The SUP Software Upgrade Protocol

Carnegie Mellon University
School of Computer Science

Steven Shafer
Mary Thompson

7 September 1989

The SUP system is a set of programs for maintaining a set of files in identical versions across a network of cooperating machines. It runs under the Mach operating system.

1. Introduction

The SUP System is a set of programs that provide for collections of files to be maintained in identical versions across a number of machines. These programs are:

SUP	The "client" program, run by users or system maintainers, which initiates the upgrade activity on a machine requesting the latest version of a collection of files. SUP will normally be run as a daemon, firing up once each night (week, etc.) to upgrade the specified file collections.
SUPFILESRV	The "file server" program, a daemon that is run by the system maintainer to service requests for files initiated by client SUP programs. The file server runs on every machine used as a "repository" of distributable versions of files. It runs continuously and listens for network connection requests by individual client processes; for each individual client request, a process is forked to service that request.
SUPSCAN	The "file scanner" program, that may optionally be run periodically to speed up execution of the file server. It pre-compiles a list of files on the file system that match the specifications for a given file collection so that the file server need not do this during each upgrade of that collection. The file scanner is normally used daily for very large file collections that are upgraded by many clients each day; it is not so useful for small file collections or for those that are upgraded by only a few client machines per day.

When SUP is run, the user designates the names of certain "file collections", sets of files, that should be upgraded. All pre-defined system collections have entries in the `coll.host` file that gives the name of the host will act as the repository for that software collection. The file server on that machine provides a list of the files included in the collection and which ones have been changed since the last time this collection was upgraded from that client machine; the client decides which of these files it needs to receive; and the files are sent from the file server. In this way, files can be installed on the repository machine once, and they will propagate to all machines upgrading from that repository as soon as the respective users execute SUP to perform the upgrade. This allows groups of cooperating users to establish sets of files that can be relied upon to be the same across a set of host machines.

SUP is intended to provide a broadcast facility for maintaining identical versions of files across multiple machines. An alternative approach is to provide a common file system; even when such a system is available, SUP is useful for maintaining frequently used files on local disk storage, and for maintaining files across the file server machines themselves.

The SUP system does not address the issues of version control, nor of maintaining compatibility of pieces of a system that are being independently changed by different users or on different machines; what SUP provides is a broadcast mechanism for issuing stable, reliable versions of software or data files, produced by a small number of maintainers and needed by a larger number of users. SUP is intended specifically to address these situations:

- *A large collection of system software prepared by a maintenance staff for use by a large user community.* For example, the CMU-CSD UNIX software used by hundreds of workstations. In such situations, the users know absolutely nothing about how to obtain such software, but they need to keep it constantly upgraded on their machines. Similarly, the maintainers cannot possibly maintain each user’s machine individually: they must rely on a broadcast mechanism such as SUP.

- *A moderate collection of software and data files shared by the members of a project.* For example, the vision libraries used by computer vision researchers at CMU. Again, a small number of maintainers on a specific machine produce and maintain this software, which is used by many users on many machines in the CMU environment. However, SUP will not support the practice of some groups that have large programs with different pieces being simultaneously modified and tested in incompatible ways by various users.

- *A user with accounts on more than one machine.* Such a user can set up SUP to automatically copy a collection of files from one machine to one or more others whenever they are changed on the repository machine.

If the system maintenance staff is involved on the repository machine a "system" file collection can be set up for which client SUP programs need know nothing but the name of the collection. Otherwise, the file collection will be "private", and client SUP programs must be specifically told the name of the repository host and the base directory name for the collection on that host.

SUP provides facilities for upgrading read-protected files (i.e. files that are not readable by the general public), data encryption of transmitted files and filenames, allowing access only to selected host machines not on the local network, automatically creating backup copies of critical files, executing command-files or programs after upgrading files that require special handling, and mailing progress and error messages to designated users. SUP is also as clever as can be about ensuring that upgraded files have the same accounting information (mode, owner, links, etc.) on the client machine as on the repository machine, and ensuring that upgrade problems don’t wipe out existing files when this is avoidable.

