ソフトウェアシンポジウム 93－8
（1993．9．9）

多空間ディレクトリシステム (MSDS)

閔京華、田中英彦
東京大学工学部電気工学科
〒113 東京都文京区本郷7丁目3番1号

従来のディレクトリシステムは、UNIXからOSF/DCEまで、ほとんど単一の木構造を用いて、ファイルで表現されたいろいろな資源の位置に関する情報を保存する。けれども、現在のコンピュータシステムの中に含まれた種々の資源は爆発的に増加している。特に大規模分散処理システムの場合はいっそう厳しくなる。これにしたがって、単一の木構造のディレクトリはますます大きくなり、複雑にもなる。このためにユーザインタフェースとしての有用性が失われる恐れが出てくる。ここでは、これに対してわれわれは“多空間ディレクトリ”(MSD) という新しい概念を提案する。MSDは複数の空間からなり、それぞれの空間はユーザのある仕事だけに対して必要もしくは有用なファイルまたはサブディレクトリの集合からなる。多空間ディレクトリシステム (MSDS) は、このアイディアに基づいて、ユーザが自由にある仕事に対して最適な小環境を構築することができる機構を提供する。この機構を用いた環境においては、ユーザが実質の仕事に専心できるので、効率を高めることができる。ユーザの仕事に関係のあり、レベル高い資源しか提供しないので、安全性を高めることもできる。

Multi-Space Directory System (MSDS)

Jing-Hua Min, Hidehiko Tanaka
Department of Electrical Engineering
Faculty of Engineering
University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan
Email: {min, tanaka}@mtl.t.u-tokyo.ac.jp

Existing directory systems such as those in UNIX and OSF/DCE take the structure of a single tree to keep track of all kinds of resources in form of files. Along with the information explosion in computer systems, especially in the case of largely scaled distributed computing systems, such single tree-structured directory becomes so large and complicated that as user interface it is no longer as friendly as before. In this paper, we introduce a new concept “multi-space directory” (MSD) which consists of multiple spaces, each space is a set of the files and subdirectories that are necessary or useful for a user's one kind of work. Multi-space directory system (MSDS), based on this idea, provides the mechanisms for a user to define his own special environment that is most suitable to one kind of his work which may be independent or cooperative, and also centralized or distributed. So he can concentrate his efforts to do what he really want and achieve high efficiency. Just providing users related high-level resources according to their work can also raises the reliability of system.

1 Introduction

In computer systems, the interfaces of many resources to users are expressed in the form of files, for example, text files, graphic files, command files, utility files, program files, include files, library files, application files, help files, device files and etc. Directory systems are used to keep track of all these kinds of files. They are often organized in hierarchical ways by using tree structures. In any existing operating system whether it is traditional OS, or network OS, or distributed OS, its directory system provides users a single tree on which all files are hung.

For a centralized system or local area distributed computing system, this single tree interface to users is thought to be acceptable and popularly used. But for a wide area distributed computing system, the problems such as the difficulties of use, cooperation, protection and management which are caused by information explosion become more and more serious. These problems form a gap between users and computer systems. To fill this gap, we introduce a new concept "multi-space directory" (MSD), and try to build multi-space directory system (MSDS) on the top of the single tree-structured directory system.

MSD consists of multiple spaces, where a space is a set of the files or subdirectories which are necessary or useful for user's one kind of work.

MSDS allow a user to built his own special environment that is most suitable to one kind of his work which may be independent or cooperative, and also centralized or distributed. In this way, the user can simplify the system operations, concentrate his efforts to the essential work and achieve a high efficiency. Hiding unrelated or low-level resources from users can raises the reliability of system.

In section 2 we discuss the strength and weakness of single tree-structured directory. In section 3 we introduce several concepts in multi-space directory such as permanent space vs. transient space, superspace vs. subspace, and real space vs. virtual space. In section 4 we present a prototype of MSDS. In section 5 we summarize the merits that can be obtained from MSD. Finally in section 6, we give a concluding remark.

