NetBSD Subfiles

William Dobbins[1] Philip Nelson[1]

[1]Western Washington University

EuroBSDcon, 2016

Subfiles: An Introduction

What is a subfile?

- A subfile is a regular file that is attached to another regular file instead of a directory.
- A regular file can have several subfiles, but a subfile can't have its own subfiles.

Subfiles: What do they look like?

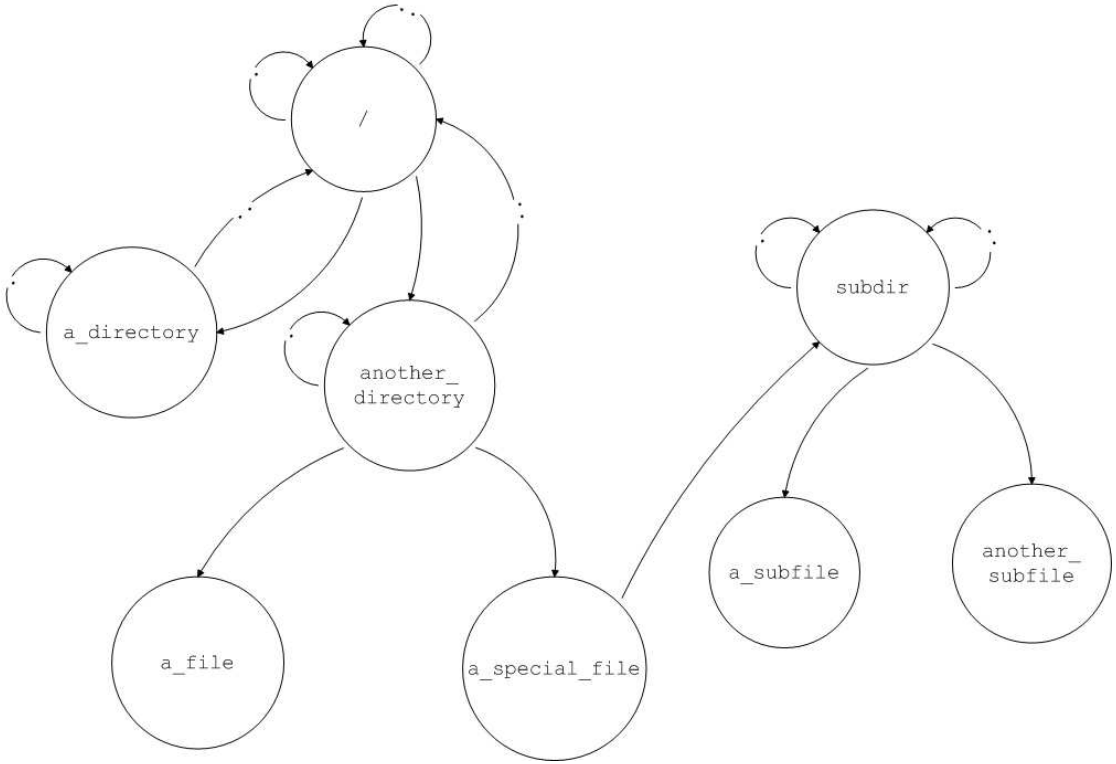

Figure 1: Subfiles in a simple filesystem

A Brief History of the Subfile

- Apple's HFS - Fork Filesystem [3]
  - Early implementations used two forks, a data fork and a resource fork.
  - Later implementations allowed multiple generic forks.
- Microsoft's NTFS - Alternate Data Streams [2]
  - Named and unnamed data streams that have a predetermined size limit.
- Solaris' UFS - Extended Attributes [1]
  - Subfiles attached to an inode that is attached to file.
  - Subfiles have unique owners and permissions and do not add to the size of the parent file.

Design

1. Only regular files shall have subfiles.
2. Subfiles shall be regular files.
3. Subfiles shall be stored in a directory (referred to as a subdir) attached to a parent file's inode.
4. The subdir shall be owned by the same user and group as the parent file.
5. The subdir's permissions shall be determined based on the parent file's permissions.
6. The subdir's . and .. fields shall point back to the subdir itself.
7. Subfiles shall not have their own subfiles.
8. The subdir shall not contain a directory.
9. There shall be a method of listing the contents of a subdir.

Subfiles: Is that what they look like?

