The Prospero File System User's Manual

Version Alpha.4.4

B. Clifford Neuman

Department of Computer Science and Engineering
University of Washington, FR-35
Seattle, Washington 98195

bcn@cs.washington.edu

Draft of June 8, 1991

1 Introduction

The Prospero file system is based on the Virtual System Model[1, 2, 3, 4]. It differs from traditional distributed file systems in several ways. In traditional file systems, the mapping of names to files is the same for all users. Prospero supports user centered naming: users construct their own view of the files that are accessible. The mapping from names to files is controlled by this view. Objects may be organized in multiple ways and the same object may may appear in different virtual systems, or even with multiple names in the same virtual system.

Prospero provides several mechanisms to support customization. Among them are the union link and the filter. Directories can contain references to files and other directories that are stored on remote nodes. This allows distribution at a much finer level of granularity than is possible in traditional distributed file systems.

2 The Directory Mechanism

In Prospero, the global file system consists of a collection of *virtual file systems*. Virtual file systems usually start as a copy of a prototype. The root contains links to files or directories[1] selected by the user. Directory links can be of several types: conventional, union, and filtered.

A conventional link is similar to a hard link in traditional file systems. It may be made to any type of object, including a directory. It maps a name for an object to the information needed to access

This research was supported in part by the National Science Foundation (Grants No. DCR-8420945 and CCR-8611390), US WEST Advanced Technologies, and Digital Equipment Corporation.

[1] To distinguish them from directories and links in traditional systems, and because they are often illusory, directories and links in Prospero are often referred to as virtual directories and virtual links.

it. As long as an unexpired link to an object exists the object may be accessed by that name. If the object moves, a forwarding pointer will allow continued access using the same name. An object is only deleted when no unexpired links to it remain.

A union link can only be made to a directory. With a union link, the objects included in the linked directory become part of the virtual directory containing the link. Thus, the contents of a virtual directory are the union of the collection of conventional links it contains, and the contents of all directories included through union links.

Filters may be attached to either type of link. A filter alters the set of links that are seen in directories whose paths pass through the filtered link. A filter can specify which links are to appear and which are to be ignored, it can change the features of individual links, or it can synthesize new links that are not in the original directory. Filters are written in C and are dynamically linked during name resolution. A filtered link contains a reference to the filter and any arguments that it takes.

3 Using the Prospero File System

This section assumes that Prospero has already been installed on your system. Section 6 describes the installation procedure.

To use Prospero you must first determine the name of the directory that contains the executables.[2] That directory contains a file called vfsetup.source, the contents of which must be read by the shell. This can be accomplished by `source`ing it. For example, if the Prospero file system binaries are stored in */usr/pfs/bin* you should either execute the following command or add it to your *.cshrc* file.

```
source /usr/pfs/bin/vfsetup.source
```

Before you can begin using the Prospero file system a virtual system must be created. Section 3.4.1 explains how the Prospero site administrator can create a new virtual system. Most Prospero sites support a *guest* virtual system which can be used until your own virtual system is created.

To use a virtual system, you must first execute the `vfsetup` command to initialize your environment. For example, if the name of your virtual system is *guest* you should execute the following command.

```
vfsetup guest
```

[2] By default, the installation directory is */usr/pfs/bin*

3.1 File Names

The "/", the ":", the "#", the "(", and the ")" are special characters in Prospero. The "/" separates components of file names, and the ":" separates the name of a virtual system from the name of a file within the virtual system. These characters should not appear in any component of a file name unless they have been quoted.

If the "#" appears at the end of the name of a virtual system, then the name preceding the "#" will be treated as an alias. For this reason, virtual systems should not have names ending in "#". It is acceptable for the names of regular files and directories to end with this character.

The "#" is also used to resolve name conflicts when the same component of a name is used by more than one object. In this case the "#" is followed by the magic number of the desired object. For this reason, unless quoted, the "#" should not be used in a component of a file name if followed by a number (including sign), and if there are no intervening non-numbers between the "#" and the end of the component.

The "#" is also used to indicate that a particular named union link is to be followed when resolving a name, and to indicate that a filter is to be applied. In these cases, the "#" is the first character of the component name and it is followed by the name of the union link to be followed or the name of the filter to be applied. For this reason, unless quoted, the "#" should not be used as the first character in a component of a file name (unless the full name of the component is "#").

