# CODE:

#include <iostream>

#include <thread>

#include <mutex>

#include <chrono>

#include <ctime>

#include<fstream>

#include<cstdio>

#include <sys/stat.h>

#include <string>

#include<pthread.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/ioctl.h>

#include <linux/fs.h>

#include <string>

#include <vector>

#include <cstdlib>

#include<algorithm>

#include <csignal>

using namespace std;

mutex mtx\_a1;

mutex mtx\_boot\_screen;

mutex mtx\_start\_time\_and\_calendar;

mutex mtx\_notepad;

mutex mtx\_calculator;

mutex mtx\_create\_file;

mutex mtx\_rock\_paper\_scissor;

mutex mtx\_copy\_file;

mutex mtx\_delete\_file;

mutex mtx\_check\_file\_info;

mutex mtx\_guessthenumber;

mutex mtx\_create\_process;

mutex mtx\_process\_message;

mutex mtx\_grantee;

mutex mtx\_interrupt;

mutex mtx\_flappy;

mutex mtx\_tic\_tac\_toe;

mutex mtx\_hangman;

mutex mtx\_mem\_match;

int az = 0;

bool isInterrupted = false;

// Signal handler function

void signalHandler(int signum) {

cout << "Received interrupt signal (" << signum << ")." << endl;

isInterrupted = true;

}

void boot\_screen()

{

mtx\_boot\_screen.lock();

cout << "WELCOME TO Sheryar Noman and Zohaib OPERATING SYSTEM..." << endl;

cout << "Hardware Requirements:" << endl;

cout << "No of cores: 8 cores" << endl;

cout << "Hard Driver Storage: 256GB" << endl;

cout << "RAM: 2GB" << endl;

this\_thread::sleep\_for(chrono::seconds(5));

cout << "OPERATING SYSTEM LOADED:)" << endl;

mtx\_boot\_screen.unlock();

}

void\* start\_time\_and\_calendar(void\*) {

az=0;

mtx\_start\_time\_and\_calendar.lock();

cout << endl << "entered calendar" << endl;

time\_t now = time(0);

char\* dt = ctime(&now);

cout << "enter 1 to break:";

cin >> az;

if (az == 1) {

mtx\_start\_time\_and\_calendar.unlock();

pthread\_exit(nullptr);

}

cout << "The local date and time is: " << dt << endl;

mtx\_start\_time\_and\_calendar.unlock();

return nullptr;

}

void\* notepad(void\*)

{

az=0;

cout<<endl<<"entered notepad"<<endl;

string a,str;

mtx\_notepad.lock();

cout << "enter 1 to break:";

cin >> az;

if (az == 1) {

mtx\_notepad.unlock();

pthread\_exit(nullptr);

}

fstream ob;

ob.open("test.txt", ios::out); // opening file in writing mode

string line;

do {

std::cout << "Enter input (or 'z' to exit): ";

std::getline(std::cin, line);

// Process input if it is not 'z'

if (line != "z") {

std::cout << "Input: " << line << std::endl;

}

} while (line != "z");

ob.close(); // closing the file

ob.open("test.txt", ios::in); // again opening the file but in reading mode

while (!ob.eof()) {

string str1;

ob >> str1; // reading word by word from file and storing in str

cout << str1 << "\n"; // printing str

}

ob.close(); // closing the file after use

mtx\_notepad.unlock();

return nullptr;

}

void\* calculator(void\*)

{

az=0;

cout<<endl<<"entered calculator"<<endl;

mtx\_calculator.lock();

cout << "enter 1 to break:";

cin >> az;

if (az == 1) {

mtx\_calculator.unlock();

pthread\_exit(nullptr);

