```
/* $Id: nfs_prot.ps,v 1.1 2000/09/15 16:21:27 rtm Exp $ */

/*
 * Sun RPC is a product of Sun Microsystems, Inc. and is provided for
 * unrestricted use provided that this legend is included on all tape
 * media and as a part of the software program in whole or part.  Users
 * may copy or modify Sun RPC without charge, but are not authorized
 * to license or distribute it to anyone else except as part of a product or
 * program developed by the user.
 *
 * SUN RPC IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE
 * WARRANTIES OF DESIGN, MERCHANTIBILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.
 *
 * Sun RPC is provided with no support and without any obligation on the
 * part of Sun Microsystems, Inc. to assist in its use, correction,
 * modification or enhancement.
 *
 * SUN MICROSYSTEMS, INC. SHALL HAVE NO LIABILITY WITH RESPECT TO THE
 * INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY SUN RPC
 * OR ANY PART THEREOF.
 *
 * In no event will Sun Microsystems, Inc. be liable for any lost revenue
 * or profits or other special, indirect and consequential damages, even if
 * Sun has been advised of the possibility of such damages.
 *
 * Sun Microsystems, Inc.
 * 2550 Garcia Avenue
 * Mountain View, California  94043
 */

const NFS_PORT          = 2049;
const NFS_MAXDATA       = 8192;
const NFS_MAXPATHLEN    = 1024;
const NFS_MAXNAMLEN     = 255;
const NFS_FHSIZE        = 32;
const NFS_COOKIESIZE    = 4;
const NFS_FIFO_DEV      = -1;   /* size kludge for named pipes */

/*
 * File types
 */
const NFSMODE_FMT  = 0170000;   /* type of file */
const NFSMODE_DIR  = 0040000;   /* directory */
const NFSMODE_CHR  = 0020000;   /* character special */
const NFSMODE_BLK  = 0060000;   /* block special */
const NFSMODE_REG  = 0100000;   /* regular */
const NFSMODE_LNK  = 0120000;   /* symbolic link */
const NFSMODE_SOCK = 0140000;   /* socket */
const NFSMODE_FIFO = 0010000;   /* fifo */

/*
 * Error status
 */
enum nfsstat {
        NFS_OK= 0,              /* no error */
        NFSERR_PERM=1,          /* Not owner */
        NFSERR_NOENT=2,         /* No such file or directory */
        NFSERR_IO=5,            /* I/O error */
        NFSERR_NXIO=6,          /* No such device or address */
        NFSERR_ACCES=13,        /* Permission denied */
        NFSERR_EXIST=17,        /* File exists */
        NFSERR_NODEV=19,        /* No such device */
        NFSERR_NOTDIR=20,       /* Not a directory*/
        NFSERR_ISDIR=21,        /* Is a directory */
        NFSERR_FBIG=27,         /* File too large */
        NFSERR_NOSPC=28,        /* No space left on device */
        NFSERR_ROFS=30,         /* Read-only file system */
        NFSERR_NAMETOOLONG=63,  /* File name too long */
        NFSERR_NOTEMPTY=66,     /* Directory not empty */
        NFSERR_DQUOT=69,        /* Disc quota exceeded */
        NFSERR_STALE=70,        /* Stale NFS file handle */
        NFSERR_WFLUSH=99        /* write cache flushed */
};

/*
 * File types
 */
enum ftype {
        NFNON = 0,      /* non-file */
        NFREG = 1,      /* regular file */
        NFDIR = 2,      /* directory */
        NFBLK = 3,      /* block special */
        NFCHR = 4,      /* character special */
        NFLNK = 5,      /* symbolic link */
        NFSOCK = 6,     /* unix domain sockets */
        NFBAD = 7,      /* unused */
        NFFIFO = 8      /* named pipe */
};

/*
 * File access handle
 */
struct nfs_fh {
        opaque data[NFS_FHSIZE];
};

/*
 * Timeval
 */
struct nfstime {
        unsigned seconds;
        unsigned useconds;
};


/*
 * File attributes
 */
struct fattr {
        ftype type;             /* file type */
        unsigned mode;          /* protection mode bits */
        unsigned nlink;         /* # hard links */
        unsigned uid;           /* owner user id */
        unsigned gid;           /* owner group id */
        unsigned size;          /* file size in bytes */
        unsigned blocksize;     /* prefered block size */
        unsigned rdev;          /* special device # */
        unsigned blocks;        /* Kb of disk used by file */
        unsigned fsid;          /* device # */
        unsigned fileid;        /* inode # */
        nfstime atime;          /* time of last access */
        nfstime mtime;          /* time of last modification */