The "(" and the ")" are used to delimit the arguments to a filter specified as part of a file name. The arguments immediately follow the name of the filter (which itself follows a "#"). If no arguments are required, the null argument list "()" must be included.

Note that in the current release, quoting of special characters is not supported so they should be avoided in the situations described. This is easily accomplished by avoiding the use of the ":" and the "/", altogether and by avoiding the use of "#" at the start of a name, at the end of a name, or where it is immediately followed by a number.

3.2 Using Existing Applications

The Prospero file system is presently implemented as a library. A number of applications have been linked with the Prospero library. The relinked versions may be found in the same directory as the other Prospero binaries, and will appear in you search path once the `vfsetup` command has been executed.

By default, names which are to be resolved using Prospero must contain or be preceded by a ":". File names that do not contain and are not preceded by a ":" are treated as native Unix file names. The default behavior can be modified by setting the PFS_DEFAULT environment variable. This may be done by using the `venable` command. When enabled, names are resolved relative to your virtual file system by default. Names that are to be treated as native Unix file names must be preceded by an "@". `vdisable` will return the default to its original state. The meanings of the

Value	Meaning
0	Never resolve names within the virtual system
1	Always resolve names within the virtual system
2	Resolve names within the virtual system if they contain a :
3	Resolve names within the virtual system by default, but treat names beginning with an @ as native file names

Table 1: Settings for the PFS_DEFAULT environment variable

values for the PFS_DEFAULT environment variable are listed in Table 1.

3.3 Finding Things

The Prospero file system provides tools that make it easier for you to keep track of and organize information in large systems. When first created, your virtual file system is likely to contain links to directories that organize information in different ways. As the master copy of each of these directories is updated, you will see the changes. You may customize these directories. The changes you make to a customized directory are only seen from within your own virtual system, but changes made to the master copy will also be visible to you. See section 3.6 for instructions on customizing a directory.

Users are encouraged to organize their own projects and papers in a manner that will allow them to be easily added to the master directory. For example, users should consider creating a virtual directory that contains pointers to copies of each of the papers that they want made available to the outside world. This virtual directory may appear anywhere in the user's virtual system. Once set up, a link may be added to the master author directory. In this manner, others will be able to find this directory. Once added to the master directory, any future changes will be immediately available to other users.

To add a link to the master copy of any of the shared directories, send a message to your site administrator. The address should be *pfs-administrator* on the primary system for the site. If you are using a virtual system stored at the University of Washington, the address would be *pfs-administrator@cs.washington.edu*.

3.4 The Commands

This section steps though the process of creating and using a virtual file system. It assumes that the Prospero file system has been installed on the system being used. Later sections explain how to install the Prospero file system at a new site and on individual systems.

Most of the commands that are specific to the Prospero file system take a debug option. The form

Value	Meaning
0 (default)	Prompt for required input
1 (-v)	Explain what is required when asking for input
2	List each action taken
3	Stop before each step
4	Stop before each step and explain the action

Table 2: Verbosity levels for newvs

is *-D#* where # is an optional integer and specifies the level of detail. The higher the integer, the greater the detail. By itself, *-D* sets the debugging level to 1.

3.4.1 Creating a Virtual File System

A virtual file system is created using the `newvs` command. The `newvs` command is for use by the site administrator[3]. To have a virtual system created, send a message to *pfs-administrator* at your site.

```
newvs [-v#] [-e] [host [name [home [owner [desc_file]]]]]
```

The `newvs` command is used to create a new virtual system. If called with no arguments, the user is prompted for the system on which the new virtual system is to reside, the name of the virtual system, the home directory, the owner, and the name of the description file.

The name of the virtual system is the default name with which it will appear in the local site's master list of virtual systems. This name is not automatically exported beyond the local site.

`newvs` will create the virtual system, assign a global name to it, and add the selected name in the master list of virtual systems for the local site. It will also copy the links from the *prototype* virtual system to the newly created one. The *-e* option will suppress this copying, and will leave the new virtual system empty. As a final step, `newvs` will optionally write a description file that may be read by `vfsetup`.