}

double num1, num2;

char op;

cout << "Enter first number: ";

cin >> num1;

cout << "Enter operator (+, -, \*, /): ";

cin >> op;

cout << "Enter second number: ";

cin >> num2;

switch (op) {

case '+':

cout << num1 << " + " << num2 << " = " << num1+num2 << endl;

break;

case '-':

cout << num1 << " - " << num2 << " = " <<num1-num2 << endl;

break;

case '\*':

cout << num1 << " \* " << num2 << " = " << num1\*num2 << endl;

break;

case '/':

if (num2 == 0) {

cout << "Error: cannot divide by zero" << endl;

} else {

cout << num1 << " / " << num2 << " = " << num1/num2 << endl;

}

break;

default:

cout << "Error: invalid operator" << endl;

break;

}

mtx\_calculator.unlock();

return nullptr;

}

void\* create\_file(void\*)

{

az=0;

cout<<endl<<"entered cerate file"<<endl;

mtx\_create\_file.lock();

cout << "enter 1 to break:";

cin >> az;

if (az == 1) {

mtx\_create\_file.unlock();

pthread\_exit(nullptr);

}

string filename;

cout << "Enter the name of the file to be created: ";

cin >> filename;

ofstream file(filename); // create a new file with the given filename

if (file.is\_open()) {

cout << "File created successfully." << endl;

// write some content to the file

file << "Hello, this is some content written to the file." << endl;

file << "You can write anything you want here." << endl;

file.close(); // close the file

} else {

cout << "Error creating file." << endl;

}

mtx\_create\_file.unlock(); return nullptr;

}

void\* rock\_paper\_scissor(void\*)

{

az=0;

cout<<endl<<"entered rock\_paper\_scissor()"<<endl;

mtx\_rock\_paper\_scissor.lock();

srand(time(nullptr)); // seed the random number generator

cout << "enter 1 to break:";

cin >> az;

if (az == 1) {

mtx\_rock\_paper\_scissor.unlock();

pthread\_exit(nullptr);

}

int playerChoice;

int computerChoice = rand() % 3 + 1; // generate a random number between 1 and 3

cout << "Welcome to Rock, Paper, Scissors!\n";

cout << "1. Rock\n";

cout << "2. Paper\n";

cout << "3. Scissors\n";

cout << "Enter your choice: ";

cin >> playerChoice;

cout << "You chose ";

switch (playerChoice) {

case 1:

cout << "Rock.\n";

break;

case 2:

cout << "Paper.\n";

break;

case 3:

cout << "Scissors.\n";

break;

default:

cout << "an invalid option.\n";

}

cout << "The computer chose ";

switch (computerChoice) {

case 1:

cout << "Rock.\n";

break;

case 2:

cout << "Paper.\n";

break;

case 3:

cout << "Scissors.\n";

break;

}

if (playerChoice == computerChoice) {

cout << "It's a tie!\n";

} else if (playerChoice == 1 && computerChoice == 3 || playerChoice == 2 && computerChoice == 1 || playerChoice == 3 && computerChoice == 2) {

cout << "You win!\n";

} else {

cout << "You lose!\n";

}

mtx\_rock\_paper\_scissor.unlock();

return nullptr;

}

void\* copy\_file(void \*)

{

mtx\_copy\_file.lock();

// implementation of copy\_file function

mtx\_copy\_file.unlock();

return nullptr;

}

void\* delete\_file(void\*) {

az=0;

cout<<endl<<"entered delete\_file()"<<endl;

mtx\_delete\_file.lock();

if (az == 1) {

mtx\_delete\_file.unlock();

pthread\_exit(nullptr);

}

const char\* file\_name = "hello.cpp";

int result = std::remove(file\_name);

if (result != 0) {

std::perror("Error deleting file");

} else {

std::puts("File successfully deleted");

}

mtx\_delete\_file.unlock();

return nullptr;

}

void\* check\_file\_info(void\*) {

az=0;

cout<<endl<<"entered checkfileinfo()"<<endl;

mtx\_check\_file\_info.lock();

string filename;

if (az == 1) {

mtx\_check\_file\_info.unlock();

pthread\_exit(nullptr);

