# Exporting Home Directories on Demand to PCs

*David Clear and Alan Ibbetson* – University of Kent at Canterbury, UK
*Peter Collinson* – Hillside Systems

## ABSTRACT

Our university has run a UNIX service, both for teaching and research, for over fifteen years. Cost considerations have made us retain timesharing hosts (with X terminals), rather than migrate to desktop workstations. We have come to PCs only in the last five years, and only as poor relations to the main timesharing service, due to a lack of manpower.

Our public PCs do not have hard disks, instead they mount system filestore over NFS. Until recently the only per-user filestore was on local 3½" floppy disks and it was left to the users to back up and manage their own data. As the PC service has gained in importance, the demand for centrally managed filestore has prompted the development of a system to give users access to their own personal secure filestore from any public PC on campus. Moreover, the use of mixed-platform teaching has made it a requirement that a user's PC filestore be visible from their UNIX environment.

This paper describes an implementation of a distributed and relatively secure system for providing NFS based personal filestore on a PC. We use an export-on-demand mechanism. A single home directory is exported to just one PC rather than whole partitions being globally exported. The service has been in use since January 1994.

### Introduction

The University of Kent at Canterbury (UKC) has an active computing population of 8,000 out of a total of 11,000 members. This is high for the UK because all our undergraduates have access to email and News. Also, there is an increasing insistence by the Faculties that assessments be word-processed rather than hand written. We offer both UNIX and IBM PC services. Only a few well-funded researchers have desk-top UNIX workstations, with the bulk of our service running from X terminals into large Sun timesharing servers. Most PCs, especially those in public areas, run DOS/Windows and do not have a hard disk. Instead they obtain application binaries over NFS [1] from per-subnet fileservers. Until recently, the only permanent filestore we offered for PCs was floppy disk.

We have always wanted to integrate the UNIX and PC user filestore, so that users see the same home directory from both environments, but we are not aware of even moderately secure methods of exporting UNIX filestore to PCs. We feel that blindly exporting whole disk partitions read/write over NFS to large collections of public access PCs is unacceptable. Our hopes that off the shelf solutions would become available have not been realised. PC distributed filestore, such as NetWare [2] or LANmanager [3], is not based on UNIX servers. On the other hand, UNIX based secure distributed filestore such as the Andrew File System [4, 5] is not available on DOS PCs, despite futures indications from the OSF DCE community. The University of Mitchigan's IFS project [6], though offering the potential for integrated filestore, is sponsored research and specific to IBM mainframes.

This paper presents our implementation of an authenticated per-PC NFS export of a user's home directory. Once we could rely on the identity of the user sitting at a PC, it also proved straightforward to offer access to chargeable UNIX resources, such as networked laser printers.

### Public Access to Anonymous PCs

The Computing Service at UKC has historically been based on mainframes, then on minis running UNIX. We succeeded in ignoring PCs until five years ago but the initial slow growth in demand has exploded in the last two years. We are currently responsible for about 1,000 IBM compatible PCs, most of which are in public access rooms spread around the campus. All the PCs are connected to the campus network, which comprises 30 ethernet segments interconnected by IP routers and an FDDI backbone. A site licence arrangement throughout the UK higher education sector for PC-NFS [7] has made it our recommended file sharing and communications package. Although we have no control over the protocols used on local subnets, it is a part of the University's strategic plan that the infrastructure should only carry IP. This reduces the cost of maintaining the campus network and also simplifies its management.

Around 80% of our users see the Computing Service just as their provider of PC services. They have access to powerful PC applications, including electronic mail, and can launch the PCs into either our central services (X, UNIX, VMS, CWIS) or on to the global Internet via one of our UNIX hosts. We have tried hard to ensure that whenever a user sits in front of one of our PCs the picture they see is familiar and consistent. The key to this consistency has been to make the PCs diskless and have their system files served from one of a handful of UNIX fileservers over PC-NFS.

We have gained a number of advantages by avoiding the use of local hard disks. Software updates have become simply a matter of using *rdist* to propagate changes from a master server. This is what gives us a consistent user interface on all campus PCs. The servers export the PC software read-only, to prevent users from accidentally (or otherwise) modifying or damaging system filestore. Viruses are not a significant issue at our site, apart from checking files before mounting them on the servers. The final advantage is that the installation of a new PC only takes a few minutes. This approach is not new, Project Athena [8] drew similar conclusions for a distributed UNIX environment.