2 Strength and Weakness of Single Tree-Structured Directory

Single tree-structured directory has been proved to be effective in centralized file systems and also adaptable to distributed file systems. From UNIX to OSF/DCE, this kind of directory systems are well designed and implemented. From the view of constructing an integrated computing environment, single tree-structured directory plays an important role in organizing large, heterogeneous collections of data. It was directly used as user interface and was thought to be friendly in the past.

But now with the advent of networking, the scale of computer systems grows rapidly, information explosion has appeared in computer systems. Although a large single tree can do well in keeping track of all information, as user interface it becomes difficult for the following reasons:

(1) Tracking a tree mostly depends on human memory. The larger and more complicated the tree is, the more difficult to remember and easier to forget. This brings a low ratio of using information.

(2) Pouring all kinds of information everywhere to users all at once makes users lost, which discourages users to make further tries.

(3) For a user or a group of users who do distributed computings, it is inconvenient to find information of his own or his fellows somewhere else in this too big single tree. This results in the difficulty of cooperation.

(4) The fact that at any time a user can access all the accessible files whether related to his current work or not may lead to a high probability of destroying some files by misop-

erations.

(5) In the background of a very large single tree-structured directory, resource managements are very difficult.

For example, given the OSF/DCE environment, three writers (wtr1, wtr2, wtr3) in different cells cooperate to write a book. Each has his own this project directory (pj1) under the cell directory he belongs. They all face the same global directory which follows the X.500 standard, as shown in figure 1. The writers may be not as skillful at using computers as programmers. They just want to edit their book cooperately. They only pay attention to the editing, formatting, printing and management of their text files. Giving all possible resources to the writers is not only useless but also boring and dangerous. Even the programmers who are skillful at using computers are often disturbed by the resources unrelated or not directly related to their current work.

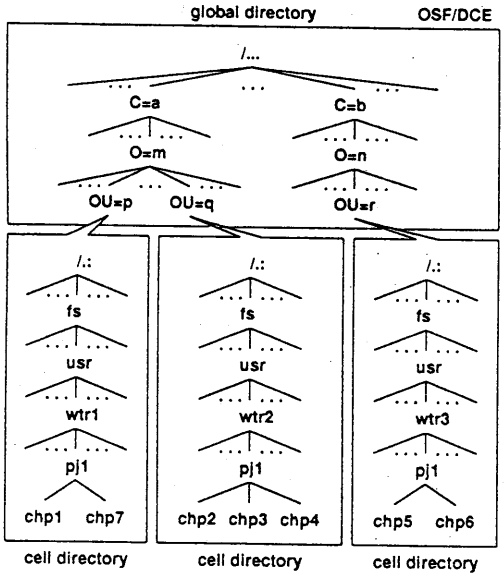

Figure 1: OSF/DCE directory system as an example

To overcome these difficulties, a new approach, rather than the single tree-structured directory, is needed.

3 Muti-Space Directory

3.1 Concept

Multi-space directory (MSD) consists of multiple spaces, where a space is a set of the files or subdirectories which are necessary or useful for user's one kind of work. Multi-space directory system (MSDS) is built on the top of a single-space directory system (SSDS), i.e. the single tree-structured directory system, to implement the idea of MSD (see figure 2). MSD user interface enhances the old one and is more friendly. We don't intend to abandon the old directory user interface. What we want to do is to filter it by MSDS according to the user's requests and provide the optimum working environment.

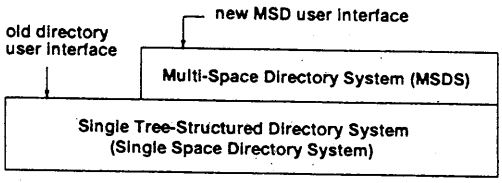

Figure 2: Position of MSDS

The idea of MSD originates from the consideration that a good directory interface to a user is one that just provides the necessary or useful resources according to the sort of the user's work. Let's look at the example shown in figure 1 again. The three writers are in a general environment so that they can access the global resources if having the rights. However, they only need a very small set of resources for their project. Seeing other unrelated or low-level resources may disturb their attention and searching required ones in the global set of resources may cut down their working efficiency. So it is desired to have a relative small environment special for their this project. MSDS makes this possible. Using MSDS, a space can be created into which the three pj1s (the directories containing text files) and the local cmd (the directory con-

taining common command files) and the local edit (the directory containing editor and formator files) are mapped (see figure 3).

