```
:::::::::::::: ls1.c ::::::::::::::
#include       <stdio.h>
#include       <sys/types.h>
#include       <dirent.h>
/**
 **     ls version 1.0
 **             purpose  list contents of directory or directories
 **             action   if no args, use .  else list files in args
 **/
main(int ac, char *av[])
{
       if ( ac == 1 )
              do_ls( "." );
       else
              while ( --ac ){
                     printf("%s:\n", *++av );
                     do_ls( *av );
              }
}

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..
 */
main( int ac, char *av[] )
{
       while ( --ac )
              dostat( *++av );
}
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*/
       }
}
```

```
:::::::::::::: stat2.c ::::::::::::::

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

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

/* #define     TAGGED_STYLE   */
#define LS_STYLE

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

dostat( filename )
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 );
}

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(), *filemode();
#ifdef TAGGED_STYLE

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

        printf("\t  mode: %s\n", filemode(info_p->st_mode) );
        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"    , filemode(info_p->st_mode) );
        printf( "%4d "  , (int) info_p->st_nlink);
        printf( "%-8s " , uid_to_name(info_p->st_uid) );
        printf( "%-8s " , gid_to_name(info_p->st_gid) );
        printf( "%8ld " , (long)info_p->st_size);
        printf( "%.12s ", 4+ctime(&info_p->st_mtime));
        printf( "%s\n"  , filename );
#endif

}

/*
 * utility functions
 */
```

```
char *
filemode( int mode )
/*
 *      returns string of mode info
 *      default to ------- and then turn on bits
 */
{
        static  char    bits[11];
        char    type;

        strcpy( bits, "----------" );

        switch ( mode & S_IFMT ){                       /* mask for type */
                case S_IFREG:  type = '-';    break;  /* stays a dash         */
                case S_IFDIR:  type = 'd';    break;  /* put a d there */
                case S_IFCHR:  type = 'c';    break;  /* char i/o dev         */
                case S_IFBLK:  type = 'b';    break;  /* blk. i/o dev         */
                default:       type = '?';    break;  /* fifo, socket..*/
        }
        bits[0] = type ;

        /* do SUID, SGID, and SVTX later */

        permbits( mode>>6 , bits+1 );                   /* owner        */
        permbits( mode>>3 , bits+4 );                   /* group        */
        permbits( mode    , bits+7 );                   /* world        */

        return bits;
}

permbits( int permval, char *string )
/*
 *      convert bits in permval into chars rw and x
 */
{
        if ( permval & 4 )
                string[0] = 'r';
        if ( permval & 2 )
                string[1] = 'w';
        if ( permval & 1 )
                string[2] = 'x';
}

#include       <pwd.h>
char  *
uid_to_name( short uid )
/*
 *      returns pointer to logname associated with uid, uses getpw()
 */
{
        struct  passwd *getpwuid(), *pw_ptr;

        if ( ( pw_ptr = getpwuid( uid ) ) == NULL )
                return "Unknown" ;
        else
                return pw_ptr->pw_name ;
}

#include       <grp.h>
char *
gid_to_name( short gid )
/*
 *      returns pointer to group number gid. used getgrgid(3)
 */
{
        struct group *getgrgid(), *grp_ptr;

        if ( ( grp_ptr = getgrgid(gid) ) == NULL )
                return "Unknown" ;
        else
                return grp_ptr->gr_name;
}
```