}

cout<<"enter filename to check file\_info";

cin>>filename;

struct stat file\_info;

// get file information

if (stat(filename.c\_str(), &file\_info) == 0) {

// output file information

cout << "File name: " << filename << endl;

cout << "File size: " << file\_info.st\_size << " bytes" << endl;

cout << "Last access time: " << ctime(&file\_info.st\_atime) << endl;

cout << "Last modification time: " << ctime(&file\_info.st\_mtime) << endl;

cout << "File permissions: " << (file\_info.st\_mode & (S\_IRWXU | S\_IRWXG | S\_IRWXO)) << endl;

} else {

cout << "Error: could not get file information" << endl;}

mtx\_check\_file\_info.unlock();

return nullptr;

}

void\* guessthenumber(void\*) {

az=0;

cout<<endl<<"entered guessthenumber"<<endl;

mtx\_guessthenumber.lock();

srand(time(NULL)); // seed the random number generator with current time

int secretNumber = rand() % 100 + 1; // generate a random number between 1 and 100

int guess;

int tries = 0;

cout << "Welcome to Guess the Number!" << endl;

cout << "I'm thinking of a number between 1 and 100. Can you guess what it is?" << endl;

do {

cout << "Enter your guess: ";

cin >> guess;

tries++;

if (guess < secretNumber) {

cout << "Too low. Guess again." << endl;

} else if (guess > secretNumber) {

cout << "Too high. Guess again." << endl;

} else {

cout << "Congratulations! You guessed the number in " << tries << " tries." << endl;

}

} while (guess != secretNumber);

mtx\_guessthenumber.unlock();

return nullptr;

}

void create\_process() {

mtx\_create\_process.lock();

// implementation of create\_process function

mtx\_create\_process.unlock();

}

void process\_message() {

mtx\_process\_message.lock();

// implementation of process\_message function

mtx\_process\_message.unlock();

}

void grantee() {

mtx\_grantee.lock();

// implementation of grantee function

mtx\_grantee.unlock();

}

void interrupt() {

mtx\_interrupt.lock();

// implementation of interrupt function

mtx\_interrupt.unlock();

}

void \* tic\_tac\_toe(void\*)

{

mtx\_tic\_tac\_toe.lock();

char board[3][3] = {{' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '}};

char player = 'X';

int row, col;

cout << "Welcome to Tic Tac Toe!" << endl;

for (int i = 0; i < 9; i++) {

// Print the current board

cout << " 1 2 3" << endl;

cout << "1 " << board[0][0] << " | " << board[0][1] << " | " << board[0][2] << endl;

cout << " ---+---+---" << endl;

cout << "2 " << board[1][0] << " | " << board[1][1] << " | " << board[1][2] << endl;

cout << " ---+---+---" << endl;

cout << "3 " << board[2][0] << " | " << board[2][1] << " | " << board[2][2] << endl;

// Get the player's move

cout << "Player " << player << ", enter your move (row column): ";

cin >> row >> col;

// Check if the move is valid

if (row < 1 || row > 3 || col < 1 || col > 3) {

cout << "Invalid move. Please try again." << endl;

i--;

continue;

}

if (board[row-1][col-1] != ' ') {

cout << "That space is already taken. Please try again." << endl;

i--;

continue;

}

// Update the board with the player's move

board[row-1][col-1] = player;

// Check if the game is over

if (board[0][0] == player && board[0][1] == player && board[0][2] == player ||

board[1][0] == player && board[1][1] == player && board[1][2] == player ||

board[2][0] == player && board[2][1] == player && board[2][2] == player ||

board[0][0] == player && board[1][0] == player && board[2][0] == player ||

board[0][1] == player && board[1][1] == player && board[2][1] == player ||

board[0][2] == player && board[1][2] == player && board[2][2] == player ||

board[0][0] == player && board[1][1] == player && board[2][2] == player ||

board[0][2] == player && board[1][1] == player && board[2][0] == player) {

cout << "Player " << player << " wins!" << endl;

break;

