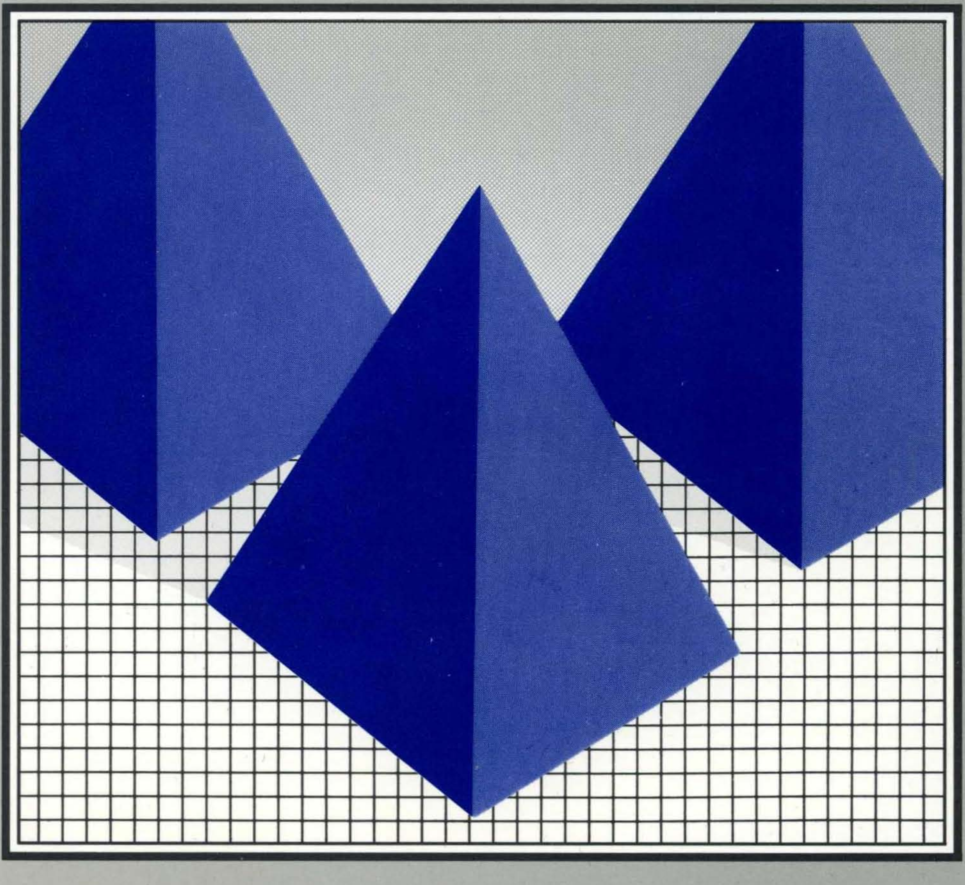
HEWLETT-PACKARD

HP 9000 Series 300 and 800 Computers

Using NFS Services



HP 9000 Series 300 and 800 Computers

Using NFS Services

HEWLETT
PACKARD

Manual Part Number: B1013-90000
Printed in U.S.A., September 1989

Notice

Hewlett-Packard Co.
3404 E. Harmony Rd.
Fort Collins, CO 80525 U.S.A.

Printing History

First Edition . . . . September 1989

Contents

1

Documentation Overview

Before reading this manual, you should be familiar with HP-UX and have access to *HP-UX Reference* manuals.

Note The information contained in this manual applies to both the Series 300 and Series 800 HP 9000 computer systems. Any differences in installation, configuration, operation, or troubleshooting are specifically noted.

You will find this manual helpful if you are using NFS Services but have no administrative responsibilities.

Contents

Refer to the following list for a brief description of the information contained in each chapter and appendix.

Index

The index provides a page reference to the subjects contained within this manual.

Conventions

This manual uses the following format for all entry instructions and examples.

Bold Text	emphasizes the word or point.
`Computer Text`	specifies a literal entry. You should enter the text exactly as shown.
Italic Text	indicates you should enter information according to your requirements.

Example:

`domainname`	*`domain name`*
Enter the word domainname	Enter the name of your YP domain.

Documentation Guide

For More Information	Read
ARPA Services: Daily Use	*Using ARPA Services*
ARPA Services: System Administration	*Installing and Administering ARPA Services*
C Programming Language	*C Programming Guide*, Jack Purdum, Que Corporation, Indianapolis, Indiana *The C Programming Language*, Brian W. Kernighan, Dennis M. Ritchie; Prentice-Hall, Inc.
Commands and System Calls	*ARPA/Berkeley Services Reference Pages* *NFS Services Reference Pages* *NS Services Reference Pages*
HP 92223A Repeater	*HP 92223A Repeater Installation Manual*
HP-UX: Installation	*HP-UX Installation Manual*
HP-UX: Operating System (HP 9000)	*HP-UX Concepts and Tutorials* *HP-UX Installation Manual/HP 9000 Series 300* *Installing and Updating HP-UX/HP 9000 Series 800* *HP-UX Reference* Manuals *HP-UX System Administrator's Manual/HP 9000 Series 800* *HP-UX System Administrator's Manual/HP 9000 Series 300* *Beginner's Guide* series for HP-UX *Introducing UNIX System V*
HP-UX: System Administration	*HP-UX Administrator's Manual/HP 9000 Series 800* *HP-UX System Administrator's Manual/HP 9000 Series 300*

For More Information	Read
LAN Hardware: Installation	*HP 98643A LAN/300 Link LANIC Installation Manual* *LAN Cable and Accessories Installation Manual*
Networking: General Information	*Networking Overview*
NFS Services: Common Commands	*Using NFS Services*
NFS Services: Programming and Protocols	*Programming and Protocols for NFS Services*
NFS Services: System Administration ■ Configuration ■ Installation ■ Maintenance ■ Network Lock Manager ■ Remote Execution Facility (REX) ■ Troubleshooting ■ Virtual Home Environment ■ Yellow Pages	*Installing and Administering NFS Services*
NS: System Administration	*Installing and Administering NS Services*
ARPA: System Administration	*Installing and Administering ARPA Services*
C2 Security	*HP-UX System Security Manual* *HP-UX Beginner's Guide* *A Beginner's Guide to Using Shells/HP 9000 Series 300/800 Computers*

2

NFS Services Overview

HP's NFS (Network File System) Services product allows many systems to share the same files. It is an independent networking product, not a distributed operating system. NFS differs from distributed operating systems by not limiting its use to specific hardware and software. Rather, it operates on heterogeneous nodes and in operating systems from a variety of vendors. Explicit file transfers across the network to your local node are unnecessary. Since access techniques are transparent, remote file access remains similar to local file access.

With NFS all network nodes are either **clients** or **servers** or both.

- A **client** is any node or process that accesses a network service.

  An NFS client can also be configured as any combination of an NFS server, YP (Yellow Pages) client, or YP server. (A YP server **must** also be configured as a YP client.)

- A **server** is any node that provides one of the network services. A single node can provide more than one service.

  An NFS server can also be configured as any combination of an NFS client, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

- Servers are passive in that they always wait for clients to call them.

  The degree to which clients **bind** to their server varies with each of the network services. However, the client always initiates the binding. The server completes the binding subject to access control rules specific to each service.

- NFS servers are **stateless**; they do not maintain information relating to each client being served. Each file request goes to the appropriate server with the parameters attached to it locally (e.g., read and write privileges). An advantage of servers being stateless is that you can reboot servers without adverse consequences to the client.

NFS Services

The NFS Services product includes the following components.

- NFS Remote File Access
- Remote Execution Facility (REX)
- Remote Procedure Calls (RPC)
- Remote Procedure Call Protocol Compiler (RPCGEN)
- External Data Representation (XDR)
- Network Lock Manager
- Network Status Monitor
- Yellow Pages (YP)
- Virtual Home Environment (VHE)

The NFS, REX, Lock Manager, and YP functionalities are built on top of RPC and XDR library routines.

Note You must recompile programs that access remote directories. Otherwise, these programs will not be able to access remote directories mounted through NFS since the old directory routines use a *read* call instead of a *getdirentries* call to access those directories.

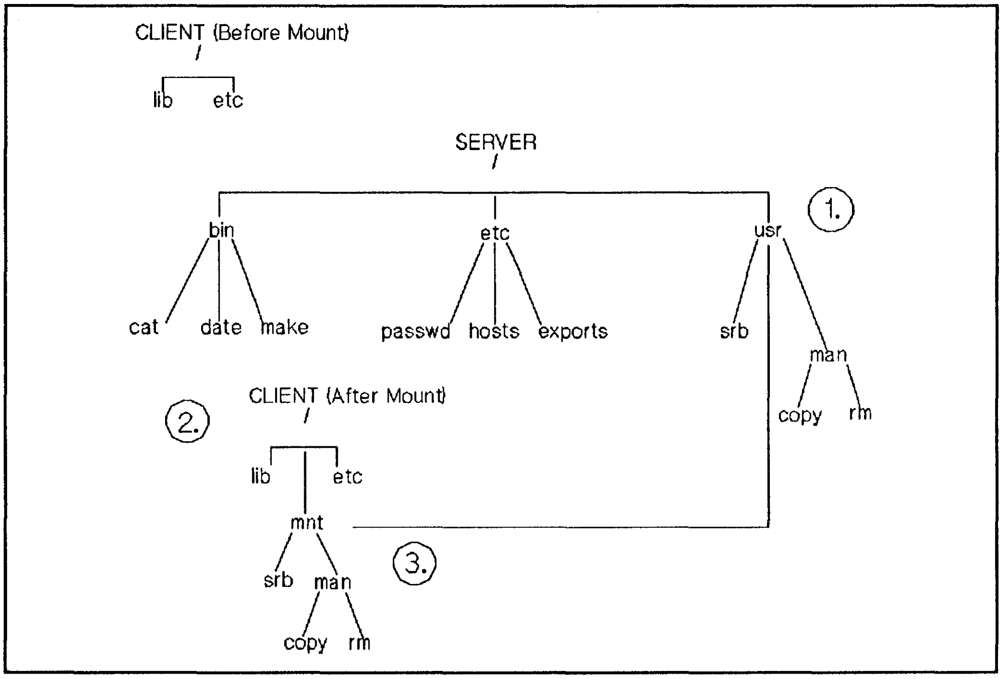
NFS Remote File Access

Before the client can access remote files,

- on the server, the super-user must export the file system (i.e., make it available) to the client and
- on the client, the super-user must mount (import) the file system.

