Topics: Directories, File Attributes, Bits, File Operations

Approach: Write our own versions of Unix programs

Featured Commands:
ls, ls -l

Main Ideas:
A directory is a list of file names
File attributes can be numerical values
File attributes can be yes/no settings
Bitwise operations (|, &)

Agenda

stdio	fopen/getc/putc/fclose vs open/read/write/close
ls1	opendir(), readdir(), closedir() not sorted, and does not support -l
file attributes	owner, dates, size, type, permissions look at chmod(), chown(), chgrp(), touch
stat1	stat() system call
stat2	coding yes-no attributes as bits binary, octal, bit operations
ls2	combine ls1 and stat2 still not sorted, and what about ls -l filename

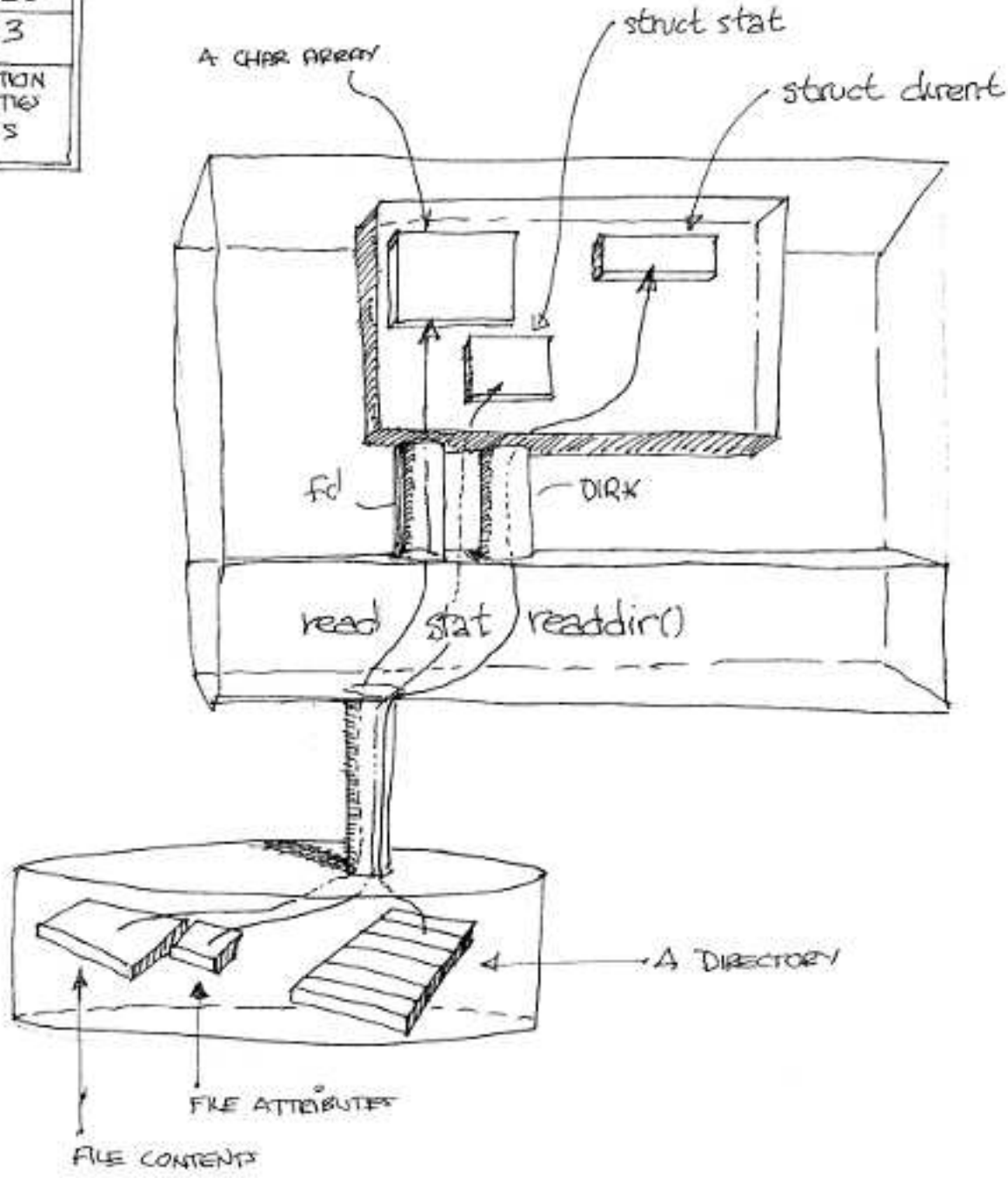
Sections

0. First, one more item about buffering:
   * people asked after class about using open,read,write,close vs fopen, getc, putc, fclose -- what's the difference?
   * Answer - the standard library adds buffering a FILE points to a struct containing a (a) ptr to a buffer, (b) some pointers to keep track of where we are, (c) file descriptor, (d) type of open(read/write/both)

