```
:::::::::::::: hello_single.c ::::::::::::::
/* hello_single.c - a single threaded hello world program */

#include  <stdio.h>
#define NUM     5

main()
{
        void    print_msg(char *);

        print_msg("hello");
        print_msg("world\n");
}
void print_msg(char *m)
{
        int i;
        for(i=0 ; i<NUM ; i++){
                printf("%s", m);
                fflush(stdout);
                sleep(1);
        }
}


:::::::::::::: hello_multi.c ::::::::::::::
/* hello_multi.c - a multi-threaded hello world program */

#include  <stdio.h>
#include  <pthread.h>

#define NUM     5

main()
{
        pthread_t t1, t2;               /* two threads */
        void    *print_msg(void *);

        pthread_create(&t1, NULL, print_msg, (void *)"hello");
        pthread_create(&t2, NULL, print_msg, (void *)"world\n");
        pthread_join(t1, NULL);
        pthread_join(t2, NULL);
}
void *print_msg(void *m)
{
        char *cp = (char *) m;
        int i;
        for(i=0 ; i<NUM ; i++){
                printf("%s", m);
                fflush(stdout);
                sleep(1);
        }
        return NULL;
}
```

```
:::::::::::::: incprint.c ::::::::::::::
/* incprint.c - one thread increments, the other prints */

#include  <stdio.h>
#include  <pthread.h>

#define NUM     5
int     counter = 0;

main()
{
        pthread_t t1;                   /* one thread */
        void     *print_count(void *); /* its function */
        int       i;

        pthread_create(&t1, NULL, print_count, NULL);
        for( i = 0 ; i<NUM ; i++ ){
                counter++;
                sleep(1);
        }
        pthread_join(t1, NULL);
}
void *print_count(void *m)
{
        int i;
        for(i=0 ; i<NUM ; i++){
                printf("count = %d\n", counter);
                sleep(1);
        }
        return NULL;
}

:::::::::::::: twordcount1.c ::::::::::::::
/* twordcount1.c - threaded word counter for two files. Version 1 */

#include  <stdio.h>
#include  <pthread.h>
#include  <ctype.h>

int      total_words ;

main(int ac, char *av[])
{
        pthread_t t1, t2;             /* two threads */
        void     *count_words(void *);

        if ( ac != 3 ){
                printf("usage: %s file1 file2\n", av[0]);
                exit(1);
        }
        total_words = 0;
        pthread_create(&t1, NULL, count_words, (void *) av[1]);
        pthread_create(&t2, NULL, count_words, (void *) av[2]);
        pthread_join(t1, NULL);
        pthread_join(t2, NULL);
        printf("%5d: total words\n", total_words);
}
void *count_words(void *f)
{
        char *filename = (char *) f;
        FILE *fp;
        int  c, prevc = '\0';

        if ( (fp = fopen(filename, "r")) != NULL ){
                while( ( c = getc(fp)) != EOF ){
                        if ( !isalnum(c) && isalnum(prevc) )
                                total_words++;
                        prevc = c;
                }
                fclose(fp);
        } else
                perror(filename);
        return NULL;
}
```

```
:::::::::::::: twordcount2.c ::::::::::::::
/* twordcount2.c - threaded word counter for two files.        */
/*                 version 2: uses mutex to lock counter   */

#include  <stdio.h>
#include  <pthread.h>
#include  <ctype.h>

int             total_words ;    /* the counter and its lock */
pthread_mutex_t counter_lock = PTHREAD_MUTEX_INITIALIZER;

main(int ac, char *av[])
{
        pthread_t t1, t2;             /* two threads */
        void     *count_words(void *);

        if ( ac != 3 ){
                printf("usage: %s file1 file2\n", av[0]);
                exit(1);
        }
        total_words = 0;
        pthread_create(&t1, NULL, count_words, (void *) av[1]);
        pthread_create(&t2, NULL, count_words, (void *) av[2]);
        pthread_join(t1, NULL);
        pthread_join(t2, NULL);
        printf("%5d: total words\n", total_words);
}
void *count_words(void *f)
{
        char *filename = (char *) f;
        FILE *fp;
        int  c, prevc = '\0';

        if ( (fp = fopen(filename, "r")) != NULL ){
                while( ( c = getc(fp)) != EOF ){
                        if ( !isalnum(c) && isalnum(prevc) ){
                                pthread_mutex_lock(&counter_lock);
                                total_words++;
                                pthread_mutex_unlock(&counter_lock);
                        }
                        prevc = c;
                }
                fclose(fp);
        } else
                perror(filename);
        return NULL;
}
```

```
:::::::::::::: twordcount3.c ::::::::::::::
/* twordcount3.c - threaded word counter for two files.