Access to remote files is the same as for local files. You need to include either the complete path name starting with / (slash) or the path name relative to the current directory.

EXAMPLE:



1. The super-user edits the server's */etc/exports* file to make the */usr* file system available to the client.

   ```
   server super-user% cat /etc/exports
   /usr client_name
   ```

2. On the client, the super-user creates a mount point */mnt* (empty directory) and mounts the file system.

   ```
   client super-user% mkdir /mnt
   client super-user% mount server:/usr /mnt
   ```

3. The client reads the files in the */mnt* directory.

   ```
   client% more /mnt/man/copy
   ```

Two very important features of NFS Remote File Access are **named pipes** and **device files**. The following sections explain the details of these two features.

Named Pipes

A named pipe is a special type of object in the HP-UX file system. A named pipe is one of the many ways in HP-UX that unrelated processes can communicate. HP-UX processes executing **on the same client system** are able to communicate using named pipes. You can use named pipes via normal file operations, e.g. *open()*, *close()*, *read()*, *write()*. Typically, one process will open the named pipe for reading and another process will open it for writing.

To illustrate named pipes, consider the following example:

EXAMPLE:

> C1 and C2 are processes executing on system C. Also assume host C has mounted file system / from host S on */mnt*. C1 opens */mnt/FIFO* for reading and C2 opens */mnt/FIFO* for writing. C1 can now read what C2 wrote to the named pipe.
>
> Next, assume a third process (process D3) is running on another client D which also has / from S mounted on /mnt (on system D), and it opened */mnt/FIFO* for reading. Is process D3 able to read what process C2 wrote to this named pipe? No, because no actual NFS activity occurs between the

NFS client and NFS server for named pipe reads and writes. These are handled entirely by the client.

Note In certain cases there would be NFS activity. For example, if you do a *chown(2)* on the named pipe, the request will go to the server to change the owner.

---

mknod()

Named pipes are created with *mknod()*. Any user can create a named pipe with *mknod()*. (Use of *mknod()* to create device files requires super-user privileges.)

Note If you attempt to make a directory or a network special file over NFS, *mknod()* will fail and will return with *errno* set to EINVAL.

Device Files

Device files are another type of object in the file system, and are used to access physical or conceptual devices attached to the system. NFS device files always refer to a device **attached to the local system** and can generally be used where a local device file would be used. Like named pipes, device files are operated on through normal file system operations. For example, to write to the system console, you can write to the file */dev/console*.

To illustrate the use of device files, consider the following:

EXAMPLE:

> System C is an NFS client of the NFS server System S, and has mounted file system / from host S on */mnt* (a super-user on System C executed the command *mount S: / /mnt*). If a process on System C attempts to write to */mnt/dev/console*, a device file representing the system console, the output will go to the system console on System C, not on System S. If a process on System S attempts to write to */dev/console*, which is the same "file" that System C wrote to, it will actually write to the console on System S.

NFS Mounts with Device Files

NFS device files are not secure. Therefore, the system administrator has the option of turning off device file access on a per-NFS mount basis. The administrator uses the *-o nodevs* option to the *mount(1m)* command to turn off device file access.

EXAMPLE:

```
mount -o nodevs nfserver:/servermountpoint /clientmountpoint
```

Note The *nodevs* option does not turn off support of named pipes.

Mounting From NFS Device Files

You may mount a local disk that is represented by a remote NFS device file.

EXAMPLE:

```
   mount /mnt/nfs/dev/dsk/0s0 /localmntpt
```

Access to the newly mounted file system will proceed as if the disk had been mounted from a local device file.

Note Access to the local disk's mounted file system will not be affected even if the NFS file system is unmounted.

Normally when unmounting a file system, you can give either the name of the device file or the name of the mount point. However, if the NFS server is down or the NFS file system is down, you must give the mount point to unmount the local disk.

EXAMPLE:

You would enter the following to unmount a local disk:

```
umount /localmntpt
```

instead of:

```
umount /mnt/nfs/dev/dsk/0s0
```

The latter case will not fail if the NFS server is down, but it will hang until the server comes back up as any other NFS access does.

Remote Execution Facility (REX)

The Remote Execution Facility allows you to execute commands on a remote host. REX is similar to the Berkeley service remote shell (*remsh(1)*) with two major differences:

- Your environment is simulated on the remote host
- You can execute interactive commands on the remote host

Remote Procedure Call (RPC)

NFS Services consists of remote programs composed of remote procedures called from the client nodes on the network. Optimally, a remote procedure computes results based entirely on its own parameters. Thus, the procedure (and therefore, the network service) is not tied to any particular operating system or hardware.

NFS clients access server information and processes by making a remote procedure call. RPC allows a client process to execute functions on a server via a server process. Though these processes can reside on different network hosts, the client process does not need to know about the networking implementations.

The client first calls an RPC function to initiate the RPC transaction. The client system then sends an encoded message to the server. This message includes all the data needed to identify the service and user authentication information. If the message is valid (i.e., calls an existing service and the authentication passes) the server performs the requested service and sends a result message back to the client.

Remote Procedure Call Protocol Compiler (RPCGEN)

RPCGEN is a Remote Procedure Call compiler. You use it to convert applications running on a single computer to ones that run over a network. It is also used to assist in writing Remote Procedure Call applications simply and directly. With RPCGEN, your development time will be reduced and you will spend less time coding and debugging network interface code.

You produce three of the files required to convert an application to run on a network. These files are:

- **protocol description file**
- **client side file**
- **server side function file**

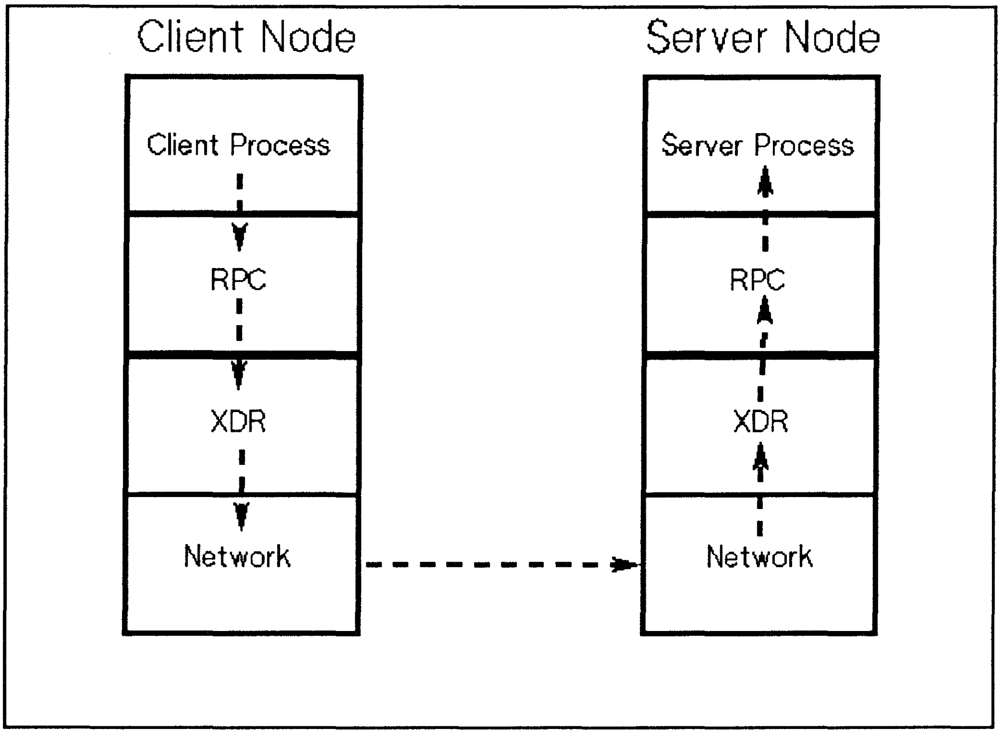
RPCGEN accepts remote program interface definitions (the protocol description file) written in RPC and produces the following C output files, which you may use as a starting point, rewriting as necessary:

- **header file**
- **client side subroutine file**
- **server side skeleton file**
- **XDR (External Data Representation) routine file**

If you wish to use the RPCGEN compiler to write RPC applications, refer to the "RPCGEN Programming Guide" chapter in the *Programming and Protocols for NFS Services* manual.

External Data Representation (XDR)

RPC uses the eXternal Data Representation functionality to translate machine dependent data formats (i.e., internal representations) to a universal format used by all network hosts using RPC/XDR. Thus, XDR enables heterogeneous nodes and operating systems to communicate with each other over the network.



RPC and XDR Data Transfer

Note: This figure does not correspond to the ISO Model.

