# An example source module to accompany...
#
# "Using POSIX Threads: Programming with Pthreads"
# by Brad nichols, Dick Buttlar, Jackie Farrell
# O'Reilly & Associates, Inc.
#
# cv/Makefile
#
# makefile for the example programs
#
CC = gcc
CFLAGS = -g
THREAD_CFLAGS = ${CFLAGS} -lpthread
OBJ = cvsimple.o inc_count.o watch_count.o

all: cvoriginal cvsimple
cvsimple: $(OBJ)
        $(CC) $(THREAD_CFLAGS) $(OBJ) -o @$
cvoriginal: cvoriginal.o
        $(CC) $(THREAD_CFLAGS) cvoriginal.o -o @$
clean:
        rm -f $(OBJ) *~ *# core cvsimple cvoriginal.o cvoriginal
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#define NUM_THREADS 3
#define TCOUNT 10
#define COUNT_THRES 12

int count = 0;
int thread_ids[3] = {0,1,2};
pthread_mutex_t count_lock=PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t count_hit_threshold=PTHREAD_COND_INITIALIZER;

void *inc_count(void *idp)
{
    int i=0;
    int *my_id = idp;

    for (i=0; i<TCOUNT; i++) {
        pthread_mutex_lock(&count_lock);
        count++;
        printf("inc_counter(): thread %d, count = %d, unlocking mutex\n", *my_id, count);
        if (count == COUNT_THRES) {
            printf("inc_count(): Thread %d, count %d\n", *my_id, count);
            pthread_cond_signal(&count_hit_threshold);
        }
        pthread_mutex_unlock(&count_lock);
        usleep(1000*(i+1));
    }
    return(NULL);
}

void *watch_count(void *idp)
{
    int *my_id = idp;
    printf("watch_count(): thread %d\n", *my_id);

    pthread_mutex_lock(&count_lock);
    while (count < COUNT_THRES) {
        pthread_cond_wait(&count_hit_threshold, &count_lock);
        printf("watch_count(): thread %d, count %d\n", *my_id, count);
    }
    pthread_mutex_unlock(&count_lock);
    return(NULL);
}

int main(void)
{
    int
pthread_t threads[3];

pthread_create(&threads[0], NULL, inc_count, (void *)&thread_ids[0]);
pthread_create(&threads[1], NULL, inc_count, (void *)&thread_ids[1]);
pthread_create(&threads[2], NULL, watch_count, (void *)&thread_ids[2]);

for (i = 0; i < NUM_THREADS; i++) pthread_join(threads[i], NULL);
return 0;
inc_counter(): thread 0, count = 1, unlocking mutex
watch_count(): thread 2
inc_counter(): thread 1, count = 2, unlocking mutex
inc_counter(): thread 1, count = 3, unlocking mutex
inc_counter(): thread 0, count = 4, unlocking mutex
inc_counter(): thread 0, count = 5, unlocking mutex
inc_counter(): thread 1, count = 6, unlocking mutex
inc_counter(): thread 0, count = 7, unlocking mutex
inc_counter(): thread 1, count = 8, unlocking mutex
inc_counter(): thread 0, count = 9, unlocking mutex
inc_counter(): thread 1, count = 10, unlocking mutex
inc_counter(): thread 0, count = 11, unlocking mutex
inc_counter(): thread 1, count = 12, unlocking mutex
inc_count(): Thread 1, count 12
watch_count(): thread 2, count 12
inc_counter(): thread 0, count = 13, unlocking mutex
inc_counter(): thread 1, count = 14, unlocking mutex
inc_counter(): thread 0, count = 15, unlocking mutex
inc_counter(): thread 1, count = 16, unlocking mutex
inc_counter(): thread 0, count = 17, unlocking mutex
inc_counter(): thread 1, count = 18, unlocking mutex
inc_counter(): thread 0, count = 19, unlocking mutex
inc_counter(): thread 1, count = 20, unlocking mutex
/ *----------------------------------*
  Constants for the shutdown code
 *----------------------------------*/
#define NUM_THREADS  6
#define TCOUNT 18
#define COUNT_THRES 18
/*------------------------------*/
   function prototypes
/*------------------------------*/
void *watch_count(void *idp);
void *inc_count(void *idp);
/**------------------*
  global data
 */
int we_are_done = 0;
int count = 0;
int thread_ids[NUM_THREADS];
pthread_mutex_t done_lock=PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t count_lock=PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t count_hit_threshold=PTHREAD_COND_INITIALIZER;
extern int we_are_done;
extern int count;
extern int *thread_ids;
extern pthread_mutex_t done_lock;
extern pthread_mutex_t count_lock;
extern pthread_cond_t count_hit_threshold;
/* thread routine that does the work */