        nfstime ctime;          /* time of last change */
};

/*
 * File attributes which can be set
 */
struct sattr {
```

```
        unsigned mode;  /* protection mode bits */
        unsigned uid;   /* owner user id */
        unsigned gid;   /* owner group id */
        unsigned size;  /* file size in bytes */
        nfstime atime;  /* time of last access */
        nfstime mtime;  /* time of last modification */
};


typedef string filename<NFS_MAXNAMLEN>;
typedef string nfspath<NFS_MAXPATHLEN>;

/*
 * Reply status with file attributes
 */
union attrstat switch (nfsstat status) {
case NFS_OK:
        fattr attributes;
default:
        void;
};

struct sattrargs {
        nfs_fh file;
        sattr attributes;
};

/*
 * Arguments for directory operations
 */
struct diropargs {
        nfs_fh  dir;    /* directory file handle */
        filename name;          /* name (up to NFS_MAXNAMLEN bytes) */
};

struct diropokres {
        nfs_fh file;
        fattr attributes;
};

/*
 * Results from directory operation
 */
union diropres switch (nfsstat status) {
case NFS_OK:
        diropokres reply;
default:
        void;
};

union readlinkres switch (nfsstat status) {
case NFS_OK:
        nfspath data;
default:
        void;
};

/*
 * Arguments to remote read
 */
struct readargs {
        nfs_fh file;            /* handle for file */
        unsigned offset;        /* byte offset in file */
        unsigned count;         /* immediate read count */
        unsigned totalcount;    /* total read count (from this offset)*/
};

/*
 * Status OK portion of remote read reply
 */
struct readokres {
        fattr   attributes;     /* attributes, need for pagin*/
        opaque data<NFS_MAXDATA>;
};

union readres switch (nfsstat status) {
case NFS_OK:
        readokres reply;
default:
        void;
};

/*
 * Arguments to remote write
 */
struct writeargs {
        nfs_fh  file;           /* handle for file */
        unsigned beginoffset;   /* beginning byte offset in file */
        unsigned offset;        /* current byte offset in file */
        unsigned totalcount;    /* total write count (to this offset)*/
        opaque data<NFS_MAXDATA>;
};

struct createargs {
        diropargs where;
        sattr attributes;
};

struct renameargs {
        diropargs from;
        diropargs to;
};

struct linkargs {
        nfs_fh from;
        diropargs to;
};

struct symlinkargs {
        diropargs from;
        nfspath to;
        sattr attributes;
};


typedef opaque nfscookie[NFS_COOKIESIZE];

/*
 * Arguments to readdir
 */
struct readdirargs {
        nfs_fh dir;             /* directory handle */
        nfscookie cookie;
        unsigned count;         /* number of directory bytes to read */
};

struct entry {
        unsigned fileid;
        filename name;
        nfscookie cookie;
```

```
	entry *nextentry;
};

struct dirlist {
	entry *entries;
	bool eof;
};

union readdirres switch (nfsstat status) {
case NFS_OK:
	dirlist reply;
default:
	void;
};

struct statfsokres {
	unsigned tsize;	/* preferred transfer size in bytes */
	unsigned bsize;	/* fundamental file system block size */
	unsigned blocks;	/* total blocks in file system */
	unsigned bfree;	/* free blocks in fs */
	unsigned bavail;	/* free blocks avail to non-superuser */
};

union statfsres switch (nfsstat status) {
case NFS_OK:
	statfsokres reply;
default:
	void;
};

/*
 * Remote file service routines
 */
program NFS_PROGRAM {
	version NFS_VERSION {
		void
		NFSPROC_NULL(void) = 0;

		attrstat
		NFSPROC_GETATTR(nfs_fh) =	1;

		attrstat
		NFSPROC_SETATTR(sattrargs) = 2;

		void
		NFSPROC_ROOT(void) = 3;

		diropres
		NFSPROC_LOOKUP(diropargs) = 4;

		readlinkres
		NFSPROC_READLINK(nfs_fh) = 5;

		readres
		NFSPROC_READ(readargs) = 6;

		void
		NFSPROC_WRITECACHE(void) = 7;

		attrstat
		NFSPROC_WRITE(writeargs) = 8;

		diropres
		NFSPROC_CREATE(createargs) = 9;

		nfsstat
		NFSPROC_REMOVE(diropargs) = 10;

		nfsstat
		NFSPROC_RENAME(renameargs) = 11;

		nfsstat
		NFSPROC_LINK(linkargs) = 12;

		nfsstat
		NFSPROC_SYMLINK(symlinkargs) = 13;

		diropres
		NFSPROC_MKDIR(createargs) = 14;

		nfsstat
		NFSPROC_RMDIR(diropargs) = 15;

		readdirres
		NFSPROC_READDIR(readdirargs) = 16;

		statfsres
		NFSPROC_STATFS(nfs_fh) = 17;
	} = 2;
} = 100003;
```