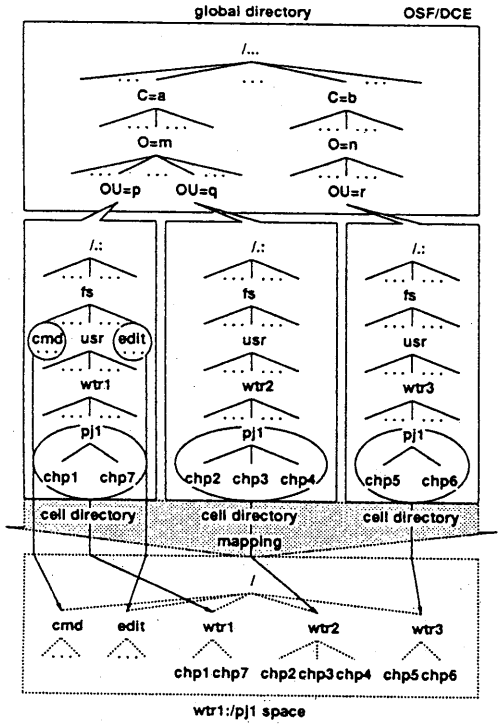

Figure 3: Mapping in MSD

The pj1 space for the wtr1 (wtr1:/pj1) is created by using the following command:

```
wtr1> create_space pj1 \
-t /cmd=/.:/fs/cmd /.:/edit/fs/edit \
   /wtr1=/.../C=a/O=m/OU=p/fs/usr/wtr1/pj1 \
   /wtr2=/.../C=a/O=m/OU=q/fs/usr/wtr2/pj1 \
   /wtr3=/.../C=b/O=n/OU=r/fs/usr/wtr3/pj1

or

wtr1> create_space pj1 -f pj1.map
```

where, create_space is command name; pj1 is space name; -t and -f are option names meaning "table" and "file"; The part after -t are the mapping table in which an item takes the form of X=Y where X is the destination name and Y is the original name of the file or subdirectory mapped into the space; The part after -f are the file pj1.map that contains the mapping table.

When the wtr1 entered the wtr1:/pj1 space, he just faces the mapped resources. The mapped commands and utilities function as the same as before. But their effective scopes are limited in the wtr1:/pj1 space. For example, in the wtr1:/pj1 space, the command "ls /" lists the contents under its root, and the command "ls /wtr?" lists the contents of the three directories (/wtr1, /wtr2, and /wtr3), as shown in figure 4. The position of the root "/" changes from space to space. The other two writers (wtr2 and wtr3) can also create their own pj1 spaces (wtr2:/pj1 and wtr3:/pj1) that are the same as the wtr:/pj1 space. In this way, they can easily share each other and work cooperately.

```
wtr1:/pj1> ls /
cmd/  edit/  wtr1/  wtr2/  wtr3/

wtr1:/pj1> ls /wtr?
wtr1:
chp1  chp7

wtr2:
chp2  chp3  chp4

wtr3:
chp5  chp6

wtr1:/pj1>
```

Figure 4: User interface of a space in MSD

SSDS gives users resources at most. MSDS gives users resources at optimum. This is the most difference between them.

3.2 Permanent space and transient space

The spaces created by create_space command remain existing until deleted by delete_space command. They are called permanent spaces. In contrast, some existing commands or utilities, say editor, also can create spaces. In editor spaces, being edited files and editor commands are only included. When exiting from

an editor, the space disappears. This kind of spaces are called transient space.