Network Lock Manager

NFS Services includes the Network Lock Manager and the Network Status Monitor. The Network Lock Manager supports file locking and synchronized access to shared files via *lockf* and *fcntl* for NFS. The Network Status Monitor is used by the Network Lock Manager to maintain the stateful locking service within the stateless NFS environment. It allows applications to monitor the status of other computers and systems.

Yellow Pages (YP)

The Yellow Pages (YP) is an **optional** service containing a collection of cooperating YP server processes that provide YP clients access to data. You can administer all the databases from one YP **master server** since it propagates data across the network to other YP servers. YP includes the following features.

- YP manages unlimited databases. Typically these include files in */etc*: *group*, *hosts*, *netgroup*, *networks*, *passwd*, *protocols*, *rpc*, and *services*.

  For example, programs previously read */etc/hosts* to find an Internet address that corresponds to a host name. When you added a new node to the network, you had to add a new entry to every node's */etc/hosts* file. Now programs can use YP to obtain information from other YP servers.

- Since the YP master server propagates all **maps** (databases) to the **slave servers**, a YP client receives consistent information regardless of which YP server it accesses.
- If a remote node running a YP server process crashes, YP client processes can obtain YP services from another YP server.
- Since the YP interface uses RPC and XDR, the service is available to other vendors.

YP Advantages

YP has several advantages.

- YP enables you to automatically keep user IDs and group IDs consistent among all the nodes participating in NFS file sharing.

  Without YP, you have to manually keep these IDs consistent for NFS.

- YP provides the convenience of centrally administering the */etc* files: *password*, *group*, *hosts*, *netgroup*, *networks*, *rpc*, *services*, and *protocols*.

  Without YP, you must administer these files on each node individually.

YP Disadvantages

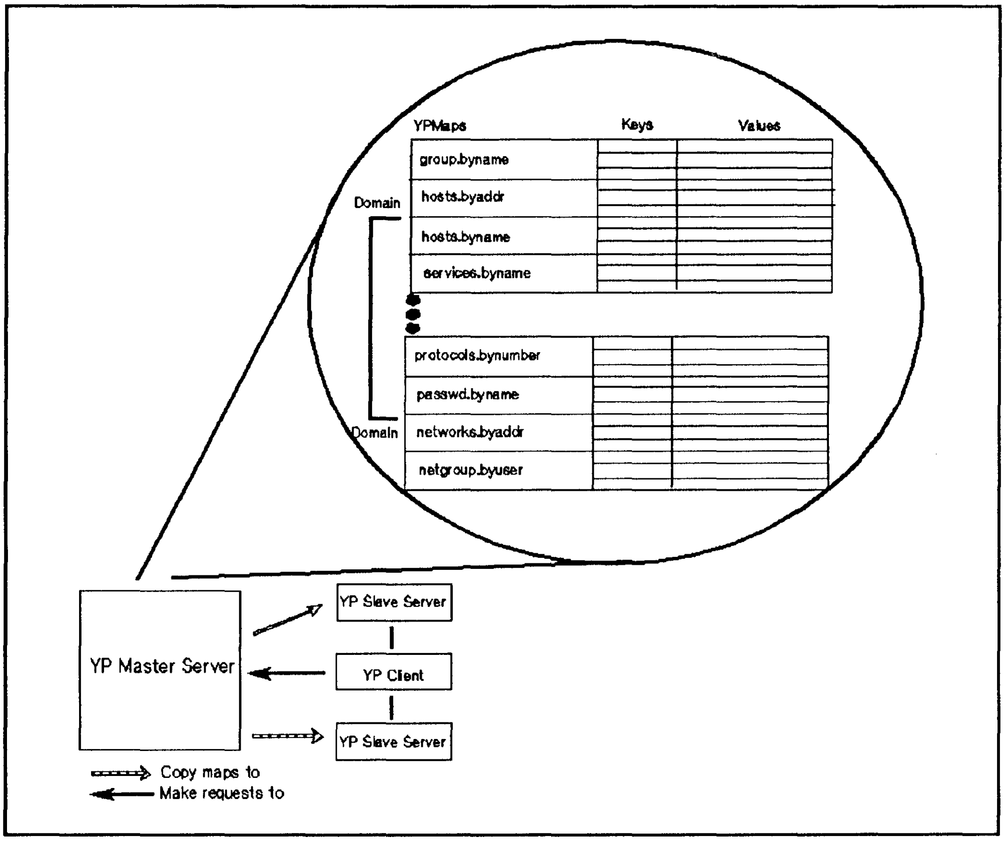
YP has the following disadvantages.

- Since YP provides YP clients access to data via the network, YP clients may observe slower performance than if the data were accessed from local files. For example, with YP, logging in may take more time if the YP server is busy.
- If any of the YP servers are unstable, remote access to files may be slower since the YP client may have to rebind to another YP server. If no other YP server is available, users may not be able to login to their nodes without access to the YP's *passwd* map.
- YP does not make changes visible to all users unless the changes are made on the YP master server.
- The YP slave servers do not immediately see the changes made to the YP master server maps. The updated maps become consistent among all YP servers only after each slave server successfully copies the maps via *ypxfr(1M)*.

Note If you configure the BIND Name Server, it will be used instead of YP for host name and address resolution. YP will still be used for all other information such as passwords. See "Configuring and Maintaining the BIND Name Server" in the *Installing and Administering ARPA Services* manual.

YP Concepts

Refer to the following figure and subsections for a summary of how components within Yellow Pages work together: maps, YP domains, YP servers (masters and slaves), and YP clients.



Yellow Pages Structure

YP Maps

The YP system stores information in YP **maps** (databases). Each map contains a set of keys and associated values: one key per value and one value per key. (A value may be a string of characters with imbedded blanks or tabs). For example, in the *passwd.byname* map, all the login names are the keys and their matching lines from */etc/passwd* are the values.

Each map has a unique **map name** that programs use to access the map. Programs must know the format of the data in the map. Many of the maps are derived from ASCII files such as */etc/hosts*, */etc/group*, and */etc/passwd*. The map format is usually identical to the ASCII file format.

Note If using YP for the first time to provide the information stored in the standard maps' ASCII files, you **must** recompile any applications that read data from those files using standard C library routines.

This recompilation ensures the files can obtain data from the YP maps. If you do not recompile the applications, they will access only the local files. If the local files are not as current as the YP maps, the applications may not work correctly.

YP Servers and YP Clients

YP servers are nodes that provide access to YP maps via the network. These maps are in */usr/etc/yp* subdirectories named after the appropriate YP domains. (See the next section, "YP Domains.")

YP clients are nodes that request access to YP maps from a YP server.

1. A YP client that is not bound sends a broadcast to all YP servers on the network.

2. The YP client **binds** to the first YP server that responds. (Each YP client binds to one YP server per YP domain.)

3. If the request is the YP client's first attempt to access data, the YP client remembers which YP server responded to the request. Subsequent requests by this YP client go directly to this YP server.

4. If the bound YP server is down or unavailable, the YP client automatically rebinds to the first YP server that responds to another broadcast.

Note A YP client can also be configured as any combination of a YP server, NFS client, or NFS server.

A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

YP Domains

A **YP domain** is a logical grouping of the set of maps contained on YP servers. You can have different YP domains for multiple sets of nodes on the LAN without worrying about the maps interfering with each other.

- Each one of the nodes within the same YP domain must have the same domain name.
- Maps using the same name in different YP domains can have different contents.

You implement a YP domain as a subdirectory of */usr/etc/yp* on each YP server; the name of this subdirectory is the name of the YP domain. For example, maps in the *research* YP domain would be in */usr/etc/yp/research*. (Note, YP domain names are case sensitive.)

The */etc/netnfsrc* file usually contains the default YP domain name. You can change the default by executing the *domainname(1)* command or by editing */etc/netnfsrc* and then rebooting the system.

YP Masters and YP Slaves

Only two types of nodes have YP databases: master and slave servers.

The **YP master server** is the node on which YP maps are built from ASCII files; it therefore, contains the master databases (maps) which other YP servers (slaves) copy. Note, the YP master server may also provide YP clients access to YP maps.

Note You should **create and modify YP databases only on the YP master server**; otherwise, all YP databases will not be consistent across the YP servers.

The **YP slave servers** are the nodes that receive the propagated maps from the YP master server. In turn, they provide YP clients access to YP maps.

Though a YP server may be the master for one map and the slave for another, random assignment of maps to YP master servers may cause confusion. Therefore, only one YP server should be the master for all maps within a YP domain.

Virtual Home Environment (VHE)

Virtual Home Environment (VHE) is an HP-developed service that allows you to configure your login environment on remote nodes to mirror the login environment on your home node. (Home node refers to the node on which your home directory physically resides.) VHE is an optional service that is available to any HP-UX system that has the NFS product. It may also be used with other UNIX systems that support symbolic links and NFS.

If you find that you never need to work from a remote node, you may want to skip this section.