2. How to Use SUP to Upgrade a File Collection

You will normally want to run SUP periodically, say every day or every week, using the *at* program, to upgrade important file collections such as UNIX system software. You can also execute SUP directly at any time. The SUP command looks like this:

$ *sup [flags]* [supfile]

These are the flags and arguments:

-*a* (all files) Normally, SUP only upgrades files if they are out-of-date or missing on your machine. If you specify the -*a* flag, then all files in the collection will be upgraded regardless of whether they meet these conditions. This is useful for recovering from catastrophic file deletions or disk crashes.

-*b* (backup) If this flag is given, or the *backup* supfile option is specified, the contents of regular files on the local system will be saved before they are overwritten with new data. The file collection maintainer can designate specific files to be worthy of backing up whenever they are upgraded. However, such backup will only take place if you specify this flag or the *backup* option to allow backups for a file collection on your machine. The backup mechanism will create a copy of the current version of a file immediately before a new copy is received from the file server; the copy is given the same name as the original file but is put into a directory called *BACKUP* within the directory containing the original file. For example, */usr/sas/src/foo.c* would have a backup copy called */usr/sas/src/BACKUP/foo.c.* There is no provision for automatically maintaining multiple old versions of files; you would have to do this yourself.

-*B* overrides and disables the -*b* flag and the *backup* supfile option.

-*d* (delete) Files that are no longer in the collection on the repository will be deleted if present on the local machine. This may also be specified in a supfile with the *delete*option.

-*D* This flag overrides and disables the -*d* flag and the *delete* supfile option.

-*e* (execute) Sup will execute commands sent from the repository that should be run when a file is upgraded. If the -*e* flag is omitted, Sup will print a message that specifies the command to execute. This may also be specified in a supfile with the *execute* option.

-*E* overrides and disables the -*e* flag and the *execute* supfile option.

-*f* (file list) If you specify -*f*, then SUP will print a list of all the files in the specified collections, indicating which ones need to be upgraded, which are directories, and which require backup copies to be created. The upgrade itself will not be performed.

-*l* (local) Normally, SUP will not upgrade a collection if the repository is on the same machine. This allows users to run upgrades on all machines without having to make special checks for the repository machine. If the -*l* flag is specified, collections will be upgraded even if the repository is local.

-*m* (mail) Normally, SUP prints log messages on its standard output and diagnostic output. With the -*m* flag, messages will be mailed to you instead. This also allows the *notify* option to be effective (see the discussion of supfiles below) to send mail to someone else.

-*N* (network debugging)
The -*N* flag causes SUP to produce tons of output lines describing the progress and status of the network communications with the name server and file server.

-*P* (debugging ports)
The -*P* flag tells SUP to use an alternate set of TCP ports from the normal ports. This is a debugging aid for system maintenance.

-*o* (old files) This flag overrides the *noold* option if it appears in the supfile. The -*a* flag also overrides the *noold* option. See "Supfiles" below for details.

-O overrides and disables the *-o* flag and the *old* supfile option.

-s (system collections)
Normally, you will specify a specific *supfile* to describe the file collections you want. If you use the *-s* flag, then a specific supfile will be used that describes the system software file collections. The supfile for system software is */usr/cmu/lib/supfiles/coll.list*.

-t (time) If you specify *-t*, then SUP will simply print out the last time and date at which each specified file collection was successfully upgraded; it will not perform any actual upgrade. This is a diagnostic aid in case you suspect the upgrades aren't working.

-v (verbose) Normally, SUP only produces messages to tell what's gone wrong (if anything does go wrong). With the *-v* flag, messages will be produced to tell what is happening when things are going normally as well.

supfile You must specify the name of a supfile, unless you use the *-s* flag to indicate the system software supfile. The supfile, described below, specifies which file collections you want to upgrade.

For example, to upgrade system software on your machine, type:

$ *sup -s*

or, to have detailed log messages mailed to you:

$ *sup -v -m -s*

Because SUP runs with your user-id, you will not be able to upgrade the system software unless you are logged onto that account when you execute SUP.

Each file collection being upgraded must have a *base directory* on your machine, which will normally contain all the files in the collection. Within this directory there should be a subdirectory called *sup* that will be used by the SUP program; it will be created automatically if you do not create it. SUP will put files into this directory as needed.

