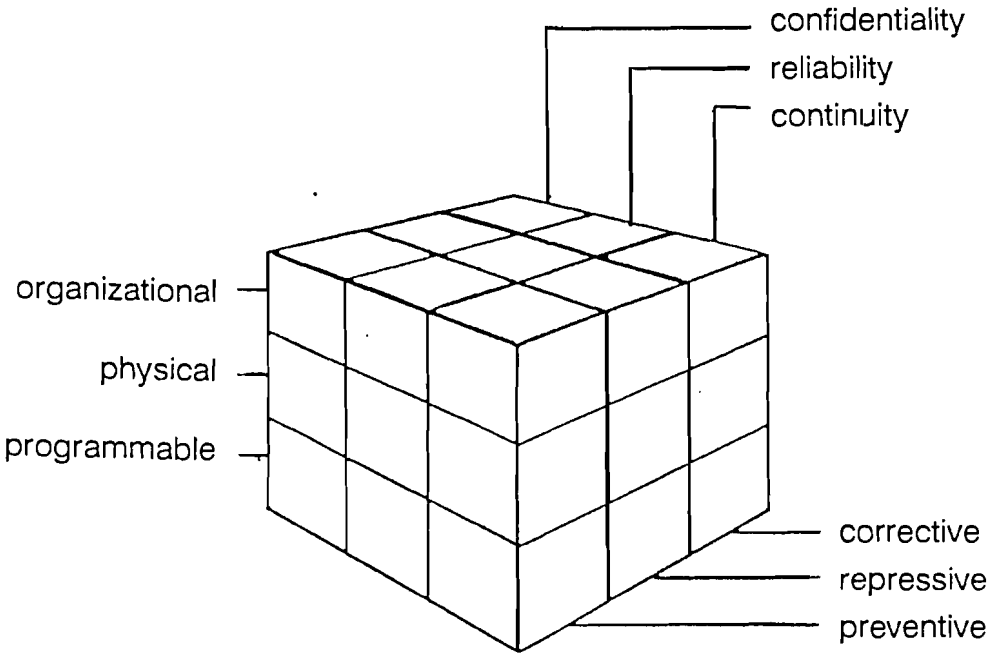

Figure 3.2 : *nature and sort of measures on the category breaches*

3.3 Principles for safe system design

The development of secure, general-purpose computer systems has been extremely difficult and has not been achieved up to this moment. In the absence of a complete methodology for the production of such systems, experience in the

development of protection mechanisms for computer systems has provided a number of useful principles that can guide the design of such mechanisms (Kent, 1981).

economy of mechanism
The simplest design which achieves the desired effect is to be chosen. Unnecessary complex designs may introduce unintended access-paths which make it possible to get round the protection. The more complex the system is, the more difficult it is to test it for it's correct working.

fail-safe defaults
The system should be safe initially. This means that access decisions should be based on explicit permissions, and not on exclusion. This leads to a conservative design method in which arguments should be carried along, why something has to be accessible, instead of why not.

complete mediation
Every access to each object should be controlled. Not any activity or access should be allowed for unauthorized users. This control demands the existence of a network-wide access-control-base.

open design
The protection of a network should not depend on the being secret of its protection mechanism. A safe system design must also remain safe in case the structure and protection systems are public. Only a bad protection system gets safer by keeping secret the structure and methodologies.

separation of privilege
This principle is based on the fact that a system is safer, if more keys have to be used side by side. This principle can be applied to all kinds of aspects of protection, for instance for encryption keys, the recognition of users and access to special tasks.

least privilege
Users and programs must have those permissions, which they actually need to be able to perform their tasks. This can cut down the damage that is created by intrusion or mistakes.

least common mechanism
Users should share as least as possible with each other. Every shared object represents a potential information path between users, and is protection sensitive. Furthermore shared objects should meet the demands of all users, which involves more complex systems.

psychological acceptability
The protection system should have such a human-machine-interface that users automatically apply the available protection mechanisms.

3.4 Security and user-friendliness

There exists some kind of paradox between security and user-friendliness. Security means to build-in access controls and provide security actions for protecting information. User-friendliness demands an open system with a minimum of obligatory operations.

Network users will look for their comfort in operating the network. They want an easy access to the system and their files. They don't want to be bothered with waiting for security tasks or unaccessible information. One example of users looking for comfort is the user who changed his password 13 times successively to be able to keep using his own password, where the system obliged him to change his password periodically and remembers the last 12 passwords. Security will have to fight the user's comfort.

The user's conduct lays a commitment on the security system. The security system may not rely on the user, but has to oblige the user to adept security operations. On the other hand the security measures are to be as user-friendly as possible, without disturbing the effects of the measure itself. The user should be relieved as much as possible from extra operations. Security should be treated as transparent as possible. The user should be well informed about security and his restrictions on the system.

4 PROGRAMMABLE SECURITY

In the previous chapter we treated the information security in general. We have divided the security measures which have to be taken, into three sorts: organizational, physical and programmable measures. In this chapter we shall deal with the programmable security measures. For this sake we shall consider how we can fill the programming layer of our cube model with security measures. Subsequently we shall describe and work out a number of the most important techniques. Respectively we will treat selective access, monitoring, encryption and backup and recovery.

We suppose here that a reliable transfer of information via the network takes place. Partly this is a security aspect in the context as we see it. On the other hand we can demand from a network an error free transfer of information.

4.1 The programmable layer in the cube model

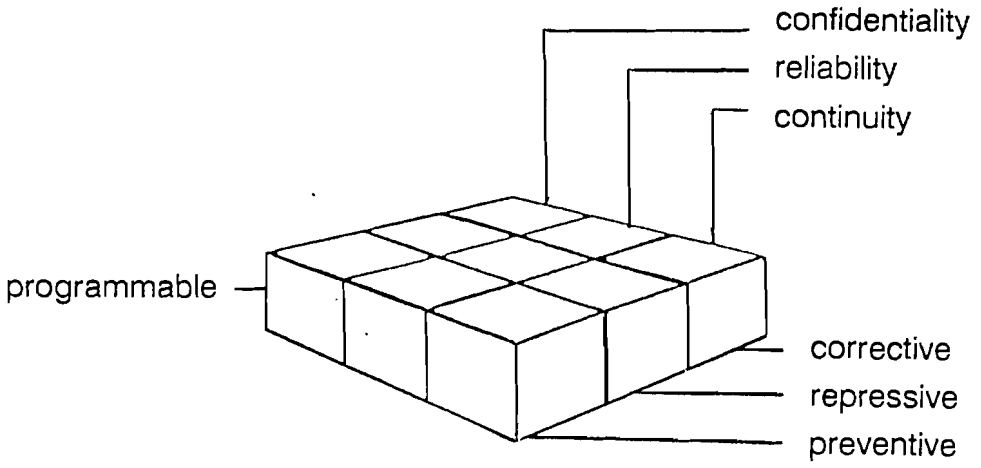

Figure 4.1 : *the programmable layer of the security model*

Figure 4.1 represents the programmable layer of the cube model. For a good security each of these 9 minicubes must be filled as good as possible with security techniques, which protect the system against the relating type of breach by means of the nature of the security measure. Now we shall have a closer look at each of these minicubes.

CONFIDENTIALITY

preventive: prevent undesired reading or use of information.
Here we can think of:

- access control to the system;
- selective access to information sources;
- to counter passive wiretapping in the shape of encryption of communication signals;
- the control of printer output: output should not just disappear or be printed anywhere.

repressive: cut down the damage in case the reliability is damaged.
This aspect can be divided into two parts:

- take notice that the confidentiality is damaged. This requires an actual overview of used files and by whom.
- take care to prevent repetition by investigating the abuse. In the system we have to be able to recover the location where the leak was. This requires a log-book of the activities of the users during a certain period.

corrective: repair the damage as good as possible after the damaging of the confidentiality.
By a quick recover of the leak the confidential information possibly yet can be recovered. However this requires a fast signalization. The signalizing has to take place by up to date monitoring.

RELIABILITY

preventive: prevent the undesired change of information.
Some possible measures are:

- access control to the system;
- selective access to information sources;
- counter active wiretapping by encryption of communication signals;
- the encrypted storage of information: encryption of stored data;
- use of control numbers by applications;
- use of expiration date; before this date the information can not be changed.

repressive: cut down the damage on modification of information.
It is important that modification of information is signalized. The damage which arises by processing of modified information has to be prevented. Detection of the modification can be done by means of control numbers and by monitoring.

corrective: repair the damage as good as possible after discovering the modification of the information.
After the modification of information a recovery will be necessary in order to restore this information.

CONTINUITY

preventive: prevent that the information is destroyed or can not be used anymore.

When information is destroyed we do not only have to deal with unauthorized users, but also authorized users have to be protected against themselves. Access can be denied to unauthorized users by access control and selective access to information sources.

In order to minimize mistakes if authorized users the following measures can be taken:

- it has to be impossible or at least difficult to destroy more files simultaneously;
- there has to be a feedback to the user whether destruction is really intended;
- special notice has to be paid to matters like formatting storage media.

repressive: cut down the damage if anyhow destruction of data should occur.

In order to cut down the damage we should be able to save as much as possible of the destroyed information. For this sake the following methods are available:

- the not yet actual destruction of the information during the session, but only on termination of the session. Many mistakes are experienced during the session.
- the use of the means to be able to fetch back files or pieces of files, which not yet actually have been written over.

corrective: repair the damage after the destruction of the information.

After total or partial destruction of information a recovery will be needed in order to restore the information again.

Figure 4.2 schematically represents the described techniques.

4.2 Selective access

The most important technique for the security of information is giving selectively access to parts of the network. We can look at what places in the network access control can take place, what authorizations we can give at this points and what criteria are used for access selection (section 4.2.1). When distinction of persons is used as a access criterion, we must be able to authenticate these persons (section 4.2.2).

	confidential	reliable	continuity

preventive	- access control
	- selective access
	- encryption of communication
	- printer control
		- access control
		- selective access
		- encryption of :
		  - communication
		  - information
		- control numbers
		- expiration date
			- access control
			- selective access
			- directory security
			- remove feedback
			- format security

repressive	- current monitoring - logbook monitoring	- control numbers - monitoring	- virtual remove - operating system utilities
corrective	- current monitoring	- recovery	- recovery

Figure 4.2 : *techniques for programmable security*

4.2.1 ACCESS CONTROL

The parts of the network where access control can take place are called access points. On these access points we cannot only give access to the specific part of the system, but also provide or deny specific access authorizations. The criteria by means of which selection takes place, are called access criteria.

ACCESS POINTS

We can introduce access control in various points in the system. These points are:

1) **workstation**
   The workstation is the base for computer use. When we give selectively access to use the workstation we create the possibility to deny each use of the system to unauthorized users.

2) **network node**
   Access control in network nodes creates the possibility to mask parts of the network. By this we can mask as well any use of the network as the permission for a limited use of the network.
3) **volume**
   By a volume we understand here a file structure on a storage medium. This may be a floppy disk, a (part of a) hard disk or a tape. Access control tomes offers the possibility to mask complete file structures for general use.
4) **directory**
   By a directory we understand here a network node in a file structure. Selective access to directories offers the possibility to mask any separated part of the file structure.
5) **file**
   By a file we understand here the smallest quantity of separated information which is accessible by the user. This may be as well data as software. Access control on files completes the possibility to mask individual files.
6) **record**
   A record is the smallest set of information known by the system. It is also possible to allow access control here. However, often it will be a task of the application program to create conditions for this.

ACCESS AUTHORIZATIONS

The access authorizations determine what is allowed and what is not in this specific part of the system. The most obvious sorts of authorizations can be indicated on files and directories.

file authorizations
There are five authorizations which refer to files. These are:
- executing the file;
- reading the file;
- modifying or changing the file;
- appending to the end of files;
- removing the file (only once!).

directory authorizations
Per directory we can give certain authorizations to the user. As authorizations we can distinguish:
- executing files;
- reading files;
- modifying or changing files;
- appending to the end of files;
- removing files;

- looking at the contents of a directory;
- preparing sub-directories;
- removing sub-directories;
- preparing new files;

- changing file attributes;
- changing directory attributes;
- modify the authorizations of the users.

Access authorities which are valid on a certain access point, can be used on a higher level access point to authorize the total underlying network. So the possible access authorizations expand with the level of access point.

ACCESS CRITERIA

The access criteria are criteria by means on which the decision is made for giving access to the relating part of the system. As access criteria we can distinguish:

- **users**
  Using users as a criteria of access selection is well-known. Often we want to award to different users different authorizations. We can regard all users individually, or we can divide them in a limited number of groups. Better is a situation where users can be regarded individual as well as member of one or more groups. In this case each one keeps his individual status, but yet it is simply possible to share certain information inside a group.

- **hardware**
  We can also regard terminals and peripheral devices as criteria to get access to parts of the network. Sometimes certain terminals or peripheral devices can be regarded as relatively unsafe because they are in locations which are secured minimally, e.g. in public computer rooms. These terminals and peripheral devices also can be considered individually as well as in groups.

- **time of the day**
  Access can also take place as function of the time. For example in times which are outside the working hours we can limit the possibilities of the system or even deny any access.

- **applications**
  Application programs which use certain information can be regarded as criterion. Certain information may only be used by specific applications.

4.2.2 AUTHENTICATION

The most often used access criterion is the distinction between users. To be able to provide different users with different authorizations it is necessary to be able to distinguish the different users. We have to identify the users, to determine the identity of the user. For this sake the users often have a user identification (userid). The user is known by the system by giving his userid, but still is not recognized. Because userid's often are known, there must be a control whether the userid actually belongs to the user. The user need to be authenticated.

Authentication can be achieved in three manners:
- by physical features;
- by personal possession;
- by knowledge or memory.

physical features are:
- eyeball;
- fingerprint;
- geometry of the hand;
- signature;
- speech (recognition).

