Operating System Lab Final Project
“Operating System Simulator”

Instructor:
Sir Hassan Ahmed

Group Members

Fatima Nasir	23L-0714	4A
Meerab Awais	23F-0515	4A

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES,
CHINIOT- FAISALABAD

Table of Contents

Introduction:

This project is a simple version of an operating system made in C++. It lets you run different tasks like Notepad, Calculator, and file operations just like a real OS. It also shows how tasks are created, run, paused, and stopped, while managing memory and CPU usage. The goal is to help understand how real operating systems work behind the scenes.

Objectives:

- Make a basic OS simulator using C++.
- Run multiple tasks like a real OS.
- Handle memory and CPU for each task.
- Show how processes are started, paused, or ended.
- Include both user and system (kernel) control features.

Methodology:

- The OS starts with a boot screen showing the name.
- User sets memory and CPU details.
- Each task runs in its own window/process.
- Tasks are added to a queue and scheduled using rules (like first come, round robin).
- User can pause or stop any task.
- User mode allows task execution, while kernel mode lets you manage the system.
- Memory is given when a task starts and released when it ends.
- File-based tasks save their data on disk.

Code:

1. Main.cpp:

```
#include <iostream>

#include <thread>

#include <chrono>

#include <mutex>

#include <pthread.h>

#include <unistd.h>

#include <sys/wait.h>

#include <fstream>
```

```
#include <vector>
#include <algorithm>
#include <unistd.h>
#include <dirent.h>
#include <sys/types.h>
#include <signal.h>
#include <string.h>

using namespace std;

double ram = 0, cores = 0, storage = 0;
mutex mtx;



struct Process {
    int pid;
    double arrival_time;
    double priority;
    double burst_time;
};

bool compare_arrival_time(const Process& a, const Process& b) {
    return a.arrival_time < b.arrival_time;
}

bool compare_priority(const Process& a, const Process& b) {
    return a.priority < b.priority;
}
void* calculator(void* p) {
```

```
    system("gnome-terminal -- bash -c './calculator; exec bash'");
    return NULL;
}

void* calendar(void* p) {
    system("gnome-terminal -- bash -c './calender; exec bash'"); // Replace "calendar" with the name of your calendar program executable
    return NULL;
}

void* clock(void* p) {
     system("gnome-terminal -- bash -c './clock; exec bash'"); // Replace "clock" with the name of your clock program executable
    return NULL;
}

void* copyFile(void* p) {
    system("gnome-terminal -- bash -c './copyFile; exec bash'"); // Replace "copyFile" with the name of your copyFile program executable
    return NULL;
}

void* createFile(void* p) {
    system("gnome-terminal -- bash -c './createFile; exec bash'");// Replace "createFile" with the name of your createFile program executable
    return NULL;
}

void* deletefile(void* p) {
   system("gnome-terminal -- bash -c 'cd $(pwd); ./deletefile; exec bash'");
```

```
 // Replace "Deletefile" with the name of your Deletefile program executable
    return NULL;
}

void* Fileproperties(void* p) {
    system("gnome-terminal -- bash -c './Fileproperties; exec bash'"); // Replace "Fileproperties" with the name of your Fileproperties program executable
    return NULL;
}

void* moveFile(void* p) {
    system("gnome-terminal -- bash -c './moveFile; exec bash'"); // Replace "moveFile" with the name of your moveFile program executable
    return NULL;
}

void* notepad(void* p) {
    system("gnome-terminal -- bash -c './notepad; exec bash'"); // Replace "notepad" with the name of your notepad program executable
    return NULL;
}

void* rename(void* p) {
    system("gnome-terminal -- bash -c './rename; exec bash'"); // Replace "rename" with the name of your rename program executable
    return NULL;
}

void* song(void* p) {
    system("gnome-terminal -- bash -c 'cd $(pwd); ./song; exec bash'");
```

```
 // Replace "song" with the name of your song program executable
    return NULL;
}

