Information Processing of Japan Vol. 9, 1969

The Organization of a File System and On-Line Interaction

TAKASHI MASUDA*, SHIGERU MOTOBAYASHI*, TAKASHIGE KUBO*, IKUZO YOSHIDA, YASUBUMI YOSHIZAWA* AND NOBUMASA TAKAHASHI*

1. *Introduction*

System design of a time-sharing system involves many problems to be considered in relation to a file system.

We have implemented a time-sharing system with two-dimensional addressing feature. It has been implemented on a HITAC 5020 (65 K words, 1 word= 32 bits) by adding DAT (dynamic address translator), which has a segmentation and paging mechanism.

This paper describes some characteristics of this system with respect to the following points:

(1) Password algorithm

(2) File system

The file system has a tree hierarchical structure. The file structure, file search technique, common file processing, etc. are reported on. For the identification of each user, an "account name" is used and various types of accounting are executed on-line in terms of this conception.

(3) Mechanism for extending the system on-line from the terminal.

In the following sections, this system is referred to as 5020 TSS.

2. *Password Algorithm*

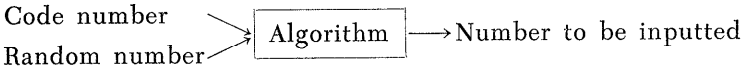
In a time-sharing system, for the privacy protection and accounting purpose, it is important to have a criterion on which a decision can be made, when a person proposes to use a terminal, as to whether he should be granted as the user of the terminal or not. Usually a password code is assigned to each user and it is inputting following the name of the user. The password code is not recorded at the terminal. If, however, this code is once glanced at by any stealing person standing behind the user, that is the end of it. In the 5020 TSS, a sensible method of privacy protection as described below is adopted.

Each user is assigned with a 4-digit number as his password code. At the start of using a terminal, the terminal user inputs his name (specified by the account name as described in 3) symbolically, and then a 10-digit random number

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Proceing Society of Japan), Vol. 9, No. 6 (1968), pp. 326-334.

* Hitachi Central Research Laboratory, Kokubunji, Tokyo.

will be outputted on the terminal. The terminal user combines the 4-digit code number and the 10-digit random number together and calculates in his mind in accordance with a certain algorithm unique to him to obtain a number to be inputted. This number is printed at the terminal.



Let us denote by i the code number, by j the random number, and by k the number to be inputted. Numbers printed at the terminal are j and k, and from i and j, k is determined uniquely, but from j and k, i can not be found. Since the algorithm used by the user is unique to him, it is almost impossible, if there are a number of hard copies of j and k, to predict i from them. Further, it may be considered not to print k in the method described above.

3. *File System*

3.1. Basic concept

In the following descriptions, direct access memory (such as disc) is assumed as file memory.

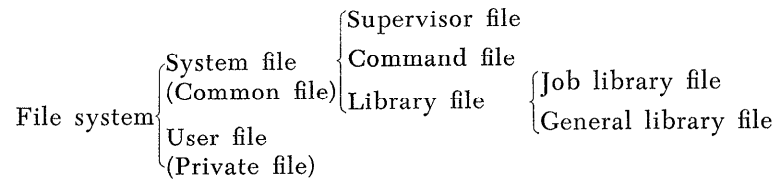
All programs or data other than wired down ones are arranged in files and registered on the file memory.

The 5020 TSS has a segmentation and paging mechanism as hardware, and in this system, any file is called symbolically as necessary during program execution and, at this point, the called file is assigned as a segment (dynamic linking), except some of wired down programs or data which are assigned previously as some segments, though most of them are called still symbolically. Files controlled by the file system are all the files which are called symbolically by dymamic linking described above.

The file system of the 5020 TSS has a tree hierarchical structure. In this section, the file system structure, file search technique, common (public) file processing, accounting method, etc. will be explained.

3.2. File system structure

The file system of the 5020 TSS has the structure as shown below.



Brief descriptions of these files are as follows.

(a) Supervisor file

All supervisors other than wired down ones are registered on the file memory and for which dynamic linking is applied.

(b) Command file

The command is a control statement which is used from the terminal to control start or stop of a job, creating, updating or outputting of a file, compilation of a source program, execution of an object program, or the like. Each command is registered as one file.

