CSC 4304 - Systems Programming
Fall 2010

Lecture - VI

Files & Directories

Tevfik Koşar

Louisiana State University
September 9th, 2010

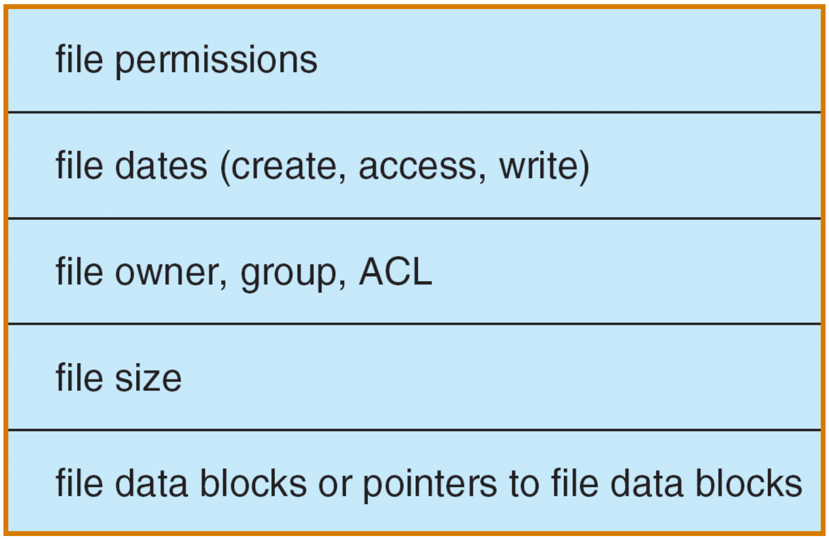
File Systems

- Provides organized and efficient access to data on secondary storage:

  1. Organizing data into files and directories and supporting primitives to manipulate them (create, delete, read, write etc)
  2. Improve I/O efficiency between disk and memory (perform I/O in units of blocks rather than bytes)
  3. Ensure confidentiality and integrity of data

  - Contains file structure via a File Control Block (FCB)
    - Ownership, permissions, location..

A Typical File Control Block



file permissions

file dates (create, access, write)

file owner, group, ACL

file size

file data blocks or pointers to file data blocks

File Properties

- Objectives
  - Additional Features of the File System
  - Properties of a File.

```
struct stat {
    mode_t   st_mode; /* type & mode */
    ino_t    st_ino; /* i-node number */
    dev_t    st_dev; /* device no (filesystem) */
    dev_t    st_rdev; /* device no for special file */
    nlink_t  st_nlink; /* # of links */
    uid_t    st_uid;          gid_t     st_gid;
    off_t    st_size; /* sizes in byes */
    time_t   st_atime; /* last access time */
    time_t   st_mtime; /* last modification time */
    time_t   st_ctime; /* time for last status change */
    long     st_blk_size; /* best I/O block size */
    long     st_blocks; /* number of 512-byte blocks allocated */
  };
```

Stat Functions

- Three major functions:

```
#include <sys/types.h>
#include <sys/stat.h>
int stat(const char *pathname, struct stat *buf);
int fstat(int filedes, struct stat *buf);
int lstat(const char *pathname, struct stat *buf);
```

☑ stat returns info about a named file

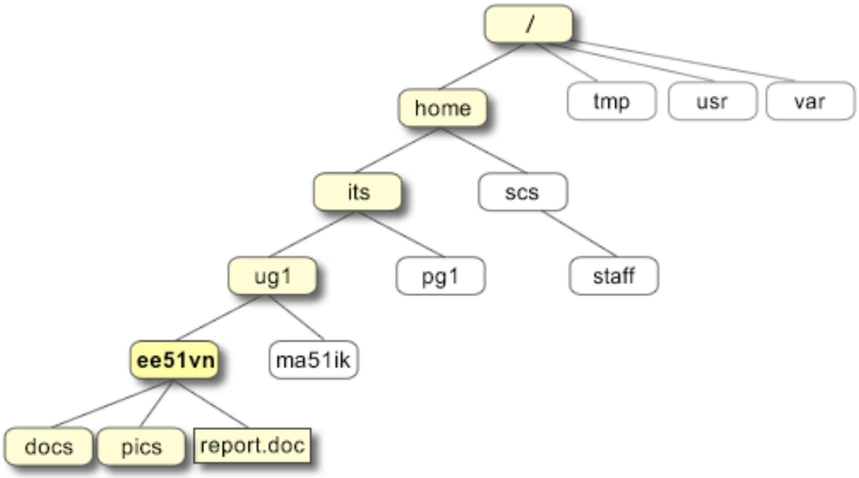
☑ fstat returns info about an already open file