void* tictactoe(void* p) {
    system("gnome-terminal -- bash -c './tictactoe exec bash'"); // Replace "tictactoe" with the name of your tictactoe program executable
    return NULL;
}

void* towerOfHonoi(void* p) {
    system("gnome-terminal -- bash -c './towerOfHonoi; exec bash'"); // Replace "towerOfHonoi" with the name of your towerOfHonoi program executable
    return NULL;
}

void* video(void* p) {
     system("gnome-terminal -- bash -c 'cd $(pwd); ./song; exec bash'");
    return NULL;
}

void* game(void* p) {
    system("gnome-terminal -- bash -c './game; exec bash'");
    return NULL;
}



void dis() {
```

```
DIR* dir = opendir("/proc");
if (!dir) {
    cerr << "Failed to open directory /proc.\n";
    return;
}

struct dirent* ent;
vector<pid_t> pids;
while ((ent = readdir(dir)) != nullptr) {
    if (ent->d_type != DT_DIR) continue;
    const string pid_str = ent->d_name;
    if (pid_str.find_first_not_of("0123456789") != string::npos) continue;
    const pid_t pid = stoi(pid_str);
    char cmdline_path[64];
    sprintf(cmdline_path, "/proc/%d/cmdline", pid);
    FILE* cmdline_file = fopen(cmdline_path, "r");
    if (!cmdline_file) continue;
    char cmdline[1024];
    const size_t len = fread(cmdline, 1, sizeof(cmdline), cmdline_file);
    fclose(cmdline_file);
    if (len == 0) continue;
    cmdline[len] = '\0';
    if (strstr(cmdline, "./") == cmdline) {
        pids.push_back(pid);
        cout << pid << " " << cmdline << endl;
    }
}
closedir(dir);
```

```
    if (pids.empty()) {
        cout << "No processes found.\n";
        return;
    }

    int pid;
    cout << "Enter the PID of the process you want to terminate: ";
    cin >> pid;

    int ret = kill(pid, SIGKILL);
    if (ret == -1) {
        perror("Failed to send signal to process");
        return;
    }

    cout << "Signal sent to process.\n";
}

void display_res() {
int arr2[9];
for(int i=0;i<9;i++)
{
arr2[i]=0;
}
int p=0;
DIR* dir = opendir("/proc");
    if (!dir) {
        cerr << "Failed to open directory /proc.\n";
        return;
```

```
    }

    struct dirent* ent;
    vector<pid_t> pids;
    while ((ent = readdir(dir)) != nullptr) {
        if (ent->d_type != DT_DIR) continue;
        const string pid_str = ent->d_name;
        if (pid_str.find_first_not_of("0123456789") != string::npos) continue;
        const pid_t pid = stoi(pid_str);
        char cmdline_path[64];
        sprintf(cmdline_path, "/proc/%d/cmdline", pid);
        FILE* cmdline_file = fopen(cmdline_path, "r");
        if (!cmdline_file) continue;
        char cmdline[1024];
        const size_t len = fread(cmdline, 1, sizeof(cmdline), cmdline_file);
        fclose(cmdline_file);
        if (len == 0) continue;
        cmdline[len] = '\0';
        if (strstr(cmdline, "./") == cmdline) {
            pids.push_back(pid);
arr2[p]=pid;
p++;
        }
    }

    closedir(dir);
            if((ram - 0.5*p) <= 0)
{
            cout<<"No More Memmory\n Delete Some process\n";
```

```
kill(arr2[1], SIGKILL);
}
    cout << "RAM : " << ram - 0.5*p << endl;
    //cout << "Cores : " << cores << endl;
    cout << "Storage : " << storage - 0.10*p << endl;
}

int main() {
    cout<<"YOO operating system"<<endl;
    sleep(3);
    cout<<"Yoo we getting started!!"<<endl;
    int choice;
    int hightime = 5;
    pthread_t calc_thread,
calendar_thread,clock_thread,copyFile_thread,createFile_thread,deletefile_thread,Fileproperties_thread,m
oveFile_thread,notepad_thread,rename_thread,song_thread,video_thread,tictactoe_thread,towerOfHonoi_
thread,game_thread;
    cout << "Enter RAM: ";
    cin >> ram;
    cout << "Enter storage: ";
    cin >> storage;

    // Initialize the queues
    int high[9];
    for (int i = 0; i < 9; i++)
    {
        high[i] = 0;

    }
    while (choice != -1) {
```

```
cout << "Choose an option:\n";
cout << "01. Calculator\n";
cout << "02. Calendar\n";
cout << "03. Clock\n";
cout << "04. Copy File\n";
cout << "05. Create File\n";
cout << "06. Delete File\n";
cout << "07. File Properties\n";
cout << "08. Move File\n";
