```
#include <pthread.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

static int done = 0;
static int cleanup_pop_arg = 1;
static int cnt = 0;

static void
cleanup_handler(void *arg)
{
        printf("Called clean-up handler\n");
        cnt = 0;
}

static void *
thread_start(void *arg)
{
        time_t start, curr;
        printf("New thread started\n");
        pthread_cleanup_push(cleanup_handler, NULL);

        curr = start = time(NULL);

        while (!done) {
                pthread_testcancel();           /* A cancellation point */
                if (curr < time(NULL)) {
                        curr = time(NULL);
                        printf("cnt = %d\n", cnt);  /* A cancellation point */
                        cnt++;
                }
        }

        pthread_cleanup_pop(cleanup_pop_arg);
        return NULL;
}

int
main(int argc, char *argv[])
{
        pthread_t thr;
        int s;
        void *res;

        s = pthread_create(&thr, NULL, thread_start, NULL);

        sleep(2);           /* Allow new thread to run a while */

        printf("Canceling thread\n");
        s = pthread_cancel(thr);

        s = pthread_join(thr, &res);

        if (res == PTHREAD_CANCELED)
                printf("Thread was canceled; cnt = %d\n", cnt);
        else
                printf("Thread terminated normally; cnt = %d\n", cnt);

        exit(EXIT_SUCCESS);
}
```