☑ lstat returns info about a symbolic link, not the referenced file

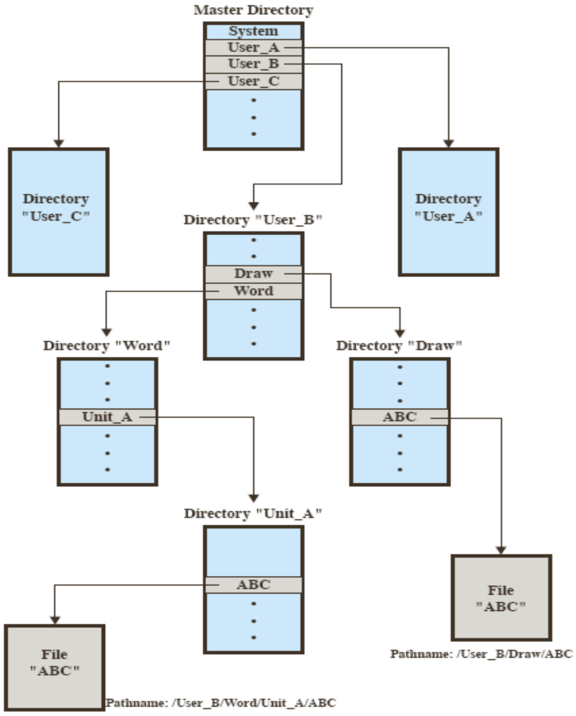
Directories

- Directories are special files that keep track of other files
  - ✓ the collection of files is systematically organized
  - ✓ first, disks are split into partitions that create logical volumes (can be thought of as “virtual disks”)
  - ✓ second, each partition contains information about the files within
  - ✓ this information is kept in entries in a **device directory** (or volume table of contents)
  - ✓ the directory is a symbol table that translates file names into their entries in the directory
    - it has a logical structure
    - it has an implementation structure (linked list, table, etc.)

Unix Directory Tree Structure



Directory Implementation



Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Directories

- Directory is a special file that contains list of names of files and their inode numbers
- to see contents of a directory:

```
$ls -1ia .
9535554 .
9535489 ..
9535574 .bash_history
9535555 bin
9535584 .emacs.d
9535560 grading
9535803 hw1
9535571 test
9535801 .viminfo
```

Example

Example inode listing

```
$ ls -iaR demodir
865 .       193 ..      277 a       520 c      491 y

demodir/a:
277 .       865 ..      402 x

demodir/c:
520 .      865 ..      651 d1        247 d2

demodir/c/d1:
651 .      520 ..      402 xlink

demodir/c/d2:
247 .      520 ..      680 xcopy
```

Directories - System View

- user view vs system view of directory tree
  - representation with “dirlists (directory files)”
- The real meaning of “A file is in a directory”
  - directory has a link to the inode of the file
- The real meaning of “A directory contains a subdirectory”
  - directory has a link to the inode of the subdirectory
- The real meaning of “A directory has a parent directory”
  - “..” entry of the directory has a link to the inode of the parent directory

Link Counts

- The kernel records the number of links to any file/directory.
- The *link count* is stored in the inode.
- The *link count* is a member of *struct stat* returned by the *stat* system call.

Change Links

- What will be the resulting changes in directory tree?
- rename y c/d1/y.old
- cp a/x c
- ln a/x c/d2/x

Implementing “pwd”

1. “.” is 247
   chdir ..

2. 247 is called “d2”
   “.” is 520
   chdir ..

3. 520 is called “c”
   “.” is 865
   chdir ..