#include <stdio.h>
#include <pthread.h>
#include "constants.h"
#include "global-data-extern.h"

void *inc_count(void *idp) {
  int i=0, save_state, save_type;
  int *my_id = idp;

  for (i=0; i<TCOUNT; i++) {
    pthread_mutex_lock(&done_lock);
    if (we_are_done) {
      printf("inc_count(): thread %d we are done shutting down (0:%d)\n", *my_id,i);
      pthread_mutex_unlock(&done_lock);
      return (NULL);
    } else
      pthread_mutex_unlock(&done_lock);
    pthread_mutex_lock(&count_lock);
    count++;
    printf("inc_count(): thread %d, count = %3d, pass=%2d unlocking mutex\n", *my_id, count, i);
    if (count == COUNT_THRES) {
      printf("inc_count(): Thread %d, count %d\n", *my_id, count);
      pthread_cond_signal(&count_hit_threshold);
    }
    pthread_mutex_unlock(&count_lock);
    usleep((unsigned) (600*(NUM_THREADS + 1 - *my_id)));
  }
  return (NULL);
}
/* function to watch for the condition variable and initiate the shutdown runs on one thread. */

#include <stdio.h>
#include <pthread.h>
#include "constants.h"
#include "global-data-extern.h"

void *watch_count (void *idp)
{
    int i=0, save_state, save_type;
    int *my_id = idp;

    printf("watch_count(): thread %d\n", *my_id);

    pthread_mutex_lock(&count_lock);

    while (count < COUNT_THRES) {
        pthread_cond_wait(&count_hit_threshold, &count_lock);
        printf("watch_count(): thread %d, count %d\n", *my_id, count);
    }
    pthread_mutex_lock(&done_lock);
    we_are_done++;
    printf("watch_count(): shut down in progress\n");
    pthread_mutex_unlock(&done_lock);
    pthread_mutex_unlock(&count_lock);

    return(NULL);
}
/***/
* An example source module to accompany...
* 
* "Using POSIX Threads: Programming with Pthreads"
*     by Brad nichols, Dick Buttlar, Jackie Farrell
*     O’Reilly & Associates, Inc.
* 
* cvsimple modified by RAK
* */

#include <stdio.h>
#include <pthread.h>
#include "constants.h"
#include "global-data.h"
#include "functions.h"
int main(void)
{
    int i;
    pthread_t threads[NUM_THREADS];

    for(i=0; i<NUM_THREADS; i++) thread_ids[i] = i;
    for(i=0; i<(NUM_THREADS-1); i++)
        pthread_create(&threads[i], NULL, inc_count, (void *)&thread_ids[i]);

    pthread_create(&threads[NUM_THREADS-1], NULL, watch_count, (void *)&thread_ids[NUM_THREADS-1]);

    for (i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }
    printf("all threads done\n");

    return 0;
}
inc_count(): thread 0, count = 1, pass= 0 unlocking mutex
inc_count(): thread 1, count = 2, pass= 0 unlocking mutex
inc_count(): thread 2, count = 3, pass= 0 unlocking mutex
inc_count(): thread 3, count = 4, pass= 0 unlocking mutex
inc_count(): thread 4, count = 5, pass= 0 unlocking mutex