The *-v* option is followed by an integer and sets the verbosity level. The meaning of the verbosity levels are presented in Table 2.

3.4.2 Initialization and Changing Virtual Systems

```
vfsetup [-n host path , [-r,v] name , -f file]
```

[3]Systems which are running the release as distributed are part of the University of Washington site. The site administrator is *pfs-administrator@cs.washington.edu*. This applies even if your system is running a server.

The `vfsetup`[4] command sets up the selected virtual system. It adds the appropriate directories to the search path and sets all necessary environment variables.

`vfsetup` can be called in several ways. With no arguments, it reads the file *~/.virt-sys* and uses the information found to access the virtual system description. The *-v* option takes the name of the virtual directory containing the system description. the *-n* option takes the name of a host and the physical name of the virtual directory on that host that contains the virtual system description. The *-f* option takes the name of a Unix file that is to be read in place of *~/.virt-sys*.

It is also possible to set up a virtual system by specifying its name. If the *-r* option is specified, the name is taken to be the default name for the virtual system at the local site. If the name is specified without a modifier, the name is looked up in the */VIRTUAL-SYSTEMS* directory of the presently active virtual system (the site default is used if no virtual system is presently active). For example, if the name of your virtual system is *guest* you can set up the virtual system using following command.

```
vfsetup guest
```

3.4.3 Creating Directories

```
vmkdir directory
```

`vmkdir` creates a virtual directory with the selected name and adds a link from its parent directory.

3.4.4 Adding and Deleting Links

```
vln [-u, -s, -m] [-e[am], -n host] oldname newname
```

`vln` adds a new link to a directory. *oldname* is an existing name for the object to which the link is to be made. *newname* is the name of the new link. The *-u* option indicates that the new link is to be a union link. The *-s* option is used to specify a symbolic link.

The *-e* and *-n* options require that the name of the host containing the target is specified. The *-n* option indicates that the native information for the target has been specified. *host* is the name of the host on which the target resides and *oldname* is the name of the target on that host. If the *-s* option has also been specified, then *host* is the name of the virtual system to which oldname is relative.

[4] Because it changes environment variables, this command only has an affect when its output is read by the shell. If *vfsetup.source* has been `source`ed, then `vfsetup` is an alias which will call the *vfsetup* executable in the appropriate manner.

The *-e* (external) option indicates that the object resides on a host that does not run Prospero. *host* is the name of the host on which the object resides, and *oldname* is the name needed to access that host using the selected access method (e.g. if the access method is anonymous FTP, the name is relative to the anonymous FTP directory on that host).

The -e option can optionally specify the access method to be used and information needed by that method. This information is specified by appending it to the *-e* option itself (e.g. *-eNFS,/u1*). If no access method is specified, the default is *AFTP,BINARY*, which will work for most files accessible by anonymous FTP. Contact *info-prospero@cs.washington.edu* if you need more information on this option.

If the standard input to `vln` has been redirected, the input will be searched for a line of the form "Virtual-system-name: vs-name". If found, *oldname* will be relative to the virtual system which has been read from (closed with) the input. If the *-m* option has been specified, the input will be additionally searched for a line for the form "Virtual-file-name: filename". If found, *filename* will be used in place of *oldname*. The reading of the standard input can be suppressed by specifying the *-a* option, in which case the currently active virtual system will be used.

```
vrm link
```

`vrm` removes the named link from a directory. It is important to note that `vrm` only removes the link. The object will continue to exist if there are any additional links to it. If there are none, then the object will become subject to garbage collection at a future time.

3.4.5 Listing Directories

If the `venable` command (see section 3.2) has been executed to set the default name resolution mechanisms to the virtual file system, then the `ls` command may be used to list virtual directories. Otherwise, virtual directories may be listed using the `vls` command.

```
vls [-A,-c,-f,-u,-v] [-a[attribute]] [path]
```

`vls` takes the virtual path name for a file or directory. If the path is for a directory, the links within that directory are displayed. If the path is for any other type of object, then the information for the named link is displayed.

By default, `vls` displays for each link the local component of the path name and the information needed to access the object. In most cases, this is a host and a name relative to that host. A "*" preceding the host name indicates that a filter is associated with the link. The *-v* option causes the object type, and the type of each field to be displayed, and it lists the filters associated with the link. It also prevents the truncation of fields that are too long to be cleanly displayed without the *-v* option.