VHE Advantages

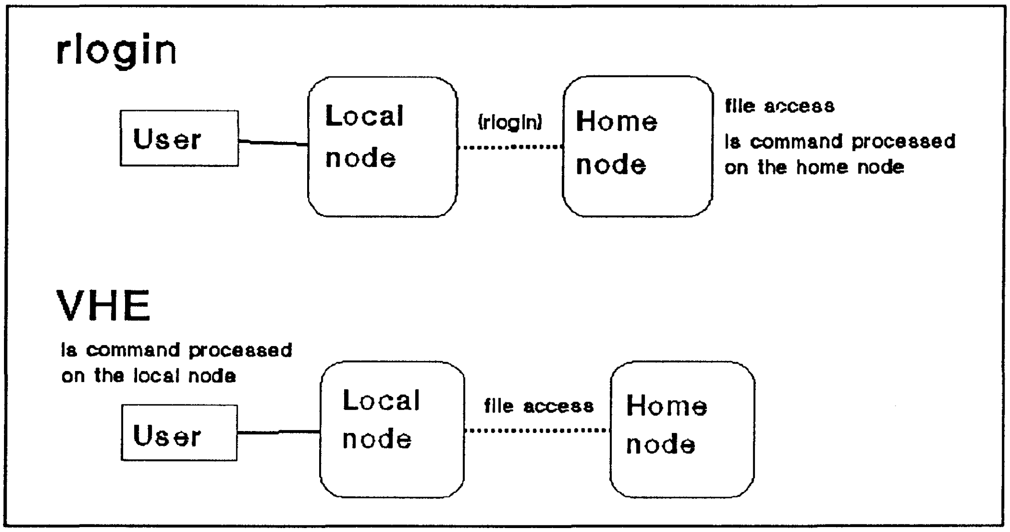
VHE's major advantage is that you can sit down at any remote node (assuming you have login permission), login, and enter into the work environment that is associated with the login on your home node (your home directory as specified in */etc/passwd*). This includes:

- home shell configuration (i.e., whichever shell you are configured to use on your home node appears when you login to a remote node).
- access to files on the file systems exported for VHE on any computers connected with VHE on the network to which you have a login and file access permission.
- use of previously defined aliases (only for C or K shells) and shell variables.
- use of customized shell scripts (assuming shells operate similarly on your home node and the node you are currently using).
- use of compiled files under your home directory from your home node (assuming your home node and the node you are logged into are of the same architecture and operating system).

Thus, VHE allows you to minimize the number of computer interfaces you must learn to be productive on the various computers that are running NFS on your network and **you are no longer tied to a particular computer to complete your work tasks.**

Another advantage of VHE is that it distributes computational work more efficiently between nodes than ARPA/Berkeley terminal emulation services such as *telnet* or *rlogin*. Unlike *telnet* or *rlogin*, VHE does not return to your home node, that contains your home environment login, to execute tasks.

Instead, VHE takes advantage of the computing capacity of the machine you are currently using. For example, if you use VHE on a node other than the home node and perform an *ls* command of a directory on the home node, the *ls* command is executed from the **local** */bin* directory. VHE does not return to your home node's */bin* directory to execute the *ls* command. The following figure illustrates this concept.



Comparison: VHE and rlogin Performing an ls command

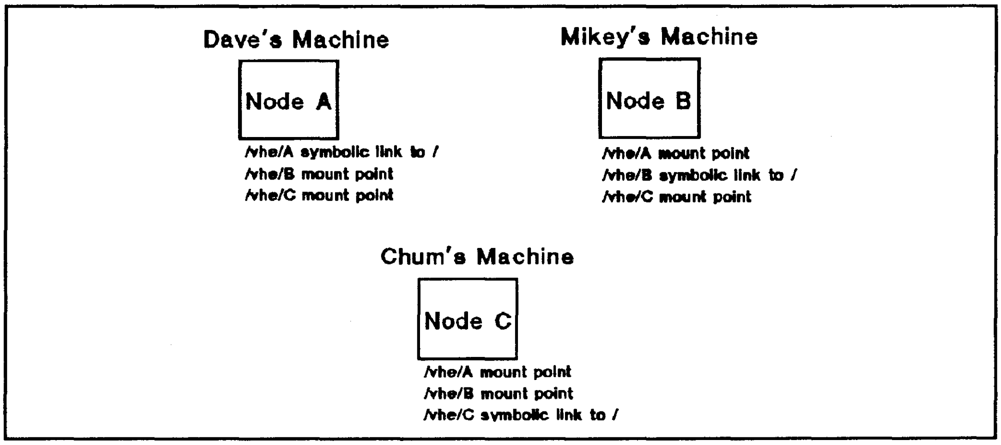
VHE Disadvantages

VHE has the following disadvantages.

- Though you can edit source code files originating from different types of computers on the network, you will not be able to **execute** object code files from a computer of a different architecture using VHE. For example, consider the following: you are currently working on an HP 9000 Series 300 and running VHE, and your home node is an HP 9000 Series 800 computer. If you try to execute an object code file on the HP 9000 Series 300 from the Series 800 computer it will not succeed. However, you can execute a script from the Series 800 computer.
- If you specify pathnames or hardware attributes in your host's *.profile* or *.login* files, you may have to modify these files to use VHE effectively. For example, the *.login* file needs to prompt for the terminal type if you plan to use VHE from more than one terminal or display type. If you do not already have this capability, then look in the sample */etc/d.login* or */etc/d.profile* files for samples of how to do this.

How VHE Works

The following diagram illustrates the directory structure of nodes in a network using VHE.



Directory Structures of Nodes Using VHE

Each node is connected to the others via NFS Services. In the picture, each node is a home node for a different user (Dave, Mikey and Chum). Each user has a customized work environment set up by the login process. Directories on each home node correspond to each of the remote nodes. For example, on node A there is a directory */vhe/B* that corresponds to node B. Using these directories as mount points, a mount is done by each node to each remote node. (The definitions of mounts and mount points are included in the "Glossary." More detailed information is contained in the "NFS Configuration and Maintenance" chapter in the *Installing and Administering NFS Services* manual).

Using VHE gives each node access to file systems located on the remote nodes. To maintain consistency when an individual is logged in to his or her home node, a symbolic link (a pointer) points to the host's root directory.

In a single node HP-UX configuration, the */etc/passwd* file contains the directory that becomes the home directory for the user upon logging in. For use with VHE, */etc/passwd* is edited such that all of the home directories are prefixed with a mount point or a symbolic link. When the login program performs a *cd* to the user's home directory, the *cd* and subsequent requests are made to the user's home node via NFS Services unless logging in on your home node.

Example Grouping

In the */etc/passwd* file, the appropriate mount point or symbolic link is added to the beginning of the pathname of the home directory for each user. The example below shows how the lines in */etc/passwd* would look for the users Dave, Mikey and Chum as shown above.

```
dave::117:100:Dave:/vhe/A/users/dave:/bin/csh
mikey::118:100:mikey Pom :/vhe/B/users/mikey:/bin/sh
chum::119:200:chum Pom:/vhe/C/users/chum:/bin/ksh
```

No matter which node Dave logs in on, his home directory is */users/dave* on node A. When scripts such as *.login* or *.cshrc* are executed, they define the execution environment as customized by Dave. His files, shell variables and aliases are available just as if he had physically logged in on node A.

Because VHE is not a virtual terminal program, when Dave executes processes, they are executed on the node he is logged into. If he is on node B, processes are executed on node B, not his native host A. For example, consider the following. Dave is working at node B and his system administrator has configured VHE to be running. Dave enters the following command on node B:

```
cc testfile.c
```

The *cc* from node B's */bin* directory is executed, but *testfile.c* is used from Dave's current working directory on node A.

3

Common Commands

This chapter describes how to access files using NFS. It also explains how to use common NFS and Yellow Pages (YP) commands.

Note All references to **servers** and **clients** apply to NFS servers and clients unless preceded by **YP**.

Key Terms

Term	Definition
Client	■ A node that requests data or services from other nodes (servers). ■ A process that requests other processes to perform operations. **Note:** An NFS client can also be configured as any combination of an NFS server, YP client, or YP server. (A YP server **must** also be configured as a YP client.)
Export	To make a file system available to remote nodes via NFS.
File System	A directory structure used to organize files.
Host	A node that has primary functions other than switching data for the network.
Internet Address	A four-byte quantity that is distinct from a link-level address and is the network address of a computer node. This address identifies both the specific network and the specific host on the network.
Key (YP)	A string of characters (no imbedded blanks or tabs) that indexes the values within a map so the system can easily retrieve information. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.
Map (YP)	A file consisting of logical records; a search key and related value form each record. YP clients can request the value associated with any key within a map. **YP map** is synonymous with **YP database.**
Map Nickname (YP)	A synonym for the YP map name when using certain YP commands.

Term	Definition
Master Server (YP)	The node on which one or more YP maps are constructed from ASCII files. These maps are then copied to the YP slave servers for the YP clients to access.
Mount	To obtain access to a remote or local file system or directory (import).
Mount Point	The name of the directory on which a file system or part of a file system is mounted.
Node	A computer system that is attached to or is part of a computer network.
Server	■ A node that provides data or services to other nodes (clients) on the network. ■ A process that performs operations as requested by other processes. **Note:** An NFS server can also be configured as any combination of an NFS client, YP client, or YP server. (A YP server **must** also be configured as a YP client.)
Value (YP)	A unit of information stored in YP maps; each value has a corresponding key (index) so the system can easily retrieve it. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.
Yellow Pages (YP)	An optional network service composed of databases (maps) and processes that provide YP clients access to the maps. The YP service enables you to administer these databases from one node. YP may or may not be active; check with your system administrator.

Term	Definition
YP Client	■ A node that requests data or services from YP servers. ■ A YP process that requests other YP processes to perform operations. **Note:** A YP client can also be configured as any combination of a YP server, NFS client, or NFS server. (A YP server **must** also be configured as a YP client.)
YP Database	See "Map (YP)."
YP Domain	A logical grouping of YP maps (databases) stored in one location. YP domains are specific to the YP network service and are not associated with other network domains.
YP Map	See "Map (YP)."
YP Password	The password for a user's login ID that exists in the YP *passwd* map. The password is the same one as the user password, but is administered through the YP. You do not have to have a password to access the YP databases.
YP Server	■ A node that provides data (maps) or services to other nodes (YP clients) on the network using YP. ■ A YP process that performs operations as requested by other YP processes. **Note:** A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

NFS Commands

Use this section to understand how to access files via NFS and how to use the following NFS commands. Refer to the *NFS Services Reference Pages* for detailed explanations of the commands and all of their options.