}

if (i == 8) {

cout << "It's a tie!" << endl;

break;

}

// Switch to the other player

if (player == 'X') {

player = 'O';

} else {

player = 'X';

}

}

mtx\_tic\_tac\_toe.unlock();

return nullptr;

}

void \* hangman(void\*)

{

az=0;

mtx\_hangman.lock();

const int MAX\_WRONG = 8;

vector<string> words;

words.push\_back("GUESS");

words.push\_back("HANGMAN");

words.push\_back("DIFFICULT");

srand(static\_cast<unsigned int>(time(0)));

random\_shuffle(words.begin(), words.end());

const string THE\_WORD = words[0];

int wrong = 0;

string soFar(THE\_WORD.size(), '-');

string used = "";

cout << "Welcome to Hangman. Good luck!\n";

while ((wrong < MAX\_WRONG) && (soFar != THE\_WORD)) {

cout << "\n\nYou have " << (MAX\_WRONG - wrong) << " incorrect guesses left.\n";

cout << "\nYou've used the following letters:\n" << used << endl;

cout << "\nSo far, the word is:\n" << soFar << endl;

char guess;

cout << "\n\nEnter your guess: ";

cin >> guess;

guess = toupper(guess);

while (used.find(guess) != string::npos) {

cout << "\nYou've already guessed " << guess << endl;

cout << "Enter your guess: ";

cin >> guess;

guess = toupper(guess);

}

used += guess;

if (THE\_WORD.find(guess) != string::npos) {

cout << "That's right! " << guess << " is in the word.\n";

for (int i = 0; i < THE\_WORD.length(); ++i) {

if (THE\_WORD[i] == guess) {

soFar[i] = guess;

}

}

} else {

cout << "Sorry, " << guess << " isn't in the word.\n";

++wrong;

}

}

if (wrong == MAX\_WRONG) {

cout << "\nYou've been hanged!";

} else {

cout << "\nYou guessed it!";

}

cout << "\nThe word was " << THE\_WORD << endl;

mtx\_hangman.unlock();

return nullptr;

}

void\* kernel\_mode(void\*)

{

cout<<endl<<"DONOT MODIFY KERNEL ELSE YOUR SYSTEM CAN BLAST!!!!"<<endl;

pthread\_exit(nullptr);

int fd = open("/dev/sda", O\_RDONLY); // open a device file

if (fd < 0) {

cout << "Failed to open device file" << endl;

pthread\_exit(nullptr);

}

int ret = ioctl(fd, BLKGETSIZE64, 0); // issue an ioctl call to the device file

if (ret < 0) {

cout << "ioctl call failed" << endl;

} else {

cout << "Device size: " << ret << endl;

}

close(fd); // close the device file

pthread\_exit(nullptr);

}

void\* memory\_match(void\*)

{

az=0;

mtx\_mem\_match.lock();

cout<<"enter 1 to stop the process:";

cin>>az;

if(az==1){if (az == 1) {

mtx\_mem\_match.unlock();

pthread\_exit(nullptr);

}

}

const int size = 4;

char board[size][size] = { {'A', 'B', 'C', 'D'},

{'E', 'F', 'G', 'H'},

{'A', 'B', 'C', 'D'},

{'E', 'F', 'G', 'H'} };

bool revealed[size][size] = { false };

int row1, col1, row2, col2;

int pairsFound = 0;

srand(time(NULL));

while (pairsFound < size \* size / 2) {

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

if (revealed[i][j]) {

cout << board[i][j] << " ";

} else {

cout << "\* ";

}

}

cout << endl;

}

cout << "Enter row and column of first card: ";

cin >> row1 >> col1;

while (revealed[row1][col1]) {

cout << "Card already revealed. Enter row and column of first card: ";

cin >> row1 >> col1;

