```
/* sto: segin's STOrage format
 * If there are any linker errors with strnlen,
 * please compile the provided strnlen.c with this.
 */

/* If compiling with Digital Mars C or any other comiler capable of
 * generating Win32s binaries, please define DIGITAL_MARS_C with it.
 */

/* TODO: Use mmap() on the archive. */

/* If using Turbo C 2.0 */
/* #define MSDOS */

#define _GNU_SOURCE || _MINIX
#include <sys/types.h>
#include <sys/stat.h>
#ifndef MSDOS
#ifndef DIGITAL_MARS_C
#include <unistd.h>
#endif
#endif
#ifdef DIGITAL_MARS_C
#include <windows.h>
#endif

/* BCC compiler, NOT gcc, intel cc, etc. */
#if __BCC__
#if __AS386_16__
typedef unsigned long int uint32_t;
typedef unsigned int uint16_t;
#else /* 32-bit code */
typedef unsigned int uint32_t;
typedef unsigned short int uint16_t;
#endif
#else
#ifndef MSDOS
#include <stdint.h>
#endif
#endif
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#ifndef MSDOS
#ifndef DIGITAL_MARS_C
#include <strings.h>
#endif
#include <utime.h>
#endif
#ifdef MSDOS
typedef unsigned long int uint32_t;
#endif

typedef struct
{
        char magic[4];
        uint32_t os;
        uint32_t entries;
} sto_header;

typedef struct
{
        uint32_t fexist;
        uint32_t flen;
        char fname[256];
        uint32_t os;
        uint32_t fattrib;
        uint32_t fowner;
        uint32_t fgroup;
        time_t fatime;
        time_t fmtime;
        time_t fctime;
} sto_fentry;

#define STO_MAGIC "STO!"
#define OS_UNIX 1
#define OS_DOS  2 /* I don't see much hope of this ever being used */
#define OS_WIN  3
#define OS_DEADBEEF     0xDEADBEEF

static struct stat ostat;
#ifndef MSDOS
static struct utimbuf tb;
#endif

void badarg(void *mem)
{
        puts("usage: sto [c|x] archive.sto [files ...]");
        free(mem);
        exit(1);
}

#ifdef __WIN32__
sync(){}
#endif

#if defined(__WIN32__) || defined(MSDOS) || defined(__BCC__)
#include "strnlen.c"

/* Return the name-within-directory of a file name.
   Copyright (C) 1996,97,98,2002 Free Software Foundation, Inc.
   This function is part of the GNU C Library. */

char *
basename (filename)
     const char *filename;
{
  char *p = strrchr (filename,
#ifdef MSDOS
  '\\'
#else /* UNIX, Linux, ELKS, etc. */
  '/'
#endif /* MSDOS */
  );

  return p ? p + 1 : (char *) filename;
}
#endif

#ifdef MSDOS

/* MS-DOS does not have a sync() function, and
 * has no buffering system except SMARTdrive.
 * so call up smartdrv.exe with /c to flush
 * the buffers.
 *
 * We make sure that no nasty "Bad command or file name"
 * appears as well :)
 */

void sync(void)
{
        char *path;
        path = searchpath("SMARTDRV.EXE");
        if (path == NULL) return;
        system("smartdrv /c");
}

/* The C library provided with Turbo C 2.0 doesn't have bzero.
 * Just map the call to memset.
 */
#endif /* MSDOS */
#if __WIN32__ || MSDOS
void bzero(void *buf, size_t len)
{
        memset(buf,0,len);
        return;
}
#endif

int main(int argc, char **argv)
{
        FILE *fd;
        FILE *archive;
        int fstatus;
        int x,y;
        char *mem;
        sto_header header;
        sto_fentry fentry;

        /* Doing some basic size checks */
        if (sizeof(uint32_t) != 4) {
                puts("uint32_t not properly defined!");
                free(mem);
                exit(1);
        }

        mem = malloc(2);

        bzero(fentry.fname,256);

        /* Die if not enough arguments */
        if (argc < 2) {
                badarg(mem);
        }

        /* Die if first argument is longer than one letter */
        if (strnlen(argv[1],2) != 1) {
                badarg(mem);
        }

        /* Check if first arg is c or x, if not, die */
        switch(argv[1][0]) {
                case 'c':
                        /* Someone didn't specify a new archive name
                         * and a file to add
                         */
                        if (argc < 4) {
                                badarg(mem);
                        }
                        /* Here we add support for gzipping .sto archives
                         * with piping the .sto archive to gzip via
                         * stdout.