These methodologies generally have not yet been developed to such an extent that they actually can be regarded as being safe. For the recognition of these features always (expensive) hardware is needed, by which the costs increase too much in many cases.

personal possession can be:
- magnetic card;
- chip card.

A disadvantage of these methodologies is that they often can be duplicated, are lost, forgotten or stolen. So surely they are not reliable for 100%, and thus they are not really suited for complete security. In combination with one of the other methods, however, they can render their services. Here also additional hardware is needed for the input of these special recognition signs, albeit this hardware is a lot cheaper than the hardware for the physical features. There even are possibilities to use a floppy-disk for this purpose, when we can make them copy-save, for example with little burnt-in holes (Wood, 1985).

knowledge or memory can be tested in the shape of:
- passwords;
- answers to questions.

This method is the most used form, possibly in combination with one of the

previous methods. The advantage of this method is that no extra hardware is needed for it.

PASSWORDS

Passwords are the most widely used authentication means. Passwords should be difficult to guess, easy to remember, frequently changed, well-guarded secrets.

Passwords can be created in three ways:
- selected by the user;
- awarded by the system manager;
- generated by the system.

Passwords have to be **remembered** in a simple way, because else the user will forget them or will search for other help like writing it down, what of course has to be prevented. On the other hand passwords should be **difficult to guess**. This demands a certain minimum length. Easy to guess passwords like first names, telephone numbers, car licence numbers and the most common words must be forbidden as password. The obligatory use of control characters and/or digits, punctuation marks, e.d., also yields safer passwords (Herschberg, 1986).

Passwords should be **changed regularly** to
- cut down the time which is available to the "cracker" in order to "crack" a password;
- cut down the time which enables the user with a false password to use this password.

The system should ask for a new password regularly and else it should deny access. There also has to be a control to prevent the reuse of passwords, such that a user by frequently changing his password can not go on using the same password.

Passwords should be stored in the system in an **encrypted** form. It has to be impossible to read, edit, move passwords or something like that. Only the system manager must be able to destroy them. Passwords should never be made visible; not during typing and not during changing.

A user should only get a limited number of **chances** in order to input his password correctly. After a number of attempts to input the proper password, the system must detect the failure and has to take actions. Possible actions are:
- logging of the failure;
- blocking of the userid:
    - for some time;
    - till this is admitted again explicitly;
- blocking of the workstation:
    - for some time;

- till this is admitted again explicitly.

Networks with access via the telephone network have to use dial-back procedures as extra authentication of the workstation.

Are there still possibilities to make passwords after all these requirements?
A method for making passwords which are quite easy to remember and difficult to guess is the concatenation of two completely independent words and then to add a control character to this, or to make a deliberate spelling mistake. For example *car* and *room* can make *carrroom*.

4.3 Monitoring

Monitoring is the logging of the events in the system. It is not only a security instrument, but it is also indispensable as a diagnosis instrument for a smooth operation of the system. Monitoring consists of the gathering of information about the use of workstations, activities of users, use of files and particularly their mutual relations. We can split up the monitoring which is of importance for the security, into two categories:

- up to date information about users and their occupations;
- registration in a log-book of the activities in the network.

up to date information
The up to date information aims to frighten the users to experiment with forbidden actions. For this sake the following data have to be gathered:

- logged-in users and on which station;
- per user an overview of activities and open files.

logbook information
The logbook information aims to be able to recover afterwards whether unallowed actions have been committed, and to be able to trace the source. For this sake the following data have to be gathered:

- what did the user do in the system:
  - when did he log in and out and where;
  - which activities have been run;
  - which files have been used, and what has been done with these;
- who has been using a certain file or activity;
- who has been using the network during a certain period and where.

The gathered data are made accessible by some commands. Care must be taken when dealing with this monitor information. The information in principle is intended for the system manager and security employees. When considering the fact also to put parts of this information at the disposal of users, the consequences of this have to be examined thoroughly.

4.4 Encryption

Cryptography is a method of secret writing. It is a way of hiding the content of a message through a position-scrambling process or through some other method of transformation. The original message is typically called **plaintext** and the transformed message is called **ciphertext**.

The objective of cryptography is to scramble the message in such a way that an unauthorized and unintended recipient of the ciphertext can not recover the underlying plaintext, while an authorized and intended recipient can easily do so.

Transforming plaintext to ciphertext is called **enciphering** or **encryption**. The inverse is called **deciphering** or **decryption**. The method used for transformation is known as an encryption algorithm. It expresses the set of rules for performing the transformation. Cryptographic algorithms use a combination of functions operating on some data (plaintext), to transform the data to ciphertext.

We can apply encryption in two areas for the security of networks:

1) **encryption of stored data**
   In order to secure data which has been stored in the system or in removable media, we can encrypt it. A simple method of encryption is the mere compressing of the information. Compressed text is also a primary security against curious looks and undesired change. Encryption of data protects the confidentiality and in a limited way the reliability. Information may be changed, but because there is not looked at what is changed and in what way, changes with a special purpose in mind are not possible, while detection is simple.

2) **encryption of communication signals**
   The encryption of communication signals is a security against wiretapping. Wiretapping is the unwanted taking part in the listening by the tapping of electromagnetic signals. We distinguish 2 types of wiretapping:
   - **passive wiretapping:** there is only a taking part in the listening over the line: only the confidentiality comes into play;
   - **active wiretapping:** there is not only a taking part in the listening, but also messages can be deleted, changed or inserted: also the reliability comes into play.

The risks of wiretapping with LAN's are probably not so big. LAN's mostly are cabled with a twisted pair of coaxial cable and often do not occur in public field. It strongly depends on the area of distribution of the network and the shielding of the cables against unauthorized persons whether wiretapping can take place.

Encryption can be implemented by software as well as by hardware. Encryption of stored data can well be done by means of software. For encryption of communication signals however, soon a hardware encrypter/decrypter will be needed, because otherwise the encryption brings along a large overhead.

In the field of encryption very much information is available. So we shall not deal with this subject here further. For further information we refer to the manifold literature. There is a very good introduction in Tanenbaum, 1981, pages 386 - 417.

4.5 Back-up and recovery

The programmable efforts in order to enable back-up and recovery particularly are a support for the organizing problem to have good back-up's at one's disposal. In the programmable field this means the creation of utilities for the simple storing and fetching of data. With this we should think of a number of four facilities:

- backup facilities for the user;
- recovery facilities for the user;
- backup facilities for the system manager;
- recovery facilities for the system manager.

5 INTRODUCTION TO WATERLOO PORT

In the continuation of this thesis we will take a closer look at the security of one special local area network: the Waterloo Port network from Waterloo Microsystems inc., Ontario, Canada (version 2.40).

Waterloo Port is a network program. It serves to enable communication of workstations (IBM PC's, PC XT's, PC AT's and their compatible's) with each other. Waterloo Port, further indicated as Port, is not a DOS application program, but uses its own operating system. However, it supports DOS applications.

In this chapter we will introduce the reader to Waterloo Port. In the first paragraph we will describe the inside of Port: the multi-process structure. This paragraph can be skipped for readers who only are interested in Port's security measures. However this paragraph is needed for understanding the part about introducing monitoring in Port (chapter 7). The second paragraph is an introduction to the user interface from Port, in which special tools for interaction with users are shortly described.

5.1 The PORT structure

The Port system evolved out of the thoth system. The thoth system was developed as a part of a software research experiment in operating systems and program structuring at the university of Waterloo, Canada. The principal goals were to demonstrate the feasibility of using multiple processes in a program as a means of structuring programs, called multi-process structuring, and to investigate the feasibility of designing and implementing portable systems.

The first goal of the experiment was achieved by designing and implementing an operating system and support application programs that both provide and used multi-process structuring. The second goal was achieved by designing the system so as to be transportable and transporting the system to several different machines. The software system produced by this experiment was called thoth (Cheriton, 1982).

The Port operating system structure is given in figure 5.1. The structure is known as a **kernel structure**. It consists of a kernel, system processes, (ordinary) processes and managed resources. Parts of the operating system are implemented as processes, the **system processes**. The (ordinary) **processes** are the tasks which the operating system must perform. The **managed resources** are I/O devices, the processor, memory, an interval timer e.d..

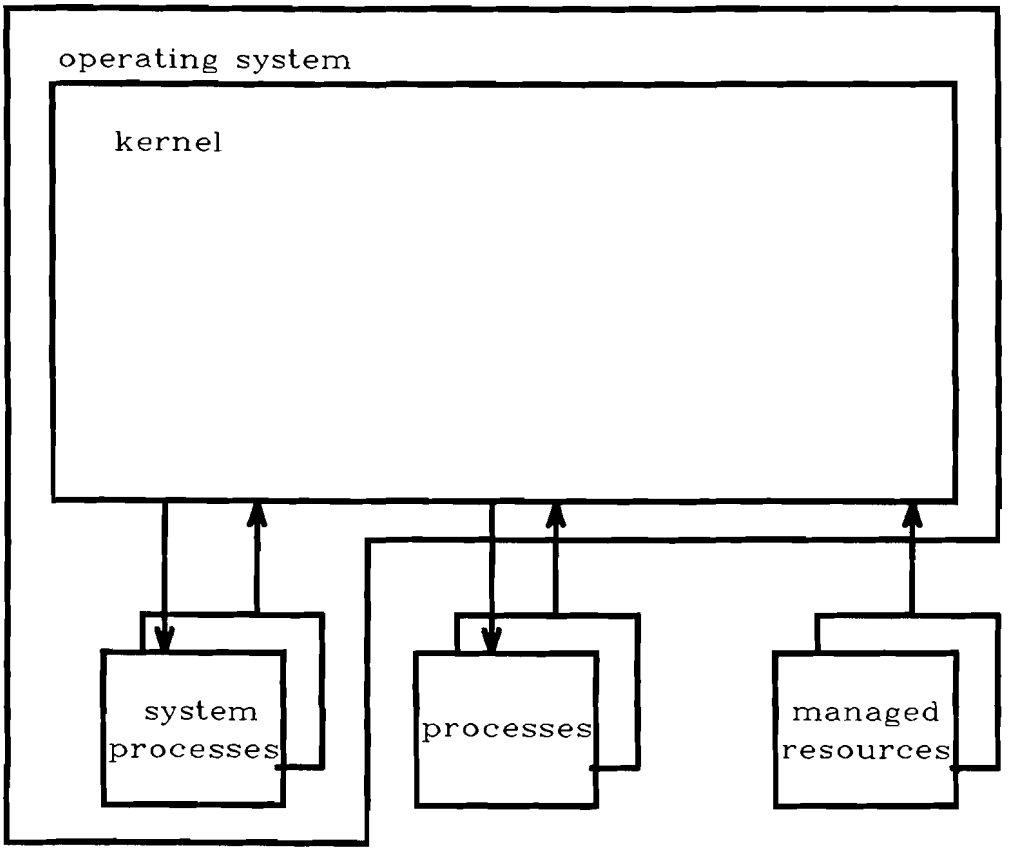

Figure 5.1 : *Port's operating system structure*

The **kernel** implements processes and operations for interprocess communication as well as process and device management. The kernel is an object manager for processes, i.e. processes are the objects. Thus, the kernel implements all process operations, including operations for process creation and interprocess communication. The kernel provides:

- inter process communication;
- process identification;
- process management;
- device management;
- dynamic allocation of memory and processes.

PORT MESSAGE-PASSING PRIMITIVES

Port has a couple of message-passing primitives for inter process communication. These primitives exchange messages between local processes. The messages are variable in length and may be up to 64 kilobytes. Typically they are only 1 to 40 bytes long.

There are two basic message-passing primitives:
- send primitive;
- receive primitive.

The **send primitive** has the syntax:

ok = send (msg, reply_msg, receiver_id)

This primitive sends a message (msg) to the process identified by receiverid and blocks the sending process until that process returns a message in reply_msg. The sending process unlocks and the send fails if the process identified by receiver_id does not exist (receiver_id is invalid) or the process identified by receiver_id dies (receiver_id becomes invalid). The sending process is **send-blocked** until the process identified by receiver_id receives msg, and then becomes **reply-blocked** until the receiver_id returns a message in reply_msg (figure 5.2).

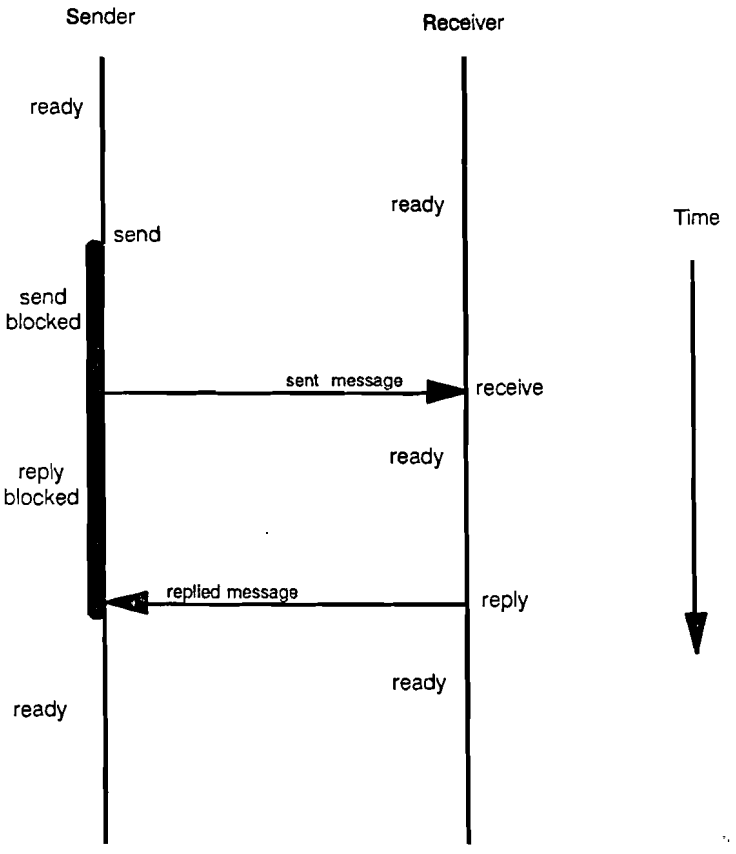

Figure 5.2 : *message pass with send primitive*

The **receive primitive** has the syntax:

ok = receive (msg, sender_id)

The sending/receiving process blocks until the process identified by sender_id places a message in msg. The process unblocks and the receive fails (ok=0) if the process identified by sender_id does not exist (sender_id is invalid) or the process identified by sender_id dies (sender_id becomes invalid) (figure 5.3).