If you want to find out what files need to be upgraded, type:

$ *sup -f* ***supfile***

This will list all the files in the collection, indicating which need to be upgraded (and why), which are directories, and which would have backup copies created if the upgrade were performed. If you simply want to find out the date and time of the last successful upgrades, type:

$ *sup -t* ***supfile***

This will print the collection name and time of the last upgrade for each collection described in the indicated supfile. For the system collections, you can use:

$ *sup -s -t*

2.1 Supfiles

When you execute SUP, you specify a *supfile* either by name or by using the *-s* flag. This file is a list of the file collections you wish to upgrade.

The supfile is a text file, with each file collection described on a single line. The line begins with the name of the collection, and may then have a number of options separated by spaces. The options are:

base=*directory* The name of the base directory on this machine for this file collection, when you don't want to use the default (*/usr/**collection***).

host=*hostname* The name of the host machine acting as the repository for this file collection, used for "private" file collections. For "system" file collections, this is unnecessary because the this information is kept in /usr/cs/lib/supfiles/coll.host.

hostbase=*directory* The name of the base directory on the repository machine for the file collection (see "How to Set Up a File Collection" below). This is also used only for "private" file collections; for "system" file collections, the information is obtained automatically by the file server. The default name of the host base directory is */usr/**collection***.

login=*accountid* The name of the account to be used by the file server, when the default is not acceptable (i.e. it doesn't have the necessary file access privileges to read the file collection). The default at CMU is the "Anonymous FTP" account if no data encryption is used, or the owner of the directory *sup/**collection*** within the repository machine's base directory if encryption is used.

password=*password*
The password used for the login account specified by the "login" option. This password will be transmitted in encrypted form over the network.

crypt=*key* An optional encryption key for filenames and file data for this file collection. If used, the same key must be specified on the repository machine or the upgrade will not take place. This key is used for filenames and data only -- not for transmission of the passsword in the "password" option. The use of data encryption also implies an alternate default account for the file server (see the "login" option above).

notify=*mailaddress* The account name to which mail is to be addressed for log messages for this file collection, when the *-m* flag is given to SUP. The account name can be a complete mail address, possibly including a network host name such as *sas@cmu-cs-ius*.

noexec This option prevents the automatic execution of command files on your machine with the upgrade is finished (see "What SUP Does" below).

backup This option enables backup copies of critical files to be created by SUP as specified in "What SUP Does" below.

nodelete This option prevents SUP from deleting files on your machine if they are deleted from the file collection on the repository machine.

noold This option tells SUP not to check on files in the collection that have not been modified or created since the last upgrade. This allows SUP to run much faster on large file collections, but prevents it from deleting files on the client machine if they are deleted from the repository, or from restoring files that have been accidentally deleted from the client machine. This option is normally useful only for very large file collections. Its function subsumes that of the *nodelete* option. The *noold* flag will be ignored when either the *-a* or *-o* flag is specified to SUP; this allows a complete file check to be performed with the *-o* flag when needed to recover from, say, accidental deletion of critical files.

Here is an example of a supfile:

```
cmu  backup
vision  crypt=pupil backup notify=vision@cmu-cs-ius
sas  host=ius hostbase=/usr/sas/sun login=sas password=foo crypt=bletc
```

This supfile specifies the following file collection upgrades:

cmu The "cmu" file collection, which is a "system" file collection, with local base directory */usr/cmu* (the default). Backup copies of critical files will be created during upgrading.

vision The "vision" file collection, also a "system" file collection, with local base directory */usr/vision*. Backups will be created. The data will be encrypted with key *pupil*, and log messages will be sent to the *vision* account on *cmu-cs-ius*.

sas A "private" file collection will be upgraded from the *cmu-cs-ius* machine. The remote base directory is */usr/sas/sun*, but the local base directory is */usr/sas*. The file server will login on account *sas* with password *foo*; the data will be encrypted with key *bletch*.

3. What SUP Does

An upgrade involves several steps:

1. SUP first reads the specified supfile, checking it for errors and recording all the specifications and options.

2. If any collections do not specify a *host* name, SUP will look in /usr/cs/supfiles/coll.host to find out the names of the hosts for these collections. In the preceding example, the name server would be asked to supply the host names for the *cmu* and *vision* file collections.