watch_count(): thread 5

inc_count(): thread 4, count = 6, pass= 1 unlocking mutex
inc_count(): thread 3, count = 7, pass= 1 unlocking mutex
inc_count(): thread 2, count = 8, pass= 1 unlocking mutex
inc_count(): thread 1, count = 9, pass= 1 unlocking mutex
inc_count(): thread 0, count = 10, pass= 2 unlocking mutex
inc_count(): thread 0, count = 11, pass= 1 unlocking mutex
inc_count(): thread 3, count = 12, pass= 2 unlocking mutex
inc_count(): thread 4, count = 13, pass= 3 unlocking mutex
inc_count(): thread 2, count = 14, pass= 2 unlocking mutex
inc_count(): thread 4, count = 15, pass= 3 unlocking mutex
inc_count(): thread 3, count = 16, pass= 4 unlocking mutex
inc_count(): thread 1, count = 17, pass= 2 unlocking mutex
inc_count(): thread 0, count = 18, pass= 2 unlocking mutex
inc_count(): Thread 0, count 18

all threads done
| inc_count(): thread 0, count = 1, pass= 0 unlocking mutex |
| inc_count(): thread 1, count = 2, pass= 0 unlocking mutex |
| inc_count(): thread 2, count = 3, pass= 0 unlocking mutex |
| inc_count(): thread 3, count = 4, pass= 0 unlocking mutex |
| inc_count(): thread 4, count = 5, pass= 0 unlocking mutex |
| watch_count(): thread 5 |
| inc_count(): thread 4, count = 6, pass= 1 unlocking mutex |
| inc_count(): thread 3, count = 7, pass= 1 unlocking mutex |
| inc_count(): thread 2, count = 8, pass= 1 unlocking mutex |
| inc_count(): thread 1, count = 9, pass= 1 unlocking mutex |
| inc_count(): thread 4, count = 10, pass= 2 unlocking mutex |
| inc_count(): thread 0, count = 11, pass= 1 unlocking mutex |
| inc_count(): thread 3, count = 12, pass= 2 unlocking mutex |
| inc_count(): thread 2, count = 13, pass= 2 unlocking mutex |
| inc_count(): thread 1, count = 14, pass= 2 unlocking mutex |
| inc_count(): thread 4, count = 15, pass= 3 unlocking mutex |
| inc_count(): thread 3, count = 16, pass= 3 unlocking mutex |
| inc_count(): thread 2, count = 17, pass= 3 unlocking mutex |
| inc_count(): thread 4, count = 18, pass= 4 unlocking mutex |
| inc_count(): thread 0, count = 19, pass= 4 unlocking mutex |
| inc_count(): thread 3, count = 20, pass= 5 unlocking mutex |
| inc_count(): thread 2, count = 21, pass= 5 unlocking mutex |
| inc_count(): thread 1, count = 22, pass= 5 unlocking mutex |
| inc_count(): thread 4, count = 23, pass= 6 unlocking mutex |
| inc_count(): thread 0, count = 24, pass= 6 unlocking mutex |
| inc_count(): thread 3, count = 25, pass= 6 unlocking mutex |
| inc_count(): thread 2, count = 26, pass= 7 unlocking mutex |
| inc_count(): thread 4, count = 27, pass= 7 unlocking mutex |
| inc_count(): thread 0, count = 28, pass= 7 unlocking mutex |
| inc_count(): thread 1, count = 29, pass= 8 unlocking mutex |
| inc_count(): thread 3, count = 30, pass= 8 unlocking mutex |
| inc_count(): thread 4, count = 31, pass= 8 unlocking mutex |
| inc_count(): thread 0, count = 32, pass= 9 unlocking mutex |
| inc_count(): thread 4, count = 33, pass= 10 unlocking mutex |
| inc_count(): thread 3, count = 34, pass= 10 unlocking mutex |
| inc_count(): thread 2, count = 35, pass= 10 unlocking mutex |
| inc_count(): thread 1, count = 36, pass= 11 unlocking mutex |
| inc_count(): thread 4, count = 37, pass=11 unlocking mutex |