4. 865 is called “demodir”
   “.” is 193
   chdir ..

Stat Struct

- Objectives
  - Additional Features of the File System
  - Properties of a File.

```
struct stat {
    mode_t   st_mode; /* type & mode */
    ino_t    st_ino; /* i-node number */
    dev_t    st_dev; /* device no (filesystem) */
    dev_t    st_rdev; /* device no for special file */
    nlink_t  st_nlink; /* # of links */
    uid_t    st_uid;        gid_t    st_gid;
    off_t    st_size; /* sizes in byes */
    time_t   st_atime; /* last access time */
    time_t   st_mtime; /* last modification time */
    time_t   st_ctime; /* time for last status change */
    long     st_blk_size; /* best I/O block size */
    long     st_blocks; /* number of 512-byte blocks allocated */
};
```

Dirent Struct

- dirent : file system independent directory entry

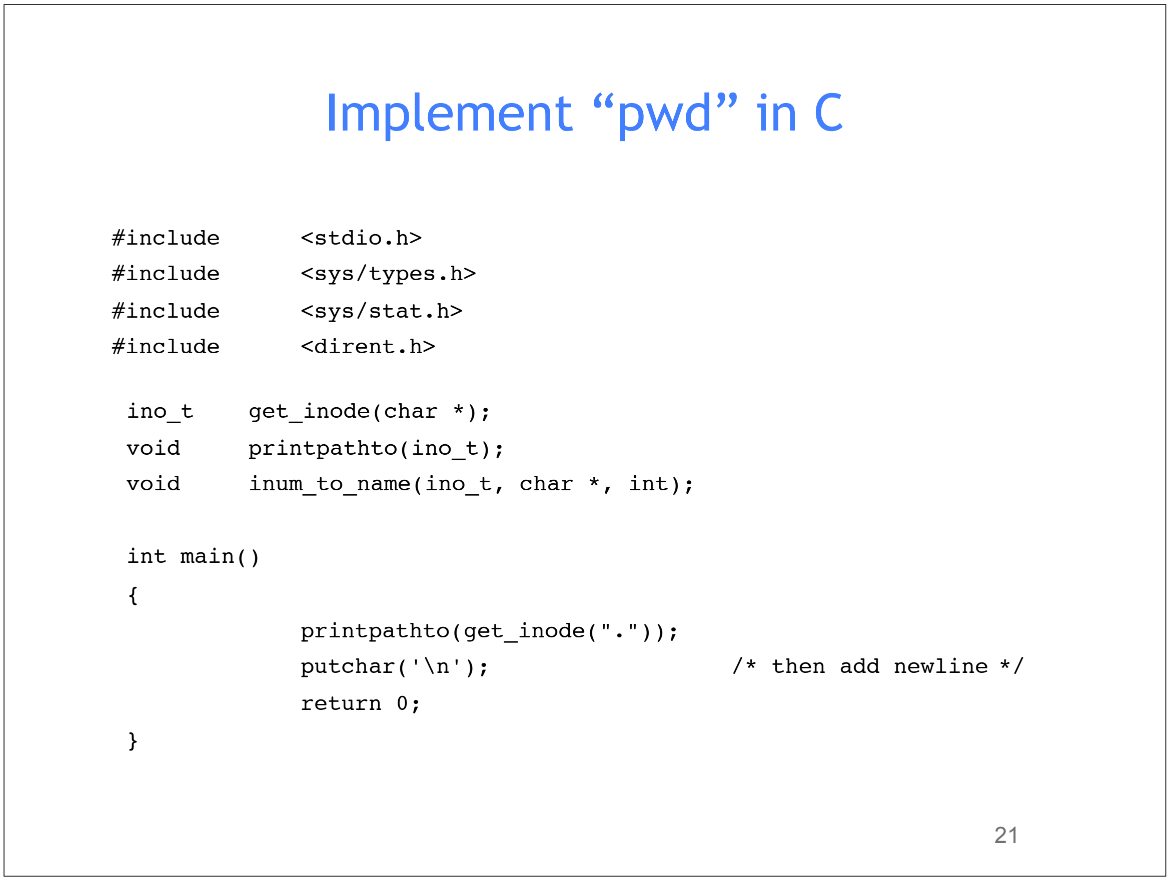
```
struct dirent{
     ino_t  d_ino;
     char   d_name[];
     ….
};
```