(c) Library file

There are two types of libraries, general library and job library. The general library is called symbolically from a general user program or system program. It contains compilers, too. For the 5020 TSS, PL/IW, a subset of PL/I, has been developed as terminal language. It can be used from the terminal by PLIW command. In the PLIW routine as command file, only the linking program to the compiler registered in library is contained. With such an arrangement, and by the use of file search principles described in 3.3, the PLIW compiler can be called symbolically from general user programs. The job library is a program which singly does one closed job. It is linked from a JOB command only. Any job library having high generality is upgraded to a command.

(d) User file

Under identification of each user, each file for the user is registered independently.

As seen from these descriptions, the file system on the file memory spreads four large branches from the root:

- Supervisor
- Command
- Library
- User

Discrimination between two types of libraries is made for each library within a library file. Each branch is followed by a first-stage "directory". The directory itself is one file and cotains a series of file names (including the next-stage directory name). File names contained in one directory are in the same level in the hierarchy of the tree. Each directory is followed by directories in any number of stages and finally always by a procedure file or data file. The procedure file or data file proper as separated from directories is called "entity" of file.

Account Directory and Terminal File Directory

In the 5020 TSS, the tree hierarchial structure is utilized also for identification of each user. The identification of each user is done by a symbolic qualified name, which is called an "account name". Each stage in the tree hierarchy corresponding to the account name is called an "account directory" and contains areas in which accounting statistics for system use are to be set. Each user is identified by a corresponding acount directory in the last stage. The account

directory in the last stage is called a "terminal account directory". The terminal account directory is followed by file directories for users, one for each, and finally always by the entity of file.

The procedure file or data file other than directory is preceded by a directory which is called a "terminal file directory". In the terminal file directory, various characters of the entity of file are entered, some of which are as follows.

- Whether a common file which is common to all users, a shared file which is shared by some users, or completely a private file (in this system, all procedures are reentrant).
- Access right (read/write/execute) and lock (the 5020 TSS has the ring protection mechanism by key and lock for each segment).
- Lock of programs accessible to this file.
- Size of the entity of file.
- Address to the entity of file.
- Generation date and purge date.
- Owner name.
- User names accessible to this file (they can be specified in terms of node names in intermediate stages of account directories), and indicators to indicate whether linked at present or not.

The terminal file directory has areas for various statistics. A schematic of the hierarchy is shown in Fig. 1.

The terminal account directory contains each user's password code, "user class", etc. The user class shows the accessible range for each user. All com-

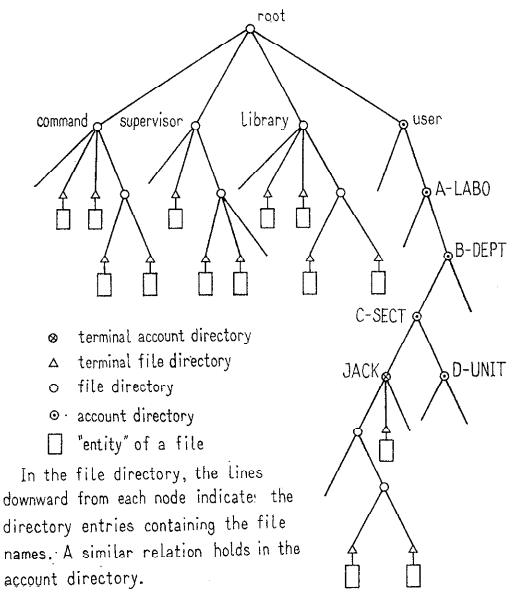


Fig. 1. Structure of file system.

mands are not usable by all users, and what commands can be used by a user is determined by the user class of that user.

Account directories are utilized for on-line accounting. For instance, in Fig. 1, pieces of accounting information on the terminal usage of JACK are accumulated not only in the stage of JACK on the account directory, but also in appropriate account directory entries, that is, C-SECT, B-DEPT and A-LABO. If JACK desires to use the terminal, he must identify himself by inputting

A-LABO. B-DEPT. C-SECT. JACK

Commands for the File System

Typical commands for a file system construction and updating are as follows.

(a) COPY and DUPLICATE commands

The COPY command is used to copy the entity of file (including the terminal file directory) from any one place on the directory to any other place.

The DUPLICATE command is used for copying directories alone: the entity of file remains uncopied and only the terminal file directory is connected to several directories simultaneously.