- *rpcinfo(1M)* see page 8
- *rup(1)* see page 10
- *rusers(1)* see page 10
- *showmount(1M)* see page 12
- *on(1)* see page 13

NFS Remote File Access

NFS allows many users to share the same files. Since access techniques are transparent, remote file access remains similar to local file access.

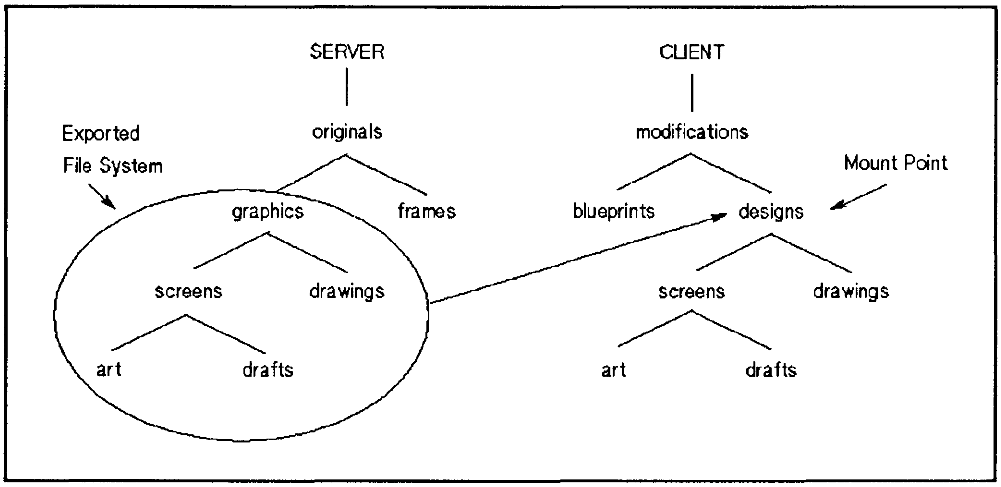
The super-user must perform two actions before you can access remote files via NFS.

- On the server, export the file system (i.e., make it available) to the client
- On the client, mount (import) the file system

Access to remote files is the same as for local files. You need to include either the complete path name starting with / (slash) or the path name relative to the current directory.

Note **If operating in an HP-UX cluster environment** and accessing a *CDF* (context dependent file) via an NFS mount, the *CDF* member is chosen based on the context of the NFS server, not the accessing node. Since this access method may return unexpected results, HP recommends you do **not** use *CDF*s **with NFS.**

EXAMPLE:



Example NFS Remote File Access	Example Entry
Edit the *drafts* file on the **server.**	`server%  vi /originals/graphics/screens/drafts`
Edit the *drafts* file on the **client.**	`client%  vi /modifications/designs/screens/drafts`
While in the *frames* directory, the **server** copies the *art* file from the *screens* directory.	`server%  cp /originals/graphics/screens/art art`
While in the *blueprints* directory, the **client** copies the *art* file from the *screens* directory.	`client%  cp /modifications/designs/screens/art art`
While in the *screens* directory, the **server** copies the *art* file to the *frames* directory.	`server%  cp art /originals/frames`
While in the *screens* directory, the **client** copies the *art* file to the *blueprints* directory.	`client%  cp art /modifications/blueprints`

rpcinfo

Execute *rpcinfo(1M)* to determine which remote programs are registered with a system's *portmap(1M)* daemon.

By providing a host name, you can list the registered RPC programs on a specific host. If you do not specify a host name, *rpcinfo(1M)* defaults to the local host.

To list the program, version, port numbers, and the protocol, use the *-p* option.

EXAMPLE: **Execute:** `rpcinfo -p` **node_7**

System Response:

```
program vers proto  port
   100003   2   udp   2049  nfs
   100004   2   udp   1028  ypserv
   100004   2   tcp   1027  ypserv
   100004   1   tcp   1027  ypserv
   100007   2   tcp   1028  ypbind
   100007   1   udp   1037  ypbind
   100001   3   udp   1069  rstatd
   100002   1   udp   1073  rusersd
   100002   2   udp   1073  rusersd
   100005   1   udp   1076  mountd
   100008   1   udp   1078  walld
   100012   1   udp   1080  sprayd
```

To see if a particular remote program and version is available using UDP, use the *-u* option.

EXAMPLE: **Execute:** `rpcinfo -u` **node_2 mountd 1**

System Response: This response indicates that the *portmap(1M)* daemon on node_2 knows about program #100005 and that it is available.

```
program 100005 version 1 ready and waiting
```

rup

Execute *rup(1)* to list host information, including how long they have been running, how many users are logged on to them, and their load average. By providing a host name, you can list information about a specific host.

Executing *rup(1)* without providing a host name causes an RPC broadcast. The local node collects responses until the RPC times out (quits). This process generally takes about two minutes.

EXAMPLE: **Execute:** `rup` *node_1 node_2 node_3 node_4*

System Response: The last three columns of this response show the load averages for 1, 5, and 15 minute intervals.

```
node_1   up            15:53,     load average: 0.11, 0.17, 0.15
node_2   up  2 days,   19:42,     load average: 0.00, 0.01, 0.01
node_3   up  21 days,  11:34,     load average: 1.66, 1.68, 1.60
node_4   up            19:24,     load average: 0.14, 0.18, 0.14
```

To sort the display by "up time," use the *-t* option.

EXAMPLE: **Execute:** `rup -t`

System Response:

```
collecting responses...
node_7       up  21 days,  11:28,   load average: 1.16, 1.42, 1.52
11.2.33.44   up 12 days.   22:15,   load average: 1.08, 0.82, 0.57
node_8       up 7 days,    18:27,   load average: 0.12, 0.09, 0.09
node_12      up 6 days,    21:20,   load average: 0.10, 0.08, 0.09
55.6.77.88   up 3 days,    3 mins,  load average: 0.00, 0.01, 0.01
node_6       up 2 days,    22:49,   load average: 0.00, 0.00, 0.02
99.0.11.22   up            18:14,   load average: 0.00, 0.00, 0.05
33.4.55.66   up            0 min    load average: 0.14, 0.04, 0.02
```

rusers

Execute *rusers(1)* to list the host names and users logged in for all remote nodes. By providing a host name, you can list information about a specific remote node.

Executing *rusers(1)* without providing a host name causes an RPC broadcast. The local node collects responses until the RPC times out (quits). This process generally takes about two minutes.

EXAMPLE: **Execute:** `rusers`

System Response: This response displays the host name or internet address in the first column and the users in the second column.

```
77.8.99.00    root
node_6        user_4 user_3 user_8 user_11
node_3        u_2
node_1        u_7
11.2.33.44    root root
node_2        root u_5 root
node_16       test_user
node_9        rootx root u_7 root
node_7        root
```

You can list more extensive information by using the *-l* command: user, host, tty (terminal), login date and time, and idle time (in minutes and seconds).

EXAMPLE: **Execute:** `rusers -l` **node_8 node_4**

System Response: The last two columns in this response show the login date and time followed by the idle time.

```
rootx    node_8:console     Apr 07 14:00     20:29
user_3   node_4:tty03       Apr 12 08:09     :23
```

showmount

Execute *showmount(1M)* to list all the clients that have remotely mounted a file system. By providing a host name, you can specify the host. If you do not specify a host name, *showmount(1M)* defaults to the local host. For example, you might want to determine which nodes have your file systems mounted.

To print all remote mounts in a *client:directory* format, use the *-a* option. The directory listed is the root of the file system that was mounted.

EXAMPLE: **Execute:** `showmount -a`

System Response: This response displays the client followed by the directory.

```
node_4:/tmp
node_7:/
node_2:/tmp
node_12:/usr/tmp/sys_rick
node_6:/tmp/y
node_8:/
```

To print a list of exported file systems, use the *-e* option.

EXAMPLE: **Execute:** `showmount -e` **node_7**

System Response:

```
export list for node_7:
/             node_31 node_32 node_11 node_6
/users/proj   node_8 node_12
```

on

Use the *on* command to execute commands on a remote host. When executing the *on* command, you specify:

- a host on which to run the remote command
- the command to run
- arguments for the command

The *on* command then simulates your current environment on the server by passing your environment variables and information about your current working directory to the remote host. The *rexd* daemon on the server mounts the file system that contains your current working directory if it is not already mounted on the server. After the environment is simulated, the command executes in the simulated environment on the remote host.

Note Your environment is simulated on the remote host but not completely recreated. Execution of a given command on a remote host will not always produce the same results as the executing the command on your local computer. The simulated environment and the environment's limitations are discussed in the "Environment Simulation" in the *Installing and Administering NFS Services* manual.

The syntax of the *on* command is as follows:

```
on [-i | -n] [-d] host [ command [argument ] ....]
```

Host specifies the name of the host on which to execute *command.* There must be an entry for *host* in the local computer's host data base.

Command specifies the command to execute on *host.* If *command* is not specified, *on* will start a shell on *host.*

You may specify three options (*-i*, *-n*, *-d*). The *-i* option must be used when invoking interactive commands, the *-n* option must be used when running commands in the background with job control, and the *-d* option is used when you wish to receive diagnostic messages.

Use of the *-d* option with either *-i* or *-n* is permitted.

EXAMPLE:

```
on -i -d host
```

or

```
on -n -d host
```

You cannot use the *-i* and *-n* options at the same time.