- **`ino_t get_inode(char *fname);`**

```
// returns inode number for the file

{
     struct stat info;

     if ( stat(fname, &info) == -1 ){
          fprinf(stderr, “Cannot stat!”);
          exit(1);
     }
     return info.st_ino;
}
```

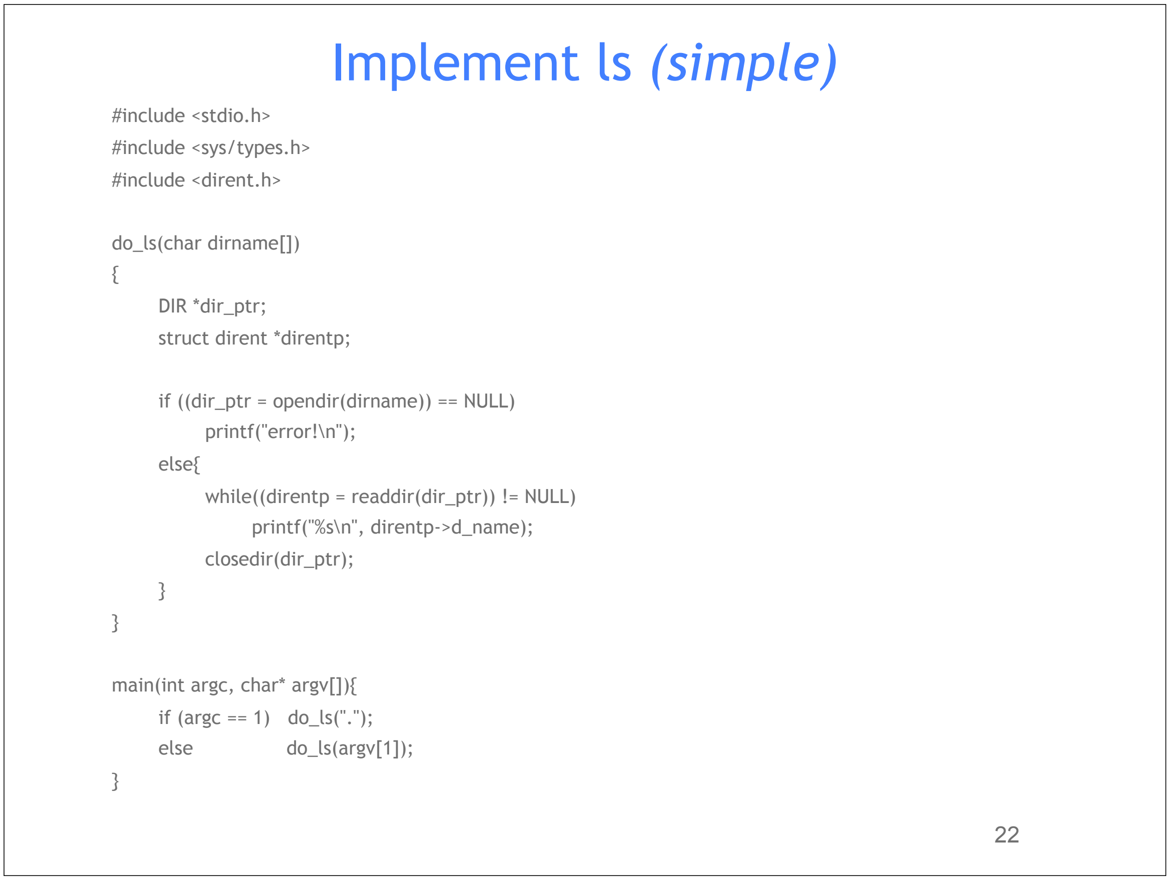
```
•  void printpathto( ino_t this_inode )
//  prints path leading down to an object with this inode

{
    ino_t    my_inode;
    char     its_name[BUFSIZ];

    if (get_inode("..") != this_node)
    {
         chdir(“..”);                  /* up one dir    */
          inum_to_name(this_inode, its_name, BUFSIZ);    /* get its name  */
         my_inode = get_inode(“.”);
         printpathto(my_inode);
         printf(“%s”, its_name);
     }
 }
```

```
•  void inum_to_name(ino_t inode_to_find, char *namebuf, int buflen)
/*
 *  looks through current directory for a file with this inode
 *  number and copies its name into namebuf
 */
{
    DIR      *dir_ptr;                /* the directory */
    struct dirent    *direntp;         /* each entry     */

    dir_ptr = opendir( "." );
    if ( dir_ptr == NULL ){
        fprintf(stderr, "cannot open a directory\n");
        exit(1);
    }

    //search directory for a file with specified inum

    while ( ( direntp = readdir(dir_ptr) ) != NULL ){
        if (direntp->d_ino == inode_to_find)
        {
            strcpy( namebuf, direntp->d_name, buflen);
            namebuf[buflen-1] = ‘\0’
            closedir( dir_ptr );
            return;
        }
    }
    fprintf(stderr, "error looking for inum %d\n", inode_to_find);
    exit(1);
}
```