The *-u* option indicates that union links are not to be expanded. By default, union links are expanded, and the results of that expansion displayed. To see which union links are included in a directory, the *-u* option must be specified.

There are cases when a directory might include more than one link with the same name. One way this can happen is if the directory contains union links. By default, only the first link with a particular name is displayed. The *-c* option tells `vls` to display all links, including those with conflicting names. The name of conflicting links will be followed by a "#" and a number that allows them to be uniquely identified.

The *-f* option causes the display of union links which could not be expanded. This option is presently set by default.

The *-a* option indicates that the attributes associated with each file are to be displayed. It also forces the verbose option. If only a particular attribute is desired, the attribute can be specified as part of the *-a* option itself (e.g. *-aFORWARDING-POINTER*). The *-a* option by itself displays all attributes. The *-A* option is similar to the *-a* option, but it only lists the attributes associated with the link itself, not those associated with the object referenced by the link[5].

3.4.6 Moving Around

```
vcd [-u] path
```

The `vcd`[6] command allows one to change the virtual working directory. If no argument is specified, the home directory is assumed. Otherwise, paths starting with "/" are treated as relative to the root of the virtual file system, and other paths are treated as relative to the current working directory. ".." specifies the directory above the current working directory along the active path from the root.

The *-u* option allows one to change ones virtual working directory to a directory included through a named union link.

If the `venable` command (see section 3.2) has been executed to set the default name resolution mechanisms to the virtual file system, then the `cd` command may also be used to change virtual directories.

```
vwd
vwp
```

The `vwd` prints the name of the current virtual directory relative to the root of the virtual system. The `vwp` command prints the information describing its physical storage location.

[5] When using `vls` to list the results of an Archie query, the *-c* and the *-A* options should be specified.

[6] Because it changes environment variables, this command only has an affect when its output is read by the shell. `vcd` is an alias which will call the *vcd* executable in the appropriate manner.

3.5 Retrieving Files

```
vget virtual-file [local-file]
```

The commands described so far allow you to move around the virtual file system, but they do not allow you to access the files that it names. If a program has been linked with the Prospero compatibility library, the program can access the file directly.

The `vget` command can be used to explicitly retrieve a file that is accessible by anonymous FTP. The *virtual-file* is the name of the file to be retrieved from the Prospero file system. *local-file* is the real name that you want the file to have in your real current working directory. If *local-file* is omitted the last component of the virtual file name will be used.

If the standard input to `vget` has been redirected, the input will be searched for a line of the form "Virtual-system-name: vs-name". If found, *virtual-file* will be relative to the virtual system which has been read from (closed with) the input. If the *-m* option has been specified, the input will be additionally searched for a line for the form "Virtual-file-name: filename". If found, *filename* will be used in place of *virtual-file*. The *-a* option can be used to suppress the searching of the standard input.

3.6 Customizing a directory

When a change is made to a directory, those changes are often visible to other users. There are times when it is desirable to create a customized view of an existing directory. This will allow changes to be made which are only visible when the directory is viewed through the current path. Such changes will not affect the view of the directory seen by others.

To create a customized view of a directory, create an empty directory, add a union link from the directory you just created to the target directory, then remove the link to the old directory and replace it with a link to the directory that was just created.

Links which are added to the customized directory will only be visible through the customized directory, but changes to the target directory will also be visible through the customized directory.

Some directories in your virtual system have already been customized. The root of your virtual system is your own. Changes in the root do not appear in the roots of other virtual systems. Each of the links from the root is also a customized directory. For example, if you add a link to the /authors directory, that link will not be visible to others. Directories at the next level, however, are not customized. Thus, if you add a link to the directory /authors/Shakespeare,William, that change will be visible to others unless you first customize that directory.

You can determine whether a directory has been customized by using the `vls -u` command. That command will show the current directory without expanding union links. Admittedly, this is a little confusing. Future releases will support the concept of an owning virtual system. This will

clear up some of the confusion by allowing automatic creation of a customized directory when one is needed.