                         */
                        if (strcmp(argv[2],"-") == 0) {
                                archive = stdin;
                        } else {
                                archive = fopen(argv[2],"wb");
                        }
                        /* Archive file can't be used, exiting */
                        if (archive == NULL) {
                                perror(argv[0]);
                                free(mem);
                                exit(1);
                        }
                        strcpy(header.magic,STO_MAGIC);
                        header.entries = argc - 3;
#ifdef MSDOS
                        header.os = OS_DOS;
#elif __UNIX__
                        header.os = OS_UNIX;
#elif __WIN32__
                        header.os = OS_WIN;
#else /* DEADBEEF! */
                        header.os = OS_DEADBEEF;
#endif /* MSDOS */
                        fwrite(&header,(sizeof header),1,archive);
                        for(x=0;x<header.entries;x++) {
                                fd = fopen(argv[x+3],"rb");
                                if (fd == NULL) {
                                        perror(argv[x+3]);
                                        fentry.fexist = 0;
                                        fentry.flen = 0;
                                        fentry.fname[0] = '\0';
                                        fstatus = fwrite(&fentry,sizeof(sto_fent
ry),1,archive);
                                        if (fstatus != sizeof(sto_fentry)) {
                                                perror(argv[2]);
                                                free(mem);
                                                exit(1);
                                        }
                                } else {
                                        stat(argv[x+3],&ostat);
                                        fentry.fexist = (char) 1;
                                        fentry.flen = ostat.st_size;
                                        fentry.fattrib = ostat.st_mode;
                                        fentry.fowner = ostat.st_uid;
                                        fentry.fgroup = ostat.st_gid;
                                        fentry.fatime = ostat.st_atime;
                                        fentry.fmtime = ostat.st_mtime;
                                        fentry.fctime = ostat.st_ctime;
                                        fentry.fname[0] = '\0';
#ifdef MSDOS
                                        fentry.os = OS_DOS;
/* If unix was defined, or is compiling under BCC for a
 * ELKS or Linux target (It won't compile for a standalone target anyways.
 *
 * BCC can cross-compile to DOS .COM, but a native compile with Turbo C is
 * better.
 */
#elif unix || (__BCC__ && !__MSDOS__)
                                        fentry.os = OS_UNIX;
#elif __WIN32__
                                        fentry.os = OS_WIN;
#else /* DEADBEEF! */
                                        fentry.os = OS_DEADBEEF;
#endif /* MSDOS */
                                        strcat(fentry.fname,basename(argv[x+3]))
;
                                        fstatus = fwrite(&fentry,sizeof(sto_fent
ry),1,archive);
                                        while (fread(mem,1,1,fd))
                                                fwrite(mem,1,1,archive);
                                }
                                bzero(fentry.fname,256);
                                fclose(fd);
                                sync();
                        };
                        break;
                case 'x':
                        /* Need archive name to extract */
                        if (argc < 3) {
                                badarg(mem);
                        }

                        /* Here we add support for gzip'd .sto archives
                         * with piping the zcat'ed the .sto.gz to stdin
                         */
                        if (strcmp(argv[2],"-") == 0) {
                                archive = stdin;
                        } else {
                                archive = fopen(argv[2],"rb");
                        }

                        if (archive == NULL) {
                                perror(argv[2]);
                                free(mem);
                                exit(1);
                        }
                        fread(&header,sizeof(sto_header),1,archive);
                        if (strncmp(header.magic,STO_MAGIC,4) != 0) {
                                puts("Bad archive!");
                                free(mem);
                                exit(1);
                        }
                        for(x=0;x<header.entries;x++) {
                                fread(&fentry,sizeof(sto_fentry),1,archive);
                                if (fentry.fexist == 0)
                                        goto end;
                                fd = fopen(fentry.fname,"wb");
                                if (fentry.fexist == 2) {
                                        fclose(archive);
                                        free(mem);
#ifdef DIGITAL_MARS_C
                                        MessageBox(0,"STO completed sucessfully.","STO"
,0);
#endif

                                        exit(0);
                                }
                                if (fd == NULL) perror(fentry.fname);
                                if (fentry.flen != 0)
                                        for(y=0;y<fentry.flen;y++) {
                                                fread(mem,1,1,archive);
                                                fwrite(mem,1,1,fd);
                                        };
                                fclose(fd);
                                /* MS-DOS has no equilvant functions
                                 * that give compatable UNIX time
                                 */
#ifndef MSDOS
                                tb.actime = fentry.fatime;
                                tb.modtime = fentry.fmtime;
                                utime(fentry.fname,&tb);
#endif
                                chmod(fentry.fname,fentry.fattrib);
                                end:
                                x=x; /* Come up with a better no-op and i'll use
 it. */
                        };
                        break;
        };
        free(mem);
#ifdef DIGITAL_MARS_C
        MessageBox(0,"STO completed sucessfully.","STO",0);
#endif
        exit(0);
}
```