3. For each collection on the list, SUP will connect to the file server on the appropriate host and carry on the upgrade process. (If the file server is on the same host machine as the client, and the base directories are the same, then no upgrade is performed.)

    a. The file server and SUP exchange setup information, including the host base directory, login account name, password, and encryption key. SUP also reports the time of the last upgrade, by looking in the file *sup/**collection**/when* (e.g. */usr/vision/sup/vision/when*) on the client machine. The file server gets the base directory name, if needed, by looking in the file */usr/cs/lib/supservers/coll.dir*, and gets the encryption key from *sup/**collection**/crypt* on the repository machine. The connection may be refused by the file server for such reasons as:

        - incorrect login name or password
        - base directory is protected against access by the specified account
        - incorrect data encryption key
        - host not allowed access

    b. Once the connection is established and access permission has been granted as above, the file server builds a list of all files and directories in the file collection. The list file *sup/**collection**/list* (see "How to Set Up a File Collection" below) is used to generate this list. The files indicated in the list file are marked with a special flag if they are indicated to have backup copies created in case of upgrade. The last modification date of each file is checked against the time of the last upgrade (reported by the client above), and if it has been modified since the last upgrade, a flag is set to indicate that the client's copy of this file is out of date. The file list, along with the backup and out-of-date flag for each file, is sent to the client. If a scan file exists as created by SUPSCAN (*sup/**collection**/scan*), then the file list is taken from the scan file instead of being constructed from the list file by the file server. In this case, the time of record for the upgrade will be the time at which the scan file was created rather than the time at which the upgrade actually occurs. If the *noold* option was specified in the supfile (and not overridden by the *-a* or *-o* flag to SUP), the list of filenames sent to the client process will exclude those files that were not created or modified since the last upgrade.

    c. The client SUP process receives the list of files and constructs a list of the files it needs. This will normally be those files that are either (1) out-of-date as indicated by the flag sent from the file server, or (2) missing from the local machine. However, if the *-a* flag was specified to SUP, then every file in the list will be selected by the client process. The list of needed files is sent to the file server. For each selected file, if the backup flag was marked and the user specified the *backup* option in the supfile, then a backup copy of the file will be created. This will be a copy of the file with "backup/" inserted in the filename just after the directory name, i.e. */usr/vision/lib/libvision.a* would have a backup copy named */usr/vision/lib/backup/libvision.a*. (The indicated directory, e.g. */usr/vision/lib/backup*, will be created if needed.) Also at this time, if the *nodelete* option was not specified, the client reads its list of files in the collection during the last successful upgrade, *sup/**collection**/last*, and deletes any files appearing in that list that are not in the current file list.

    d. The file server receives the list of files requested by the client. It checks to see that each file is on its list of files for this collection, and that each file is accessible; if a file fails either condition its name will be reported to the client as being denied to that client. Each file will be sent to the client machine, along with the owning account name, the owning group name, the 12 low-order mode bits, and the last modification time for that file. If a file has more than one link (filename), both of which are requested by the client, then the file will be sent once and the second (etc.) filename will be accompanied only by a flag saying it's a link and the name of the file actually sent. The client receives each file and processes it as described under "File Installation" below. Directories are treated similarly to files; the mode, owning account and group, and modification time are transmitted to the client machine whenever a directory is upgraded.

e. When all files have been transmitted, the file server looks at the list of command-files to be executed after upgrading for this collection (see "How to Set Up a File Collection" below). If any of the command-files or their trigger files have been upgraded, then the client is sent the filename of the command-file. The client SUP process will normally execute these files; however, if the user has specified *noexec* in the supfile, then the files will not be executed. In this case, a log message will be created and printed or mailed, telling the user to please execute these files. When creating command files to be executed by SUP, you should bear in mind that the command file might be executed several times on the same version of the trigger files.

f. Finally, if the upgrade is successfully completed, the file server reports the time (on its own clock) at which the upgrade began; the client will record this time in the file *sup/**collection**/when* to be reported as above at the start of the next upgrade. If the *nodelete* option was not specified, then the list of all files in the collection is written into *sup/**collection**/last*.

SUP and the server processes begin each connection by determining whether byte-swapping is necessary for passing integers across the network. If so, the server process performs byte-swapping on input or output of integers to the network, while the client uses its natural representation for network I/O.

3.1 File Installation

When SUP receives an ordinary file (i.e. one that is not a link to a previously sent file), it follows this procedure:

1. If the file doesn't already exist on the local machine, it's a new file. It will be created and the data will be read directly into this file.