To have a link added to the master copy of a shared directory you should send a message to *pfs-administrator* at your site, or to *pfs-administrator@cs.washington.edu*.

3.7 Attributes and Forwarding Pointers

This release includes preliminary support for attributes and forwarding pointers. This means that file attributes may be retrieved using the *-a* option to the `vls` command. User defined attributes may also be associated with a file. For the time being, such attributes must be added manually.

Forwarding pointers are also supported. If a file or directory has moved and a forwarding pointer exists, the forwarding pointer will be returned and the request retried. Like attributes, forwarding pointers must be added by hand. There are not presently any programs to add forwarding pointers or attributes. For information on how to add these by hand, send a message to *info-prospero@cs.washington.edu*.

4 Protection

Access control lists (ACLs) may be associated with directories in Prospero, and with individual links within a directory. These access control lists specify how the directory information is to be protected. They have nothing to do with the protection of the file to which a link refers. Table 3 lists the protection modes that may appear within and access control list entry.

When an entry appears in both columns, it means that the entry on the directory overrides the entry on a link. For example, a user with R access to the directory can read a link even if denied r access to the link. If a link protection bit is stored in a directory access control list, then that bit indicates the default protection associated with links in the directory that do not specify their own access control lists.

Administer allows the changing of the access control list. View allows it to be viewed. List allows one to see a directory entry using wildcard searches. Read is required to determine its binding (the file it references). If one is allowed read, but not list, then one can only retrieve the link if its exact name is specified. Modify allows one to change the binding of a link, but it does not allow one to add or delete links. Insert allows links to be added, and delete allows them to be deleted. The add and remove rights bits are a restricted form of administer. They allow an individual to add or remove a restricted set of rights. For example, ">r" allows one to grant read access to someone else without also allowing one to grant modify access.

Negative rights my be specified by preceding the rights field with a minus sign (-). The order of access control list entries is important. For negative rights to have an effect, they must precede any rights that authorize access by that individual. When new ACL entries are added, they are added

Directory		Link	
Character	Meaning	Character	Meaning
A	Administer	a	Administer
V	View	v	View
L	List	l	List
R	Read	r	Read
M	Modify	m	Modify
D	Delete	d	Delete
I	Insert	does not apply to link	
B	Administer	does not override link	
Y	View	does not override link	
>	Add rights	]	Add rights
<	Remove rights	[	Remove rights
)	Add rights	does not override link	
(	Remove rights	does not override link	

Table 3: Protection modes in Access Control Lists

at the front of the list, meaning that recent entries take precedence over older ones.

When access is checked for a link, three access control lists are checked. First, the ACL associated with the link itself is checked. If the link does not have its own ACL, then the default ACL associated with the directory is used. Next, the ACL associated with the directory is checked to see if it grants rights that override those in the link. Finally, if access has still not been granted, a special override ACL is checked. This is maintained by the system and should be used only in emergencies. Negative rights in one list does not override access granted in another.

There are several different ACL entry types. DEFAULT, SYSTEM, and DIRECTORY cause other access control lists to be included. DEFAULT is the default ACL specified by the system on which the Prospero server is running. SYSTEM is also specified separately on each Prospero server. It usually grants additional access to system administrators. DIRECTORY is the default access control list associated with the directory containing a link. It allows one to easily specify that the rights on a link are to be in addition to the default rights specified in the directory. If no rights are associated with these ACL entry types, then the rights granted are based on those in the DEFAULT, SYSTEM, or DIRECTORY access control lists themselves. If rights are specified for such entries, then the rights are the minimum of those specified, and those in the included ACL. Files and directories initially authorize access to DEFAULT and SYSTEM as defined in those lists. Users have the option of removing such access[7].

The ANY, AUTHENT, ASRTHOST, and TRSTHOST ACL types require rights to be specified. ANY matches any user. AUTHENT specifies an authentication method, and the name of the authorized individual as returned by that method. At present, this entry type is not supported.

[7]The SYSTEM access control list is separate from the OVERRIDE list which can not be removed.

The ASRTHOST method specifies a list of authorized principals in the form *user@internet-address*. If no Internet address is specified (and no atsign), then the user is matched regardless of the requesting host. Octets of the Internet address can be wildcarded, or replaced with a "%". A wildcard matches any number, and a "%" matches the number corresponding to the local host. For example, "bcn@%.%.%.*" matches the user "bcn" on the local subnet.

