METHOD AND APPARATUS FOR INSTALLING AND EXECUTING A SINGLE USER TASK IN A MULTI-USER ENVIRONMENT

Field of the Invention:

The invention relates to a method and apparatus for installing and executing programs by a computer and more specifically to a method and apparatus for installing and executing programs by a multi-user computer.

Background of the Invention

Computer systems can be broadly divided into two categories; single user systems and multi-user systems. Single user systems are those in which one user will have control of the processor for the duration of use. Multi-user systems are those in which the computer will permit several users to time-share the processor substantially simultaneously. That is, each user periodically is given a time-slice of the limited amount of computation time available, in a round-robin fashion.

In multi-user systems the computer is capable of executing an application program for one user and, when that user's time-slice of the available computation time is completed, saving the state of the processor and the state of the program so that the application program can be stopped and another user's application program begun. When the first user's time-slice occurs again, the processor will reload the application program, the state of the machine and the state of the program and resume executing the program where the computer left off during the previous time-slice.

When an application program begins to execute, it typically accesses initialization data, typically but not necessarily within an initialization file, which typically contains user specific data regarding how the program is to be executed. For example, a word processing program may have initialization data which specifies what icons the user has defined, what screen colors are to be used and how the screen is to appear geometrically. When an application is developed for use in a multi-user environment, the program is constructed to maintain separate user contexts so that the program uses specific initialization data for each user. Applications developed for single user computers do not typically have the ability to maintain separate user contexts. This is due to the

fact that an application developed for a single user computer expects to have only one user context.

With the advent of computer networks, the issue arises that what may have been a single user computer running a single user application is now being used as a network execution server, providing single user application execution for multiple users. A single user application executed from a network execution server must now keep track of the application initialization data which are created for each user.

The present invention addresses this problem thereby permitting single user applications to be executed in multi-user environments.

Summary of the Invention

The invention relates to a method of installing and executing a single user application on a multi-user system. The method includes the steps of installing a single user application using the installation program of the application, placing the application initialization data in a predetermined location in data storage, obtaining initialization data corresponding to the single user application from the predetermined location in data storage, copying the initialization data into a user specific initialization data location, and executing the single user application using the user specific initialization data. In one embodiment the step of copying the initialization data copies the initialization data into a specified user directory. In another embodiment the step of copying the initialization data includes the step of renaming the initialization data with a unique user name.

The invention also relates to an apparatus for installing and executing a single user application on a multi-user system. The apparatus includes a process installing a single user application, a process placing the application initialization data in a predetermined location in data storage, a process obtaining initialization data corresponding to the single user application from the predetermined location in data storage, a process copying the initialization data into a user specific initialization data, and a process executing the single user application using the user specific initialization data. In one embodiment the process copying the initialization data copies the initialization data into a specified directory. In another embodiment the process copying the initialization data renames the initialization data with a unique user name.

Brief Description of the Drawings

This invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

Fig. 1 is block diagram of a single user computer system on which the invention can be practiced;

Fig. 2 is a flow diagram of an embodiment of the invention;

Fig. 3 is a flow diagram of another embodiment of the invention; and

Fig. 4 is a flow diagram of an embodiment of the invention for adding or updating an entry in the initialization data.

Detailed Description of the Invention

In brief overview and referring to Fig. 1, a single user computer system on which the invention may be practiced includes a processor 10 executing an application program 18 using data input from an input/output device 26, such as a keyboard and display screen upon execution. The application program 18 is designed to access its initialization data 34, which is typically stored on a disk 42.

Normally when the application is installed, the system is first placed in installation mode, and the installation program of the application uses operating system calls to update or create initialization data in a shared or common system directory (for example \WINDOWSAPPS\). Once the installation is completed, the system is placed in execute mode, and when the application program executes, the application program can then use system calls to query the settings in the initialization data.

