CST 334: Operating Systems

Dr. Glenn Bruns

# Lab: locked data structures - solution

Here’s a solution to the locked data structures lab:

#include <stdio.h>

#include <pthread.h>

#include "mythreads.h"

typedef struct {

// state var

long cnt;

// sync var

pthread\_mutex\_t lock;

} COUNTER;

COUNTER \*cnt\_create() {

COUNTER \*cntr = (COUNTER \*)malloc(sizeof(COUNTER));

pthread\_mutex\_init(&cntr->lock, NULL);

cntr->cnt = 0;

return cntr;

}

void cnt\_incr(COUNTER \*cntr) {

pthread\_mutex\_lock(&cntr->lock);

cntr->cnt++;

pthread\_mutex\_unlock(&cntr->lock);

}

void cnt\_decr(COUNTER \*cntr) {

pthread\_mutex\_lock(&cntr->lock);

cntr->cnt--;

pthread\_mutex\_unlock(&cntr->lock);

}

long cnt\_get(COUNTER \*cntr) {

pthread\_mutex\_lock(&cntr->lock);

int val = cntr->cnt;

pthread\_mutex\_unlock(&cntr->lock);

return val;

}

/\*

\* counting application

\*/

void \*mythread(void \*arg) {

COUNTER \*cntr = (COUNTER \*)arg;

int i;

for (i = 0; i < 1e7; i++) {

cnt\_incr(cntr);

}

return NULL;

}

int main(int argc, char \*argv[]) {

COUNTER \*cntr = cnt\_create();

pthread\_t p1, p2;

printf("main: begin (counter = %ld)\n", cnt\_get(cntr));

Pthread\_create(&p1, NULL, mythread, cntr);

Pthread\_create(&p2, NULL, mythread, cntr);

// join waits for the threads to finish

Pthread\_join(p1, NULL);

Pthread\_join(p2, NULL);

printf("main: done with both (counter = %ld)\n", cnt\_get(cntr));

return 0;

}