The ASRTHOST type accepts the username asserted by the client. It is not possible to verify that the user has not modified the software to claim someone else's identity. The Internet address can generally be considered accurate, though it too can be spoofed by a knowledgeable and determined attacker. The TRSTHOST type is identical to the ASRTHOST type, but is accepted only when the request originates from a privileged port on the requesting system. Although this method might be used to provide security similar to that for the Berkeley R commands, it is not recommended that you install Prospero binaries setuid root until the sources have undergone careful scrutiny for possible security holes[8].

Listing ACLs

The `list_acl` command may be used to list the contents of an access control list.

```
list_acl [-d dir] [link-name]
```

With no arguments, it lists the ACL for the current directory. If the *-d* option is specified, the following argument specifies the directory whose ACL is to be listed. An optional link-name specifies that the directory to be listed applies to the named link within the directory.

Modifying ACLs

The `set_acl` command allows one to change the access control list associated with a link or directory.

```
set_acl [-asirKE,-n,-N] [-t type] [-d dir] [-l link] rights principals
```

The *-d* option is followed by the name of a directory. By default, the ACL for the directory is modified. The *-l* option allows one the modify the ACL for an individual link.

The *-a, -s, -i, -r, -K,* and *-E* options indicate the operation to be performed on the ACL. They correspond in order to add rights, subtract rights, insert a new entry, remove an entry, kill the entire ACL (setting it to the default), and replacing the entire ACL.

[8] By no means should vget or vcache be installed setuid root as these command write files to paths specified by the user.

If the *-t* option is specified, it must be followed by the type of the ACL entry to be added, deleted, etc. If the *-t* option is not specified, ASRTHOST is the default.

The first field following the options specifies the rights to be added or deleted. All remaining arguments are the names of the principals to be included in the particular ACL entry.

When an ACL is set to an initial value (using the *-K* or *-E* options), the SYSTEM ACL is automatically included. The SYSTEM entry can be removed by using the *-r* option in a subsequent `set_acl` command. The addition of the SYSTEM entry can be suppressed by using the *-n* option in conjunction with the original *-K* or *-E* option. Whenever rights are removed from an ACL, the system checks to make sure that the user removing the rights will be able to fix any mistakes. If the the change would result in the user being unable to make subsequent changes, the minimal rights allowing the user to make subsequent change are automatically added back. This safety mechanism may be overrided by specifying the *-N* option.

5 Filters

Filters are written in C, compiled, and dynamically linked during name resolution. Because of portability problems with the dynamic linker, filters are not included in this release, but may be available upon request.

6 Setting up a new system

This section explains how to install the Prospero file system on a new system. It assumes that the local site has already been configured, This section (Section 6) can be skipped by most users.

6.1 Building the binaries

[See the INSTALLATION file in the distribution]

6.2 Running the Server on Unix (like) Systems

If you are installing the Prospero server, a user and group ID must be established under which the directory server will run. The directory associated with the user ID should be a location in which additional information about virtual files and directories can be stored. New files which are to exist only within the Prospero file system will also be stored under this directory. It is suggested that you chose the user name *pfs* for this pseudo-user, but other names may be used as well.

Once the user ID has been set up, install the binaries. The directory in which they must be

installed is selected at compile time. As originally distributed, it is */usr/pfs/bin*. The program `pstart` should be installed setuid and setgid the pseudo-user just described.

```
pstart [hostname]
```

To start the server run `pstart`. `pstart` takes an optional host name. If specified, the host name must be the primary name for the host on which the server is running. In most cases, the server is able to determine the name on its own and there is no need to specify it as an argument.

`pstart` will connect to the directory associated with the pseudo-user, it will check to make sure that the user id is set appropriately, and it will exec the directory server with the appropriate arguments.