In order for a single user application to be executed on a multi-user computer system, it is necessary that different initialization data be maintained for each user on the multi-user system. To accomplish this, in one embodiment of the present invention as shown in Fig. 2, a subdirectory is created by the system manager for each user of the system (for example \USERS\USER1\) when the user's account is created and an initialization subdirectory is created in each user subdirectory (for example \USERS\USER1\WINDOWSAPPS\). It should be noted that it is not necessary that the name of the location of the application subdirectory (\USERS\USER1\WINDOWSAPPS\) under the user subdirectory have the same name as the system common directory (\WINDOWSAPPS\).

When the application begins to execute it attempts to locate an entry in the application initialization data (for example, APPLI.INI) by calling the operating system (Step 100). To find this file, the application makes a call to the operating system which first determines whether the system is in execute or install mode (Step 108). If the system is in execute mode, the call to the operating system attempts to provide a path to the user specific initialization data location (for example \USERS\USER1\WINDOWSAPPS\ by determining whether a user specific version exists (Step 116). If a user version does not exist, the operating system copies the version of the application initialization data (APPLI.INI) from the system directory \WINDOWSAPPS\ to the user directory \USERS\USER1\WINDOWSAPPS\ (Step 124).

If the operating system then finds (Step 132) the entry in the user specific version of the initialization data, it returns the entry to the application (Step 140). If the system fails to find the entry in the user specific version, the operating system determines (Step 148) if the version of the application initialization data in the system directory is newer than the user specific version. The operating system can determine this by examining a time-of-year-stamp which was generated when the file was updated or version number associated with the file. If the system version is not newer than the user specific version, then the entry is not present in any version and the operating system returns an error to the application (Step 156). If the system version is newer, the operating system merges the system entries into the user specific version so as not to lose the changes made by the user (Step 164). In one embodiment, the merge is performed by the operating system only if the operating system determines that a merge is permitted for this file. Whether a merge is permitted is designated on the file by a flag or other indicator attached to the system version of the file, the user specific version of the file, or stored in a separate data structure. In this manner merging may be permitted on a per file basis. The operating system then determines if the entry is in the merged file (Step 132) and proceeds as discussed previously.

If the system is not in execute mode, the operating system only determines if the entry is in the system version of the file (Step 172). If the entry is present in the system version, the system returns the entry to the application (Step 140), and if not the system returns an error to the application (Step 156).

In another embodiment, the application begins to execute as discussed previously and attempts to locate an entry in the application initialization data (for example, APPLI.INI) (Step 200). Again, to find this file, the application makes a call to the operating system which first determines whether the system is in execute or install mode (Step 208). If the system is in

execute mode, the call to the operating system returns the path to the unique user specific initialization data location (file name) located in the system directory (for example \WINDOWSAPPS\USER1APPLI.INI) and determines whether a user specific version exists in the directory (Step 216). In one embodiment, the user specific version of the initialization data is named by adding some unique identifier to the name of the initialization data. For example, the unique user specific name USER1APPLI.INI or APPLINIUSER1.INI is formed by attaching the user account name (USER 1) to application initialization data (APPLI.INI). Such a user specific name may be formed from the user's account, user's id, user's serial number or other unique user identifier. In yet another embodiment the system uses a lookup table to match a user to a specific file. If a user specific version does not exist, the operating system copies the system version of the application initialization data (APPLI.INI) within the system directory (for example \WINDOWSAPPS\) to a user specific file name by appending the user account name to the application initialization data name as described above (Step 224).

If the operating system then finds (Step 232) the entry in the user specific version, it returns the entry to the application (Step 240). If the entry is not in the user specific version, the operating system determines (Step 248) if the system version of the application initialization data is newer than the user specific version and if it is not newer than the user specific version, then the operating system returns an error to the application (Step 256) as discussed previously. Again, if the system version is newer, the operating system merges (Step 264) the system version entries into the user specific version as described previously. The operating system then determines if the entry is in the merged file (Step 232) and proceeds as discussed previously. Again if the system is not in execute mode, the operating system only determines if the entry is in the system version of the file (Step 272) and again proceeds as discussed previously.