   Thus, getc or fgets do what our utmplib do.
   The include file <libio.h> line 271 has the def

1. files have content AND properties
2. Using ls
3. A brief review of the Unix directory tree
4. Is this just like who? open, read, show, close
5. What is a directory?
6. Reading a directory
7. How close are we to a complete version of ls?
8. How do we add the -l option?
9. How does stat() work?
10. What info do we need from what stat tells us?
11. How do we decode the type and file permissions?
12. How do we convert uid to logname?
13. How do we convert gid to group?
14. Building stat2.c : format info just like ls -l
15. Can we combine stat2.c with ls1.c ?

CSCI-E28
Lecture 3
ORGANIZATION & PROPERTIES OF FILES



FACTS : THE FILE SYSTEM STORES FILES

FILES HAVE CONTENT (use read)
FILES HAVE ATTRIBUTES (use stat)

THE FILE SYSTEM STORES DIRECTORIES

DIRECTORIES HAVE CONTENT (USE READDIR)
DIRECTORIES HAVE ATTRIBUTES (USE STAT)

```
:::::::::::::: ls1.c ::::::::::::::
/** ls1.c
 **   purpose  list contents of directory or directories
 **   action   if no args, use .  else list files in args
 **/
#include        <stdio.h>
#include        <sys/types.h>
#include        <dirent.h>

void do_ls(char []);

int main(int ac, char *av[])
{
        if ( ac == 1 )
                do_ls( "." );
        else
                while ( --ac ){
                        printf("%s:\n", *++av );
                        do_ls( *av );
                }
        return 0;
}

void do_ls( char dirname[] )
/*
 *      list files in directory called dirname
 */
{
        DIR             *dir_ptr;               /* the directory */
        struct dirent   *direntp;               /* each entry    */

        if ( ( dir_ptr = opendir( dirname ) ) == NULL )
                fprintf(stderr,"ls1: cannot open %s\n", dirname);
        else
        {
                while ( ( direntp = readdir( dir_ptr ) ) != NULL )
                        printf("%s\n", direntp->d_name );
                closedir(dir_ptr);
        }
}
:::::::::::::: stat1.c ::::::::::::::
#include        <stdio.h>
#include        <sys/types.h>
#include        <sys/stat.h>

/*
 *      stat1.c         a user interface to the stat system call
 *                      for each arg, it lists all interesting
 *                      file info.  Has a lot of numbers, though..
 */

void dostat(char *);

int main( int ac, char **av)
{
        while ( --ac )
                dostat( *++av );
        return 0;
}

void dostat( char *filename )
{
        struct stat info;

        printf("%s:\n", filename );                     /* print name    */
        if ( stat(filename, &info) == -1 )              /* cannot stat   */
                perror( filename );                     /* say why       */
        else                                    /* else show info        */
        {
                printf("\t  mode: %o\n", (int) info.st_mode);   /* mode  */
                printf("\t links: %d\n", (int) info.st_nlink);  /* links */
                printf("\t owner: %d\n", (int) info.st_uid);    /* owner */
                printf("\t group: %d\n", (int) info.st_gid);    /* group */
                printf("\t  size: %ld\n",(long)info.st_size);   /* size  */
                printf("\t   mod: %ld\n",(long)info.st_mtime);  /* mod   */
                printf("\taccess: %ld\n",(long)info.st_atime);  /* access*/
                printf("\tdevice: %x\n", (int) info.st_dev);    /* device*/
        }
}
```

```
:::::::::::::: stat2.c ::::::::::::::
#include        <stdio.h>
#include        <sys/types.h>
#include        <sys/stat.h>
#include        <string.h>
#include        <time.h>

/*
 *      stat2.c         based on stat1.c but prints more stuff.
 *                      Chose one of two formats.
 */

/* #define      TAGGED_STYLE    */
#define LS_STYLE
#define TEXTDATE

#ifndef DATE_FMT
#ifdef  TEXTDATE
#define DATE_FMT        "%b %e %H:%M"           /* text format  */
#else
#define DATE_FMT        "%Y-%m-%d %H:%M"        /* the default  */
#endif
#endif

