#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

#include <time.h>

#include <unistd.h>

#include <pthread.h>

struct station {

int out\_passengers;

int in\_passengers;

pthread\_mutex\_t lock;

pthread\_cond\_t train\_arrived\_cond;

pthread\_cond\_t passengers\_seated\_cond;

pthread\_cond\_t train\_is\_full\_cond;

};

void station\_init(struct station \*station);

void station\_load\_train(struct station \*station, int count);

void station\_wait\_for\_train(struct station \*station);

void station\_on\_board(struct station \*station);

volatile int threads\_completed = 0;

void \*passenger\_thread(void \*arg)

{

struct station station = (struct station)arg;

station\_wait\_for\_train(station);

\_\_sync\_add\_and\_fetch(&threads\_completed, 1);

return NULL;

}

struct load\_train\_args {

struct station \*station;

int free\_seats;

};

volatile int load\_train\_returned = 0;

void \*load\_train\_thread(void \*args)

{

struct load\_train\_args ltargs = (struct load\_train\_args)args;

station\_load\_train(ltargs->station, ltargs->free\_seats);

load\_train\_returned = 1;

return NULL;

}

const char\* alarm\_error\_str;

int alarm\_timeout;

void \_alarm(int seconds, const char \*error\_str)

{

alarm\_timeout = seconds;

alarm\_error\_str = error\_str;

alarm(seconds);

}

void alarm\_handler(int foo)

{

fprintf(stderr, "Error: Failed to complete after %d seconds. Something's "

"wrong, or your system is terribly slow. Possible error hint: [%s]\n",

alarm\_timeout, alarm\_error\_str);

exit(1);

}

#ifndef MIN

#define MIN(\_x,\_y) ((\_x) < (\_y)) ? (\_x) : (\_y)

#endif

int main()

{

struct station station;

station\_init(&station);

srandom(getpid() ^ time(NULL));

signal(SIGALRM, alarm\_handler);

\_alarm(1, "station\_load\_train() did not return immediately when no waiting passengers");

station\_load\_train(&station, 0);

station\_load\_train(&station, 10);

\_alarm(0, NULL);

int i;

const int total\_passengers = 100;

int passengers\_left = total\_passengers;

for (i = 0; i < total\_passengers; i++) {

pthread\_t tid;

int ret = pthread\_create(&tid, NULL, passenger\_thread, &station);

if (ret != 0) {

perror("pthread\_create");

exit(1);

}

}

\_alarm(2, "station\_load\_train() did not return immediately when no free seats");

station\_load\_train(&station, 0);

\_alarm(0, NULL);

int total\_passengers\_boarded = 0;

const int max\_free\_seats\_per\_train = 50;

int pass = 0;

while (passengers\_left > 0) {

\_alarm(2, "Some more complicated issue appears to have caused passengers "

"not to board when given the opportunity");

int free\_seats = random() % max\_free\_seats\_per\_train;

printf("Train entering station with %d free seats\n", free\_seats);

load\_train\_returned = 0;

struct load\_train\_args args = { &station, free\_seats };

pthread\_t lt\_tid;

int ret = pthread\_create(&lt\_tid, NULL, load\_train\_thread, &args);

if (ret != 0) {

perror("pthread\_create");

exit(1);

}

int threads\_to\_reap = MIN(passengers\_left, free\_seats);

int threads\_reaped = 0;

while (threads\_reaped < threads\_to\_reap) {

if (load\_train\_returned) {

fprintf(stderr, "Error: station\_load\_train returned early!\n");

exit(1);

}

if (threads\_completed > 0) {

if ((pass % 2) == 0)

usleep(random() % 2);

threads\_reaped++;

station\_on\_board(&station);

\_\_sync\_sub\_and\_fetch(&threads\_completed, 1);

}

}

for (i = 0; i < 1000; i++) {

if (i > 50 && load\_train\_returned)

break;

usleep(1000);

}

if (!load\_train\_returned) {

fprintf(stderr, "Error: station\_load\_train failed to return\n");

exit(1);

}

while (threads\_completed > 0) {

threads\_reaped++;

\_\_sync\_sub\_and\_fetch(&threads\_completed, 1);

}

passengers\_left -= threads\_reaped;

total\_passengers\_boarded += threads\_reaped;

printf("Train departed station with %d new passenger(s) (expected %d)%s\n",

threads\_to\_reap, threads\_reaped,

(threads\_to\_reap != threads\_reaped) ? " \*\*\*" : "");

if (threads\_to\_reap != threads\_reaped) {

fprintf(stderr, "Error: Too many passengers on this train!\n");

exit(1);

}

pass++;

}

if (total\_passengers\_boarded == total\_passengers) {

printf("Looks good!\n");

return 0;

} else {

// I don't think this is reachable, but just in case.

fprintf(stderr, "Error: expected %d total boarded passengers, but got %d!\n",

total\_passengers, total\_passengers\_boarded);

return 1;

}

}

void

station\_init(struct station \*station)

{

station->out\_passengers = 0;

station->in\_passengers = 0;

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

pthread\_cond\_init(&(station->train\_arrived\_cond), NULL);

pthread\_cond\_init(&(station->passengers\_seated\_cond), NULL);

pthread\_cond\_init(&(station->train\_is\_full\_cond), NULL);

}

void station\_load\_train(struct station \*station, int count)

{

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

while ((station->out\_passengers > 0) && (count > 0)){

pthread\_cond\_signal(&(station->train\_arrived\_cond));

count--;

pthread\_cond\_wait(&(station->passengers\_seated\_cond), &(station->lock));

}

if (station->in\_passengers > 0)

pthread\_cond\_wait(&(station->train\_is\_full\_cond), &(station->lock));

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

}

void station\_wait\_for\_train(struct station \*station)

{

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

station->out\_passengers++;

pthread\_cond\_wait(&(station->train\_arrived\_cond), &(station->lock));

station->out\_passengers--;

station->in\_passengers++;

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

pthread\_cond\_signal(&(station->passengers\_seated\_cond));

}

void station\_on\_board(struct station \*station)

{

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

station->in\_passengers--;

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

if (station->in\_passengers == 0)

pthread\_cond\_broadcast(&(station->train\_is\_full\_cond));

}