| inc_count(): thread 0, count = 38, pass=12 unlocking mutex |
| inc_count(): thread 4, count = 39, pass=12 unlocking mutex |
| inc_count(): thread 0, count = 40, pass=13 unlocking mutex |
| inc_count(): thread 2, count = 41, pass=13 unlocking mutex |
| inc_count(): thread 1, count = 42, pass=14 unlocking mutex |
| inc_count(): thread 4, count = 43, pass=14 unlocking mutex |
| inc_count(): thread 3, count = 44, pass=15 unlocking mutex |
| inc_count(): thread 4, count = 45, pass=16 unlocking mutex |
| inc_count(): thread 2, count = 46, pass=16 unlocking mutex |
| inc_count(): thread 3, count = 47, pass=17 unlocking mutex |
| inc_count(): thread 4, count = 48, pass=18 unlocking mutex |
| inc_count(): thread 1, count = 49, pass=18 unlocking mutex |
| inc_count(): thread 0, count = 50, pass=19 unlocking mutex |
| inc_count(): thread 3, count = 51, pass=19 unlocking mutex |
| inc_count(): thread 2, count = 52, pass=19 unlocking mutex |
| inc_count(): thread 4, count = 53, pass=20 unlocking mutex |
| inc_count(): thread 4, count = 54, pass=20 unlocking mutex |
| inc_count(): thread 1, count = 55, pass=21 unlocking mutex |
| inc_count(): thread 2, count = 56, pass=22 unlocking mutex |
inc_count(): thread 0, count = 57, pass= 7 unlocking mutex
inc_count(): thread 2, count = 58, pass=10 unlocking mutex
inc_count(): thread 4, count = 59, pass=17 unlocking mutex
inc_count(): thread 3, count = 60, pass=13 unlocking mutex
inc_count(): thread 1, count = 61, pass= 9 unlocking mutex
inc_count(): thread 2, count = 62, pass=11 unlocking mutex
inc_count(): thread 3, count = 63, pass=14 unlocking mutex
inc_count(): thread 0, count = 64, pass= 8 unlocking mutex
inc_count(): thread 2, count = 65, pass=12 unlocking mutex
inc_count(): thread 1, count = 66, pass=10 unlocking mutex
inc_count(): thread 3, count = 67, pass=15 unlocking mutex
inc_count(): thread 0, count = 68, pass= 9 unlocking mutex
inc_count(): thread 3, count = 69, pass=16 unlocking mutex
inc_count(): thread 2, count = 70, pass=13 unlocking mutex
inc_count(): thread 1, count = 71, pass=11 unlocking mutex
inc_count(): thread 3, count = 72, pass=17 unlocking mutex
inc_count(): thread 2, count = 73, pass=14 unlocking mutex
inc_count(): thread 0, count = 74, pass=10 unlocking mutex
inc_count(): thread 1, count = 75, pass=12 unlocking mutex
inc_count(): thread 2, count = 76, pass=15 unlocking mutex
inc_count(): thread 0, count = 77, pass=11 unlocking mutex
inc_count(): thread 1, count = 78, pass=13 unlocking mutex
inc_count(): thread 2, count = 79, pass=16 unlocking mutex
inc_count(): thread 1, count = 80, pass=14 unlocking mutex
inc_count(): thread 0, count = 81, pass=12 unlocking mutex
inc_count(): thread 2, count = 82, pass=17 unlocking mutex
inc_count(): thread 1, count = 83, pass=15 unlocking mutex
inc_count(): thread 0, count = 84, pass=13 unlocking mutex
inc_count(): thread 1, count = 85, pass=16 unlocking mutex
inc_count(): thread 0, count = 86, pass=14 unlocking mutex
inc_count(): thread 1, count = 87, pass=17 unlocking mutex
inc_count(): thread 0, count = 88, pass=15 unlocking mutex
inc_count(): thread 0, count = 89, pass=16 unlocking mutex
inc_count(): thread 0, count = 90, pass=17 unlocking mutex
all threads done