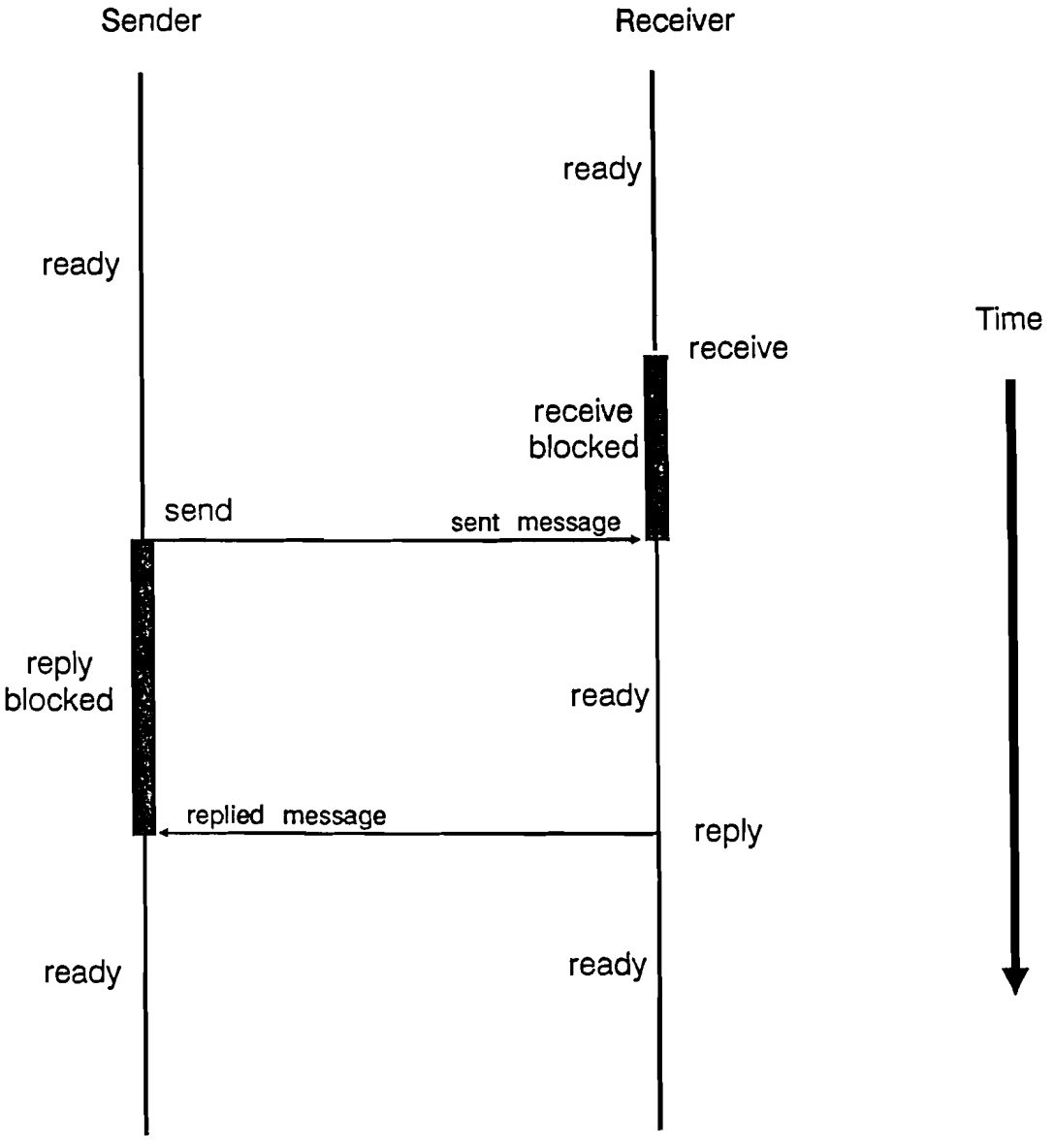

Figure 5.3 : *message pass with receive primitive*

Two other primitives are the **transfer primitives**. These primitives (transfer_from and transfer_to) copy a number of "count" bytes from/to an "others_area" in the data segment of the process identified by "other_id" to/from "my_area". The transfer fails if the process identified by other_id does not exist or the areas do not reside completely within the segments of source and destination processes. The transfers are only done while the process identified by other_id is blocked (figure 5.4).

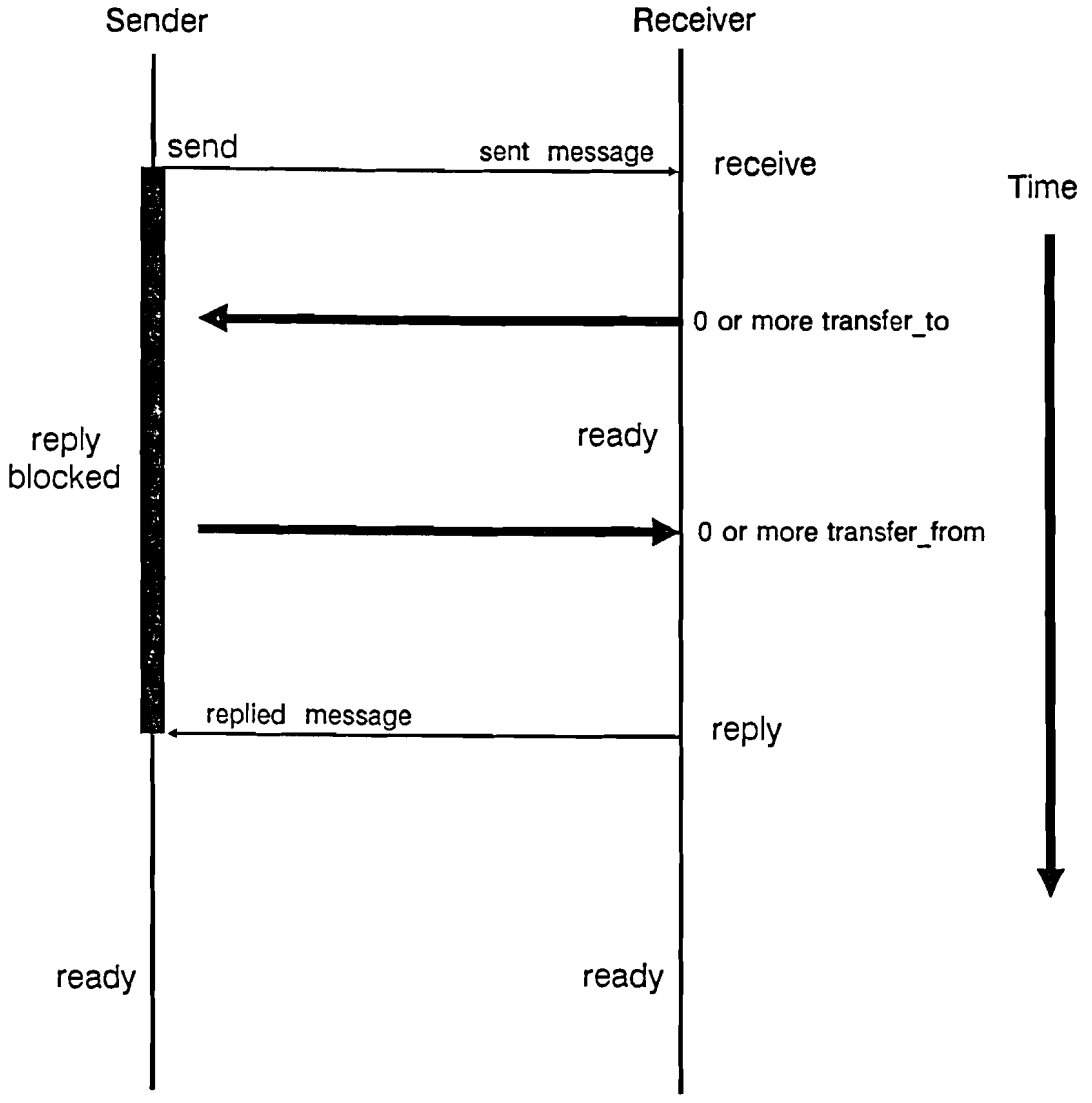

Figure 5.4 : *block transfers*

PROCESSES

As stated before, large parts of the Port operating system are implemented as system processes. A **process** is a single execution of a program that is identifiable and controllable by the operating system. The system processes of Port can be described in a number of stereotype process structures. We will describe these stereotype Port processes.

A **proprietor** process (figure 5.5) acts as a owner of some resource, i.e. only the proprietor can give access to this resource. The process implements operations for client processes, using the resource. The proprietor handles operations at first-come-first-served basis and can only handle one client process at a time. The client process is reply-blocked while the operation is being handled.

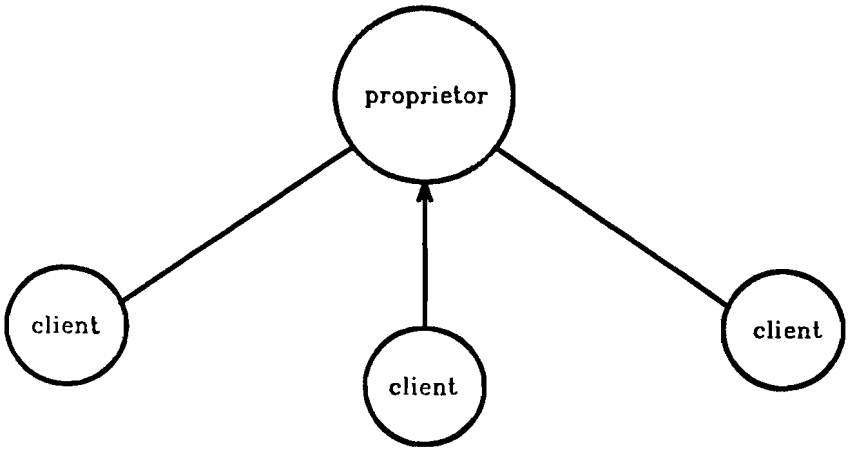

Figure 5.5 : *proprietor process*

An **administrator** process (figure 5.6) does not become "tied-up" processing a request, but can handle more then one client at a time. The administrator typically uses worker processes to perform certain tasks for it. While these worker processes are busy working, the administrator can handle other clients. This means that requests are not always handled in first-come-first-served order, because a short task is faster executed then a large one. So more then one client (or worker) can be reply-blocked at a time on the administrator.

A **worker** process is initialised by an administrator. It sends to the administrator process when his work has been completed and/or when he is ready to do some more work. The administrator later replies with a message indicating what work is to be done.

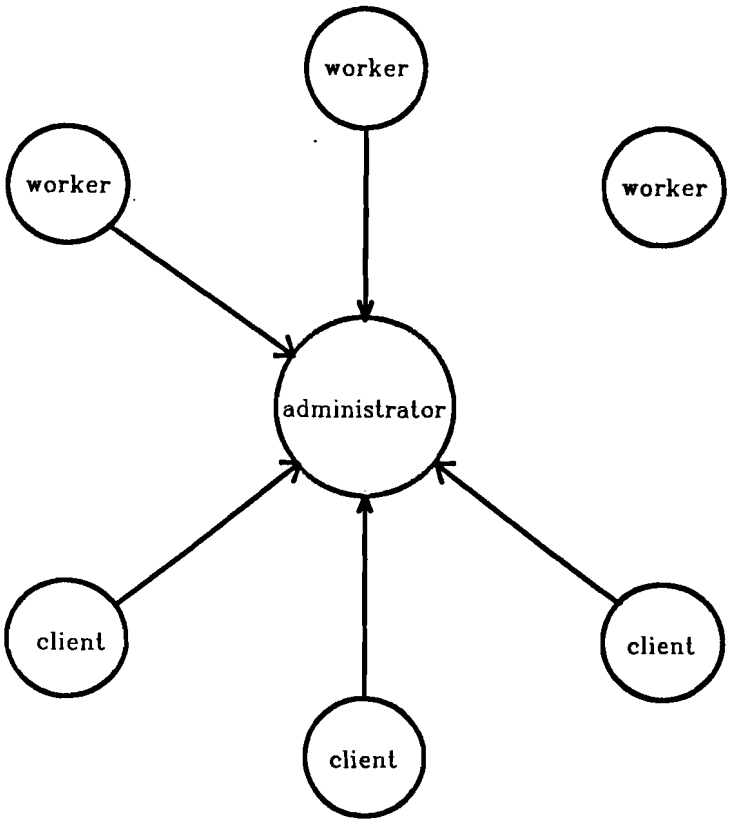

Figure 5.6 : *administrator process*

A **notifier** (figure 5.7) is a process which waits (blocks) for an event to occur. The event can be external (e.g. completion of disk activity) or internal (e.g. replied or received message). When the notifier notices an event it sends a "notification" message to a server, which usually is an administrator.

A **courier** is a special kind of notifier. It serves as a messenger between two servers. The courier sends to one server and the reply from that server is then sent to another server.

A **vulture** (figure 5.8) is also a special kind of notifier. Vultures are used to diagnose whether a client is still alive. Therefore it is receive-blocked on a client that will never send to it, because it does not know about it's existence. When the client dies, the receive fails and the vulture will send an obituary message to the administrator that was serving the client, and also started the vulture.