As with UNIX workstations, there are disadvantages in running PCs without local hard disks. Data access over the network is slow. This is not as bad as might be imagined because read-only NFS, with personal files on floppy disk, does not suffer the server bottleneck associated with synchronous writes. The migration from DOS to Windows at UKC is slow, since much of our hardware is low on memory. In future, it seems likely that local hard disks will need to be fitted to all Windows-capable PCs, if only to avoid the NFS writes associated with swap traffic. Currently, we serve up to 100 PCs from a single SPARC 2, although we have introduced some redundancy to avoid a server failure causing widespread disruption. This redundancy is provided via a ROM based boot mechanism.

All the ethernet cards in our PCs are fitted with a commercial boot ROM [9]. When the PC is booted, the ROM broadcasts a BOOTP [10] request for a server. The campus IP routers forward these packets to ensure they reach at least two of the fileservers. The servers run *bootpd* and use the PC's ethernet address to indicate, via the BOOTP response, which file the PC should load over TFTP [11]. This file is a DOS disk image and is loaded into memory as RAM drive A. Location specific information in files such as `CONFIG.SYS` is patched into this RAM drive with a vendor supplied utility, which is how DOS learns the PC's IP address, netmask and default gateway.

There are seven general PC-NFS fileservers. Which one a particular PC chooses depends on its location on the campus network. Redundancy and workload is controlled to a limited extent by system staff adding or removing entries in `/etc/bootptab` on each server. This load sharing, coupled with the unpredictability of user location on campus, makes the PC-NFS system fileservers a poor choice for holding per-user files.

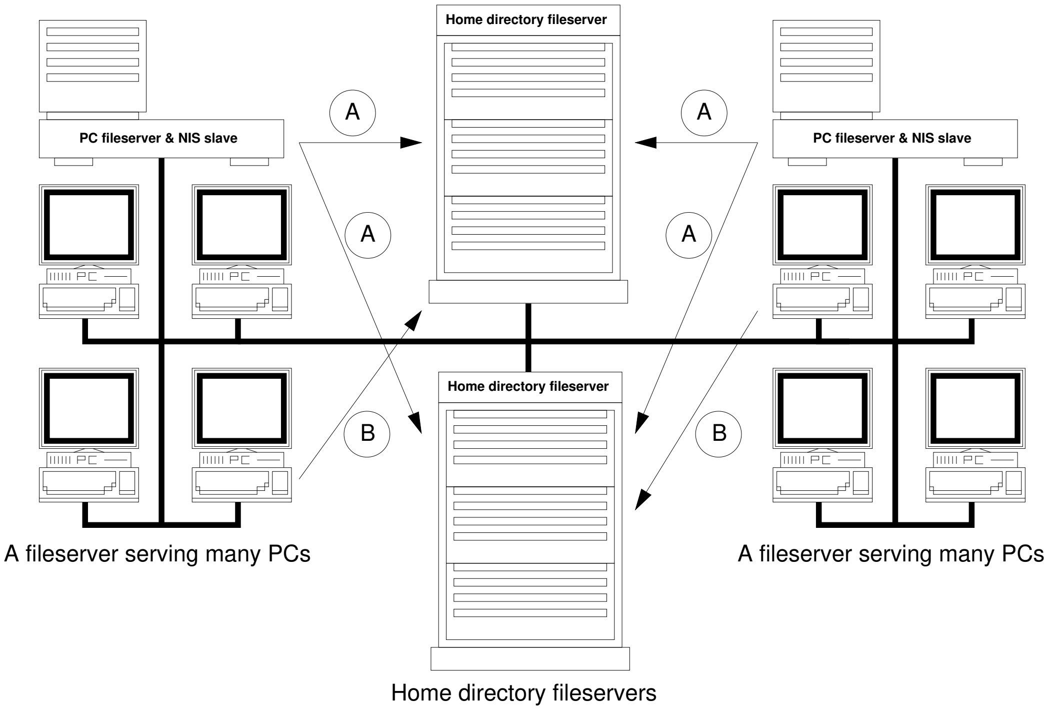


**Figure 1**: Basic Communicating Host Set

Until a year ago, we sidestepped the issue of personal filestore completely, by not offering anything other than local 3½" floppy disk. Users were left to back up and manage their own data, with sadly predictable results. Our group was therefore commissioned to provide each user with a moderately secure backed-up networked filestore.

### The Problem of Per User Filestore

The use of mixed platform teaching has made it a requirement that a user's PC filestore be visible from their UNIX environment. For a user's UNIX home directory to be accessible to PC-NFS, it must be exported from whichever timesharing host holds that UNIX account. PC-NFS provides a user authentication mechanism for distributed filestore, via the NET NAME command, but it does not provide a secure model of operation. All user filestore must be exported read/write by the server to every PC on campus. If the per-user filestore was located on the PC-NFS system fileservers, assuming the geographical problems could be resolved, it might have been just acceptable to abdicate responsibility on the grounds that 'they are only PCs'. However, our plan to coalesce a user's PC and UNIX filestore meant that insecurities could compromise our whole UNIX service.