void dostat(char *);
void show_file_info(char *, struct stat *);
char *fmt_time(time_t, char *);

int main( int ac, char **av )
{
        while ( --ac )
                dostat( *++av );
        return 0;
}

void dostat( char *filename )
{
        struct stat info;

        if ( stat(filename, &info) == -1 )              /* cannot stat   */
                perror( filename );                     /* say why       */
        else                                    /* else show info        */
                show_file_info( filename, &info );
}

void show_file_info( char *filename, struct stat *info_p )
/*
 * display the info about 'filename'.  The info is stored in struct at *info_p
 */
{
        char    *uid_to_name(), *ctime(), *gid_to_name(),
                *mode_to_letters(), m[12];
#ifdef TAGGED_STYLE

        printf("%s:\n", filename );                     /* print name    */

        printf("\t  mode: %s\n", mode_to_letters(info_p->st_mode,m) );
        printf("\t links: %d\n", (int) info_p->st_nlink);       /* links */
        printf("\t owner: %s\n", uid_to_name(info_p->st_uid) );
        printf("\t group: %s\n", gid_to_name(info_p->st_gid) );
        printf("\t  size: %ld\n",(long)info_p->st_size);        /* size  */
        printf("\t   mod: %s",   ctime(&info_p->st_mtime));
        printf("\taccess: %s",   ctime(&info_p->st_atime));
#endif
#ifdef  LS_STYLE
        printf( "%s"    , mode_to_letters(info_p->st_mode,m) );
        printf( "%2d "  , (int) info_p->st_nlink);
        printf( "%-4s " , uid_to_name(info_p->st_uid) );
        printf( "%-8s " , gid_to_name(info_p->st_gid) );
        printf( "%ld " , (long)info_p->st_size);
        printf( "%s "   , fmt_time( info_p->st_mtime, DATE_FMT ) );
        printf( " %s\n"  , filename );
#endif

}
```

```
/**** utility functions ****/

#define MAXDATELEN      100

char * fmt_time( time_t timeval , char *fmt )
/*
 * formats time for human consumption.
 * Uses localtime to convert the timeval into a struct of elements
 * (see localtime(3)) and uses strftime to format the data
 */
{
        static char     result[MAXDATELEN];

        struct tm *tp = localtime(&timeval);            /* convert time */
        strftime(result, MAXDATELEN, fmt, tp);          /* format it    */
        return result;
}
/*
 * This function takes a mode value and a char array and puts into the
 * char array the file type and the nine letters for the bits in mode.
 * NOTE: It does not code setuid, setgid, and sticky codes
 */
char *mode_to_letters( int mode, char str[] )
{
    strcpy( str, "----------" );           /* default=no perms */
    /*            123456789               positions in string */
    if ( S_ISDIR(mode) )  str[0] = 'd';    /* directory?       */
    if ( S_ISCHR(mode) )  str[0] = 'c';    /* char devices     */
    if ( S_ISBLK(mode) )  str[0] = 'b';    /* block device     */

    if ( mode & S_IRUSR ) str[1] = 'r';    /* 3 bits for user  */
    if ( mode & S_IWUSR ) str[2] = 'w';
    if ( mode & S_IXUSR ) str[3] = 'x';

    if ( mode & S_IRGRP ) str[4] = 'r';    /* 3 bits for group */
    if ( mode & S_IWGRP ) str[5] = 'w';
    if ( mode & S_IXGRP ) str[6] = 'x';

    if ( mode & S_IROTH ) str[7] = 'r';    /* 3 bits for other */
    if ( mode & S_IWOTH ) str[8] = 'w';
    if ( mode & S_IXOTH ) str[9] = 'x';
    return str;
}
#include        <pwd.h>

char *uid_to_name( uid_t uid )
/*
 *      returns pointer to username associated with uid, uses getpw()
 */
{
        struct  passwd *pw_ptr;
        static  char numstr[10];

        if ( ( pw_ptr = getpwuid( uid ) ) != NULL )
                return pw_ptr->pw_name ;
        sprintf(numstr,"%d", uid);
        return numstr;
}
#include        <grp.h>

char *gid_to_name( gid_t gid )
/*
 *      returns pointer to group number gid. used getgrgid(3)
 */
{
        struct group *grp_ptr;
        static  char numstr[10];

        if ( ( grp_ptr = getgrgid(gid) ) != NULL )
                return grp_ptr->gr_name;
        sprintf(numstr,"%d", gid);
        return numstr;
}
```