(b) FILE, DELETE and UPDATE commands

These commands can specify not only the last stage, but also any intermediate stage directory on the file system.

(c) GATHER command

The GATHER command is used to construct a tree hierarchical structure of files. This will be explained by an example. If there exist several sub-programs

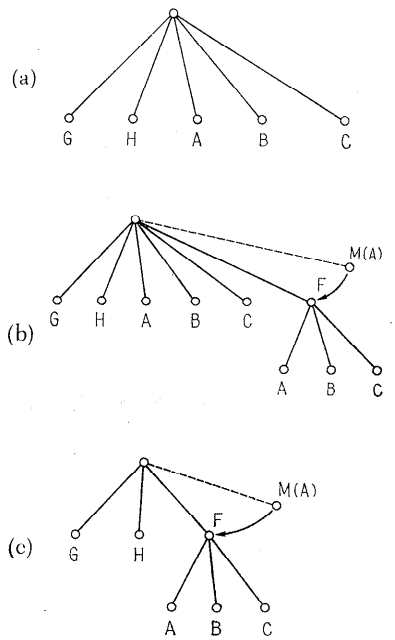


Fig. 2. Example of GATHER command.

in the same level of a tree structure, which call each other and each of which is registered as an independent file, they can be gathered together and be given a single name for the whole. Let us consider a tree structure as shown in Fig. 2 (a), where A, B and C are assumed to be programs which call each other, A being the main program among them. Inputting

GATHER (F, A, B, C)
*M ENTRY A

gives a new directory and the tree structure becomes as shown in Fig. 2 (b). Then, deleting A, B and C of the upper level results in Fig. 2 (c). For instance, by calling M from C, substantially it is possible to run A which is in the lower level. In such a case, M is called "external entry" with respect to A of the lower level. By the use of a GATHER command, it is also possible to merely gather several files or directories together to produce a new directory.

(d) PERMET, SHARE and SETCOMMON commands

These commands will be explained in relation to on-line extension of the system in 4.

(e) VSPATCH command

This command is not for updating the file on the file memory, but for "patching" some parts of the file indicated by the file name and its locations only on the virtual space. We found this command very useful for on-line debugging or on-line extension of the system.

3.3. *File search technique*

Take for example the case when, from a procedure file X in a tree structure, another file Y is called. The procedure for searching the file Y in this case is as follows.

(a) First, file names belonging to the same directory as X (file names hanging from the same node as X in the tree structure) are searched out, and if a procedure file having name Y exists in them, it is the file being sought. (It is assigned as a segment, and control transfers to it.)

(b) It may happen that name Y found by the procedure shown above in the file names belonging to the same directory as X is not for procedure file, but external entry. The possibility is seen in Fig. 2 (c). In the same figure, if M is called from G and found to be an external entry, A written in the entry of M, can be obtained. Any external entry has a corresponding directory node (in the case of this figure, F), and file searching proceeds seeking A in a level by one order lower. When A is found, it is assigned with a segment and the program runs at the level one order lower.

(c) If neither of (a) and (b) applies, that is, the file name being sought can not be found in the same level as the calling file, the first level in the general library is searched for the file name. This is regarded as library calling. By a

similar procedure to (a) and (b), search is continued within the library directory. When the file name being sought is found, control transfers to it.

(d) If non of (a), (b) and (c) applies, it is the case of an undefined file and the call is regarded as error.

These principles are applicable directly also when the file being sought is a data file, not a procedure file.

Any case so far given refers to the call by a "call name", that is, a case where the point through which one file runs presently is taken as a base point, and from the point another file is called by a single name. In addition to this, the call by a "tree name" is also possible. In this case, a call is made by a qualified name from the root, but for user's file, the head of the file directory for each user (excluding account directories) is considered as the root.

This function is used, in general, in specifying parameters of each command. For instance, in Fig. 2 (c), if F belongs to the first level directory and if program A is to be run,

RUN (F.A)

is inputted, and control transfers to A. RUN is a file belonging to a command branch, and F and A are files belonging to a user branch. In this case, F.A is called symbolically from the RUN command under specification of seeking a file belonging to a user branch and of the call by a tree name.

The specification of the the call by a tree name, however, can not be written syntactically in the general compiler language (in the case of the 5020 TSS, PL/IW). It must be registered as a separate program file and be written in the assembler language.