There are some differences between permanent spaces and transient spaces as listed below:

(1) Permanent spaces keep their resources until they are deleted explicitly by delete_space command, while transient spaces do until their creating commands or utilities finish.

(2) At the beginning of login, all existing permanent spaces can be resumed and there is no transient space.

(3) At the time of logout, all permanent spaces can be preserved and all transient spaces must be removed.

3.3 Superspace and subspace

The multiple spaces in MSD are structured in hierarchy as shown in figure 5. A space can create other spaces. The creating space is called superspace and the created spaces are called subspaces. The resources in the subspaces are subset of those in the superspace. A subspace can also be a superspace from which subsubspaces are created. The highest superspace for a user is called the root space of the user. In figure 5, the two users have different root spaces, i.e. usr1:/ and usr2:/.

The definition of space names follows the hierarchy of MSD for each user. It is very similar to the definition of file names. A full space name is a pathname to the space. But the uniquely naming scopes are under each user. So the same space names under different users refer to different spaces belonging to each user respectively. For the example in figure 5, the root space names are / and the full names of the first-level subspaces are /spc1 for both the usr1 and usr2. The full names of the second-level subspaces are /spc1/spc1.1 and /spc1/spc1.2 for the usr1. The usr2 has not the second-level subspaces. The usr1 can just use spc1.1 to point the space /spc1/spc1.1 when his current space is /spc1. This hierarchy of spaces can be browsed by using the command browse_space which will be introduced in section 4.

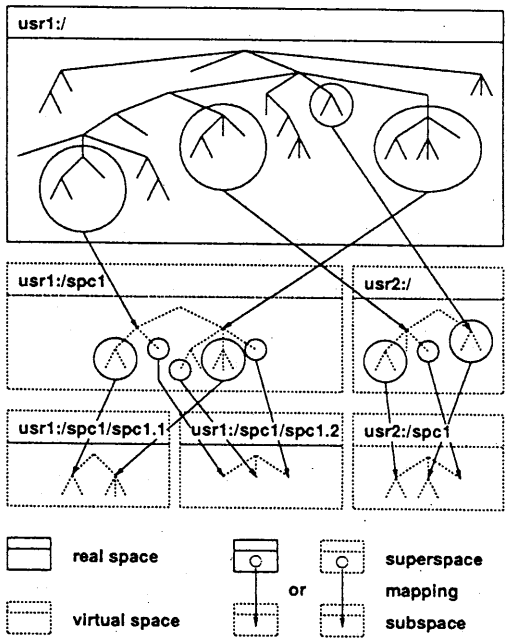

Figure 5: Hierarchy of spaces in MSD

3.4 Real space and virtual space

In MSD, there is a special space that holds the whole resources. We call it as real space. As shown in figure 5, the real space stores the entities of all resources that can be seen in the other spaces. Any spaces created directly or indirectly form the real space are virtual spaces which just keep the mappings of resources. And the entities of these mappings are also stored in the real space.

The real space is the base of MSD. Any existing single tree-structured directory such as the one in UNIX or OSF/DCE can be used as the real space. MSDS is designed and implemented on top of it. The mappings from the real space to virtual spaces are maintained and realized by MSDS.

Because the virtual spaces are defined to suit one kind of user's work at optimum, the user interface with virtual spaces is more friendly and efficient than that with the only real space.

4 Prototype of MSDS

We are trying to build a prototype of MSDS which provide a set of commands to manage and use MSD. Most commands are used for permanent spaces.

The main commands are listed and explained briefly in the following:

(1) create_space: create a permanent space by giving the space name, and the tables of the destination names and original names of the files or subdirectories to be mapped into it. The tables can be directly given in the command line, and can also be indirectly given through files that store the tables. A user can create spaces for himself under his root space, and can not create spaces for others, except superusers.

(2) delete_space: delete a permanent space by giving the space name or not. If no space name is given, the current space is deleted when it was create by the user himself. A user can only delete the spaces create by himself. When a permanent space is deleted, not only the mappings of the mapped resources are deleted but also the local resources created in the space are removed.