}

revealed[row1][col1] = true;

cout << "First card: " << board[row1][col1] << endl;

cout << "Enter row and column of second card: ";

cin >> row2 >> col2;

while (revealed[row2][col2]) {

cout << "Card already revealed. Enter row and column of second card: ";

cin >> row2 >> col2;

}

revealed[row2][col2] = true;

cout << "Second card: " << board[row2][col2] << endl;

if (board[row1][col1] == board[row2][col2]) {

cout << "Match!" << endl;

pairsFound++;

} else {

cout << "No match." << endl;

revealed[row1][col1] = false;

revealed[row2][col2] = false;

}

}

cout << "Congratulations! You found all the pairs." << endl;

mtx\_mem\_match.unlock();

return nullptr;

}

void \* alloc\_mem(void\*)

{

//to do this on kernel level see kernel mode function

mtx\_a1.lock();

int ram1=8;

cout<<"how much ram you want to allocate?:";

cin>>ram1;

if(ram1>2)

{

cout<<ram1<<" cannot be allocated as it is too high for the system.";

}

else

{cout<<"ram allocated";}

mtx\_a1.unlock();

return nullptr;

}

int main()

{

cout << "Menu:\n";

cout << "1. Display time and calendar\n";

cout << "2. Open notepad\n";

cout << "3. Open calculator\n";

cout << "4. Create a file\n";

cout << "5. Play rock-paper-scissors\n";

cout << "6. Copy a file\n";

cout << "7. Delete a file\n";

cout << "8. Check file info\n";

cout << "9. Play guess the number game\n";

cout << "10. Enter kernel mode\n";

cout << "11. Play tic-tac-toe game\n";

cout << "12. Play hangman game\n";

cout << "13. Play memory match game\n";

cout << "14. Allocate memory\n";

cout << "16. Interrupt Handling\n";

cout << "15. Exit\n";

thread t1(boot\_screen);

t1.join();

signal(SIGINT, signalHandler);

pthread\_t t2;

pthread\_create(&t2,nullptr,&start\_time\_and\_calendar,nullptr);

pthread\_join(t2,nullptr);

pthread\_t t3;

pthread\_create(&t3,nullptr,&notepad,nullptr);

pthread\_join(t3,nullptr);

pthread\_t t4;

pthread\_create(&t4,nullptr,&calculator,nullptr);

pthread\_join(t4,nullptr);

pthread\_t t5;

pthread\_create(&t5,nullptr,&create\_file,nullptr);

pthread\_join(t5,nullptr);

pthread\_t t6;

pthread\_create(&t6,nullptr,&rock\_paper\_scissor,nullptr);

pthread\_join(t6,nullptr);

pthread\_t t7;

pthread\_create(&t7,nullptr,&copy\_file,nullptr);

pthread\_join(t7,nullptr);

pthread\_t t8;

pthread\_create(&t8,nullptr,&delete\_file,nullptr);

pthread\_join(t8,nullptr);

pthread\_t t9;

pthread\_create(&t9,nullptr,&check\_file\_info,nullptr);

pthread\_join(t9,nullptr);

pthread\_t t10;

pthread\_create(&t10,nullptr,&guessthenumber,nullptr);

pthread\_join(t10,nullptr);

pthread\_t t11;

pthread\_create(&t11, nullptr, &kernel\_mode, nullptr);

pthread\_join(t11, nullptr);

pthread\_t t12;

pthread\_create(&t12,nullptr,&tic\_tac\_toe,nullptr);

pthread\_join(t12,nullptr);

pthread\_t t13;

pthread\_create(&t13,nullptr,&hangman,nullptr);

pthread\_join(t13,nullptr);

pthread\_t t14;

pthread\_create(&t14,nullptr,&memory\_match,nullptr);

pthread\_join(t14,nullptr);

pthread\_t t15;

pthread\_create(&t15,nullptr,&alloc\_mem,nullptr);

pthread\_join(t15,nullptr);

return 0;

}

# OUTPUT:



