

```

:..... 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("\t access: %ld\n", (long)info.st_atime); /* access */
        printf("\t device: %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;
}

```