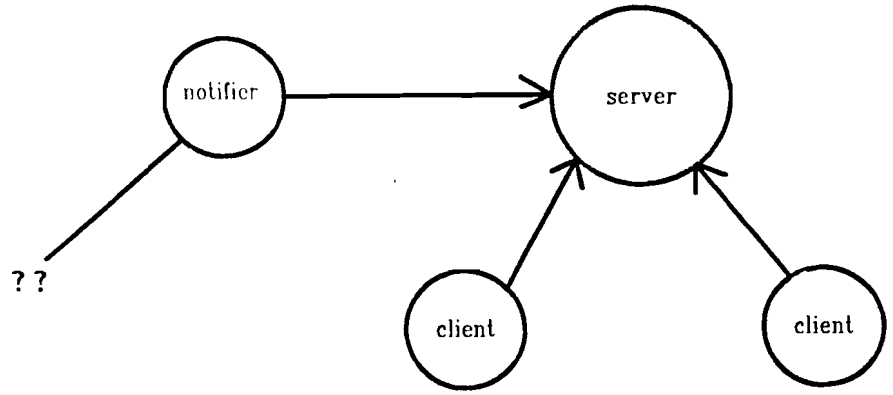

Figure 5.7 : *notifier process*

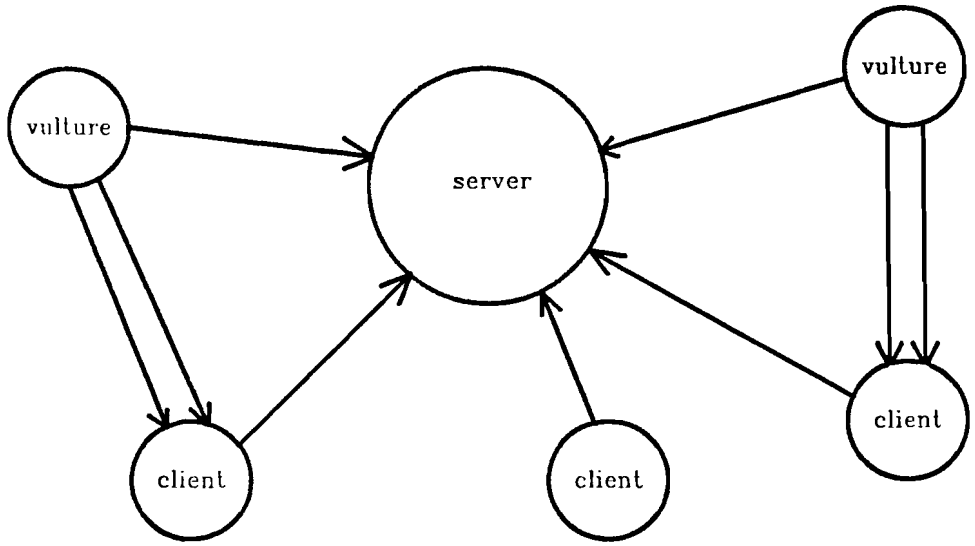

Figure 5.8 : *vulture process*

PROCESS SWITCHING MECHANISM

The process switching mechanism loads and stores the volatile environment of the process, and also determines which process be active next. Finding the next process to activate is determined by the processor allocation rule.

The **processor allocation** is handled at two levels, referred to as the micro level and macro level. The **macro level** sets process priorities according to high-level criteria involving memory scheduling and long-term processor utilization, attempting to recognize interactive programs and processor-intensive programs. The **micro level** allocates the processor by applying the processor allocation rule to the process priorities set by the macro level.

On each process switch, the micro-level process-switching routines locates the highest-priority ready process that has been ready the longest. This is done by keeping lists of waiting processes for each priority level. New waiting processes are placed at the end of the according priority list, and processes are activated from the beginning of this list. First the higher level priority lists are dealt with.

Process switching may occur because of two causes:
- a higher priority process becomes ready, either by a kernel operation invoked by the active process, or by an interrupt service routine;
- the active process blocks.

The absolute priority of a process is its team priority (macro level) plus its relative process priority. The relative priority of a process remains fixed, but the team priority of a transient team can be changed by the macro level, thus changing the absolute priority of all processes on the team.

5.2 The PORT user interface

Port has quite its own interface toward users. Instead of directories and menus Port uses rooms and icons. A **room** is a full screen size space in which a number of activities can be present as well as references to other rooms. Port shows these activities and references to other rooms in the shape of icons. An **icon** is a small square block which contain a drawing or a text. A room can contain up to 28 icons. The icon for a reference to another room is called a **door**.

ROOMS

Inside Port a room is the tool to group activities. For the various tasks which can be carried out on the system, special rooms can be created, which contain the activities that are needed for this task. There are 2 kind of rooms:

- rooms which can be controlled by the user himself and where he is allowed to make changes and create new rooms;
- rooms which are controlled by other users and which can only be used by the user, but may not (permanently) be changed by him.

An example of the first kind is the **office**, the room which is entered by the user after the access-procedure. Examples of the second kind of room are the **supply room** and the **network room**. A drawing of an example room is given in figure 5.9 (from Port user guide).

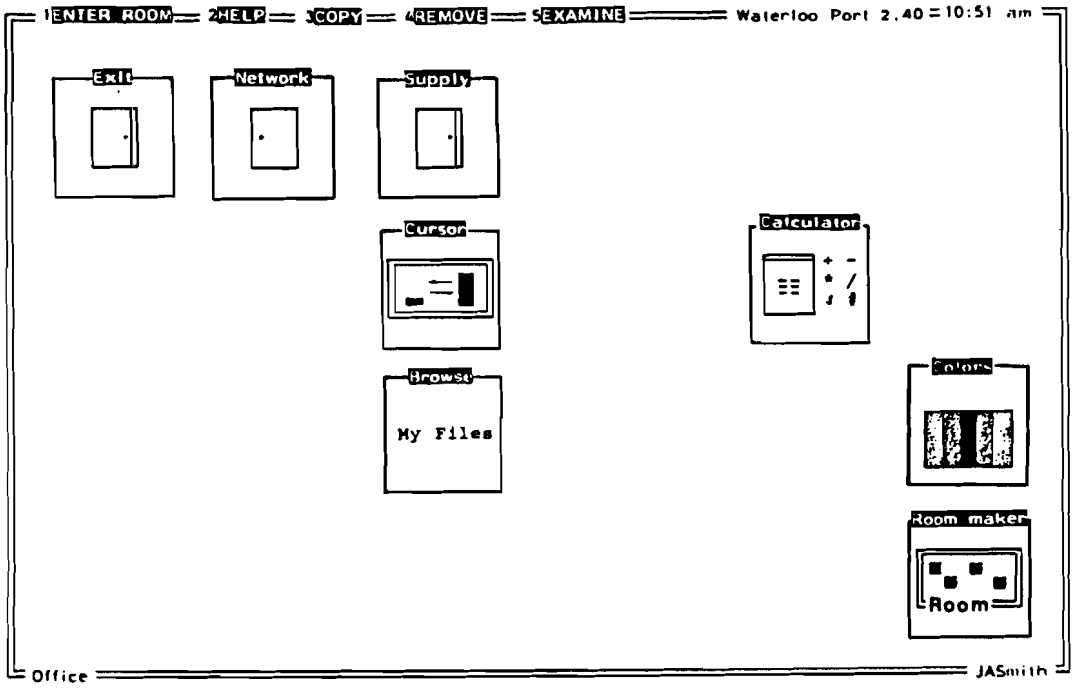

Figure 5.9 : *example of a room*

Every room at least contains a door with the label **exit**. This door brings the user back to the room from where he entered the current room. Possible other doors in the room can lead to other rooms. The rooms have not been structured in a hierarchical way, but they can be coupled in all kinds of ways. This means that a room has not to be left via the room by which it was entered, but there can be other ways. For instance it is possible to realize a circular structure with rooms (figure 5.10). From room A we can enter several rooms, for example room B. From room B we can return to room A or enter room C. When room C has a door to room A we can go back to room A directly. We also can go bach by returning to room B, and return from room B to room A. The system can only be left by the exit door in the users own office.

For clarification: rooms only contain activities and doors, and so it is **not** a directory structure, in which files are grouped.

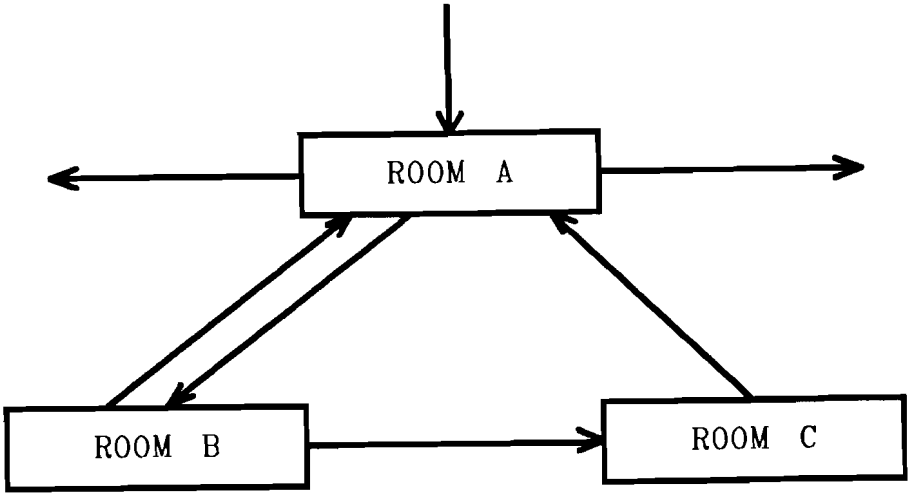

Figure 5.10 : *example of a circular room structure*

ACTIVITIES

An **activity** is a set of one or more processes which co-operate to perform a task for the user. An activity mostly activates its own window. A **window** is a variable length part of the screen which has been reserved for specific information. This window is used to execute interactively the activity and to present results.

Examples of activities are the mail-activity, which is used to prepare messages to other users and to send them or to read incoming messages, and the browser-activity, which allows to browse through the file-structure of a volume. DOS-applications like wordprocessing and databases also can be started as an activity. In figure 5.11 the windows of a browser and the calculator-activity are given as an illustration how windows look like.

Port can handle several activities at the same time. This is called **multi-tasking**. This means that a user does not have to wait until an activity is finished or closed, to be able to start a new activity. Unfortunately this is not valid for Port's DOS applications. Only one Port DOS-activity can be active, besides other Port-activities.

ICONS

Port knows 2 kinds of icons:
- icons that can start an activity;
- icons that present a reference to another room: doors.

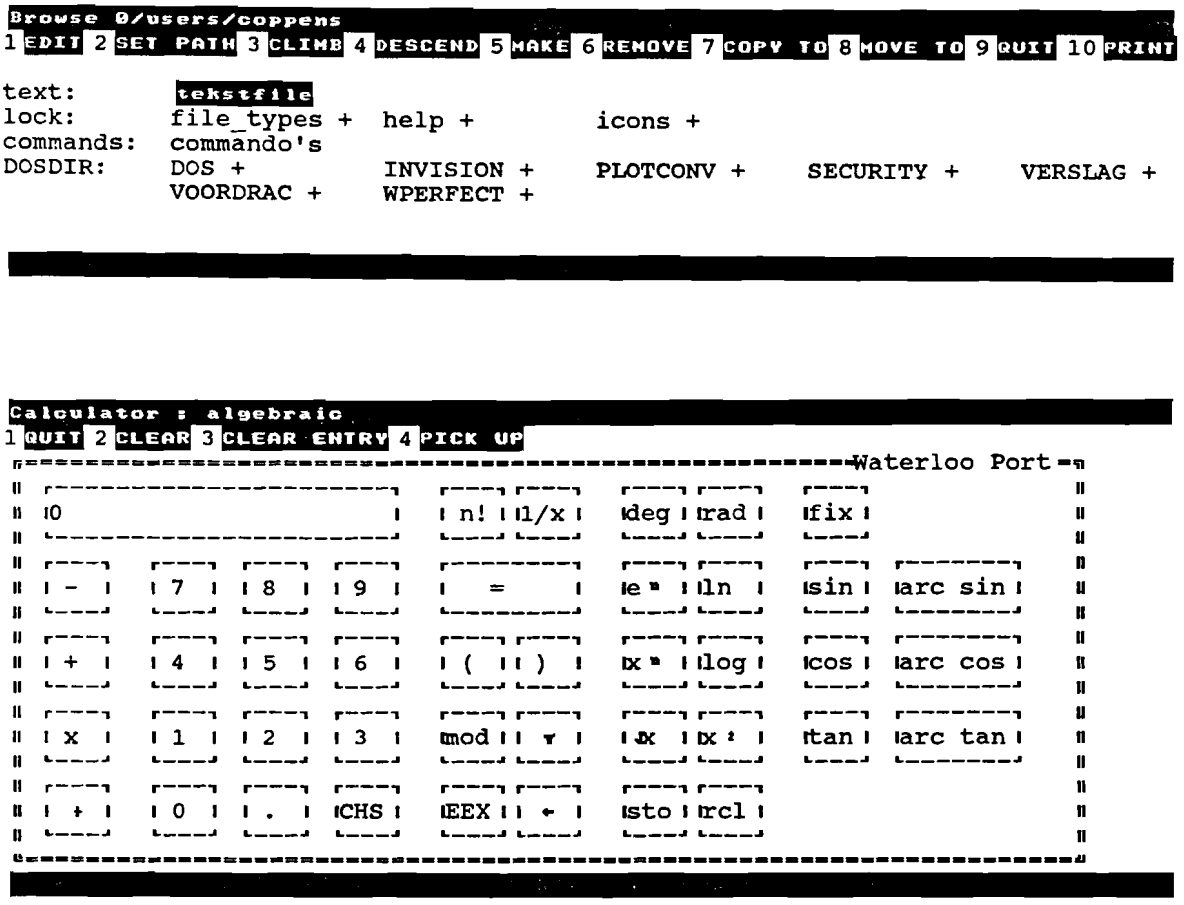

Figure 5.11 : *the browser and calculator window*

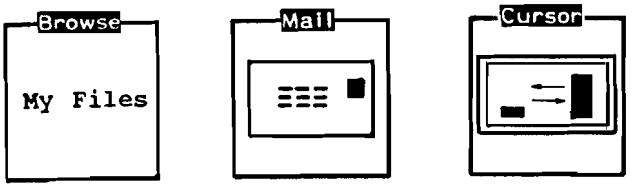

Figure 5.12 : *some typical icons*

Some typical icons are presented in figure 5.12. Icons are to be selected with a mouse or the cursor on the screen. After selection of the icon, a number of functions can be executed. The most important function is starting the activity or entering the indicated room. Other functions are the help-function, the copy-function, the remove-function and the examine-function.