As an aside, the ease with which uids can be faked to an NFS server from a PC should not be underestimated. It is fairly straightforward to find the PC memory locations holding the uid and gid used by PC-NFS just by a test and compare attack using several legitimate login names. More recently we have found that students have managed to defeat the boot ROM and run a UNIX like system such as Linux from floppy disk. From there, illegal access to any user's private filestore is but a *useradd* and *mount* away.

Figure 1 illustrates the two-level authorisation mechanism which we have implemented in an effort to avoid these obvious security problems.

As can be seen, user home directories are stored on a number of home directory fileservers. The arrowed lines represent communications between machines. There are two authorisation levels:

(A) PC fileservers are trusted hosts. Users at the PC must authenticate themselves with the PC-NFS server which will then make a privileged connection to the home directory fileserver and register the successful authentication from that particular PC.

(B) PCs are untrusted. They may only communicate with the home directory fileserver once they have been registered by the PC-NFS server.

### The Initial Menu

A home directory, or for that matter most of the PC applications, are not exported to a PC until the user has logged in.

At the end of the PC boot sequence, DOS runs the `AUTOEXEC.BAT` script from the RAM disk which was loaded by the boot ROM over TFTP from the PC-NFS server. This script offers the user a menu of just three items: start a telnet session to a campus host, register as a new user, or log in and mount a home directory.

Users are not required to authenticate themselves to the PC service if all they wish to do is use the PC as a dumb terminal. They will connect to a local timesharing host which will enforce its own access controls. Off-site telnet is only available to authenticated users, so we have some hope of tracing malefactors.

The details of new user registration are outside the scope of this paper. We use a pre-registration mechanism for students which is based on information supplied by our central administration just before the start of each academic year. Users collect their login names and choose a password by entering their surname and student number, which is supposedly secret, at least at the beginning of the year. Similar procedures seem to be quite common in the USA, for example [12], but the majority of UK universities still use a paper based system. Our experience is that automation becomes mandatory once all newly arriving students are registered for IT services, especially email.

The third menu item allows the PC user to log in and mount their home directory. The only trusted host known to the PC at this point is the local PC-NFS fileserver. Each of these servers needs access to the same password file. They also need to know, for each user, which host holds their UNIX account, as well as their home directory pathname. The initial implementation has used standard building blocks as much as possible. In particular we chose the Network Information Services (NIS) [13] for password support because implementations are provided by Sun for both UNIX and PC-NFS.

### Network Information Services

The Network Information Services, formerly called Yellow Pages, are well documented by Sun, but a brief outline is presented here for the uninitiated. The examples are chosen to focus on the specifics of our PC home directory service.

The smallest NIS entities are the key and the data. The key/data pair form the smallest addressable object, the table entry. Table entry keys are unique within a table. Multiple tables are grouped together to form domains. Therefore table names are unique within a domain and domain names are unique across a LAN: Figure 2 illustrates the model.

The highlighted entry is the *ajm1* entry from the passwd.byname table of the comp-pchome domain.

A NIS server for a domain receives its data from the domain's NIS master. There is one NIS master per domain and any number of NIS slave servers, providing resilience to failure. When a client needs to use the NIS database it broadcasts for a server. Whichever server responds first will be used and the client need not distinguish between a master and slave server.

On the server, the directory `/var/yp` is the root for all NIS data. The data for the domain 'dom' is stored under `/var/yp/dom` in *ndbm* format files.

NIS tables are transferred (pushed) from the master server to all the slave servers by the *yppush* command. This is required for each table to be updated, four in our case. The mechanism is moderately secure since the slave servers know the identity of the master server.

From a UNIX standpoint, NIS comprises four processes:

- ypserv – the NIS server daemon
- ypbind – the NIS binding daemon
- yppush – the NIS update program
- libc – the NIS programmer's interface

Within a NIS server there is one daemon process which handles both requests from clients and the updating of its own information from the master server. This daemon process is called *ypserv*.

When a client request enters *ypserv* for processing, more than a simple table lookup takes place. A table can be created with a security option, by specifying the *-s* flag to *makedbm* when building the table. This means that *ypserv* will only answer requests from clients with low numbered IP ports. It can be used for crude protection of a password table, for example. The security is easily subverted from the PC world, but it requires knowledge of the RPC programming library. The same security model is used in our PC Home Directory System and is thus an area for improvement.