Implement “pwd” in C

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

 ino_t    get_inode(char *);
 void     printpathto(ino_t);
 void     inum_to_name(ino_t, char *, int);

 int main()
 {
             printpathto(get_inode("."));
             putchar('\n');                  /* then add newline */
             return 0;
 }
```



Implement ls *(simple)*

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

do_ls(char dirname[])
{
    DIR *dir_ptr;
    struct dirent *direntp;

    if ((dir_ptr = opendir(dirname)) == NULL)
        printf("error!\n");
    else{
        while((direntp = readdir(dir_ptr)) != NULL)
            printf("%s\n", direntp->d_name);
        closedir(dir_ptr);
    }
}

main(int argc, char* argv[]){
    if (argc == 1)   do_ls(".");
    else             do_ls(argv[1]);
}
```

Get File Info

```
show_stat_info(char *fname, struct stat *buf)
{
        printf("   mode: %o\n", buf->st_mode);
        printf("  links: %d\n", buf->st_nlink);
        printf("   user: %d\n", buf->st_uid);
        printf("  group: %d\n", buf->st_gid);
        printf("   size: %d\n", buf->st_size);
        printf("modtime: %d\n", buf->st_mtime);
        printf("   name: %s\n", fname);

}
main(int argc, char* argv[]){
        struct stat info;

        if (argc > 1)
                if (stat(argv[1], &info) != -1){
                        show_stat_info(argv[1], &info);
                }

}
```

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

Get More Info

```
show_stat_info(char *fname, struct stat *buf)
{
        struct passwd *user = getpwuid(buf->st_uid);
        struct group *gr = getgrgid(buf->st_gid);

        printf("   mode: %o\n", buf->st_mode);
        printf("  links: %d\n", buf->st_nlink);
        printf("   user: %s\n", user->pw_name);
        printf("  group: %s\n", gr->gr_name);
        printf("   size: %d\n", buf->st_size);
        printf("modtime: %d\n", buf->st_mtime);
        printf("   name: %s\n", fname);

}
main(int argc, char* argv[]){
        struct stat info;

        if (argc > 1)
                if (stat(argv[1], &info) != -1){
                        show_stat_info(argv[1], &info);
                }

}
```

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <pwd.h>
#include <grp.h>
```

Summary

- Files and Directories
  - user view vs system view
  - inode listing
  - stat struct
  - dirent struct
  - implementing pwd
  - implementing ls

- Next Class: Unix Process Environment

- Project 1 out next class
- Read Ch 4 from Stevens..

Acknowledgments

- Advanced Programming in the Unix Environment by R. Stevens
- The C Programming Language by B. Kernighan and D. Ritchie
- Understanding Unix/Linux Programming by B. Molay
- Lecture notes from B. Molay (Harvard), T. Kuo (UT-Austin), G. Pierre (Vrije), M. Matthews (SC), and B. Knicki (WPI).