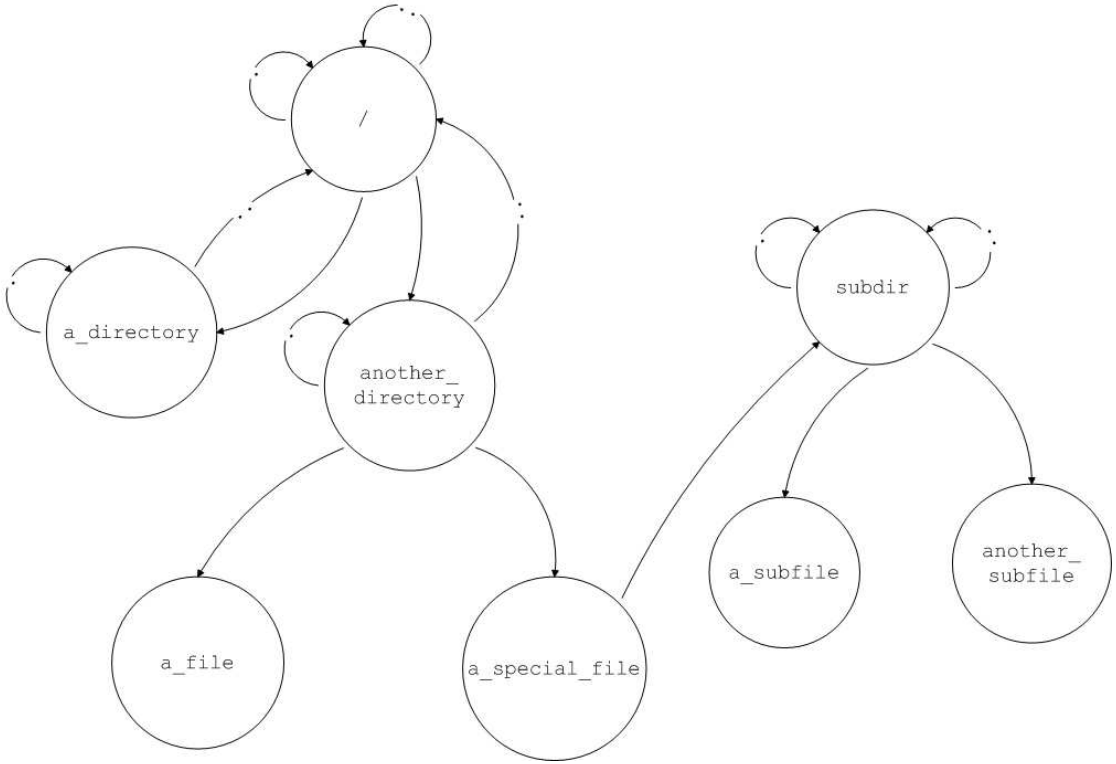

Figure 2: Subfiles in a simple filesystem

New System Call

```
int
sys_fsubfile_open(struct lwp *l,
        const struct sys_fsubfile_open_args *uap,
        register_t *retval)
{
    /* {
         syscallarg(int) fd;
         syscallarg(char *) name;
         syscallarg(int) flags;
         syscallarg(int) mode;
    } */
⋮

}
```

What does the new system call do?

- Check the file name for implied nesting ('/')
- Read or write flags
- Get the parent file vnode and lock it
- Check read and write permissions and file type for the parent file
- Check that a subdir already exists
- Pass the vnode to the FFS layer

New FFS Function

```
int
ufs_subdir_open (void *v)
{
    struct vop_subdir_open_args /* {
        struct vnode              *a_fvp;
        struct vnode              **a_svp;
        struct componentname      *a_cnp;
        kath_cred_t               a_cred;
    } */ *ap = v;
⋮

}
```

What does ufs_subdir_open() do?

- Create a new directory (subdir)
- '.' and '..' point back to the new directory
- Assign permissions to the subdir
  - Read and write are the same as the parent file
  - Execute is the logical OR of the parents' read and write permissions
- Don't enter the new directory in another directory
- Return the subdir vnode

Back to the system call

- Make sure the returned vnode is valid
- Pass the subdir vnode to open() as a starting point for lookup()

Other Changes

- mkdir() checks the the subdir_id of the parent. If the subdir_id is not 0, no new directory can be created.
- A new vnode flag indicates that a file is a subfile
- Two new stat() flags to indicate that a file is or has a subfile
- chdir() and chmod() exit if passed a subdir

How do we use these new functions?

1. subfile_open
2. open_subdir
3. subfile_unlink
4. subfile_stat
5. subfile_access

Future Work

1. libc
2. fsck
3. ar, tar, dump, and restore
4. chmod and chown
5. unlink

Questions

References

Mauro, J., and McDougall, R.
Solaris Internals (2Nd Edition).
Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

Nagar, R.
Windows NT File System Internals: A Developer's Guide.
O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1997.

Rose, C., and Hacker, B.
Inside Macintosh.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.