(3) switch_space: switch from the current space to another by giving the destination space name. A user can only switch among his spaces, and can not switch to other users' spaces, except superusers.

(4) browse_space: browse spaces under a superspace of a user by giving the user name, and the superspace name. If no user name is given, the current user browse his own spaces.

(5) add_resource: add files or subdirectories into a permanent space by giving the space name, and the tables of the destination names and original names of the files or subdirectories to be added.

(6) cut_resource: cut files or subdirectories from a permanent space by giving the space name, and the names of the files or subdirectories to be cut.

The rich existing command and utility files in the real space can be mapped into any virtual spaces with the same functions except the effective scopes that are limited in the spaces where they are called. Therefore, they are still usable and can be used more efficiently in the environment of MSD.

MSDS provides users a familiar but more convenient and more powerful environment, not a complete new and unfamiliar one.

The prototype of MSDS is not completed. Some other mechanisms are needed shuch as to implement different semantics of file sharing, to maintain the consistency of files among spaces, and to support different kinds of synchronization and cooperation among users.

5 Merits from MSD

We introduce MSD to try to achieve merits in the following four aspects: use, cooperation, protection and management.

5.1 Easy use

MSD allows users to have their own virtual spaces contributed to some kinds of their work. Just related high-level resources are included so that in these virtual spaces searching can become easier and operating can be simplified.

Because virtual spaces are mapped from an existing real directory space, the interface of MSD can be very familiar to users and the rich powerful existing software tools can still be applied as before in different virtual spaces but with smaller scopes.

5.2 Easy cooperation

Sharing resources and exchanging information are the preconditions of cooperation. In MSD, shared resources in the real space can be mapped to the virtual spaces of the sharing users. In this way, a user can easily find the shared resources of his cooperators in his own virtual space without remembering the real location of the shared resources in the real space, and vice versa, his shared resources can also be

easily seen and accessed by his cooperators in the same way. Information required for cooperation can be exchanged through the shared resources.

5.3 Easy protection

In a virtual space, a user just faces the related high-level resources for his working in this virtual space. He cannot see and also cannot access the unrelated or low-level resources although he may have the rights to access them in his root space. This greatly lowers the possibility of destroying resources by misoperations, especially for the resources outside the virtual space.

MSD provides the way that system managers can assign a user's root space to a virtual space, not allow him to enter the real space. When the user enters system by login, he just possesses the resources in his virtual root spaces. In this sense, MSD can be seen as a more flexible, explicit and high-level mechanism for resource assignment and protection.

5.4 Easy management

System managements have become more and more complicated and difficult along with the growth of system scales. Distributed managements are applied in many distributed computing systems. MSD gives supports to distributed managements. By mapping the resources to be managed in the real space to different management virtual spaces according to the management scales and the relationship between managers, we can build a set of well-structured management virtual spaces. Each manager has his own management virtual spaces and do management work in them. He also cooperates with other related managers through them as stated in section 5.2.

6 Concluding Remarks

In this paper, we have introduced a new concept "Multi-Space Directory" (MSD) which enhances the traditional single tree-structured directory to provide users more friendly and efficient resource-accessing environment, especially in the case of largely scaled distributed computing systems which is full of all kinds of global resources.

References

[1] S. Mullender, ed., *Distributed Systems*, ACM Press, 1989

[2] A. S. Tanenbaum, *Modern Operating Systems*, Prentice-Hall, 1992

[3] A. L. Ananda and B. Srinivasan, ed., *Distributed Computing Systems: Concepts & Structures*, IEEE Computer Society Press, 1991

[4] OSF, *Introduction to OSF DCE*, Prentice-Hall, 1992

[5] OSF, *OSF DCE User's Guide and Reference*, Prentice-Hall, 1993

[6] OSF, *OSF DCE Application Development Guide*, Prentice-Hall, 1993

[7] CCITT, *The Directory-Overview of Concepts, Models and Services*, Recommendation X.500 edition, 1988