2. Otherwise, the file already exists. An attempt will be made to read the file contents from the network into a temporary file, which will later be renamed to replace the destination file. SUP will try to create a temporary file in the following directories, proceeding down the list until one of the attempts succeeds:

   a. ***destination-directory*** (the directory containing the file itself)
   b. *sup* (the *sup* subdirectory of the local base directory)
   c. *.* (the local base directory)
   d. */usr/tmp*
   e. */tmp*

   If all these attempts fail, SUP will try to create the destination file itself instead of using a temporary file. If that fails, SUP will complain with a log message and go on to the next file.

3. The file data itself is read into the temporary or destination file. Interrupts are disabled during this process.

4. If the file read was a temporary file, it is renamed to the destination file. This is done via

link/unlink, if possible, unless the destination file has more than one link already on this machine. If the link/unlink fails or if the destination file has multiple links, then the temporary file is actually copied to the destination and the temporary file is unlinked.

5. The owner, group, modification time, and low-order 12 mode bits of the destination file are set to the values received from the file server. The last-access time of the file is set to the current local clock.
6. If the *-v* flag was specified to SUP, a log message is printed indicating the successful receipt of the file.

When SUP receives a new link to a file previously received, it simply tries to unlink and link unless the two names are on different file systems on the local machine. If this is the case, or if the link fails, then the previously received file is actually copied to the new name (using exactly the same process as described above for creating a temporary file if needed, etc.) The file-system comparison checks the file system of the old file itself and the directory containing the new name.

In all cases, if the directory containing a received file or link does not exist on the local machine, SUP creates it with *mkdir* (recursively if needed) before creating that file or link. The mode, owner, group, and modification time of the newly created directory will then be set to be the same as the mode of the corresponding directory on the repository machine. This accounting information will be updated whenever the directory is modified on the repository machine.

As can be seen from this description, SUP will work the most smoothly (i.e. always using link/unlink for file name changing) if it has write permission in the local base directory and in all subdirectories of that directory.

4. How to Set Up a File Collection

It takes only a little bit of effort to set up a file collection on a repository machine to be upgraded by authorized clients.

First, the file collection must be given a *name* and a *base directory*. There can be several collections with different names sharing the same base directory; for example, there might be a file collection called *cmulib* for the CMU C library, and one called *cmumathlib* for just the math routines in the CMU C library, both using */usr/cmu* as the base directory.

If it is a "private" file collection, client users must be told the name of the repository host and the name of the host base directory for use in the *host* and *hostbase* options in the supfile, described above. If it is a "system" collection instead, the system maintainers must alert the name servers of the host name (in */usr/cmu/lib/supservers/coll.host*) and the appropriate file server of the base directory name (if not the default, by putting it into */usr/cmu/lib/supservers/coll.dir*).

Within the base directory, a subdirectory must be created called *sup*. Within this subdirectory will be a further subdirectory whose name is the name of the collection (there may be several of these if several collections share the same base directory). Each collection's subdirectory will contain any or all of four files:

collection*/list* The *list file*, describing the files in the collection. The format of this file is explained below.

collection*/host* Normally, the file server allows access to a collection by all machines. If you wish to allow access only to specific remote hosts, you can list their names, one per line, in this text file. If a remote host has several alias names, it need only be listed once in this file. The name *LOCAL* can be used to allow access to all machines on the local network. This access control is in addition to the other forms of authentication provided by SUP (data encryption and UNIX file protection modes).

collection*/crypt* If you wish to apply data encryption to the filenames and file contents in this collection during upgrading, create this file and place in it the encryption key. This should be on a single line of text containing nothing else except an optional newline character. The client must supply the same key via the *crypt* option in the supfile for this file collection; the file server checks that explicitly before authorizing the upgrade to take place.

collection*/scan* To speed up execution of the file server, you may wish to create a scan file periodically. This is done by executing the SUPSCAN program:

$ *supscan [-v]* **collection [basedir]**

This will construct a list of all files matching the specifications in the list file, and record it in the scan file. When an upgrade is performed on the collection, the file server will notice that the scan file is present and use this list instead of actually building the filename list by analyzing the list file. The time of record for the upgrade will then be the time at which the scan file was created rather than the time at which the upgrade occurs. A scan file is only useful for very large file collections that are upgraded many times each day. The options for the SUPSCAN program are *-v* ("verbose") to produce messages as it scans the file list, and *basedir* if the collection is a private collection whose base directory name is not the default name.

