#include <pthread.h>

#include <iostream>

#include <errno.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <pthread.h>

#include <signal.h>

#include <mutex>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Here is a program that will simulate asking for a file, the dispatch thread1

\* will immediately get new files and will send worker threads to complete the

\* simulated execution of some task, waiting for one second 80% of the time

\* and from 7-10 seconds the other 20%. The dispatch thread keeps track of the

\* number of files. And we also calculate the average time for file access and

\* print to screen.

\*

\* @author Keith Schmitt, Andrew Lawton

\* @version 1.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void\* worker (void\* arg);

void sigHandler(int);

bool shutdown = false;

std::string fileName;

int num\_of\_files = 0;

double average = 0;

//mutex for guarding average calculation

std::mutex average\_mutex;

int main(){

pthread\_t thread1; // thread ID's

void \*result1; // return values

//used for error reporting

int status;

//signal for control -C

signal(SIGINT, sigHandler);

while (!shutdown){

sleep(1);

std::cout<<"Please enter a file name: ";

if(std::getline(std::cin, fileName) && shutdown){

break;

}

//create the thread

if ((status = pthread\_create (&thread1, NULL, worker, NULL)) != 0) {

std::cerr << "thread create error: " << strerror(status) << std::endl;

exit (1);

}

++num\_of\_files;

}

//responsible parents will wait for threads

if ((status = pthread\_join(thread1, &result1)) != 0) {

fprintf(stderr, "join error %d:%s\n", status, strerror(status));

exit(1);

}

//print out the average time

std::cout<<"Average file access time: "<<average/num\_of\_files<<std::endl;

return 0;

}

void\* worker (void\* arg) {

//thread getting data from shared memory

std::string myFile = fileName;

std::cout<<"Accessing File... "<< myFile<<std::endl;

//temp variable for recording time

int temp;

int probability = rand()%10;

//80% chance that we wait for 1 second

if (probability <= 8){

temp = 1;

}

else{

//otherwise we wait between 7 and 10 seconds

probability = rand()%4;

temp = 7 + probability;

}

sleep(temp);

std::cout<<"File Accessed: "<< myFile<<std::endl;

std::cout<<"Terminating..."<<std::endl;

std::lock\_guard<std::mutex> guard(average\_mutex);

average += temp;

sleep(2);

return NULL;

}

//catches control-C

void sigHandler (int sigNum){

if(sigNum == SIGINT){

std::cout<<" received"<<std::endl;

std::cout<<"That's it, I'm shutting you down..."<<std::endl;

sleep(1);

std::cout<<"Number of files serviced: "<<num\_of\_files<<std::endl;

shutdown = true;

}

}