Although it is not recommended, the directory server can also be started manually. You must first be logged in as (or be su'ed to) the user under whose ID you want the server to run. You can then execute `dirsrv` passing as arguments the required directory names.

```
dirsrv [-m] root shadow data aftpdir afsdir hostname
```

The *-m* (manual) option prevents the directory server from dissociating itself from the terminal. It is only useful for debugging. *root* is logical root of the system. Only files below this point (and those under aftpdir and afsdir) will be accessible through the Prospero file system. *shadow* is the name of the directory that is to contain additional information about files and directories. It should typically be the *shadow* subdirectory of the pseudo-user described above. *data* is the local directory under in which new virtual directories and their contents will be stored.

aftpdir is the name of the directory hierarchy to which anonymous FTP has access and *afsdir* is the name of the directory through which afs files may be accessed. If these access methods are not supported by your system, these arguments should be the null string.

Users can use the Prospero file system even if the server is not running, but they will be unable to access any files or directories stored locally. If `pstart` is installed setuid and setgid to the Prospero user and group IDs, then the directory server can be started by any user. You may also want to start the directory server from the system's */etc/rc* file.

As things stand, users can access files and directories created on the local system, but they can not create new virtual systems stored locally. If you want to allow virtual systems to be stored locally, then you must have the site administrator add a reference to the new system from the *pfs_storage* virtual directory.

Adding references to the new system

The remainder of this section describes the actions that are to be taken by the site administrator. Systems which are running the release as distributed are part of the University of Washington site.

The site administrator is *pfs-administrator@cs.washington.edu*. This applies even if your system is running a server. If you are not a site administrator, you can skip this section.

If you want to allow virtual systems to be stored locally, several links must be added and a new virtual directory must be created. This will only be possible if the server has **not** been configured read-only. When a new site is established (see Section 7) these links and directories are automatically created on the primary system for the site. The following steps are only required when adding additional systems.

In the following steps, HOST is the fully qualified domain name for the host to be added and PATH is the full path of the subdirectory of the pseudo-user (described above) which will store the new virtual directories. The last component will typically be *pfsdat*. If that directory does not already exist, it should be physically created.

A link must be added from the virtual directory *pfs_storage* to the pfsdat directory on the new site. While still in the master virtual system, the following steps will add this link.

```
vcd /pfs_storage
vln -n HOST PATH HOST
```

You will next have to create the *local_vsystems* virtual directory. You do this by issuing the commands:

```
vcd HOST
vmkdir local_vsystems
```

7 Setting up a new site

This section explains how to set up a new site. Systems which are running the release as distributed are part of the University of Washington site. This applies even if your system is running a server. The site administrator is *pfs-administrator@cs.washington.edu*.

If you would like to set up your own site, send a message to *info-prospero@cs.washington.edu* to obtain the appropriate distribution (several additional files). Unless you are setting up your own site, you may skip the remainder of this section (Section 7).

Before you begin, you will have to obtain a global prefix that will uniquely identify the virtual systems registered at your site. Send a message to *pfs-administrator@cs.washington.edu* to obtain this prefix.

The global prefix is part of the low level name for each virtual system. It should be thought of as an address. Users use higher level names to specify virtual systems.

Registering a prefix will allow objects created at your site to be named by others. Even if your site will not be reachable by any other sites, it is still important to register a prefix. Doing so guarantees that no other site has the same prefix. This will make it possible to connect with the rest of the global system should a connection ever be established. More information on setting up an isolated site is described in Section 7.2.

7.1 Setting up the master directories

To initially set up a new site, run the command `newpsite`. `newpsite` will construct a skeletal site configuration based on the compile time options described in the previous section.

Once the site has been set up, it will be necessary to create additional directories and add them to the prototype on which new virtual systems will be modeled.

[suggestions on what to add still to be specified]

7.2 Setting up an Isolated Site

As was already mentioned, even if your site will be isolated from others, you should still try to register a unique global prefix. If this is impossible, see Section 7.2.1 for instructions on assigning one yourself.

If your site will be isolated from other sites you will have to set up a replica of the global root. This directory must be reachable with the name "#" from your site's master list of virtual systems. To create a replica of the global root, create a directory and add nested subdirectories corresponding to each component of your sites global prefix. The last entry should be a link to your site's master list of virtual systems.

7.2.1 If you can not register a prefix

If it is not possible to contact *pfs-administrator@cs.washington.edu,* then it may still be possible to generate a unique global prefix based on a unique identifier assigned by another authority. Right now, the only names that may be turned into unique global prefixes are officially registered Internet domain names.