This is all the preparation required for a file collection to be upgraded.

Note that the default user-id for the file server is "Anonymous FTP" if no data encryption is used; if encryption is used, the default will be the owner of the subdirectory *sup/**collection*** within the base directory.

4.1 The List File

The *list file* contains a description of the files to be upgraded. It contains any number of commands, each on a separate line of text. Each line contains a keyword and a number of operands separated by spaces. All filenames in the list are evaluated relative to the collection's base directory on the repository machine for the file server, and relative to the base directory on the client machine for the client SUP process. All (except *execfile*) may be file specifications containing the shell's meta-characters *, *?*, *[...]*, and *{...}*.

The commands are:

upgrade ***filename ...***
: These files will be included in the list of files to be upgraded. If a directory name is given, all the files in that directory will be included and any subdirectories will be recursively included as well.

omit ***filename ...*** These files will not be included in the list; if a directory is specified, it will not be included nor will its contents be included. For example, the specifications *upgrade lib* and *omit lib/test* will include all files and subdirectories of *lib* except *lib/test* (and any subdirectories or files within *lib/test*).

omitany ***pattern ...*** The specified patterns are compared against the files in the upgrade list. If a pattern matches, the file is omitted. The omitany command currently supports all wild-card patterns except {...}. Also, the pattern must match the entire filename, so a leading */, or a trailing /*, may be necessary in the pattern.

backup ***filename ...*** The files will be marked for creating backup copies whenever they are upgraded (as described above). Only files can be included in this list -- not directories. As noted above, actual backup copies will only be created by SUP when these files are being upgraded, and then only if the user has specified the *backup* option in the supfile.

execute ***execfile*** *(**triggerfile ...***) ***...***
: The listed *execfiles* are command files or programs to be executed after an upgrade to perform routine installation of the upgraded files. Each *execfile* will be executed only when it is upgraded itself, or when any of its listed *triggerfiles* have been upgraded. The installation tasks performed by such files might be, for example, creating a table of contents for manual entries or for a subroutine library, or modifying a host name field within a data file.

include ***listfile ...*** The named listfiles will be read at this point. This allows a collection to contain another collection in its entirety, by simply specifying the name of the listfile for that other collection.

backup ***filename...*** The specified file(s) are marked for backup; if they are upgraded and the client has specified the *backup* option in the corresponding line of the supfile, then backup copies will be created as described above. Directories may not be specified, and no recursive filename construction is performed; you must specify the names of the specific files to be backed up before upgrading.

symlink ***filename...*** The specified file(s) are to be treated as symbolic links and will be transfered as such and not followed. By default, SUP will follow symbolic links.

rsymlink ***dirname...*** All the symbolic links in the specified subdirectories are to be transferred as such and not followed.

Blank lines may appear freely in the list file, and the order in which command lines appear within the file does not matter. Filenames must generally match as strings, e.g. with base directory */usr/vision*, "*lib*" in the *upgrade* command would not match "*/usr/vision/lib*" in the *omit* command; it would only match "*lib*" in the *omit* command. However, one special exception exists: if the base directory is specified via "dot" (.) in the *upgrade* command, the filenames within that directory need not begin with "*./*" in other commands. For example, with base directory */usr/vision*, the commands "*upgrade .*" and "*omit exp*" specify a file list including all files and directories within */usr/vision* except the subdirectory */usr/vision/exp* and its subdirectories and files.

5. Setup for SUP

You need to add the following entries to */etc/services*:

```
supfilesrv      871/tcp
supfiledbg      1127/tcp
```

A *supfilsrv* daemon should be kept running on any machine that may act as a repository for a collection. Since this includes private collections, it is advisable to have *supfilesrv* running on all machines if SUP is to be widely used. *nanny* can be used to start the *supfilesrv*.

The program *modcoll.8* is used to set up the control files for the "system" collections. See */usr/cs/man/man8/modcoll.8*

Table of Contents

1. Introduction	**1**
2. How to Use SUP to Upgrade a File Collection	**3**
2.1 Supfiles	**4**
3. What SUP Does	**7**
3.1 File Installation	**8**
4. How to Set Up a File Collection	**10**
4.1 The List File	**11**
5. Setup for SUP	**13**