Every icon has a **label**. This label contains the name of the activity which can be started by the icon or the name of the room which is referred to by the door.

Furthermore every icon has a number of **fields** in which the features of the icon have been stored. These fields are only visible while examining the icon. A field is a piece of the screen suited for input of certain information, for example a name or an instruction. There are fields for the contents of the icon, the location of the helpfile and the activity to be executed.

Depending on the activity to be executed, one or more **parameters** can belong to an icon. These parameters contain further information concerning the activity which has to be executed.

In principle icons can be copied, moved and removed. However, these features can be blocked by the system manager or the user.

FILE STRUCTURE

Inside Port there exists a multiple of **file-types**. During the creation of a file the user selects a certain type of file. Every file type has its own features and properties. The file types can roughly be divided in three categories:

- file types for **normal Port usage**:
  text, lock, commands, code 86, binary, library;
- file types for **Port's DOS usage**:
  DOSDIR, DOSV, DOS, DOSH, DOSR;
- file types for **Port's programming activities**:
  C, function, external, manifests, template, asm 86.

Underneath almost any file new files can be created. In this way we get the well-known tree-structure of files, as we know from other operating systems like DOS and UNIX. Only here the nodes are also files and may contain data.
The file types which can be placed underneath a file depend on the concerning file type. Not underneath all file types all other file types can be created. This is especially the case with Port's DOS files.

Port needs his own partitions on disk. A port partition on a floppy disk occupies the whole disk. On a hard disk one Port partition and several DOS partitions can exist together. The size of the partitions is not assigned. In figure 5.13 the possible partitions of hard disks is given.

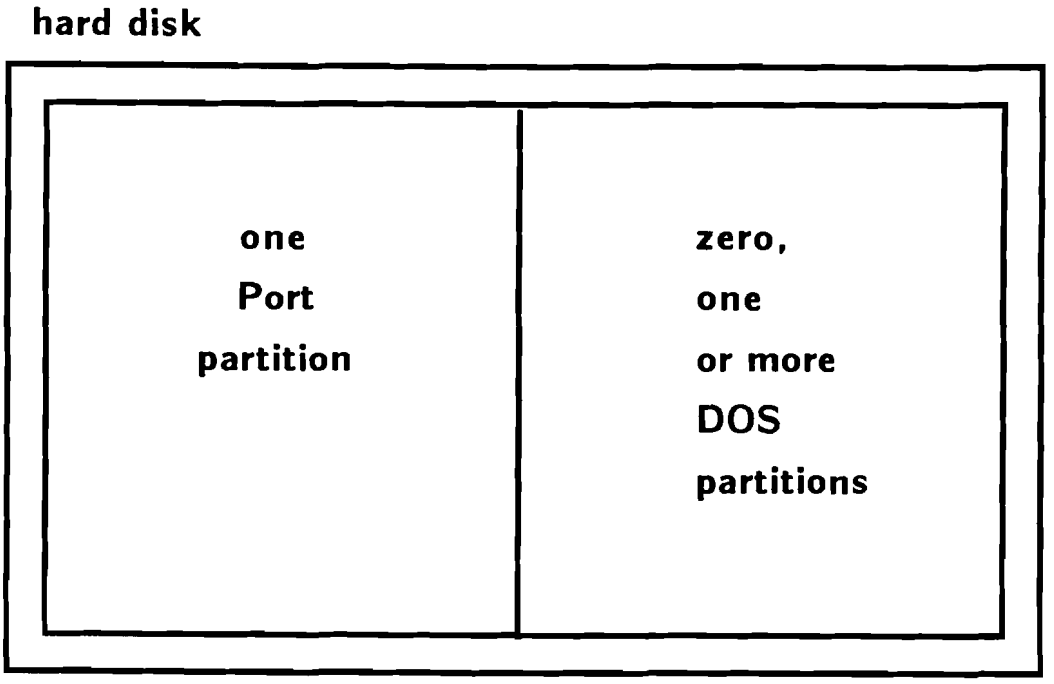

Figure 5.13 : *possible Port and DOS partitions on a hard disk*

PORT'S DOS STRUCTURE

DOS is a guest operating system within Port. It is implemented to make it possible to use DOS applications within the Port operating system. Because the use of DOS as a part of the Port operating system some DOS applications which are not programmed correctly may cause problems. For example when that application use interrupt levels which are reserved for Port itself.

Port's DOS file types can be used within Port, as well as in Port's DOS. Normal Port usage files and Port programming files can not be used in Port's DOS. Normally (MS)Dos files cannot be used within Port itself, but they can in Port's DOS. Schematically this is presented in figure 5.14.

In the previous paragraph we saw there are 5 types Port's DOS files:

DOSDIR : acts as a DOS directory
DOS : acts as a normal DOS file
DOSH : acts as a hidden DOS file
DOSR : acts as a read only DOS file
DOSV : DOSVolume, acts as a complete file structure in DOS, but as a single file in Port

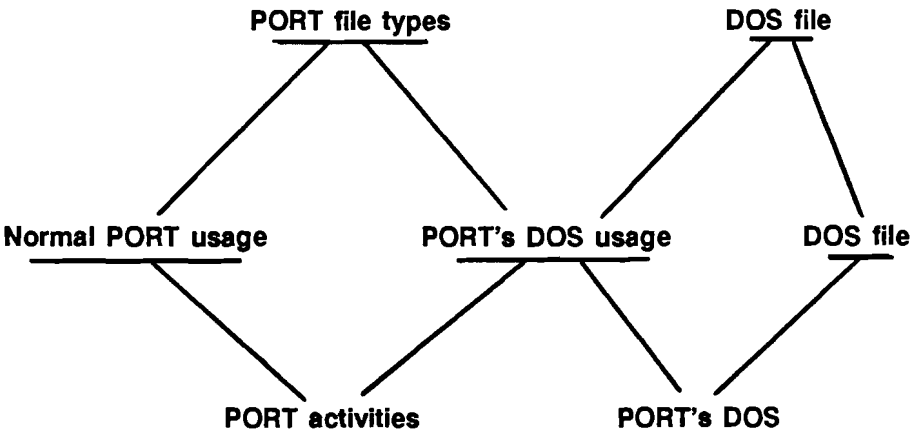

Figure 5.14 : *PORT's DOS structure*

In Port's DOS up to four DOSDIR's or DOSVolumes can be made accessible for use. These DOSDIR's or DOSVolumes can be accessed by using virtual disks E: till H:. The local floppy disks and hard disk DOS partitions also are accessible to Port's DOS.

NETWORK

Port is a network program. This means that a workstation which is embodied in the Port network, can communicate with any other workstation which has been coupled to the same network. Port knows 2 types of stations:

- **network software servers**, which contain network software and at least have one hard disk;
- **ordinary workstations**, just called workstations which may serve as a file server or printer server.

A network consists of at least one network software server (NSS). A software server must be started with a booting diskette. Workstations get there network software from a server and don't need to be booted from a floppy disk.

The use of the network manifests in the collectively use of data volumes and printers. Every volume or part of a volume as pleased can be placed at the disposal of the network, or can be withdrawn from it. Users don't notice the difference between their own volumes or network volumes.

Port knows two kinds of printers:

- **local printers** which are only accessible from the attached station;
- **network printers** which are not only available to the attached station but to all stations connected in the network.

The choice for making a printer local or network is made on installation of the printer and can be changed easily.

6 INFORMATION SECURITY WITHIN WATERLOO PORT

In this chapter we will look at the information security aspects of Port. The description will follow the structure of these security aspects. We start with the access control of the system, followed by the file protection measures. In 6.3 we will look at the possibilities to protect tasks. Encryption within Port is the subject of 6.4. Finally we will compare the security of Port with our model, to check the effectiveness of Port's security measures.

6.1 access control of Port

Access control is the measures taken to limit the accessibility of the system. Selectivity is essential for effective access control.

The access control of Port takes place at the beginning of all tasks. A workstation which is started with Port enters the entry room, the **lobby**. This lobby contains not a single activity and only one door, the **entrance** door. The entrance has the shape of a room, but in principle it is an activity, where the access control takes place. Once the user is admitted by the entrance activity, no further access control takes place.

The access control within Port has two major functions:
- selective admitting persons to make use of the system;
- identifying users to make further selective access possible to files and tasks.

The method of access control uses a **users-identification (userid)** and a **password**. Both userid and password may maximally be 16 characters long. The actual access control takes place on userid + password. If one of both or both are not correct then one returns to the lobby without acknowledge what went wrong.

The userid is shown on the screen during typing. The userid may exist of alphabetic characters, digits, points and apostrophes. There is a difference between lower- and upper-case characters.

The password is not shown on the screen during typing. It only consists of alphabetic characters. There is no difference between lower- and upper-case

characters. The password is stored into the system in an encrypted form and cannot be read by anybody, even not by the system manager. However the system manager is able to remove the password.

We can place some notes on this access protection of Port.

- The chosen method of userid and password is a simple access method. The knowledge of userid and password allows anybody to enter the system. This method ia probably chosen while it is the only method which not requires any special hardware, and so it is simply applicable anywhere.

- The used method of a visible userid and an invisible password is an obvious choice. No information is provided by the system concerning the correctness of userid or the password separately. This is a correct implementation.

- The userid mostly contains the name of the user, because this is used for identification throughout the whole system, and as such has to be recognized by other users. This means that this userid mostly is not secret, and does not really contribute to the protection.

- The most powerful userid, the one of the system manager, is equal to all Port systems and cannot be changed. So this userid has the greatest chance for unwanted use and thus must be supplied with an effective password. Fortunately this userid is owned by the person who cares most for security.

- In fact the password takes care for the access control. The more possibilities there are to form passwords, the better it is. So it is very strange that for the forming of passwords only alphabetic characters can be used, and above that there is made no difference between lower- and upper-case characters. Probably this is caused by the method of encryption to encipher the passwords.

- There are no requirements for passwords. Any password can be chosen by the user. This will lead to a number of easy to guess passwords, or even empty passwords.

- There are no requirements on periodically changing the passwords.

- A person can unrestricted keep trying to get access to the system. This is an unsafe situation. After some number of tryouts, the userid and/or terminal should be blocked and some measures should be taken to prevent this trial and error method.

6.2 File protection inside Port

The file protection inside Port takes place in the various file types. Each file type has its own protection aspects. This is an important difference between file types. For example the only difference between a DOS-file and a DOSR file is that the last one is a read only file, a file which can be read an not be (re)written. Other files are protected against editing. These are clear differences in protection.

There are two types of files that are especially aimed at information protection. These security files are the **lock-file** and the **DOSDIR-file**. The most important difference between these file types is the fact that the lock-file is a general Port file, while a DOSDIR-file behaves as a DOS-directory and especially renders its services during DOS-activities.

The lock-file and the DOSDIR-file both contain a so-called **permission-list**. An example of a permission list is given in figure 6.1. In this permission-list certain permissions are given or denied to specific users. The underlined permissions are admitted to the specific user. These permissions concern the file structure which have been situated under this specific lock-file or DOSDIR-file. The lock-file and DOSDIR-file itself contain no information beside this permission-list. The users are identified in the permission-list by their userid. There exists a special identification for all other users than mentioned in the permission-list, the **all- others** identification.

owner	**read**	**modify**	**append**
user_1	read	modify	append
user_2	**read**	modify	append
user_3	read	modify	**append**
user_4	**read**	modify	**append**
user_5	**read**	**modify**	**append**
all others	read	modify	append

Figure 6.1 : *example of a permission list*

The permissions known by the lock-files and DOSDIR-files are:

read : qualification to read files and subfiles
append : qualification to add data at the ends of files
modify : qualification to make, remove and change files

Of course these permissions can also be combined. Port knows the following combinations of permissions:

- blank : no permissions;
- read;
- append;
- read and append;
- read, modify and append : complete qualifications.

In order to determine the permissions of a user the following algorithm is used :

1) **If** the file is a security file, then proceed with step 2, **else** go up one level in the file-structure and repeat this step until a lock-file or a DOSDIR-file is found.

2) If the security file contains the userid of the user, or the all-others identification, then go to step 3, **else** go up one level in the file-structure and proceed with step 1.

3) If the userid of the user is mentioned, then the attributed qualifications are valid, **else** the qualifications which attribute to the all-others identification are valid.