Internet Domain Names. If you have an officially registered Internet domain name, it may be turned into a global prefix by reversing the order of the components, replacing the periods with slashes, and prepending "#/INET/".

```
ISI.EDU      =      #/INET/EDU/ISI
```

8 Glossary

master directory. A master directory is a directory maintained at the site level, and included through union links as part of the corresponding directory in multiple virtual systems.

virtual directory. A virtual directory is a directory in a virtual file system. The contents of a virtual directory might be determined on the fly by expanding union links, or applying filters.

global file system. The global file system is the collection of links and directories that make up the virtual file systems accessible to the user. The links and directories form a generalized directed-graph.

virtual file system. A virtual file system is the file system part of a virtual system. It consists of a root directory and all the files and directories that can be reached by traversing 0 or more virtual links. The virtual file system is a projection of the global file system as viewed from the selected root.

filter. A filter is a program attached to a virtual link. A filter can modify the results of directory queries where the path from the root of the virtual file system to the queried directory passes through the filtered link.

conventional link. A conventional link similar to a hard link in traditional file systems. It maps a name for an object to the information needed to access it.

union link. A Union link is a virtual link where the objects included in the linked directory are to become part of the directory containing the union link. The contents of a directory is the union of the set of conventional links it contains, and the contents of all directories included through union links.

virtual link. A virtual link is either a conventional link or a union link, with or without an attached filter. Where the meaning is clear from context, "virtual link" is often shortened to "link".

site. Site is a loosely defined term. It is best described as a collection of systems sharing a similar default view of the world.

local system. The local system is the physical system to which a user is logged in. "local system" is often shortened to "system" if the meaning is clear.

virtual system. A virtual system defines the context within which names are to be resolved, it includes the objects and services which are to be available, and it specifies other information that customizes one's view of the world.

view. A view is a mapping from names to objects. A virtual system implements and imposes a view. It is possible for more than one virtual system to impose the same or similar views.

9 Quick Reference

Commands affecting the Prospero file system:

```
vfsetup [-n host path , [-r,v] name , -f file]

vcd [-u] path

vwd

vls [-v] [-u] [-f] [path]

vln [-u] [-s] [-e] [-n host1] name1 name2

vmkdir directory

vrm link

vget virtual-file [local-file]

newvs [-v#] [-e] [host [name [home [desc_file]]]]

list_acl [-d dir] [link-name]

set_acl [-asirKE,-n,-N] [-t type] [-d dir] [-l link] rights principals

vdisable

venable
```

Meanings for PFS_DEFAULT:

The meanings of the values for the PFS_DEFAULT environment variable are:

```
 Value       Meaning

   0         Never resolve names within the virtual system
   1         Always resolve names within the virtual system
   2         Resolve names within the virtual system if they begin with a :
   3         Resolve names within the virtual system by default, but
             treat names beginning with an @ as native file names
```

Acknowledgments

Ed Lazowska, Hank Levy, David Notkin, and John Zahorjan helped me to refine the ideas for the virtual system model. Ed Lazowska deserves special thanks for the numerous discussions out of which many of the ideas emerged. Some of the ideas emerged from discussions with Alfred Spector and other instructors at the Arctic '88 Course on Distributed Systems. The name Prospero was suggested by Bill Bryant.

References

[1] B. Clifford Neuman. The need for closure in large distributed systems. *Operating Systems Review*, 23(4):28–30, October 1989.

[2] B. Clifford Neuman. The Virtual System Model for large distributed operating systems. Technical Report 89-01-07, Department of Computer Science, University of Washington, April 1989.

[3] B. Clifford Neuman. Workstations and the Virtual System Model. In *Proceeding of the 2nd IEEE Workshop on Workstation Operating Systems*, pages 91–95, September 1989. Also available as University of Washington Technical Report 89-10-10. Also appears in the *Newsletter of the IEEE Technical Committee on Operating Systems*, Volume 3, Number 3, Fall 1989.

[4] B. Clifford Neuman. The Virtual System Model: A scalable approach to organizing large systems (a thesis proposal). Technical Report 90-05-01, Department of Computer Science, University of Washington, May 1990.