EXAMPLE:

```
on -i -n host
```

is not permitted.

The -i Option (Interactive Mode)

The *-i* option invokes the interactive mode. This option must be specified for all interactive commands (commands which expect to be communicating with a terminal). Examples of interactive commands are *vi(1)*, *csh(1)*, and *more(1)*. If this option is specified with a non-interactive command such as *sort(1)*, it will be executed as an interactive command, but there may be no difference in behavior.

EXAMPLE:

```
on -i node_7 vi file
```

The -n Option (No Input Mode)

The *-n* option sends the remote program an end-of-file when the program reads from standard input instead of connecting the standard input (*stdin*) of the *on* command to the standard input (*stdin*) of the remote command. The *-n* option is necessary when running commands in the background with job control.

The -d Option (Debug Mode)

The *-d* option allows you to receive diagnostic messages during the start up of the *on* command. The messages may be useful in detecting configuration problems if the *on* command is failing while connecting to a given host.

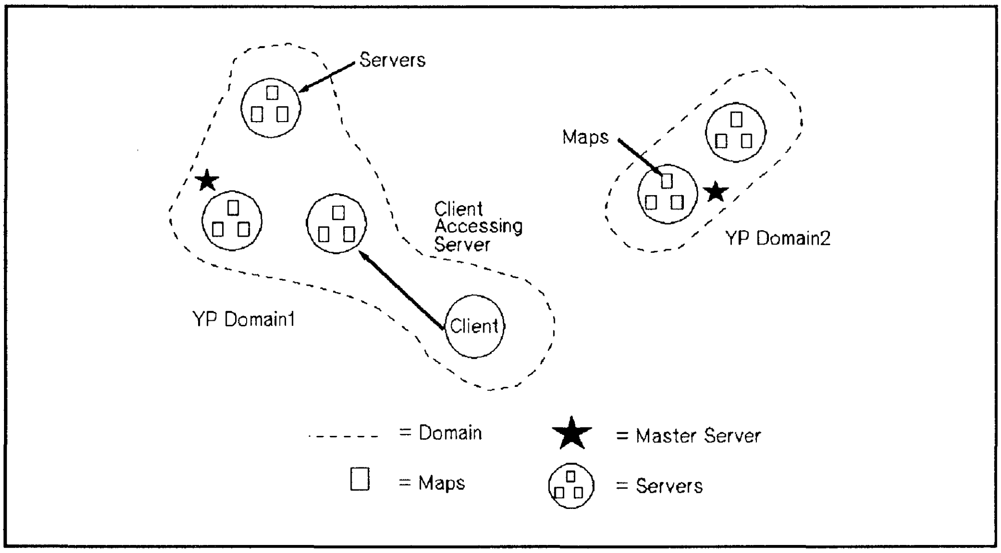
Yellow Pages Overview

The Yellow Pages (YP) is an **optional** network database service that enables YP clients to access information from any correctly configured YP server on the network. One YP master server can automatically propagate modifications across the network.

YP Maps

The YP system stores information in YP **maps** (databases) that are consistent across the nodes. Each map has a unique, case-sensitive map name that is used for accessing maps.

Each map consists of **keys** (for indexing) and **values** (data). You can use YP commands for querying for values associated with a particular key within a map and for retrieving key-value pairs within a map.



Yellow Pages Overview

YP Servers and YP Clients

YP servers store and provide access to the YP maps (databases); **YP clients** request data from the maps residing on YP servers. Since different YP servers have consistent YP maps, responses are identical no matter which YP server answers a request.

- A YP client can also be configured as any combination of a YP server, NFS client, or NFS server.
- A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

YP Domains

A **YP domain** is a logical grouping of YP maps; each YP server contains a set of maps for at least one YP domain.

YP domains enable maps with the same names to exist on one LAN; the maps are made unique by belonging to separate YP domains. With YP domains, you need not worry about the maps interfering with each other.

- Each of the nodes within a YP domain has the same YP domain name.
- Maps using the same name in different YP domains may have different contents.

YP domains are implemented as subdirectories of */usr/etc/yp* on the YP servers only; the name of each subdirectory is the name of a YP domain. For example, maps in the *research* YP domain would be in */usr/etc/yp/research*. (Note, YP domain names are case sensitive.)

YP Master and Slave Servers

Only two types of nodes have YP databases: master and slave servers.

The YP **master server** is the node on which all YP maps within a particular YP domain are created and modified. As modifications occur, the YP **slave servers** copy the maps to ensure all YP databases are alike; in turn, they provide resources to the YP clients.

Note, YP clients can bind to both YP master and slave servers.

YP Commands

Since YP hides the details of how and where data is stored, you do not need to know all the configuration details to access information. You can, however, use the following commands to determine the location and content of YP information.

- *domainname(1)*
- *ypcat(1)*
- *ypmatch(1)*
- *yppasswd(1)*
- *ypwhich(1)*

domainname

Execute *domainname(1)* to display the current YP domain name.

```
domainname
```

For example, you might need to determine the current YP domain name to define a netgroup in */etc/netgroup*. (Netgroups are network-wide groups of nodes and users defined in */etc/netgroup* on the master server.)

ypcat

Execute *ypcat(1)* to list the contents of a specified YP map. You can use either the map name or map nickname to specify the desired map.

EXAMPLE: **Execute:** `ypcat group.byname` **or** `ypcat group`

System Response: This response displays the group name, the group ID (GID), and the members of the group.

```
daemon::5:notes,anon,uucp
users::23:window,nowindow
other::1:root,daemon,uucp,who,date,dooley,sync
root::0:root
mail::6:root
sys::3:root,bin,sys,adm
rje::8:rje,shqer
bin::2:root,bin,daemon,lp
adm::4:root,adm,daemon
```

To list the map nicknames applicable to the *ypcat(1)* command, use the *-x* option.

EXAMPLE: **Execute:** `ypcat -x`

System Response:

```
Use   passwd for  map passwd.byname
Use   group for  map group.byname
Use   networks for  map networks.byaddr
Use   hosts for  map hosts.byaddr
Use   protocols for  map protocols.bynumber
Use   services for  map services.byname
Use   aliases fomap mail.aliases
Use   ethers fomap ethers.byname
```

ypmatch

Execute *ypmatch(1)* to print the data (values) associated with one or more keys in a specified YP map. You can use either the map name or map nickname to specify the desired map.

To list the map nicknames applicable to the *ypmatch(1)* command, use the *-x* option.

EXAMPLE: **Execute:** `ypmatch` **my_node** hosts.byname

System Response: This response displays the internet address (value) associated with the *hosts.byname* map for the node *my_node*.

```
11.2.33.44    my_node
```

yppasswd

The YP password is the password for a user's login ID that exists in the YP *passwd* map. It is used as the user password, but is administered through YP. Note, you are not required to have a password to access the YP databases.

If you change your password with the *passwd(1)* command, you will change only the entry in your local */etc/passwd* file if the entry exists. If your password is not in the file, the following error message occurs when using *passwd(1)*.

```
Permission denied.
```

If this error occurs, execute *yppasswd(1)*.

YP Password Guidelines

Execute *yppasswd(1)* to change or install a password associated with a specified login name in the YP *passwd* map.

The following list provides the requirements for creating and changing YP passwords. Note, these guidelines are different from those of *passwd(1)*. (Refer to the "HP NFS Services vs. Local HP-UX" appendix in this manual.)

- Only the owner or super-user can change a YP password. The super-user must know the current YP password to change another user's YP password.
- Only the first eight characters of the YP password are significant; the rest are truncated.
- A YP password must contain at least five characters if it includes a combination of either
  - uppercase and lowercase letters or
  - numeric and special characters
- A YP password must contain at least four characters if it includes a combination of uppercase letters, lowercase letters, and numeric characters.
- A YP password must contain at least six characters if it includes only monocase letters.

YP Password

Use the following steps to create or change your YP password in the YP *passwd* map.

1. Execute the *yppasswd(1M)* command.

   ```
   yppasswd user_login_name
   ```

2. The system prompts you for the old YP password even if one does not exist. If it exists, enter the old YP password; otherwise, press **RETURN**.

   Note, the YP password may be different from the one in your local */etc/passwd* file.

3. The system prompts you for the new YP password twice to ensure you enter the correct response. Enter your new YP password twice, pressing **RETURN** after each entry.

   The system now updates the master server *passwd* map.

EXAMPLE: **Execute:** `yppasswd`

System Response:

```
Old YP password:
New password:
Retype new password:
The YP passwd has been changed on host_name, the master YP passwd
server.
```

ypwhich

Execute *ypwhich(1)* to print the host name of the YP server supplying YP services to a YP client.

To list all available maps and their YP master server host names, use the *-m* option. You can use either the map name or map nickname to determine which YP server is the master server for a specified YP map.

To list the map nicknames applicable to the *ypwhich(1)* command, use the *-x* option.

EXAMPLE: **Execute:** `ypwhich -m`

System Response: This response displays the available maps and their YP master server host names.

```
services.byname         node_1
rpc.bynumber            node_1
protocols.bynumber      node_1
protocols.byname        node_1
passwd.byuid            node_1
passwd.byname           node_1
networks.byname         node_1
networks.byaddr         node_1
netgroup.byuser         node_1
netgroup.byhost         node_1
netgroup                node_1
hosts.byname            node_1
hosts.byaddr            node_1
group.byname            node_1
group.bygid             node_1
vhe_list                node_1
ypservers               node_1
```

A

HP NFS Services vs. Local HP-UX

If you have applications running on HP-UX, they may behave differently over NFS Services. Use this appendix to understand the basic differences between NFS Services and local HP-UX operations.