4. *On-Line Extension of the System*

For the 5020 TSS, the PL/IW language, which is a sub-set of PL/I, and the assembler language have been developed. These languages can be used from the terminal. On-line extension of the system by using them can be performed as follows.

According to our basic design policy, the initial system was designed to fulfil the minimum basic function (called "nucleus"). The nucleus means the minimum configuration which permits on-line extension of the system in future. It consists of supervisors in the resident part, some basic commands and a language processor, PL/IW (the assembler was developed by use of PL/IW).

Fig. 3 shows an on-line system extension process. In the same figure, PERMIT, SHARE and SETCOMMON denote each one command.

The PERMIT command is used to permit a specific file of one user to be used by other specific users (which can be specified by nodes of intermediate stages of account directories).

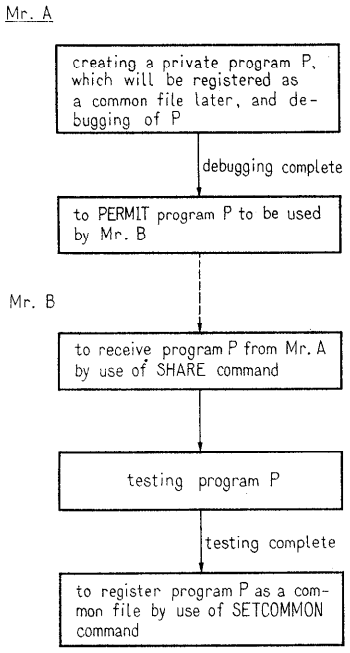


Fig. 3. On-line extension process of a system.

The SHARE command is used to bring the permitted file into the file system of the user adding a new chain on the directory only, the entity of file remaining unmoved.

The SETCOMMON command is used to register a certain file as command, library, supervisor or other common file. Only a specific person who is in charge of the control of the system is able to utilize this command.

In Fig. 3, the most important phase is the debugging of program P by Mr. A. The 5020 TSS uses the ring protection (by lock in 16 levels 0 through 15) with the segmentation mechanism incorporated in the system. General users are able to use several lock values of weaker power (for instance, the values of lock which are permitted to general users are in the range of 10 to 15, the power being weakened with increasing lock value). For any library or command having the lock value within this range, debugging of user level is possible and no problem occurs. A problem occurs first when a user desires to debug a command or the like with strong power (small in the lock value) as his private file. The lock value at the time of registration as common file is out range (greater power) of the lock value allowed for the private file of the user. Isn't the debugging at the level of the private file of the user completely possible? The best solution will be simulating some of supervisor modules which become necessary during debugging in conjunction therewith. This method, however, is not always very easy. Besides, when Mr. B receives program P from Mr. A

who says debugging complete, what method will be justified for confirming the Mr. A's saying on-line? This also could not but resort to simulation of modules which become necessary. In this respect, on-line extension of those commands, which are required to use supervisor call directly, which are required to write on tables of small lock values, which are required to rewrite file memories, etc. will rather by troublesome. Conversely, these modules with strong power should be included in the nucleus if they have high generality.

5. *Conclusion*

The 5020 TSS is now operating with 20 terminals. We have mentioned mainly about the file structure in this system. The structure is closely related to the segmentation mechanism, which is adopted in this system. As for the on-line system extension mentioned in 4, if it could be complete, productivity of software system will be very much improved. In practice, some modules can not be completed on-line without solving difficulties of simulation, etc., but with this taken into consideration, more time saving will be accomplished than the entire systen is constructed off-line. Furthermore, not only the system extension, but also implementation of a system itself from several terminals can be considered naturally, and this should be far easier with respect to the protection.

Acknowledgements

This paper owes much to discussions with professors of the University of Tokyo, in particular, Dr. H. Takahashi, Dr. E. Goto, Dr. E. Wada, Dr. M. Hosaka, and Dr. S. Osuga.

References

[1] Corbató, F. J. and V. A. Vyssotsky, Introduction and Overview of the Multics System, *Proc. of the FJCC, 27*, Sparton Books, Washington, D. C. (1965), 185–196.

[2] Daley, R. C. and P. G. Neumann, A General Purpose File System for Secondary Storage, *Proc. of the FJCC, 27*, Sparton Books, Washington, D. C. (1965), 213–229.