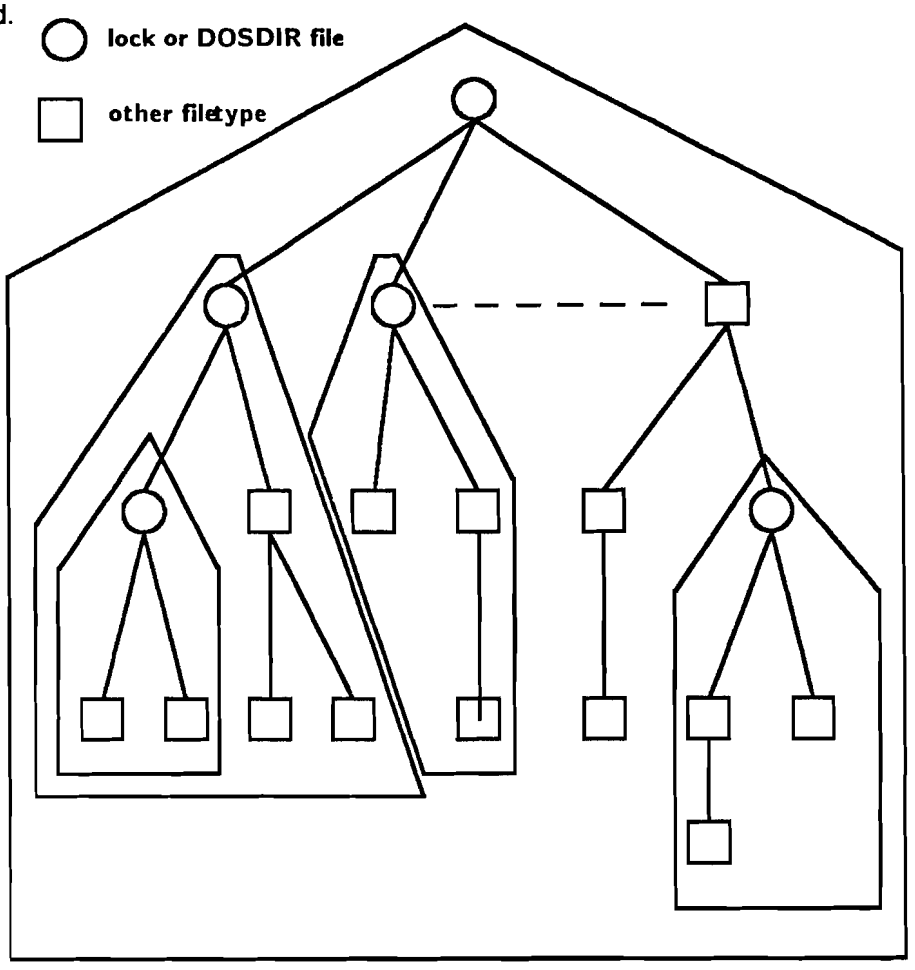

Figure 6.2 : *security file influences*

The lock-file and DOSDIR-file influences on the file-structure is graphically represented in figure 6.2.

During the creation of a lock-file or a DOSDIR-file the permission-list is empty. No specific permissions are allowed or denied. In this case the permissions are valid which have been defined in the higher lock-file or DOSDIR-file. An exception on this is the lock-file which is created during the making of a new userid. In this lock-file initially the new user is given all permissions, and all permissions are denied to all other users. The system manager has all permissions in all files.

The file protection of Port seems rather solid. A total reach of files is covered and there are safe defaults. A point of attention is the fact that a user can, by placing himself in the permission-list and by denying himself the modify qualification, lose his permissions, and he can no longer access them. Even in general, everyone who has modify permission in a lock-file or a DOSDIR-file can change the permissions of any of the users.

6.3 Task protection inside Port

Port knows 2 sorts of instructions for the execution of tasks :
- activities
- commands

Activities carry out various actions and control a window of their own. **Commands** only carry out one specific action and need not control a window of their own. This depends on the manner in which the command is started up, and some parameters.

Activities and commands can be started in two ways :
- by use of an icon ;
- by use of a file of type command.

USE OF ICONS

As a way of task protection, icons can be disposed limited. This can be done with use of the fields of the icon :

movable;
copyable;
removable;
examinable.

To restrict the number of icons that can be used, none of the available icons may be examinable. When an icon is **examinable** it can be changed totally. When the activity field, the Icon contents and the help file are changed, a total new

icon is created. In this way every other icon can be constructed. A room where we want to restrict the number of icons only may contain doors to similar rooms.

The **copyability** of an icon makes it possible to spread the icon throughout the network. Copyability in addition to examinability makes it possible to create all possible icons.

The **movable** qualification of an icon makes it movable, within its own room which certainly has no security aspects, as well as to other rooms. Moving from one own room to another has no security aspects as well, but moving an icon from a system room to one's own room is as good as a copy.

The **removable** qualification clearly has no importance for security. When limiting the number of icons, there is no need for removing icons.

The effectiveness of the restriction of the number of icons initially given to a user knows some stages. In the first stage a new user must get used to Port and will only use the icons in its own room, and those in the rooms he can reach by the doors in his room. After a while he will, when he is permitted, examine the icons and change parameters and fields. As soon as he will discover how to change the activity field, he is able to make every icon he wants. This can only be prevented by not making any icon examinable (and copyable) which this user can reach.

But even when the availability of icons is restricted, there is still the possibility of executing the instructions by using a file of the type commands.

(An exception for the possibility to create icons by the user himself are doors. It is not possible to change a standard icon into a door, or to change the parameters of a door, except for the system manager.)

USE OF COMMAND-FILES

By means of the use of a file of type command, it is possible to execute all commands and activities. So a limitation of using icons is not a sufficient protective measure. There also is need for a protection against using a command-file. In this case there is no limitation possible. Either there is a command-file available or there is none.

So when we want an effective task protection, we must take care we can prevent the availability of a file of type command. This demands two types of actions. First we must not give the user his own file-structure, because this enables him to make a command-file. Port gives the possibilities to do this. When creating a userid the system gives the choice between creating a lock-file for this userid or not. Second we must take care the user is not permitted

to create anywhere else in the system a single file. This at least means no single lock-file may contain a modify permission for the all others identification, unless the users we want to exclude are mentioned. This also means there may not be a browse icon available for a diskette station.

The limitations in using icons and files makes task protection difficult to implement. It has its effects on the total system and every user. The possibilities for the user whose available tasks are limited will be quit restricted.

However activities and commands can be masked for users by placing there load-files under a lock-file. In this way we can set up for every activity or command a user-list of their own. It is simpler to compose certain groups of activities and commands. These (groups of) activities and commands can be supplied to the users explicitly. As a standard, Port knows this protection for some system manager functions.

Probably even better is to create a lock-file for every userid and place the activities and commands this user may use in their own lock-file. Only the user has read permission in this file, all others have no permissions. This (as well as the previous solution) may cause problems with installing a new version of the operating system. But it is the only way to provide a real task protection.

6.4 Encryption within Port

Port has a command "crypt" to encipher information. It is used to encipher stored information. Enciphering and deciphering is done with the same key. This key has to be given by the user in the command line.

The key may contain up to 100 characters. A longer key is to provide a better encryption. The method of encryption is not public.

A user probably will encrypt all his files with the same key. Otherwise he has to memorize several different keys, and the file they belong to. Passwords probably will be popular keys, because they are thought of as secret and save. But the key is a part of the command line and as such visible on the screen while typing and using. This is very dangerous when the key is the password.

A better method is to put the command line in an icon, and encrypt and decrypt files by using this icon. For sure the command line must not be changed every time it's used for changing the filenames. Providing a standard icon which makes this possible is a useful addition to the icon set. Using an icon which contains the encryption key still has the disadvantage that the system manager can still read the key, and use the icon.

6.5 Conclusions

	confidential	reliable	continuity
preventive	- access control - selective access - encryption of information	- access control - selective access	- access control - selective access
repressive			
corrective		- recovery	- recovery

Figure 6.3 : *overview of Port security techniques*

In figure 6.3 an overview of the security techniques of Port is given in the programmable layer of the security model. We notice the preventive part of the model is best filled. The corrective part is only filled with recovery techniques, which we did not describe, however they are available in Port.

There are four empty minicubes in this layer. When we compare these mini cubes with their corresponding cubes in figure 4.2, we notice monitoring (both current monitoring and logbook monitoring) covers three of the four empty cubes. So implementing monitoring should be the next step to improve the security of Port. We will look at implementing monitoring in the next chapter.

7 MONITORING IN PORT

As we have seen in the previous chapter, Port has no monitoring capability at this moment.To conclude this report we will look at the possibilities to implement monitoring in Port. Therefore we will first determine where we want to use the monitoring for and try to develop a specification for monitoring. (7.1). Then we will work out a number of technical aspects needed for the realization and propose a base decomposition for the process to be developed.

In 7.3 we will describe the constituent process which gathers and stores the relevant information. Subsequently in 7.4 the constituent process which enables the user to consult the monitor information.

7.1 Specification

Monitoring is the registration of events in the system. For monitoring for security we have to register those events which interfere with the stored information and the correct working of the system. So for our purpose not every one event have to be registered. When we refer to monitoring in the future, we will mean the monitoring for security purposes.

Monitoring serves a number of targets:

- **deterrence**
  The knowledge that events which take place on the system are being registered, can restrain users from using or experimenting with illegal tasks.
- **limiting the damage**
  Monitoring can be used in two manners to limit the damage:
  1) Information about the reliability of the information can be gathered from the monitoring. This may prevent that unreliable information is used for further processing, which increases the damage.
  2) By a fast signalling of a breach of the confidentiality, there is a chance the confidential information can be recovered, or in some other way the damage can be limited, because of the breach is detected prematurely.
- **prevent repetition**
  When a breach of the security is detected, monitoring can retrieve by whom, where and when the breach took place. (We can only retrieve the userid which committed the breach, and not the actual user of the userid.)

To realize these targets we have to collect a number of data items:
- what happens to which information;
- by whom;
- where;
- when.

These data items need to be accessible in two ways:
- In a current survey where the present tasks of the system are registered;
- In a logbook which contains a registration of all performed actions on the system.

To make the data accessible, there has to be a couple of functions which can show the surveys. These functions should be user-friendly. This means that the user should be spared from unnecessary information. The system should do the selection of the information.

The monitoring naturally should be reliable. All (relevant) tasks should be registered at all times. The monitoring has to be resistant against any failure in the system.

Of course monitoring should effect the performance of the system as minimal as possible. This manifests for example in a minimal use of:
- processor time;
- memory, both temporary and permanent;
- communication over the network.

The interaction between monitoring and the user must take a similar form as other activities within Port. This means the surveys have to be given in the form of windows. Selection of an item in the window can be done with a "point" instrument, like a mouse or the cursor, and the select button. The selected item appears like usual in reverse video. There should be a default selection, preferable the most selected item, which can be combined with the upper left positioned item. The function buttons offer a number of possible actions for the selected item or the window as a whole. We must realize that within Port files only can be accessed by use of a command or an activity.

There must be two different base menus for the current survey and the logbook survey. The base menu for the current survey consists of:
- users survey;
- workstations survey;
- task survey (with parameters);
- total survey.

The total survey shows all information at once, as much as fits on the screen. Scrolling shows the rest of the window contents. It contents the start time of the task, the task with parameters, the user and the station number. The other surveys (users, workstations and task) only give a survey of the concerning category. Selection of one of the items gives the possibility to get a survey of registrations in which this item occurs (by using a function button "give constituent survey").

Because the number of registrations of the logbook is by far larger then the number of registrations for the current survey, we have to create a number of other surveys. A total survey is impossible, just as a survey of tasks with parameters. The base menu for the logbook consists of:

- users survey;
- workstations survey;
- task survey (without parameters);
- input filename;
- time schedule.

The users, workstations and tasks (without parameters) surveys act like those in the current survey, except that the surveys don't have to contain actually logged items, but they also may contain all possible items in this category.

7.2 Technical realization

We can see monitoring as a **process** which runs on the system, besides the numerous other processes in the system. In the future we will refer to monitoring as **the monitoring process**. The technical realization consists of an investigation of the technical aspects of monitoring. For this purpose we shall decompose the monitoring process in more simple (standard) processes and examine the interaction with already existing Port processes. The available information about the Port processes is quit limited and is derived from a coarse map "waterloo Port operating system".

As mentioned before, monitoring is the registering of events in the system. For this purpose information is gathered and stored in some place. Current information and logbook information demand different methods for storing:

- current information is **kept** in a temporary memory;
- logbook information is **registered** in a permanent, non-removal memory (often a hard disk).

ALLOCATION OF THE MONITOR PROCESS

The monitor process has to be present in the system all the time. As soon as the system is started up, the monitor process also should be started.

The monitor process uses both temporary as permanent memory. In principle we can realize the monitor process in two different ways:
- every station has it's own monitor process with the accompanying memories (decentralized);
- there is one central monitor process which gathers all information in a central memory.

Decentralizing the monitor process creates some problems for stations without permanent memory. But even when all stations have their own permanent memory, there is still the disadvantage the logbook information only can be accessed when the station is available for network use. This means for Port the system must be powered on and the network program must be loaded.

This demand to achieve a minimal fragmentation of the logbook information over the various stations, is a reason to chose for a central monitoring process. In principle it is possible to make a central logbook monitoring and a decentralized current monitoring. However, we will not consider this possibility in this report.

The monitor process can best be allocated in the server which supplies the user identifications. This has the following benefits:
- there is always one and only one of such a station in the network active at a time;
- mostly this will be the same station;
- current monitoring always runs on only one station while the logbook information spreads among a minimal number of stations;
- when this station should fall out other stations can not access permanent memory, because they need to be identified for that (So then there is no need for monitoring at these occasions.);
- when another userid server takes over the identification tasks, this station will also take over the monitoring.

DECOMPOSITION

When we take a closer look at the monitor process, we see we can split up the process in two constituent processes (figure 7.1):
- the **information gathering process**, which gathers, processes and stores the required information
- the **information query process**, which takes care fore the interaction with the user and the accessability of the logged data.

This are two independent processes which hardly need any mutual communication. (There is an exception for the exchange of some pointers.) The processes make use of the same resources: the logged information in temporary and permanent memory.