 *              - Version 3: one counter per file
 */

#include  <stdio.h>
#include  <pthread.h>
#include  <ctype.h>

struct arg_set {              /* two values in one arg */
               char *fname;   /* file to examine      */
               int  count;    /* number of words      */
};

main(int ac, char *av[])
{
       pthread_t      t1, t2;        /* two threads */
       struct arg_set args1, args2;  /* two argsets */
       void           *count_words(void *);

       if ( ac != 3 ){
              printf("usage: %s file1 file2\n", av[0]);
              exit(1);
       }
       args1.fname = av[1];
       args1.count = 0;
       pthread_create(&t1, NULL, count_words, (void *) &args1);

       args2.fname = av[2];
       args2.count = 0;
       pthread_create(&t2, NULL, count_words, (void *) &args2);

       pthread_join(t1, NULL);
       pthread_join(t2, NULL);
       printf("%5d: %s\n", args1.count, av[1]);
       printf("%5d: %s\n", args2.count, av[2]);
       printf("%5d: total words\n", args1.count+args2.count);
}
void *count_words(void *a)
{
       struct arg_set *args = a;     /* cast arg back to correct type */
       FILE *fp;
       int  c, prevc = '\0';

       if ( (fp = fopen(args->fname, "r")) != NULL ){
              while( ( c = getc(fp)) != EOF ){
                     if ( !isalnum(c) && isalnum(prevc) )
                            args->count++;
                     prevc = c;
              }
              fclose(fp);
       } else
              perror(args->fname);
       return NULL;
}
```

```
:::::::::::::: twordcount4.c ::::::::::::::
/* twordcount4.c - threaded word counter for two files.
 *              - Version 4: condition variable allows counter
 *                           functions to report results early
 */

#include  <stdio.h>
#include  <pthread.h>
#include  <ctype.h>

struct arg_set {              /* two values in one arg*/
               char *fname;   /* file to examine     */
               int  count;    /* number of words     */
};

struct arg_set  *mailbox = NULL;
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t  flag = PTHREAD_COND_INITIALIZER;

main(int ac, char *av[])
{
        pthread_t      t1, t2;        /* two threads */
        struct arg_set args1, args2;  /* two argsets */
        void           *count_words(void *);
        int            reports_in = 0;
        int            total_words = 0;

        if ( ac != 3 ){
                printf("usage: %s file1 file2\n", av[0]);
                exit(1);
        }
        pthread_mutex_lock(&lock);    /* lock the report box now */

        args1.fname = av[1];
        args1.count = 0;
        pthread_create(&t1, NULL, count_words, (void *) &args1);

        args2.fname = av[2];
        args2.count = 0;
        pthread_create(&t2, NULL, count_words, (void *) &args2);

        while( reports_in < 2 ){
                printf("MAIN: waiting for flag to go up\n");
                pthread_cond_wait(&flag, &lock); /* wait for notify */
                printf("MAIN: Wow! flag was raised, I have the lock\n");
                printf("%7d: %s\n", mailbox->count, mailbox->fname);
                total_words += mailbox->count;
                if ( mailbox == &args1)
                        pthread_join(t1,NULL);
                if ( mailbox == &args2)
                        pthread_join(t2,NULL);
                mailbox = NULL;
                pthread_cond_signal(&flag);   /* announce state change */
                reports_in++;
        }
        printf("%7d: total words\n", total_words);
}
```

```
void *count_words(void *a)
{
        struct arg_set *args = a;     /* cast arg back to correct type */
        FILE *fp;
        int  c, prevc = '\0';

        if ( (fp = fopen(args->fname, "r")) != NULL ){
                while( ( c = getc(fp)) != EOF ){
                        if ( !isalnum(c) && isalnum(prevc) )
                                args->count++;
                        prevc = c;
                }
                fclose(fp);
        } else
                perror(args->fname);
        printf("COUNT: waiting to get lock\n");
        pthread_mutex_lock(&lock);    /* get the mailbox */
        printf("COUNT: have lock, storing data\n");
        if ( mailbox != NULL ){
                printf("COUNT: oops..mailbox not empty. wait for signal\n");
                pthread_cond_wait(&flag,&lock);
        }
        mailbox = args;                       /* put ptr to our args there */
        printf("COUNT: raising flag\n");
        pthread_cond_signal(&flag);   /* raise the flag */
        printf("COUNT: unlocking box\n");
        pthread_mutex_unlock(&lock);  /* release the mailbox */
        return NULL;
}
```

```
:::::::::::::: tbounce1d.c ::::::::::::::
/* tbounce1d.c: controlled animation using two threads
 *      note    one thread handles animation
 *              other thread handles keyboard input
 *      compile cc tbounce1d.c -lcurses -lpthread -o tbounce1d
 */

#include       <stdio.h>
#include       <curses.h>
#include       <pthread.h>
#include       <stdlib.h>
#include       <unistd.h>

/* shared variables both threads use. These need a mutex. */

#define MESSAGE " hello "

int     row;    /* current row         */
int     col;    /* current column      */
int     dir;    /* where we are going  */
int     delay;  /* delay between moves */

main()
{
        int     ndelay;         /* new delay           */
        int     c;              /* user input          */
        pthread_t msg_thread;   /* a thread            */
        void    *moving_msg();

        initscr();              /* init curses and tty */
        crmode();
        noecho();
        clear();

        row   = 10;             /* start here          */
        col   = 0;
        dir   = 1;              /* add 1 to row number */