HP NFS Services Networking Operation	Local HP-UX Operation
Append Mode	
If two processes operating on different clients open the same file using *O_APPEND*, the *write* operation may not append data to the file.	If two processes open the same file using *O_APPEND*, the *write* operation should append information to the file.
chacl(1)	
You can only use the *-F* option. The other options of *chacl(1)* are not supported over NFS.	You can use all options locally.
Device Files	
NFS does not support remote access to device files, but does support local access to device files via NFS.	HP-UX supports local access to device files.
File Locking	
NFS supports remote file locking for NFS reads and writes in advisory mode only.	HP-UX supports local file locking in advisory and enforcement modes.
getacl(2) system call	
Is not supported over NFS.	Is supported locally.

HP NFS Services Networking Operation	Local HP-UX Operation
Group Membership	
A user may be a member of eight groups. If a user who is a member of more than eight groups attempts to access a file, the system accesses only the first eight groups for permission checking.	A user may be a member of up to 20 groups.
Long File Names	
NFS can access file names up to 255 characters in length if the remote NFS file system supports them. **Note**: The local file system truncates long file names to 14 characters, conflicting file names might occur when accessing a file system that supports long file names.	The local HP-UX file system supports both short and long file name file systems. On short file name systems, HP-UX silently truncates file names that are longer than 14 characters in length.
mknod(1M) Command	
The *mknod* command will work only with named pipes over NFS.	You can use the *mknod* command locally for all file types.
Named Pipes	
NFS named pipes cannot be used to communicate between machines in the same diskless cluster.	Named pipes can be used to communicate among clients in a diskless cluster.
Reading Directories	
You cannot use the *read(2)* call to read a remote directory, rather you should use *readdir(3)*.	You can use the *read(2)* call to read a local directory. However, to do so can restrict migration of programs to future HP-UX versions.
RFA	
Using a network special file on an NFS file system is not supported.	Is supported locally.
setacl(2) system call	
Is not supported over NFS.	Is supported locally.

HP NFS Services Networking Operation	Local HP-UX Operation
setaclentry(3) library routine	
Is not supported over NFS.	Is supported locally.
Super-user Permission	
The super-user UID *0* is mapped to *-2* by default. Anything requiring super-user permission may not work over NFS. For example, a super-user may not be able to perform the following tasks. ■ Link and unlink directories ■ Alter directories such as */*, */etc*, and */bin* ■ Use chmod to set sticky or setuid bits ■ *mknod* of device files	Super-user has permission to perform any operation locally (by definition).
System Time	
Commands that access clocks on different systems may not provide consistent times since system clocks differ. For example, if you give the *utime(2)* command a *NULL* pointer for the times value, the following process occurs: 1. The system sets the access time and modification time according to the client node clock.	Commands that access clocks on the local system provide consistent times.

HP NFS Services Networking Operation	Local HP-UX Operation
System Time *continued...*	
2. It then sends these times over to the server which changes the inode to reflect the new access and modification times. 3. The server node identifies the change in the inode and thus, modifies the inode's status change time according to its own clock. The result is a high probability of differing times between the server's access and modification times versus its status change time. **Note:** If operating in an HP-UX cluster environment, all nodes in the cluster have the same time as the root server's clock. Therefore, clock skew problems exist only if the root server's clock is different from other NFS servers.	
Unlinking	
The server does not keep state information and does not know if a process has a file open. ■ The server will unlink a file if it receives a request to do so; thus, subsequent requests for the file will result in an error.	If you open a local file and unlink it before you close the file, the file descriptor for the open file will still be valid to access the file.

HP NFS Services Networking Operation	Local HP-UX Operation
Unlinking *continued...*	
■ If a process opens a file and then unlinks it, the client renames the file so it appears to be gone. When the process quits, the client then unlinks the renamed file. ■ If the unlink request comes from a different node than from where the open request came from, the file is deleted.	
yppasswd(1) Command	**passwd(1) Command**
This command does not have a **password aging** feature. The super-user must know the current password to change another user's password. ■ A password must contain at least five characters if it includes a combination of either: uppercase and lowercase letters, numeric or special characters. ■ A password must contain at least four characters if it includes a combination of uppercase letters, lowercase letters, and numeric characters. ■ A password must contain at least six characters if it includes only monocase letters.	This command has a **password aging** feature. Super-user does not have to know the password to change another user's password. ■ Each password must have six or more characters: at least two alpha characters and at least one numeric or special character. ■ Each password must differ from the user's login name and any reverse or circular shift of that name. ■ New passwords must differ from the old by at least three characters.

HP NFS Services Networking Operation	Local HP-UX Operation
pathconf/fpathconf	
The following variables for the *pathconf/fpathconf* system calls are not supported over NFS:	All variables are supported locally for the *pathconf/fpathconf* system calls:
_PC_CHOWN_RESTRICTED variable	_PC_CHOWN_RESTRICTED variable
_PC_LINK_MAX variable	_PC_LINK_MAX variable
_PC_NAME_MAX variable	_PC_NAME_MAX variable
_PC_NO_TRUNC variable	_PC_NO_TRUNC variable
_PC_PATH_MAX variable	_PC_PATH_MAX variable
The following variables for the *pathconf/fpathconf* system calls return local information over NFS.	
_PC_MAX_CANON variable	_PC_MAX_CANON variable
_PC_MAX_INPUT variable	_PC_MAX_INPUT variable
_PC_VDISABLE variable	_PC_VDISABLE variable
The following variable for the *pathconf/fpathconf* systems calls is supported over NFS:	
_PC_PIPE_BUF variable	_PC_PIPE_BUF variable

B

Migrating from RFA to NFS

When using networks consisting of all HP systems, the Remote File Access (RFA) service provides distributed file access among Series 300 and 800 computers.

Use this appendix if you wish to translate your RFA applications to NFS applications.

Why Migrate to NFS Services?

Using NFS Services has several advantages.

- NFS works with other vendors' equipment and other operating systems.
- NFS is a defacto industry standard.
- NFS allows transparent file access.
- NFS with YP provides centrally administered databases.

Similarities

HP NFS and RFA have the following similarities.

- No remote device access
- Not all UNIX[1] semantics are fully supported

Differences

Refer to the following table for a list of differences between HP NFS and RFA.

NFS Services	RFA
You can run *setuid* programs accessing data on remote file systems.	You cannot run *setuid* programs accessing data on remote file systems.
NFS operates in a heterogeneous operating system environment.	RFA operates on HP-UX operating systems only.
Only the super-user can perform remote NFS mounts.	All users can establish access to remote file systems.
You can centrally administer your databases using YP service.	You have no centrally administered database.
All users with read access to the mount point can read the remote file system.	Only users performing *netunam* can access the remote file systems.
Read and write file caching occurs on the clients; read caching occurs on the servers.	Read and write file caching occurs on the servers; caching does not occur on the clients.
The servers are stateless (do not remember client activities) and therefore, can be rebooted without interfering with client activities. (The client can resume access to the server when it is rebooted.)	The servers have state and therefore, remember the activities in which the client is involved.

(1) UNIX (R) is a U.S. registered trademark of AT&T in the U.S.A. and other countries.

NFS Services	RFA
One mount gives you access to only one file system.	One *netunam* gives you access to all file systems under the root directory.

Changing Scripts from RFA to NFS

Changing RFA scripts to NFS requires only minor changes. You can change both shell scripts that accept different path names and those that use hard-coded path names.

Shell Scripts that Accept Different Paths

Shell scripts that accept different paths require only minor modifications.

- You must perform a remote mount of a file system or directory either
  - as part of the script or
  - before executing the script.

  Since super-user must execute mounts, the script must be *setuid root* if the mount is performed as part of the script.

Note Having setuid root scripts is a potential security problem.

  If the script's owner does not have super-user permissions, the super-user can configure */etc/checklist* to automatically mount the remote file systems at boot time. This process allows users to execute scripts without checking to see if the remote file system is accessible.

- If RFA is **not** being used, remove any calls to *netunam* from the script. Removing these calls prevents *netunam* failures from causing the scripts to fail.

Shell Scripts with Hard-coded Paths

You can handle shell scripts with hard-coded path names in two ways.

- Change the path name in the script to correspond to the NFS mount point.
- Create a path name for the NFS mount point which corresponds to the path name in the script.

To mount the remote file system either as part of the script or automatically via */etc/checklist*, you must modify the shell scripts as described above in "Shell Scripts that Accept Different Paths."

Change Pathnames

Change the path name in the script to correspond to the NFS mount point.

EXAMPLE: The script has a hard-coded path name of */net/systemB/project*, and you want to mount the remote directory */project* on */user/project* as follows.

```
mount systemB:/project /user/project
```

Now change the script to use the path name */user/project* in place of */net/systemB/project*.

Create New Pathnames

Create a path name for the NFS mount point that corresponds to the path name in the script.

EXAMPLE: The script has a hard-coded path name of */net/systemB/project* which accesses the remote directory */project*.

1. Remove the network special file */net/systemB*.
2. Create the directories */net/systemB* and */net/systemB/project*.

```
mount systemB:/project /net/systemB/project
```

The path name, therefore, remains the same.