Another feature of *ypserv* is the ability to forward host table requests to the Domain Name Service [14]. When invoked with the *-d* flag, or when the host tables are built with the *-b* flag to *makedbm*, *ypserv* takes all requests to the hosts.byname and hosts.byaddr tables and passes them to the DNS for resolution. We use this option at UKC. Smaller sites could equally well elect to use the explicit host tables and not use the DNS.

To look up an entry in a NIS table, a client program must make a connection to a NIS server. It is possible to broadcast for a NIS server but for every process on a client to do this would be inefficient. A level of indirection is therefore added.

A client process talks to a local daemon called *ypbind* to find the network address of a NIS server for the particular domain. *ypbind* maintains a list of domains and servers so that, once it has received a reply to it's own broadcast for a NIS server, it can cache and reply instantly to other client process requests. On single process operating systems such as PCs running DOS, the functions of *ypbind* are provided transparently in the NIS library.

### PC Home Directory System Components

Our site has made extensive use of the UNIX group id for many years. We place users in groups depending on their affiliation within the University. For example, *elt* is an undergraduate in Electronics ('t' for teaching) and *elr* is a researcher. For historical reasons, Computing Laboratory staff are in group *cur*.

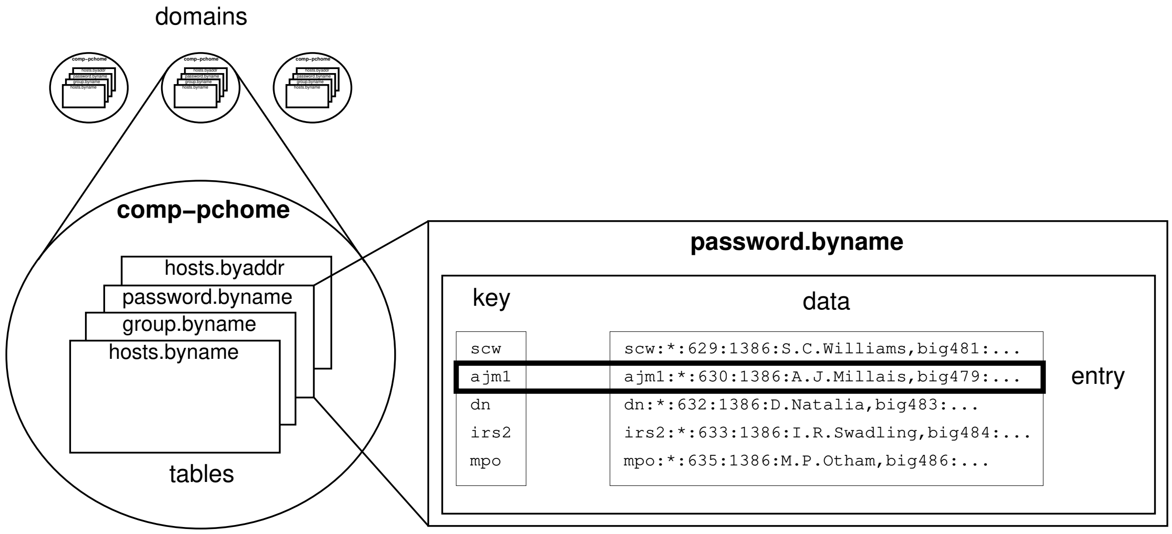


**Figure 2**: The NIS Namespace Model

User home directories have a consistent structure across all machines. An undergraduate called *fred* in Electronics will find their files in `/home/elt/fred`. Symbolic links at the group level are used to manage this hierarchy. All files in the *elt* group are placed in the same partition for ease of administration, it also aids with peer pressure when partitions fill up.

### The NIS Configuration in the PC Home Directory System

We use the NIS domain comp-pchome, which all our public PCs use to resolve host names, group names, services, and networks. The password file, which needs to be accessed by all PC-NFS servers, is available in the standard passwd.byname and passwd.byuid NIS tables in the comp-pcuser domain. Every PC user has an entry in these tables, which contain their password for the PC home directory system and their group id. However, it should be stressed that users do not ordinarily appear in `/etc/passwd` on the PC-NFS fileservers. They do not have a UNIX account there.

There is one other NIS table of interest to the system, the *group.home* table. It performs the mapping from group names to host and path names described above, and tells the system where to look for a particular group's home directories. For example, if it contained the entry:

`cur` maps to `stork:/home/cur`

and the user *dac*, who is in group *cur*, logs in, the system will locate dac's home directory as `/home/cur/dac` on host *stork*.

The NIS tables are mastered on the same host as the *rdist* master for PC applications binaries and all other PC fileservers are configured as slave NIS servers. This means that a PC uses the same fileserver for application binaries, PC-NFS authentication and printing, and NIS serving.

### The Faculty Home Directory Fileservers

User home directories are distributed between four SPARCcenter 1000 servers. They support both a general UNIX timesharing service (primarily to X terminals) and PC home directory fileserving. Filestore on a particular fileserver is allocated to users by group which, as closely as possible, maps to a faculty.

If the *group.home* NIS table points to a user's UNIX home directory, the PC and UNIX filestore is integrated. At the time of writing, we have not yet taken this final step. Each of our users has two directories on one of the SPARC1000s. There is a PC home directory under `/pchome`, pointed to by the group.home table, and a UNIX home directory under `/home`, pointed to by `/etc/passwd`. Their PC home directory is visible from their UNIX account but export restrictions protect their UNIX home directory from the PCs. We hope this is not just natural British conservatism. It also seems good engineering practice to have an extended test period before making a change which will add so much extra functionality that it cannot be withdrawn. We will be throwing the big switch real soon now.

### The PC-NFS Daemon rpc.pcnfsd

The PC-NFS authentication and printer daemon *rpc.pcnfsd* was originally supplied by SunSoft. Its functions cover:

- Authenticating usernames and passwords.
- Submitting print jobs to the local PC-NFS server.
- Cancelling print jobs.
- Reporting printer queues.

We modified the daemon to add the functions:

- Registering authenticated PCs with the home directory fileservers.
- Submitting print jobs to a user's home directory fileserver.

The change for PC printing is needed to allow us to charge for laser output. Users do not have UNIX accounts on the PC-NFS fileservers, so instead their print jobs are submitted from their home directory server.

The details of PC printing are straightforward and will not be discussed further here. Instead, we will concentrate on the mechanics of PC authentication and registration with a home directory fileserver.

The user types a login name and a password into a locally written login program on their PC. Spawning the standard PC-NFS command

`NET NAME` *login password*

sends this information to *rpc.pcnfsd* on the local fileserver. This is far less secure than a Kerberos [15] based mechanism. Although plaintext passwords are sent over the network, it is no worse than our standard telnet service. We are migrating towards UTP and managed hubs to reduce our vulnerability to eavesdropping attacks. The command can only return a success or failure indication, no text or result codes can be passed back, which limits the amount of error reporting which can be done.

The local fileserver is used as the NFS authentication host as it is geographically close. Until users have identified themselves, their home directory server is unknown. The PC-NFS daemon uses NIS for password lookup. If the password is valid, NIS is consulted again to determine, based on the user's group id, which home directory host serves their files. A privileged connection is then made to our *pcshared* service (share is the Solaris 2 term for export) on that host and the user is registered as being correctly authenticated on the PC at which they are sitting. A successful user authentication response is then returned to the PC.

At this point the *NET NAME* command returns control to the client login program. If the login has been accepted, the client makes a connection to *pcshared* on the home directory server and requests the user's home directory be exported. The export is made *only* to the client PC. If the export cannot be made or *pcshared* does not believe the client PC is authorised, an error message is returned and displayed on screen by the client login program. If all goes well, the client mounts the user's home directory using the PC-NFS command:

`NET USE H:` *host:path*

When this command succeeds, the logging in process is complete.

In order not to tie up *rpc.pcnfsd*, which is single threaded, the connection to *pcshared* is only given 10 seconds to succeed. If the time expires, a failure indication cannot be sent to the client PC. It would cause our invocation of *NET NAME* to emit a 'password incorrect' message, which would confuse the user. Instead, a success indication is returned to the PC. On attempting to request the export, the user will receive a 'share request refused' message from *pcshared*. This message will be displayed as a general error although it might be wiser to retry at this point. There is room for improvement generally in the error messages seen by users.

### The Share Daemon pcshared

Our locally written *pcshared* daemon resides on the home directory fileservers. It responds to authorisation messages from trusted PC-NFS servers and subsequently to export requests from previously authorised client PCs.

PC-NFS authentication servers have a trusted status with the *pcshared* daemon. Such trusted hosts may register client PCs, specifying a user and a home directory path, using a telnet-like protocol on a low numbered TCP port. An example authorisation message sent from a PC-NFS server for user *dac* sitting at PC *pcpra* is:

```
authorise pcpra dac /home/cur/dac
```

Once the user authentication is complete, the PC client login program uses NIS to select the host serving the user's home directory and makes a connection to *pcshared*. The PC requests that the user's home directory be exported to it, thus:

```
share dac /home/cur/dac
```

As the client was previously registered by the trusted PC-NFS server, the request is accepted and the export is performed. The full dialogue is shown in Figure 3.

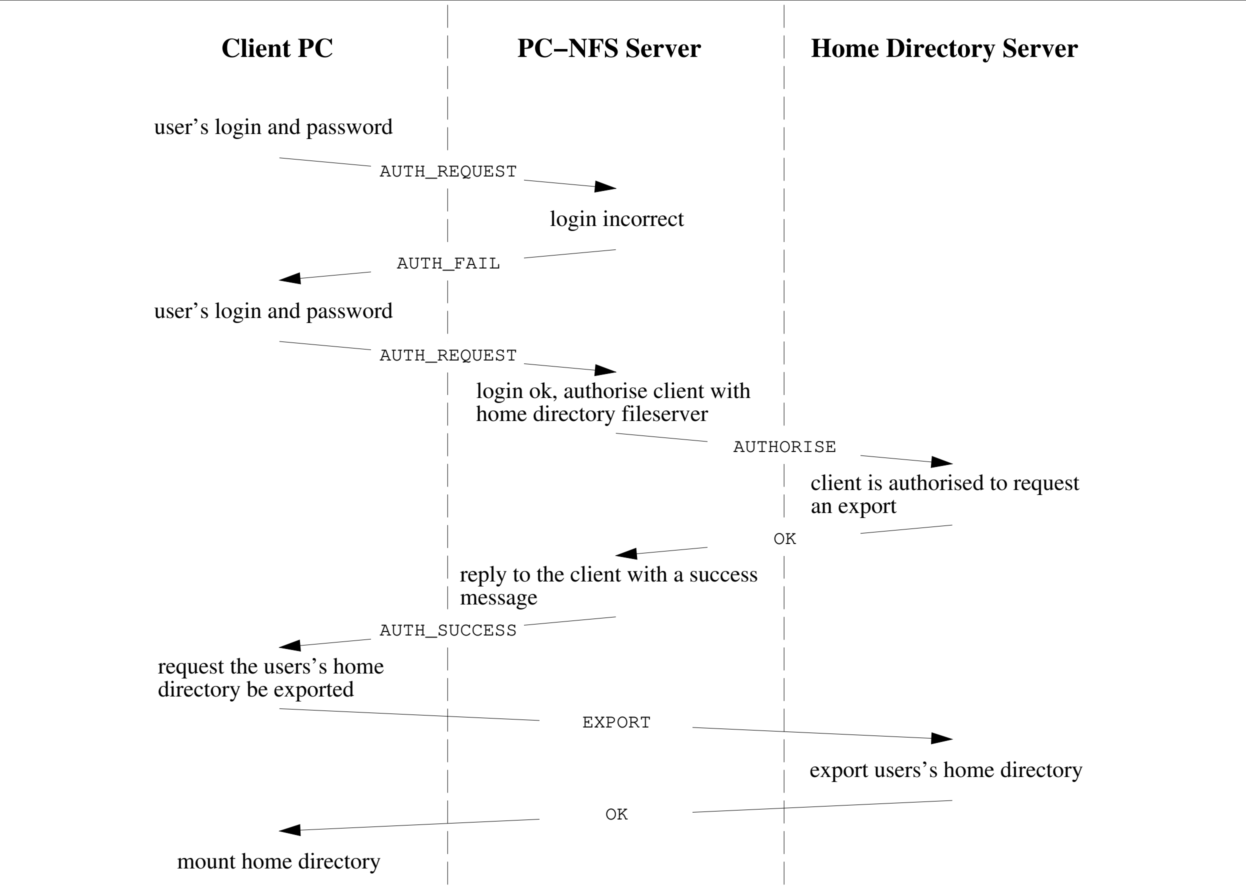


**Figure 3**: The Logging In Dialogue

The figure shows two attempts to log in. The first is with an incorrect password and the second succeeds. At first sight it might seem more natural for *rpc.pcnfsd* to directly request the home directory server to export the user's home directory. There are several reasons why the above system is used:

- The home directory server daemon, *pcshared*, will only listen to trusted hosts in the first instance. Only PC fileservers are trusted as they are secure UNIX machines with no access by users. Thus, the fileserver must talk directly to *pcshared*.
- The mechanics of exporting a directory are such that it can be a time consuming process. PC-NFS authentication is based on RPC/UDP datagrams. If *rpc.pcnfsd* does not respond quickly enough then another request will be sent by the client PC's RPC network layer. Thus *rpc.pcnfsd* must not do anything that takes too much time.
- To get around the RPC timeout problem, it might be thought that *rpc.pcnfsd* could fork and let a child process request the export from *pcshared*. The parent would then respond with a success message to the client. This would not work. The client PC needs to know when the export has completed, otherwise it will try to mount an unexported directory.
- By increasing the involvement of the PC client login program, more error conditions can be reported to the user. The simple success/fail response codes returned by *rpc.pcnfsd* are not suitable for the wide range of possible failures.

Although PC-NFS servers are trusted, a configuration file on the home directory fileserver limits which filesystems may be exported.

### Logging out

When the user finishes a PC session they are expected to run our *LOGOUT.EXE* command, which sends an `unshare` request to *pcshared* on the home directory server to remove the export. Although we try hard to remind them, users are apt to walk away without logging out – sometimes they have no choice if the PC crashes. We partially counter this by scanning the entire list of exports when a new user logs in, and cleaning up previous entries for that PC. However, this only works if both users have their home directory on the same server.

### Password Changing

A user may change their password by running *PASSWD.EXE*. This command determines the hostname of the master NIS server for the comp-pchome domain and makes a connection to the *nispwupdated* daemon on that machine.

Once the connection to *nispwupdated* is established, it sends the client a random single byte. This is used to very lightly encrypt the data stream. The goal is not to make the password change dialogue secure (only a Kerberos style system could do that), rather to make it harder to read for the casual listener.

After the random byte has been sent, the user is prompted for their username and password which is then sent, lightly disfigured, to *nispwupdated* for checking. Once a successful match is made, the user is prompted for a new password. If it is deemed to be acceptable [16], it is put into the password file. After the password dbm files have been updated, they are *yppush*'ed to the NIS slave servers by the daemon.

### Disk Quota Reporting

Users may find out their disk quota by using the *QUOTA.EXE* command, which connects to the *pcquotad* service on their home directory server. Disk quotas are not considered sensitive information, so the client need only send the user's login name to *pcquotad* which will respond with the user's quota data. The client program then nicely formats this data and displays it on screen.

## Experiences to Date

We presently have 1750 users using 8 Gb of filestore. There are 4 SPARCcenter 1000s, which serve home directories to a community of over 1,000 PCs as well as providing a general UNIX service, and 7 PC-NFS servers. In our hands, *yppush* has proved relatively unreliable. It is invoked by the NIS master to propagate the maps to the slaves every time a user changes their password. Whether the RPC mechanism is inherently unreliable or race conditions arise when several users change their passwords at once is unclear. The observed effect is that, after changing their password, a user may be able to log in only at PCs served by a subset of the slave PC-NFS servers. Running *yppush* a second time by hand usually clears the problem.

## Future Directions

The current implementation of the PC Home Directory System as described has several failings. With hindsight, we would make improvements in the following areas: serving passwords to the network in a secure way, making a cleaner login sequence, and providing more resilience on a server crash.

### Network Password Serving

The use of NIS to serve passwords has the disadvantage that it is not secure. Access may be gained to the encrypted password strings by anyone with socket and RPC programming knowledge. Another problem with NIS is that, when a single password is updated, the entire password table is shipped to all NIS slaves. As some fileservers sit on the same subnets as public PCs, the password file is vulnerable to *netwatch* style programs.

A way forward would be to use a bespoke password file transport mechanism. This would take a similar form to how NIS does things (with a master host) but with the following differences:

- TCP connections could be used instead of UDP as used by NIS. We suspect that this would make the transfer of passwords more robust.
- Only password file differences would be sent out to the password server slaves. This removes the ability for a user to trigger the movement of the entire password file.
- The password file transfer could be encrypted, to thwart eavesdroppers.

The use of such a method would be compatible with the current system as only *rpc.pcnfsd* uses NIS to look up user passwords.

### A Cleaner Login Sequence

The *rpc.pcnfsd* interface for login authentication is not particularly good at reporting errors. A possible alternative to this would be to remove the user authentication and export requests from *rpc.pcnfsd* altogether and build a new login daemon to handle that. The difficulty is that PC-NFS on the client PC must run the `NET NAME` program to authenticate the user. A new model which could accommodate this could be:

1. The user on the client PC runs `LOGIN.EXE` which makes a connection to the login daemon on the PC-NFS fileserver.
2. The user then has a dialogue similar to that of *nispwupdated*, with lightly encrypted network traffic, to pass across the username and password.
3. The login daemon checks the username and password and, if incorrect reports it as such.
4. If the password is correct the login daemon has the user's home directory exported by talking to a *pcshared* style process on the user's home directory host. Notice that, as this is a fully synchronous system, the trusted host is free to ask for the export to be performed itself.
5. Once exported, the login is complete so the login daemon registers that the user has logged in with *rpc.pcnfsd*. Then it returns a success code to the client.
6. The client runs `NET NAME` to perform the mandatory PC-NFS logging in functions. *rpc.pcnfsd* can simply reply with a positive return code as it has been told that the user has already correctly logged in. No password need be given or checked.

The advantages of this system are that

- The home directory export request is made to the home directory server by the trusted PC-NFS host. There is no authorisation step.
- There is a fully synchronous error reporting path. This makes error trapping and reporting trivial. It removes the need for the Message Send Protocol [17], which we use (somewhat untidily) at present.
- At no point is *rpc.pcnfsd* tied up, thus leaving it to handle print requests as normal.

### Persistent Exports

At the moment, if a home directory server reboots for any reason, the export table is lost. To the user this would be seen as an 'access denied' message from the fileserver once it came back to life.

This could be better handled by having *pcshared* record the exports it has made in a private file which, on startup, it could read to pre-configure itself. If a fileserver were to reboot the home directories would be re-exported and the users would not be forced to log back in.

## Availability

The PC home directory software is available by anonymous ftp from `ftp.ukc.ac.uk` in directory `pub/pchome`.

## Author Information

David Clear obtained his bachelors degree in Computer Science from the University of Kent in 1990. Since then he has worked for the Computing Service Division, specialising in UNIX systems support. Reach him at the Systems Group, Computing Laboratory, University of Kent, Canterbury CT2 7NF, England or at dac@ukc.ac.uk.

Alan Ibbetson graduated in Biochemistry at the University of London in 1972, but quickly moved into computing. His masters degree is for research in message passing operating systems. He has been involved in UNIX support since 1980. Since 1990 he has managed a team of six people, including David, providing UNIX services to the campus at UKC. Reach him at ali@ukc.ac.uk.

Peter Collinson gained a Ph.D in Computer Science from the University of Essex in 1973. He then moved to the University of Kent as a Lecturer in Computer Science. He is responsible for making UNIX a major feature of computing activity on the Kent Campus. In 1980, UKC became one of the first sites in the UK to offer UNIX cycles to all computer users on campus. He left UKC in 1989 to start his own consultancy, Hillside Systems. He retains links with the University and is an Honorary Fellow of the Computing Laboratory. His email address is pc@hillside.co.uk and WWW page is http://www.hillside.co.uk/.

## References

[1] ''Design and Implementation of the Sun Network File System'', Sandberg et al., Proc. USENIX, Summer, 1985.

[2] NetWare sales literature, Novell Inc.

[3] LANmanager sales literature, MicroSoft Corporation.
[4] ''Scale and Performance in a Distributed File System'', J.H. Howard et al., ACM Transactions on Computer Systems, Vol 6, No 1, February 1988.
[5] ''Scalable, Secure and Highly Available Distributed File Access'', M. Satyanarayanan, IEEE Computer, Vol 23, No 5, May, 1990.
[6] ''Institutional File System Overview'', T. Hanns, AIXTRA: The AIX Technical Review, pp 25-32, January, 1992.
[7] ''PC-NFS Administration Guide'', 801-3977-10, March, 1993, SunSelect.
[8] ''Berkeley UNIX on 1000 Workstations: Athena Changes to 4.3BSD'', G. W. Treese, Proc. USENIX, Winter 1988.
[9] ''Release notes for the TCP/IP Boot PROM Version 1.47'', D. Koppen, EDV-Beratungs-GmbH, May, 1993.
[10] ''Bootstrap Protocol (BOOTP)'', B. Croft and J. Gilmore, RFC 951, 1985.
[11] ''The TFTP Protocol'', K. R. Sollins and N. Chiappa, RFC 783, 1981.
[12] ''The Athena Service Management System'', M. A. Rosenstein et al., Proc. USENIX, Winter 1988.
[13] ''The Network Information Service'', in System & Network Administration, SunOS 4.1 manual set, 800-3805-10, March 1990, Sun Microsystems.
[14] ''Domain Names – Concepts and Facilities'', P. Mockapetris, RFC 1034, 1987.
[15] ''Kerberos Version 5 Revision 5.1'', J. Kohl and B.C. Neuman, MIT Project Athena, Cambridge MA, September 1992.
[16] ''Foiling the Cracker: A Survey of, and Improvements to, Password Security'', D. V. Klein, Proc. UKUUG Summer 1990 Conference, London.
[17] ''Message Send Protocol 2'', R. Nelson and G. Arnold, RFC 1312, 1992.