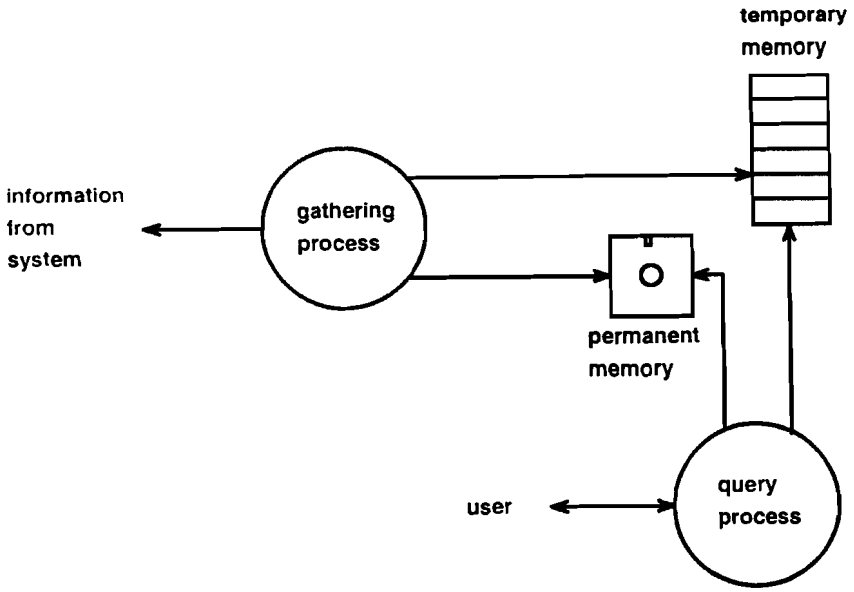

Figure 7.1 : *base decomposition of the monitor process*

The **information gathering process** must gather the required information. The required information is a number of fixed types of information. As soon as a change in the system takes place, the gathering process receives a message. These messages are handled in two ways:

- it registers the changes (when they are relevant) in the logbook. For this purpose it writes the information (sequential) to permanent memory.
- it adapts the current information in the temporary memory to the new situation.

The **information query process** provides the user with a couple of functions, by which he can recall the logged data. As a result of such a function, the query process accesses the temporary or permanent memory and processes the data to achieve the demanded surveys. It also provides the interaction between monitor process and user.

By dividing the monitor process in an information gathering process and an information query process, we have to make a division of their functions. There are two possibilities:

- the information gathering process collects the data and creates the diverse lists needed for the query process;
- the information gathering process only registers the collected data in a simple way, while the query process processes the data to provide the desired surveys.

When we have to generate a large number of different surveys, the first possibility is quit unpractical. Some disadvantages are:

- demands more processing time for storing;
- demands more memory space;
- creates a lot of surveys which might be used at all.

The second possibility has the disadvantage the surveys have to be put together at the moment of demand. This will provide longer response times, especially for the logbook surveys.

Regarding the specification demands of performance it is clear we have to chose for the second possibility: create the surveys on demand. It is much more acceptable to wait for the results of processing a demand then have longer response times every time.

The information to be registered must be kept in memory. It is important how this is done. From one side the information to be registered must take minimal memory space. From the other side the data must be stored in an orderly way to be easy to access. From the point of memory space, variable length data fields are attractive, however for an easy accessability fixed length data fields are preferable.

To be able to make a good choice between variable and fixed length data fields, the data structure of the information to be registered must be known.

Every registration forms a **record**. A record contains:
- a time indication;
- a station number;
- a user name;
- a task name with parameters.

The **time indication** consists of two parts:
- date yy / mm / dd year/month/day
- time hh : mm : ss hour/minute/seconds

The **station number** is a number between 001 and 255, consisting of three digits.

The **user name** is a string of maximal sixteen characters.

We distinguish three kinds of **tasks**:
- activities;
- commands;
- DOS-activities.

DOS-activities are in principle activities, though we do not only want to log a DOS-activity is started, but also what DOS-activity and what files are being used. This part will not be further developed in this report.

The syntax of **activities** and **commands** is:

activity_name <option> <unlabeled_argument>
command_name <option> <labeled_argument> <unlabeled_argument>

The parameters option, labeled_argument and unlabeled_argument can occur zero, one or more times in the task. A survey of all standard activities and commands is given in the appendix.

The data structure of these names and parameters is as follows:

- **activity_name** and **command_name** are strings, varying from 4 to 23 characters.
- **options** consist of a "+" sign, followed by one or more characters. Options may be abbreviated to their first letter. The number of options vary per activity and command, with a maximum of 8. When an option is omitted, a suitable default is used.
- **labeled_arguments** have the form:
  label = string
  The label may be abbreviated to the first letter. The string length may vary from 1 to 255 characters. The number of labeled argument differs per activity and command, with a maximum of 8. Labeled arguments are usually optional.
- **unlabeled_arguments** are strings of characters. Unlabeled arguments can be all kinds of information, for example userids, file names or plain text. There is no maximum length for unlabeled arguments. There may belong up to 32 unlabeled arguments to a single command or activity.
- **file names** are a kind of an unlabeled argument. They are described by their full name, preceded by the station-address name. File names may be up to 255 characters.

From this data structure appears that we can not suffice with fixed length information fields. Because the varying number of parameters and the variable length, these parameters must be stored in variable length information fields.

The time indication, the station number, the user name and the task name can be stored in fixed length information fields. We can use a **pointer** to refer to the variable parameter fields.

Because the partition between the mutual parameters and between the mutual parameter records is not clear, we must partition the parameters and records by special characters:

- end parameter character (EPC)
- end record character (ERC)

We have to make sure these characters can't occur in the parameters themselves

The structure of the information storage is graphically represented in figure 7.2.

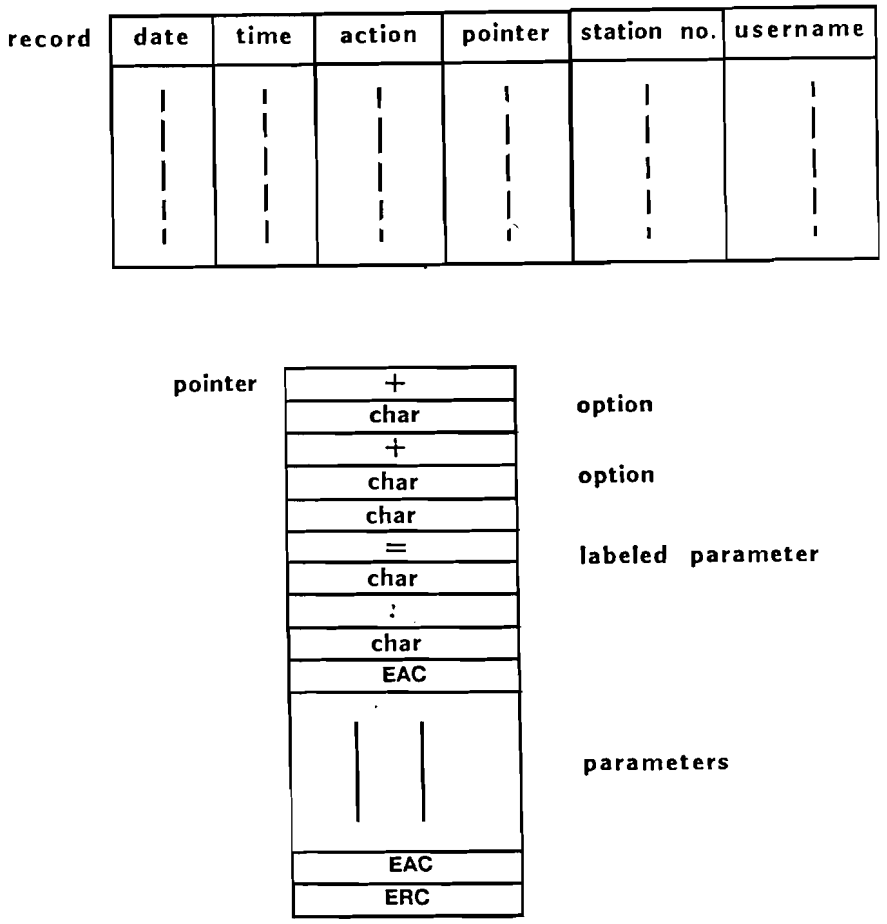

Figure 7.2 : *data recording structure*

SELECTION AND COMPRESSION

It is not necessary to register all activities and commands. Here we can distinguish the current information from the logbook information. How we make the decision which tasks to log and which not is not clear yet. In principle we can say that tasks which use stored files must be logged.

To limit the amount of information to register we can compress the diverse items.

- The **date field** is not important for the current information and may be compressed for the logbook information. The year digits can be left out easily, because they are of no importance. To register the date only when

changes occur, or by starting up the monitor process, saves some characters per record but makes the access more difficult.

- From the **time field** the seconds characters are of minimal importance and may be left out of the registration.

- Because the **station number** is a number between 1 and 255 the internal registration in Port can be achieved in 1 byte.

- **User names** can be compressed by connecting user names to a code. When we make this code 1 byte long, we can handle 256 user names. Because this may be to limited we can better take two bytes.

- **Activity names** and **command names** may be compressed as well. We can easily compress the names by abbreviating long names to 7 a 10 characters. Another possibility is to connect every activity name and command name to a code. Because there are about 20 standard activity and 50 standard commands we may code them to a single byte.

- The **pointer** will use 2 or 3 bytes, dependent on the type of implementation.

In this way we can limit the fixed length of a record to 10 or 11 bytes (figure 7.2).

time	station no.	user name	task name	pointer
4	1	2	1	2 a 3

Figure 7.3 : *fixed length record after compression*

The parameters from the tasks place us for bigger problems.

- The **options** are easy to compress to the "+" sign and their first character (2 bytes).

- The **labeled arguments** can be compressed to the first letter of the label and the "=" sign, followed by a string. The string, up to 255 characters, may be a pathname.

- The **unlabeled arguments** can be all kinds of names, under which filenames, and text. The names have a maximum length and might be compressed. The text in the tasks often will be of minor importance for security, and may be omitted.

On the possibilities to compress filenames and other unlabeled arguments some further examination has to done.

7.3 The information gathering process

Every station has its own operating system with activities and commands. So **locally** has to be observed that a task is started. This local notifier should intercepts the task name and his parameters by continuously locking at the tasks. Eventually selection of tasks may take place here. Tasks which should be logged are send to the **central** monitor process together with the user name and the station number. This delivers the following decomposition (figure 7.4).

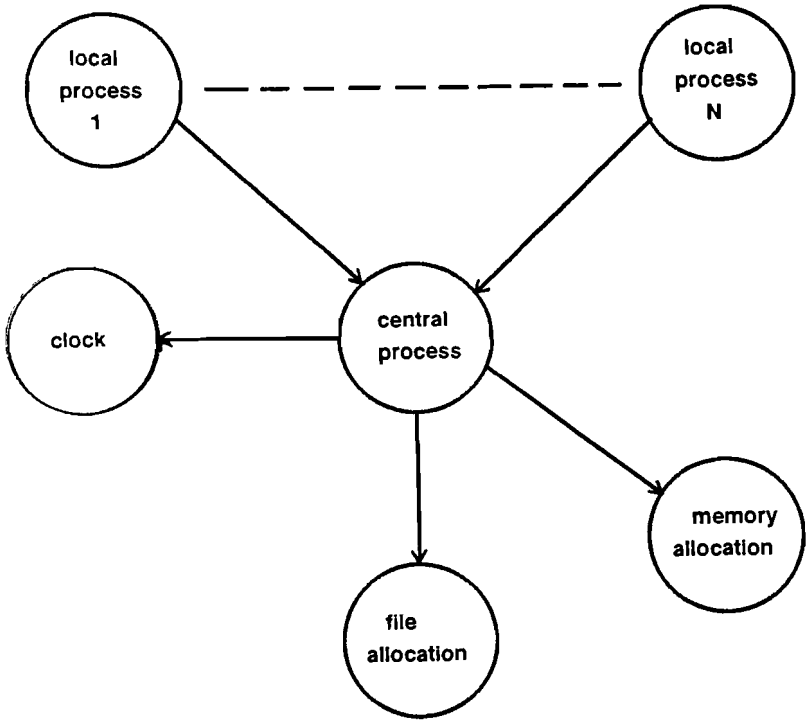

Figure 7.4 : *the information gathering process*

The **clock process** delivers on demand of the central process the date and time.

The **file allocation process** takes care for the access of the permanent memory and consists of two standard Port processes: the access-coordinator and the space-proprietor. Because the exact message primitives of these processes are not known, we can't describe the interaction between these processes.

The **memory allocation process** takes care for the access of the temporary memory. About this part of the Port structure is very little known. We may assume some kind of such a process is present within Port.

In principe, the **central process** is a control process. This process gets information delivered from the local processes and handles this information. (The station which accommodates the central monitoring process itself also accommodates a local gathering progress.)

There is a difference between the start of a task and the end of it. An "end task" is only used to delete information from the current monitoring memory. A "start task" message is used to log the task in both temporary (current) and permanent (logbook) memory. The messages are written down in a structured way.

The central process has a proprietor structure: it serves only one local process at a time. Because all local processes send their messages with the same priority, the longest waiting local process message will be handled. To prevent problems which may arise at the local processes (a new message to be sent, while the previous isn't processed yet), the central process has to be fast enough to handle the messages, or messages must be buffered locally.

The **local gathering process** notices the start and end of a task. When it intercepts a task signal, it collects the station number and user name, and sends the complete information to the central process. Between the (remote) local agents and remote clients are located, which take care of the network communication.

The local gathering process can be subdivided in a local control process, a communication process (remote agent), a task notifier and user name and station number couriers. The coherence between these processes is shown in figure 7.5.

The local control process initializes the other local processes. It acts as a administrator process. It waits for a message of the action notifier, that a task is started or ended. After this the control process push the user name and station number couriers to provide the user name and station number. When this information arrives the entirety is send as a message to the central process. For this, the process identifier of the central process must be known to the local process.

The selection and compression of tasks can take place in two locations: local or central. Local selection has the disadvantage of larger local processes. Central selection has the disadvantage of a lot of unnecessary communication over the network.

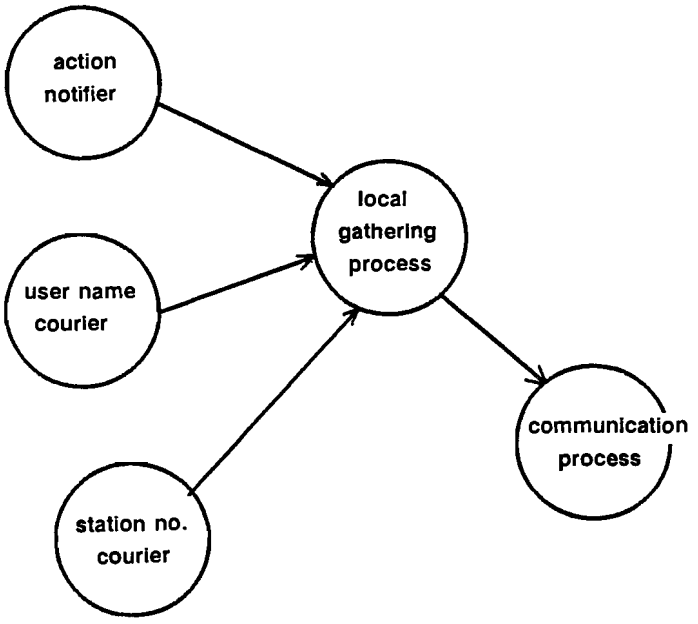

Figure 7.5 : *the local gathering process*

When the system goes down the information must stay accessible. The problems arise when files become unaccessible because they are not closed down when the system went down. The possibility of two identical files which are both kept up to date, and one file is only opened when the other file is closed, is not reasonable in this case. The logbook files are far too large to keep a duplicate. When the system is powered down there is still time to close the file. The problems arise when the system hangs itself.

When the monitor process is closed down, the pointer to the logbook must be saved. we can place this pointer (together with other ones) in some fixed place in the file, for example the first record.

7.4 The information query process

The query process forms the interface between the user and the registered information. It serves the user with a number of windows where various sorts of information can be asked for. The various information surveys are collected from data in temporary and permanent memory.

We can decompose the query process into a client server, a current query process and a logbook query process (figure 7.6).

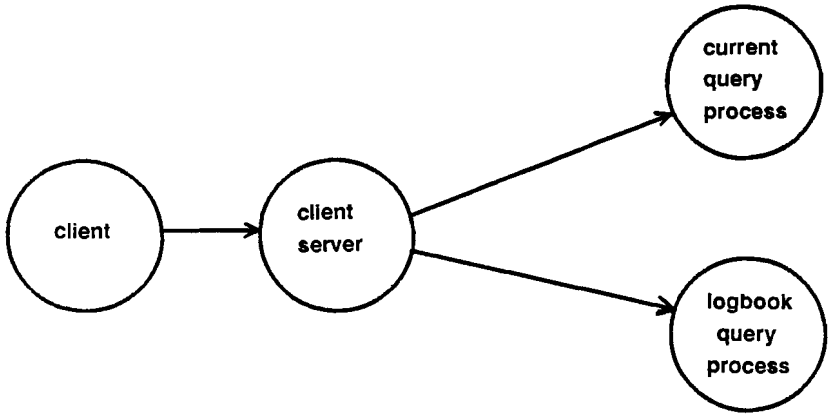

Figure 7.6 : *the query process*

The **client server** handles the interaction with the client. For this it takes care of the input from and output to the monitor windows, and sends special assignments to the current or logbook query process.

The **current query process** and the **logbook query process** process the assignments of the client server process. For this they determine what exactly is demanded, collect the necessary information from the resources and sends the information to the client in an ordered way.

The difference between the current query process and the logbook query process is, besides the type of memory to access, the quantity of information to process. The current query process processes little information, which is kept up to date all the time. The logbook query process processes much more information and doesn't need to react on current changes. The actual use of the logbook process is the detection of already happened events.

The divided query processes get a request from the client server process, for example, give a survey of the user names. The current query process searches in memory all user name fields and collects the names. These names have to be processed to remove duplicates. For this purpose we can use standard commands

like **sort** and **unique**. The logbook query process gets a list of all possible user names out of the system, for example by browsing the userid files.

A following demand can be what a specific user has done on the system. For this, in both processes the user name fields have to be searched for this user name. When the name is found the appropriate record is read and the search is continued till the end. The information in the record is send to the client server.

Backwards searching in memory and direct reflecting of the found information is especially useful for the logbook query process, because we are mainly interested in the last tasks (who last changed this file ?). This means the query process needs a **pointer** from which it can derive the place of the last record. This pointer best can be fetched from the gathering process.

When we search backwards, we need a method to **detect** the end of the information records. This is the beginning of the stored information. We can think of three methods to achieve this:

- a fixed starting pointer, known to the query process;
- a variable starting pointer, to be asked for at the gathering process;
- known, dummy start fields, known to the query process and stored at the initialization.

For the search for separate filenames we can make use of the pointers in the data structure. The last pointer will be read, checked whether it is a real pointer and not a dummy, and the concerning parameter fields are read (till the end parameter character EPC). After this the next (previous) pointer is read, etc.

The query processes need to be able to generate a number of lists, in which the items of a category are mentioned. For the realization of these lists we have the following suggestions.

The **current query process** "walks through" the concerning fields to make a survey of the current lists, with the held of commands like sort and unique.

The **logbook query process** uses three standard lists:

- A list of the **commands** and **activities** to log can easily be set up. This list is only changed by a software update, where this list also can be updated.

- A list of **station numbers** can be fetched from the system. There is an icon which lists the available station numbers, so the complementary information must be available as well.

- A list of **user names** is available by browsing the userids, and there may be even an easier way to achieve this.

CONCLUSIONS AND RECOMMENDATIONS

The conclusions are divided in three parts:
- security in LAN's;
- security in Port;
- implementation of monitoring in Port.

SECURITY IN LAN's

* Security of local area networks is still in it's infancy. LAN's rely heavily on their protective environment. They are not resistent against serious hackers (Paans, 1987).

* Preventive security measures mostly prevail over repressive and corrective security measures. The latter however should not be forgotten.

* In general, technical security measures must be supported with effective organizational measures.

* For each installation of a LAN, the required security should be determined by a risk-analysis. A LAN must provide the freedom to adept the required security measures an skip others.

* It is very hard to achieve a sufficient information security for work-stations. When it is possible to by-pass the network software to access data, only total encryption of all data provides confidentiality, while reliability and continuity can not be achieved.

SECURITY IN PORT

* Port is minimally secured.

* Authentication is poor because there is:
  - not a single obligation for passwords with regard to length, type of characters to use and prohibited passwords;
  - no possibilities for an obligatory periodical change of password;
  - no limitation in the number of chances to gain access to the system.

* The information protection looks solid, however:
  - permissions may become unclear because of the layered structure of security files;
  - there is no possibility to create user groups and mention them in security files, which is rather user-unfriendly;
  - the ownership of files is relative: creators can be banned from their own files.

* Task protection is well possible in Port with the help of security files, but not promoted in the manuals.

* Encryption is rather insufficient implemented because:
  - the crypt command does not function very well, regarding to key control and encryption results;
  - encryption of communication signals is not possible.

* There is not a single repressive measure available in Port.

IMPLEMENTATION OF MONITORING IN PORT

* Monitoring is only useful when the number of logs is minimized to the essential ones and the system gives the user exact that information he needs.

* Logged data should be treated in a data-base way (in records and data fields) in order to achieve clear surveys.

* Implementing monitoring in Port seems well possible. We lacked the detailed information about the process structure to finish the job.

* Compression of the information fields yields an enormous economy in memory space. Some investigation in the compression of parameters should further restrict the record length.

REFERENCES

Aalders, J.C.H., I.S. Herschberg, A. van Zanten,
Handbook for Information Security,
North-Holland Publishing, Amsterdam, the Netherlands, **1985**.

Bautz, J.F., A. Brouwer, A.J.F.M. Jongenelen,
Gegevensbescherming,
Kluwer, Deventer, the Netherlands, **1987**.

Cheriton, David R.,
The THOTH system: Multi-Process Structuring and Portability,
North Holland, Elsevier Science Publishing Co., Inc., Amsterdam, the Netherlands, **1982**.

Herschberg, I.S.,
The hackers' comfort,
Communicatie: Beheer, Bevelliging en Controle, R. Paans, ed.,
NGI, Amsterdam, **1986**.

Kent, Stephen T.,
Security in computer networks,
Protocols and Techniques for communication networks, F.F. kuo, ed.,
Prentice/Hall International Editions, Englewood Cliffs, New Jersey, **1981**.

Paans, R., I.S. Herschberg,
Computer Security: The Long Road Ahead,
Computers & Security, Vol.6, page 403 - 416, **1987**.

Tanenbaum, Andrew S.,
Computer Networks,
Prentice/Hall International Editions, Englewood Cliffs, New Jersey, **1981**.

Wood, Michael B.,
Computer Access Control,
NCC Publications, Manchester, England, **1985**.

APPENDIX : SYNTAX OF PORT's COMMANDS AND ACTIVITIES

1 STANDARD COMMANDS

admin/clear_archive_bit file1 file2 ... or <PATHNAMES

admin/erase_userid userid1 userid2 ... + file or <USERIDS +file

admin/nem_userid userid1 userid2 ... or <USERIDS
+file office=office_file password=initial_password

admin/nw_adresses +all +configured

admin/set_dos_info arg1 arg2 arg3 ... arg10

archive

backup <PATHNAMES (backup_root) (+file) +don't_clear_archive_bits
+suppress_prompt

change <TEXT or file=pathname
pattern replacement match=number

compare <TEXT file

copy source_file destination_file

count <TEXT

crypt <TEXT key

dearchive (index=index_file)

delay seconds

diff file1 file2 +ignore_blanks

dump file (+interactive or +patch)

ec <COMMANDS arg1 arg2 ... +suppress_output

edit file line=number

eu <ARGUMENTS cmd1 cmd2 ... +suppress_output

files root or <PATHNAMES
+full_pathnames +types (+immediate) (+changes) since=yy/mm/dd:hh:mm
before=yy/mm/dd:hh:mm

find <TEXT pattern +all_except

format_tape

fspace <PATHNAMES

fstats file1 file2 ... +terse or
<PATHNAMES +terse or
file_system=number or
file_system=remote_server_name

if <ARGUMENTS cmd1 cmd2 ... +suppress_output +not

list file1 file2 ... or <PATHNAMES

list_tape_files >index_file or +header

make file1 file2 ... or <PATNAMES
size=n label=l +suppress_output

move from_root to_root +suppress_output

network/attach type=files or type=printer
network_name+service_name (local_name+pathname)

network/detach service_name

network/station_adress (new_station_address)

network/stats +terse +diagnostics

print file or <TEXT
printer=printer_name copies=n line-spacing=m

printer/cancel printer=printer_name or +all

printer/default printer_name

printer/purge printer=printer_name or +all

printer/restart printer=printer_name or +all

remove file1 file2 ... or <PATHNAMES
+suppress_output

restore backup-file or <BACKUP_PATHNAMES data=data_file

screen_saver number_of minutes or +off

select <TEXT column1 column2

shut_down file_system=number

sort file or <TEXT
column=M length=N +descending

split <TEXT root_file number=count +lines

time +standard +date_only

translate_path #special_path1 #special_path2 ...

unique <TEXT +number

window <TEXT title=activity_name +scroll

write (+bold or +reverse or +underline) arg1 arg2 ...

2 ACTIVITIES

backup

browse pathname
subfunctions: edit
make
remove
copy to
move to
print

calculator title_line calucaltor_type display_format

characters

clock

colors

create file_system media_type

cursor

execute command window size visible scroll

locked

mail

pc_dos memory-size printers drives access-modes boot-drive

read_floppy floppy_image_file diskette_drive

restore backup_file

room_maker

saver +off number_of_minutes

shutdown file_system=file_system_number

user-keys

write_floppy floppy_image_file deskette_drive

GLOSSARY

activity
An activity inside Port is a set of one or more processes which co-operate to perform a task for the user.

directory
A directory is a network node in a file structure.

door
A door inside Port is an icon with a small door as contents. This refers to the room of which the name is in the label.

exit
The exit inside Port is the door which leads to the room from where the current room has been entered. The exit from used office leads to the lobby and this can also be regarded as the exit from Port.

field
A field inside port is a small piece of the screen (and in a data structure) (often a part of a line) that is suited for the input of certain information, e.g. a filename or an instruction.

file
A file is the smallest quantity of separated information which is accessible by the user.

icon
An icon (picture) inside Port is a small square block in which a drawing or a text can be found. These small blocks indicate activities or references to other rooms.

label
The label of an icon is the name of the icon. The label is the heading text of the icon.

lobby
The lobby is the entry room of Port, in which a workstation enters after the workstation is started up with Port.

local area network
A local area network (LAN) is a collection of mutually connected computer systems and peripherals which can exchange data, with a limited geographical distribution.

monitoring
Monitoring is the registration of events in the system.

multi-tasking
Multi-tasking means that a workstation can execute several tasks simultaneously in an active, pseudo-parallel way.

network room
The network room inside is the room which enables the user to "browse" through the file structures of the available volumes. These are all the volumes which have been put at the disposal of the network.

office
The office inside Port is the owner's own room, which he enters after the access procedure. In his own office the user has the right of property, and is able to create other own rooms.

process
A process is a single execution of a program that is identifiable and controllable by the operating system.

record
1) A record is the smallest set of information stored in the system.
2) A record is a collection of data fields which belong to one another.

room
A room inside Port is a full screen sized space in which a number of activities and doors can be present.

supply room
The supply room inside Port is the room which contains all icons which have been put at the disposal of the network. The user can copy from the supply room those icons he want to his own rooms.

volume
A volume is a file structure on a storage medium. This may be a floppy disk, a (part of a) hard disk or a tape.

window
A window inside Port is a variable length part of the screen that is reserved for certain specific information.

workstation
A workstation is a computer system which is especially meant for personal use, though multi-user capabilities may be possible.

INDEX