Referring to Fig. 4, if an application attempts to update or add an entry to the application initialization data by calling the operating system, the operating system first determines if the system is in the execute or install mode (Step 308). If the system is not in the execute mode, the system version is updated (Step 314). However, if the system is in execute mode, the system first is checked to determine whether a specific user version exists (Step 120). If it does exist, the user specific file initialization data is updated (Step 138). Otherwise, the system version is copied to the user version (Step 146) prior to the entry being added to the user specific version (Step 138). The steps required to determine the location of the user specific files are as discussed above.

For an operating system to be used with the invention, it is necessary that the operating system either have the capability of returning the necessary initialization data location information to application program or that it be modified to return the necessary initialization data location information to the application program when presented with a request. Thus the operating system must be able to return the user specific location of the initialization data if the system is in execution mode or the shared location of the initialization data if the system is in installation mode. For example, in the WINDOWS operating system (Microsoft Corp., Redmond, Washington) the operating system request GetWindowsDir must be modified to return the correct loaction depending upon whether the system is in installation or execution mode. It is also necessary that the system calls which are used to query or set the initialization entries be modified. Examples of such calls in the WINDOWS operating system are GetPrivateProfileString, GetProfileString, GetProfileSection, and WritePrivateProfileString.

The operating system routines which are used to get or set initialization information need to be able to determine whether the system is in installation or execution mode and respond accordingly. If the system is in installation mode, these calls behave normally, retreiving information from and writing information to the shared location. However, if the system is in execution mode, these routines must determine if a user specific version of the initialization data exists, and if not create a copy in the user specific location from the shared location. An enhancement which supports the merging of new entries from the system version of the initialization data into the user specific version of the initialization data must also be present.

It should be noted that although the above embodiment is described in terms of initialization data being maintained in a directory, the method and apparatus work equally effectively for other directory like structures such as a registry.

Having described preferred embodiments of the invention, it will now become apparent to one of skill in the art that other embodiments incorporating the concepts may be used. It is felt, therefore, that these embodiments should not be limited to disclosed embodiments but rather should be limited only the spirit and scope of the following claims.

CLAIMS



What is claimed is:

1. A method of installing and executing a single user application on a multi-user system comprising the steps of:

obtaining initialization data corresponding to said single user application;

copying said initialization data into a user specific initialization data; and

executing said single user application using said user specific initialization data.

2. The method of claim 1 wherein said copying said initialization data includes the step of copying said data into a user specific location.

3. The method of claim 2 wherein the user specific location is a directory.

4. The method of claim 3 wherein said user specific directory is a user directory.

5. The method of claim 1 wherein said copying of said initialization data includes the step of renaming said user specific initialization data with a unique user name.

6. The method of claim 5 wherein said unique user name comprises a user identifier appended to the name of said application initialization data.

7. A method of installing and executing a single user application on a multi-user system comprising the steps of:

obtaining initialization data corresponding to said single user application, said initialization data comprising a new entry;

determining if user specific initialization data exists; and

if said user specific initialization data exists, merging said new entry with said user specific initialization data.

8. A system for installing and executing a single user application on a multi-user system comprising:

This Sheet Has Been Replaced



THIS SHEET HAS BEEN REPLACED

a process obtaining initialization data corresponding to said single user application;

a process copying said initialization data into a user specific initialization data; and

a process executing said single user application using said user specific initialization data.

9. The system of claim 8 wherein said process copying said initialization data copies said file into a user specific location.

10. The system of claim 9 wherein said user specific location is a directory.

11. The system of claim 10 wherein said specified directory is a user directory.

12. The system of claim 8 wherein said process copying said initialization data renames said user specific initialization data with a unique user name.

13. The system of claim 12 wherein said unique user name comprises a user identifier appended to the name of said application initialization data.

This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED