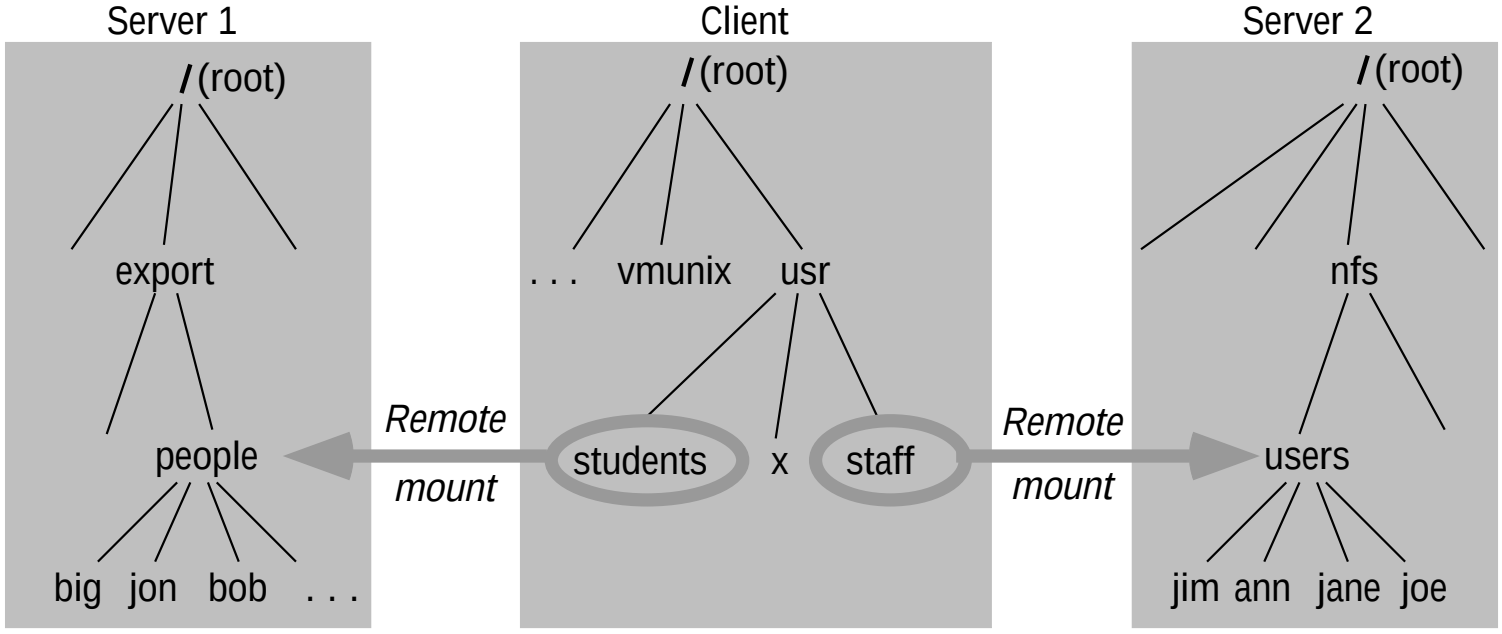
Figure 8.1 Local and remote file systems accessible on an NFS client.



Note: The file system mounted at */usr/students* in the client is actually the sub-tree located at */export/people* in Server 1; the file system mounted at */usr/staff* in the client is actually the sub-tree located at */nfs/users* in Server 1.

Figure 8.2 NFS software architecture.

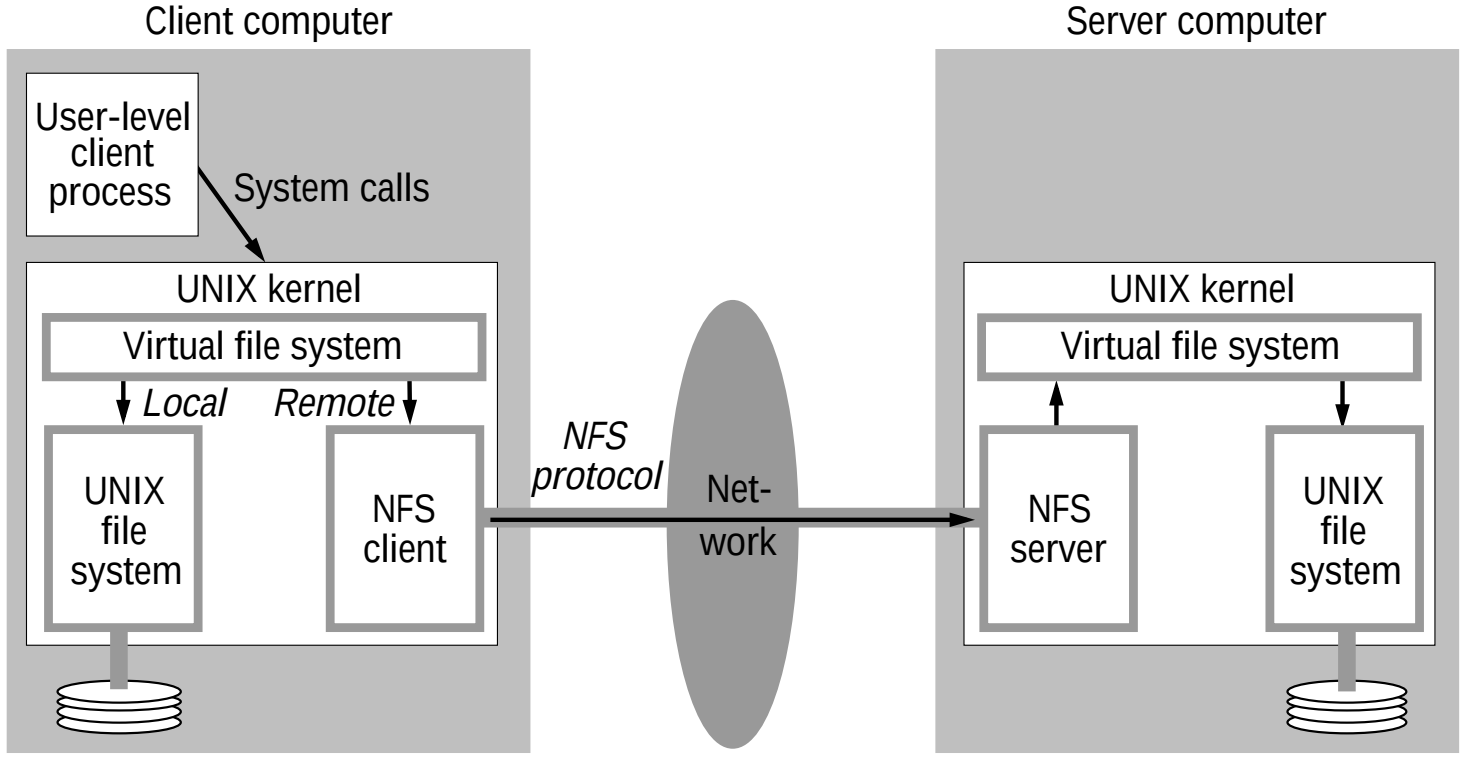


Figure 8.3 NFS Server operations.

---

lookup(dirfh, name) → *fh, attr*

Returns a file handle and attributes for the file *name* in the directory *dirfh*.

create(dirfh, name, attr) → *newfh, attr*

Creates a new file *name* in directory *dirfh* with attributes *attr* and returns the new file handle and attributes.

remove(dirfh, name) → *status*

Removes file *name* from directory *dirfh*.

getattr(fh) → *attr*

Returns file attributes of file *fh*. (Similar to the UNIX *stat* system call.)

setattr(fh, attr) → *attr*

Sets the attributes (mode, user id, group id, size, access time and modify time of a file). Setting the size to 0 truncates the file.

read(fh, offset, count) → *attr, data*

Returns up to *count* bytes of data from a file starting at *offset*. Also returns the latest attributes of the file.

write(fh, offset, count, data) → *attr*

Writes *count* bytes of data to a file starting at *offset*. Returns the attributes of the file after the write has taken place.

rename(dirfh, name, todirfh, toname) → *status*

Changes the name of file name in directory dirfh to toname in directory todirfh.

Figure 8.3 continued NFS Server operations.

link(newdirfh, newname, dirfh, name) → *status*

Creates an entry *newname* in the directory *newdirfh* which refers to file *name* in the directory *dirfh*.

symlink(newdirfh, newname, string) → *status*

Creates an entry *newname* in the directory *newdirfh* of type *symbolic link* with the value *string*. The server does not interpret the *string*, but makes a symbolic link file to hold it.

readlink(fh) → *string*

Returns the string that is associated with the symbolic link file identified by *fh*.

mkdir(dirfh, name, attr) → *newfh, attr*

Creates a new directory *name* with attributes *attr* and returns the new file handle and attributes.

rmdir(dirfh, name) → *status*

Removes the directory empty *name* from the parent directory *dirfh*. Fails if the directory is not empty.

readdir(dirfh, cookie, count) → *entries*

Returns up to *count* bytes of directory entries from the directory *dirfh*. Each entry contains a file name, file id, and an opaque pointer to the next directory entry, called a *cookie*. The *cookie* is used in subsequent *readdir* calls to start reading from the subsequent entry. A *readdir* with a 0 value for the *cookie* reads from the first entry in the directory.

statfs(fh) → *fsstats*

Returns file system information (such as block size, number of free blocks, and so on) for the file system containing a file *fh*.

Figure 8.4 Distribution of processes in the Andrew file system.

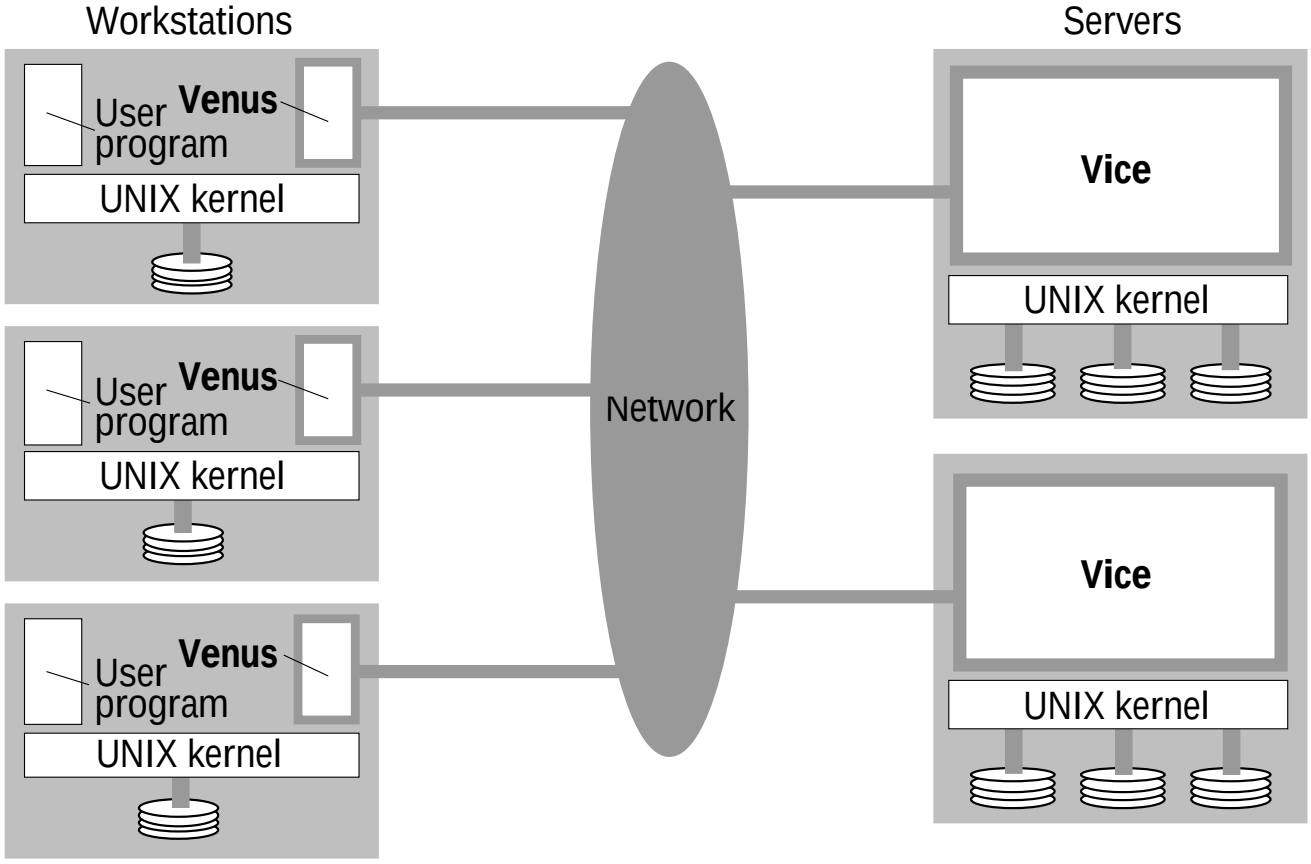


Figure 8.5 File name space seen by clients of the Andrew File System.

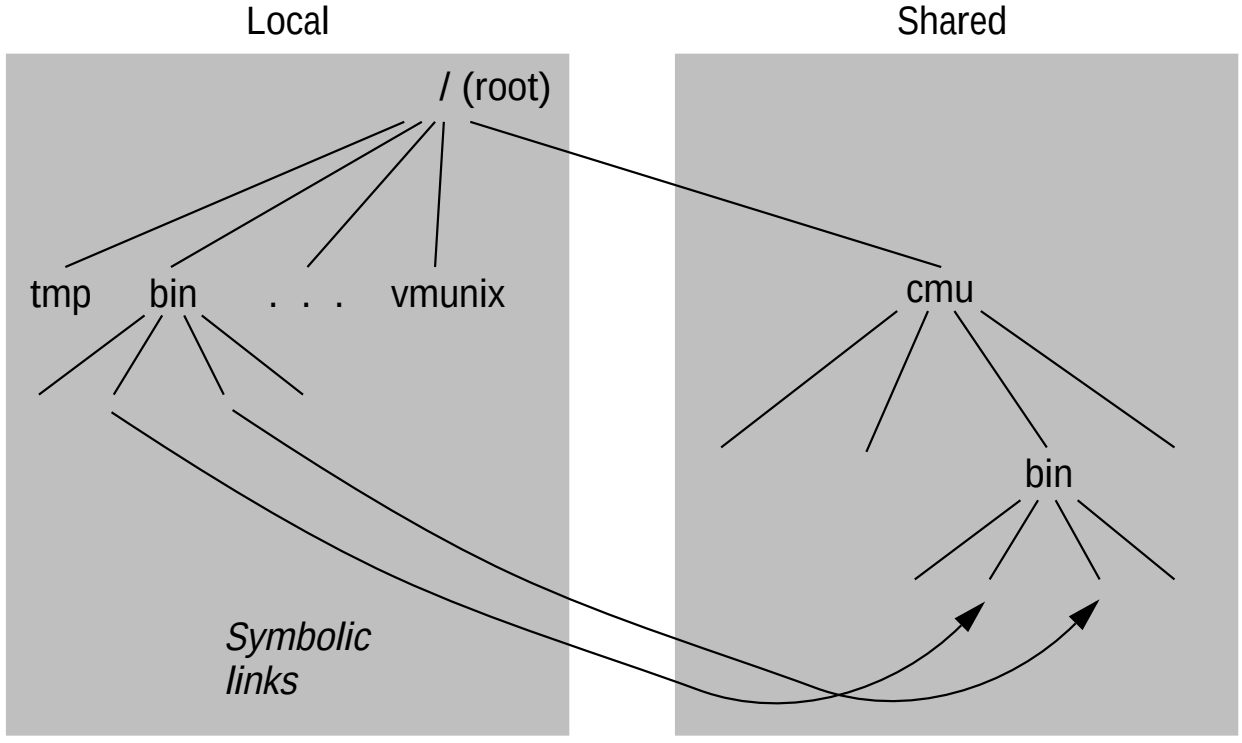


Figure 8.6 System call interception in the Andrew file system.

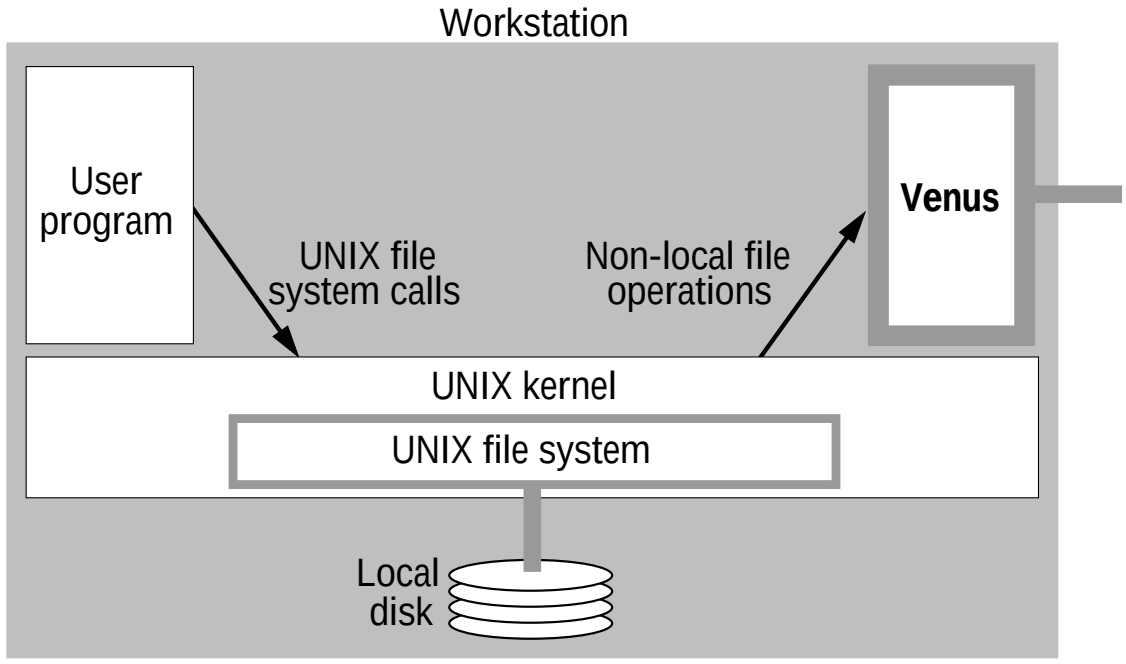


Figure 8.7 Implementation of file system calls in AFS..

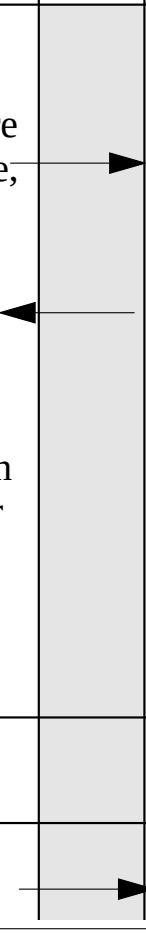
User process	*UNIX kernel*	*Venus*	*Net*	*Vice*
open(FileName, mode)	If FileName refers to a file in shared file space, pass the request to Venus. Open the local file and return the file descriptor to the application.	Check list of files in local cache. If not present or there is no valid callback promise, send a request to the Vice server that is custodian of the volume containing the file. Place the copy of the file in the local file system, enter its local name in the local cache list and return local name to UNIX.		Transfer a copy of the file and a *callback promise* to the workstation. Log the callback promise
read(FileDescriptor, Buffer, length)	Do a normal UNIX read operation on local copy.			
write(FileDescriptor, Buffer, length)	Do a normal UNIX write operation on local copy.			

Figure 8.8 The main components of the Vice service interface.

Fetch(fid) $\rightarrow$ *attr, data*

Returns the attributes (status) and, optionally, the contents of file identified by the *fid* and records a callback promise on it.

Store(fid, attr, data)

Returns the attributes and, optionally, the contents of a specified file and records a callback promise on it.

Create() $\rightarrow$ *fid*

Creates a new file and records a callback promise on it.

Remove(fid)

Deletes the specified file.

SetLock(fid, mode)

Sets a lock on the specified file or directory. The mode of the lock may be shared or exclusive. Locks that are not removed expire after 30 minutes.

ReleaseLock(fid)

Unlocks the specified file or directory.

RemoveCallback(fid)

Informs server that a Venus process has flushed a file from its cache.

BreakCallback(fid)

This call is made by a Vice server to a Venus process. It cancels the callback promise on the relevant file.

Note: Directory and administrative operations (*Rename, Link, Makedir, Removedir, GetTime, CheckToken*, etc.) are not shown.