File System I/O

CST 357/457 – Systems Programming
Michael Ruth, Ph.D.
Associate Professor
Computer Science & I.T.
mruth@roosevelt.edu

Objectives

- Discuss file system IO including file metadata, stat, permissions & ownership
- Discuss directory including entries, creation, removal, opening directory streams, and reading directories
- Explain links including hard links and symbolic links and their system calls
- Discuss copying and moving files

Files & Their Metadata

- Recall that each file is referenced by an ***inode*** addressed by an ***inode number***
  - The inode is both a physical object located on the disk and a logical entity complete with a data structure in the kernel
  - The inode stores the ***metadata*** associated with a file
    - You can obtain the inode number using –i flag with ls command

The stat family

- UNIX provides a family of functions to obtain the metadata of a file:
  - `#include <sys/types.h>`
  - `#include <sys/stat.h>`
  - `#include <unistd.h>`

  - `int stat(const char *path, struct stat *buff)`
  - `int fstat(int fd, struct stat *buff)`
  - `int lstat(const char *path, struct stat *buff)`

struct stat

```
struct stat {
   dev_t st_dev;            /* Device ID containing file */
   ino_t st_ino;            /* inode number */
   mode_t st_mode;          /* permissions*/
   nlink_t st_nlink;        /* number of hard links */
   uid_t st_uid;            /* user ID of owner */
   gid_t st_gid;            /* group ID of owner */
   dev_t st_rdev;           /* device ID (special file) */
   off_t st_size;           /* total size in bytes */
   blksize_t st_blksize;    /* block size for file system */
   blkcnt_t st_blocks;      /* number of blocks */
   time_t st_atime;         /* last access time */
   time_t st_mtime;         /* last modification time */
   time_t st_ctime;         /* last status change time */
 };
```


stat Ex

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
   struct stat sb;
   int ret;

   if (argc < 2){ /* error */ }

   ret = stat(argv[1], &sb);

   if (ret) {/* error */ }

   printf("%s is %ld bytes\n",argv[1],sb.st_size);

}
```

stat Ex 2

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
    struct stat sb;
    int ret;

    if (argc < 2){ /* error */ }

    ret = stat(argv[1], &sb);

    if (ret) {/* error */ }

    printf("File Type: ");
```


Stat Ex 2 (cont)

```
switch(sb.st_mode & S_IFMT) {
    case S_IFBLK:
        printf("block device node \n");
        break;
    case S_IFCHR:
        printf("character device node \n");
        break;
    case S_IFDIR:
        printf("Directory\n");
        break;
    case S_IFIFO:
        printf("FIFO\n");
        break;
    case S_IFLNK:
        printf("Symbolic Link\n");
        break;
    case S_IFREG:
        printf("Regular File\n");
        break;
    case S_IFSOCK:
        printf("Socket\n");
        break;
    default:
        printf("Unknown\n");
        break;
}}
```


Permissions

- Stat can be used to obtain the permissions, two other calls set those values
  - `#include <sys/types.h>`
  - `#include <sys/stat.h>`
  - `int chmod(const char *path, mode_t mode)`
  - `int fchmod(int fd, mode_t mode)`

Ownership

- In stat, the st_uid and st_gid fields provide the file's owner/group, we can set them using the following:
  - `#include <sys/types.h>`
  - `#include <unistd.h>`

  - `int chown(const char *path, uid_t owner, gid_t group)`
  - `int lchown(const char *path, uid_t owner, gid_t group)`
  - `int fchown(int fd, uid_t owner, gid_t group)`

Directories

- In UNIX, a directory is a simple file consisting of a list of directory entries
  - Each entry is a inode-name mapping AKA Link
  - Every directory contains two special directories:
    - . -> right here
    - .. -> parent (except for root directory)

Current Working Directory

- Every process has a current directory which is called its current working directory (cwd)
  - Starting point for resolving relative pathnames
  - A process can obtain & change its cwd

Obtaining/Changing CWD

```
#include <unistd.h>

char * getcwd(char *buff, size_t
size);

int chdir(const char *path);
int fchdir(int fd);
```

- We often get CWD so we can return to it later…
  - If this is the case, use the fd as it is more efficient

Creating Directories

```
#include <sys/stat.h>
#include <sys/types.h>

int mkdir(const char *path,
mode_t mode)

Final perm:
    (mode & ~umask & 01777)
```


Removing Directories

```
#include <unistd.h>

int rmdir(const char *path);

NOTE:
    there is no recursive
    version!
```

Reading a Directory's Contents

- To read a directory's contents, you need a ***directory stream*** represented by ***DIR***
  - `#include <sys/types.h>`
  - `#include <dirent.h>`
  - `DIR * opendir(const char *name)`

Directory Stream

- A directory stream is basically a file descriptor with some metadata
  - We can obtain the fd if necessary
  - `#define _BSD_SOURCE /*or _SVID_SOURCE */`
  - `#include <sys/types.h>`
  - `#include <dirent.h>`
  - `int dirfd(DIR *dir);`

Reading a Directory's Contents (2)

- Once you have a directory stream, you can read the directory entries:

```
#include <sys/types.h>
#include <dirent.h>

struct dirent * readdir(DIR *dir)

struct dirent {
   ino_t d_ino; /* inode number */
   off_t d_off; /* offset to next dirent */
   unsigned short d_reclen; /* length of record */
   unsigned char d_type; /* type of file */
   char d_name[256]; /* filename */
}
```

Closing a Directory Stream

```
#include <sys/types.h>
#include <dirent.h>

int closedir(DIR *dir)
```


Links

- A link is essentially a name for an inode
  - We can create more than one link to the same inode
    - Only on the same file system though
    - We call these ***hard links***
  - Another type of link is not a filesystem mapping, but a higher level pointer than is interpreted at runtime
    - We call these ***symbolic links***
    - May be relative or absolute
    - May point to nonexistent files (***dangling symlink***)

Hard Links

```
#include <unistd.h>

int link(const char *oldpath,
const char *newpath)
```

Symbolic Links

```
#include <unistd.h>

int symlink(const char
*oldpath, const char
*newpath)
```


Unlinking & Remove

```
#include <unistd.h>

int unlink(const char *path)

#include <stdio.h>

int remove(const char *path);
```


Copying & Moving Files

- There is no system call for copying a file
  - We just open the old file, open the new file, copy the file byte by byte, and close both
- For moving, there is a system call:
  - `#include <stdio.h>`
  - `int rename(const char *old, const char *new);`

Summary

- Discussed file system IO including file metadata, stat, permissions & ownership
- Discussed directory including entries, creation, removal, opening directory streams, and reading directories
- Explained links including hard links and symbolic links and their system calls
- Discussed copying and moving files

Questions?