        delay = 200;            /* 200ms = 0.2 seconds  */

        if ( pthread_create(&msg_thread,NULL,moving_msg,MESSAGE) ){
                fprintf(stderr,"error creating thread");
                endwin();
                exit(0);
        }

        while(1) {
                ndelay = 0;
                c = getch();
                if ( c == 'Q' ) break;
                if ( c == ' ' ) dir = -dir;
                if ( c == 'f' && delay > 2 ) ndelay = delay/2;
                if ( c == 's' ) ndelay = delay * 2 ;
                if ( ndelay > 0 )
                        delay = ndelay ;
        }
        pthread_cancel(msg_thread);
        endwin();
}

void *moving_msg(char *msg)
{
        while( 1 ) {
                usleep(delay*1000);    /* sleep a while      */
                move( row, col );      /* set cursor position */
                addstr( msg );         /* redo message               */
                refresh();             /* and show it        */

                /* move to next column and check for bouncing       */

                col += dir;            /* move to new column  */

                if ( col <= 0 && dir == -1 )
                        dir = 1;
                else if (  col+strlen(msg) >= COLS && dir == 1 )
                        dir = -1;

        }
}
```

```
:::::::::::::: tanimate.c ::::::::::::::
/* tanimate.c: animate several strings using threads, curses, usleep()
 *
 *      bigidea one thread for each animated string
 *              one thread for keyboard control
 *              shared variables for communication
 *      compile cc tanimate.c -lcurses -lpthread -o tanimate
 *      to do   needs locks for shared variables
 *              nice to put screen handling in its own thread
 */

#include       <stdio.h>
#include       <curses.h>
#include       <pthread.h>
#include       <stdlib.h>
#include       <unistd.h>

#define MAXMSG  10             /* limit to number of strings */
#define TUNIT   20000          /* timeunits in microseconds */

struct  propset {
                char    *str;   /* the message */
                int     row;    /* the row     */
                int     delay;  /* delay in time units */
                int     dir;    /* +1 or -1    */
        };

pthread_mutex_t mx = PTHREAD_MUTEX_INITIALIZER;

int main(int ac, char *av[])
{
        int            c;              /* user input          */
        pthread_t      thrds[MAXMSG]; /* the threads         */
        struct propset props[MAXMSG]; /* properties of string       */
        void           *animate();     /* the function               */
        int            num_msg ;       /* number of strings   */
        int          i;

        if ( ac == 1 ){
                printf("usage: tanimate string ..\n");
                exit(1);
        }

        num_msg = setup(ac-1,av+1,props);

        /* create all the threads */
        for(i=0 ; i<num_msg; i++)
                if ( pthread_create(&thrds[i], NULL, animate, &props[i])){
                        fprintf(stderr,"error creating thread");
                        endwin();
                        exit(0);
                }

        /* process user input */
        while(1) {
                c = getch();
                if ( c == 'Q' ) break;
                if ( c == ' ' )
                        for(i=0;i<num_msg;i++)
                                props[i].dir = -props[i].dir;
                if ( c >= '0' && c <= '9' ){
                        i = c - '0';
                        if ( i < num_msg )
                                props[i].dir = -props[i].dir;
                }
        }

        /* cancel all the threads */
        pthread_mutex_lock(&mx);
        for (i=0; i<num_msg; i++ )
                pthread_cancel(thrds[i]);
        endwin();
        return 0;
}
```

```
int setup(int nstrings, char *strings[], struct propset props[])
{
        int num_msg = ( nstrings > MAXMSG ? MAXMSG : nstrings );
        int i;

        /* assign rows and velocities to each string */
        srand(getpid());
        for(i=0 ; i<num_msg; i++){
                props[i].str = strings[i];    /* the message */
                props[i].row = i;             /* the row     */
                props[i].delay = 1+(rand()%15);       /* a speed     */
                props[i].dir = ((rand()%2)?1:-1);     /* +1 or -1    */
        }

        /* set up curses */
        initscr();
        crmode();
        noecho();
        clear();
        mvprintw(LINES-1,0,"'Q' to quit, '0'..'%d' to bounce",num_msg-1);

        return num_msg;
}

/* the code that runs in each thread */
void *animate(void *arg)
{
        struct propset *info = arg;           /* point to info block */
        int     len = strlen(info->str)+2;    /* +2 for padding      */
        int     col = rand()%(COLS-len-3);    /* space for padding   */

        while( 1 )
        {
                usleep(info->delay*TUNIT);

                pthread_mutex_lock(&mx);      /* only one thread     */
                   move( info->row, col );    /* can call curses     */
                   addch(' ');                /* at a the same time  */
                   addstr( info->str );              /* Since I doubt it is */
                   addch(' ');                /* reentrant           */
                   move(LINES-1,COLS-1);      /* park cursor         */
                   refresh();                 /* and show it         */
                pthread_mutex_unlock(&mx);    /* done with curses    */

                /* move item to next column and check for bouncing  */

                col += info->dir;

                if ( col <= 0 && info->dir == -1 )
                        info->dir = 1;
                else if (  col+len >= COLS && info->dir == 1 )
                        info->dir = -1;
        }
}
```