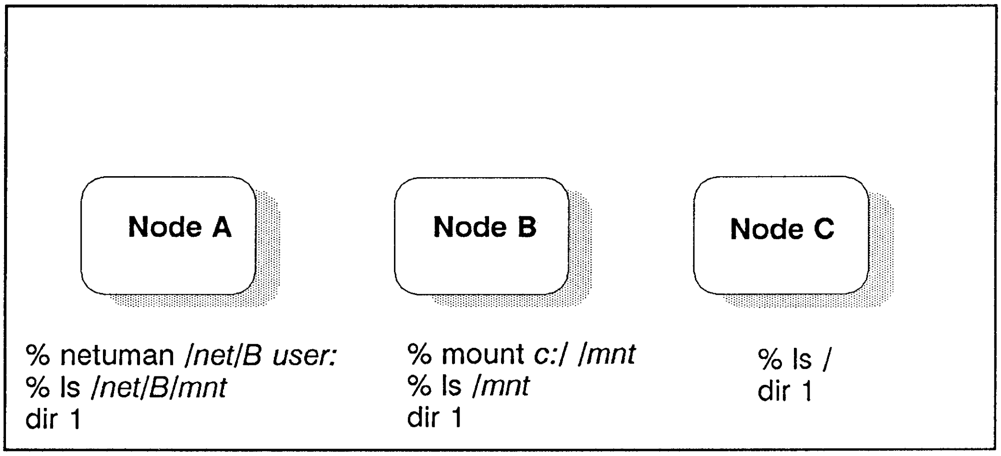
Note For RFA, access to the remote system is via a network special file. Creating an NFS mount point with the same name as the network special file for the remote system could cause confusion. Problems will not occur if

- the system does not use RFA and if
- you remove the network special file.

All remote access will then be via mount points that have the same names as the network special files that were removed.

RFA through NFS

RFA functions are operational through NFS. For example, an RFA node can access a remote directory by going through an NFS client. Note, however, **NFS functions cannot operate through RFA** because NFS parses the path names differently than RFA. Therefore, an NFS node cannot access a remote node by going through RFA.



```
% netuman /net/B user:     % mount c:/ /mnt      % ls /
% ls /net/B/mnt            % ls /mnt             dir 1
dir 1                      dir 1
```

EXAMPLE: RFA remote mount through an NFS client

1. Node B performs an NFS mount to Node C.
2. Node A then performs a netunam to Node B.
3. Node A lists (ls) the contents of Node C's root directory.

Glossary

A

Alias	A term for referencing alternate networks, hosts, and protocols names.
ARPA	Advanced Research Projects Agency A U.S. government agency that was instrumental in developing and using the original ARPA Services networking standards.

B

Bind	■ Process by which a client locates and directs all requests for data to a specific server. ■ Process of establishing the address of a socket that allows other sockets to connect to it or to send data to it.

C

CDF	Context Dependent File. A hidden directory that contains all the versions of a file needed by the different cnodes.

Client	■ A node that requests data or services from other nodes (servers). ■ A process that requests other processes to perform operations. **Note:** An NFS client can also be configured as any combination of an NFS server, YP client, or YP server. (A YP server **must** also be configured as a YP client.)
Clock Skew	A difference in clock times between systems.
Cluster	One or more workstations linked together with a local area network (LAN), but consisting of only one file system.
Cnode	Any node operating in an HP-UX cluster environment, including diskless nodes and the root server.

D

Daemon	Background programs that are always running, waiting for a request to perform a task.
Diskless Cnode	A node in an HP-UX cluster that uses networking capabilities to share file systems, but does not have a file system directly attached.

E

Escape Sequence (YP)	Characters used within files to force inclusion and exclusion of data from YP databases. The escape sequences are as follows. ■ + (plus) ■ - (minus) ■ +@netgroup_name ■ *-@netgroup_name*
Export	To make a file system available to remote nodes via NFS.

F

File System	A directory structure used to organize files.

G

GID	A value that identifies a group in HP-UX.
Global (YP)	A means of access in which the system always reads YP maps rather than the local ASCII files.

H

Hard Mount	A mount that causes NFS to retry a remote file system request until it succeeds, you interrupt it (default option), or you reboot the system.
Home Node	A term used in Virtual Home Environment (VHE) to refer to the machine on which a user's home directory physically resides.
Host	A node that has primary functions other than switching data for the network.
Host Node	A term used in Virtual Home Environment (VHE) to refer to the node a user is logged in to. This node environment is set up from the configuration files found on the user's home node.

I

Import	To obtain access to a remote file system from an outside source; to mount.
Internet Address	A four-byte quantity that is distinct from a link-level address and is the network address of a computer node. This address identifies both the specific network and the specific host on the network.

Interrupt-able Mount	A mount that allows you to interrupt an NFS request by pressing an interrupt key. (Though the interrupt key is not standardized, common ones include **CTRL - C** and **BREAK.**)

K

Key (YP)	A string of characters (no imbedded blanks or tabs) that indexes the values within a YP map so the system can easily retrieve information. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.

L

Local (YP)	A means of access in which the system first reads the local ASCII file. If it encounters an escape sequence, it then accesses the YP databases.

M

Map (YP)	A file consisting of logical records; a search key and related value form each record. YP clients can request the value associated with any key within a map. **YP map** is synonymous with **YP database.**
Map Nickname (YP)	A synonym for the YP map name when using certain YP commands.
Master Server (YP)	The node on which one or more YP maps are constructed from ASCII files. These maps are then copied to the YP slave servers for the YP clients to access.
Mount	To obtain access to a remote or local file system or directory (import).

Mount Point	The name of the directory on which a file system is mounted.

N

Netgroup	A network-wide group of nodes and users defined in */etc/netgroup*.
NFS	Network File System.
Network Lock Manager	A facility for locking files and synchronizing access to shared files.
Network Status Monitor	A daemon running on all network computers to maintain stateful locking service within NFS. It also allows applications to monitor the status of other computers.
Node	A computer system that is attached to or is part of a computer network.

P

Propagate	To copy maps (data) from one YP server to another.
Protocol	The rules and steps by which servers and clients exchange data and control information.

R

Remote Execution Facility (REX)	A facility which allows a user to execute commands on a remote node.
Remote Procedure Call (RPC)	A call made by clients either to access server information or to request action from servers.

Remote Procedure Call Protocol Compiler (RPCGEN)	A remote procedure call compiler used to help programmers write RPC applications by automatically generating necessary programs and code fragments.
Root Server	The only node in an HP-UX cluster that has file systems physically attached to it.

S

Server	■ A node that provides data or services to other nodes (clients) on the network. ■ A process that performs operations as requested by other processes. **Note:** An NFS server can also be configured as any combination of an NFS client, YP client, or YP server. (A YP server **must** also be configured as a YP client.)
Slave Server (YP)	A node that copies YP maps from the YP master server and then provides YP clients access to these maps.
Soft Mount	An optional mount that causes access to remote file systems to abort requests after one NFS attempt.
Stateless	Servers do not maintain (preserve) information relating to each file being served. Each file request moves across the network with the parameters attached to it locally (e.g., read and write privileges).
Steady State	Servers maintain (preserve) information relating to each file being served. For YP, the information contained in a YP map is consistent among all YP servers within a given YP domain (i.e., is not in the process of being updated).

U

UID	A value that identifies a user in HP-UX.
Unmount	To remove access rights to a file system or disk that was mounted via the *mount(1M)* command.
Update	The HP-UX command that installs software onto the system.

V

Value (YP)	A unit of information stored in YP maps; each value has a corresponding key (index) so the system can easily retrieve it. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.
VHE	See "Virtual Home Environment."
Virtual Home Environment (VHE)	A network service that allows users to log in at host nodes and utilize their home nodes' execution environments.

X

External Data Representation (XDR)	A protocol that translates machine-dependent data formats (i.e., internal representations) to a universal format used by other network hosts using XDR.

Y

Yellow Pages (YP)	An optional network service composed of databases (maps) and processes that provide YP clients access to the maps. The YP service enables you to administer these databases from one node. YP may or may not be active; check with your system administrator.
YP Client	■ A node that requests data or services from YP servers. ■ A YP process that requests other YP processes to perform operations. **Note:** A YP client can also be configured as any combination of a YP server, NFS client, or NFS server. (A YP server **must** also be configured as a YP client.)
YP Database	See “Map (YP).”
YP Domain	A logical grouping of YP maps (databases) stored in one location. YP domains are specific to the YP network service and are not associated with other network domains.
YP Map	See “Map (YP).”

YP Password	The password for a user's login ID that exists in the YP *passwd* map. The YP password is the same one as the user password, but is administered through the YP. You do not have to have a YP password to access the YP databases.
YP Server	■ A node that provides data (maps) or services to other nodes (YP clients) on the network using YP. ■ A YP process that performs operations as requested by other YP processes. **Note:** A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

Index

Reader Comment Card

HP 9000 Series 300 and 800 Computers
Using NFS Services
B1013-90000

We welcome your evaluation of this manual. Your comments and suggestions will help us improve our publications. Please tear this card out and mail it in. Use and attach additional pages if necessary.

Please circle the following Yes or No:

Is this manual well organized?	☐ Yes	☐ No
Is the information technically accurate?	☐ Yes	☐ No
Are instructions complete?	☐ Yes	☐ No
Are concepts and wording easy to understand?	☐ Yes	☐ No
Are examples and pictures helpful?	☐ Yes	☐ No
Are there enough examples and pictures?	☐ Yes	☐ No

Comments:

Date____________________________

Name:____

Title:____

Company:_____________________________________

Address:______________________________________

City & State: __________________________________

HEWLETT PACKARD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 37 LOVELAND,CO

POSTAGE WILL BE PAID BY ADDRESSEE

Hewlett-Packard Company
Colorado Networks Division
3404 East Harmony Road
Fort Collins, CO 80525

ATTN: Network Usability Department

Fold Here

Tape Please do not staple Tape

HEWLETT
PACKARD

HP Part Number
B1013-90000
Printed in U.S.A

E0989

B1013-90000