cout << "09. Notepad\n";
cout << "10. Rename File\n";
cout << "11. Play Song\n";
cout << "12. Play Video\n";
cout << "13. Run Tic Tac Toe\n";
cout << "14. Run Tower of Honoi\n";
cout << "15. Run Game 3\n";
cout << "Default. Terminate Functions\n";

cout << "-1. Exit\n";
cin >> choice;

switch (choice) {
case 1:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&calc_thread, NULL, calculator, NULL);
```

```
        pthread_join(calc_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        display_res();
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
case 2:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&calendar_thread, NULL, calendar, NULL);
        pthread_join(calendar_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
```

```
        cerr << "Error creating process.\n";
    }
    break;
}
case 3:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&clock_thread, NULL, clock, NULL);
        pthread_join(clock_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
case 4:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&copyFile_thread, NULL, copyFile, NULL);
```

```
        pthread_join(copyFile_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
case 5:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&createFile_thread, NULL, createFile, NULL);
        pthread_join(createFile_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
```

```
    }
    break;
}
case 6:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&deletefile_thread, NULL, deletefile, NULL);
        pthread_join(deletefile_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
case 7:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&Fileproperties_thread, NULL, Fileproperties, NULL);
        pthread_join(Fileproperties_thread, NULL);
```

```
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
case 8:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&moveFile_thread, NULL, moveFile, NULL);
        pthread_join(moveFile_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
```

```
    break;
}
case 9:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&notepad_thread, NULL, notepad, NULL);
        pthread_join(notepad_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
case 10:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&rename_thread, NULL, rename, NULL);
        pthread_join(rename_thread, NULL);
        exit(0);
```

```
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
case 11:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&song_thread, NULL, song, NULL);
        pthread_join(song_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
```

```
}
case 12:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&video_thread, NULL, video, NULL);
        pthread_join(video_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
case 13:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&tictactoe_thread, NULL, tictactoe, NULL);
        pthread_join(tictactoe_thread, NULL);
        exit(0);
    }
```

```
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
case 14:
{
    pid_t pid = fork();
    if (pid == 0) {
        // Child process
        pthread_create(&towerOfHonoi_thread, NULL, towerOfHonoi, NULL);
        pthread_join(towerOfHonoi_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        // Parent process
        wait(NULL);
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}
```

```
        case 15:
{
    pid_t pid = fork();
    if (pid == 0) {
        pthread_create(&game_thread, NULL, game, NULL);
        pthread_join(game_thread, NULL);
        exit(0);
    }
    else if (pid > 0) {
        wait(NULL);
        display_res();
        break;
    }
    else {
        cerr << "Error creating process.\n";
    }
    break;
}

        case 17:
            dis();
            break;
        case 18:
            display_res();
            break;
        default:
        {
            //
            DIR* dir = opendir("/proc");
```

```
if (!dir) {
    cerr << "Failed to open            directory /proc.\n";
    break;
}

int arr2[9], p = 0;
for (int i = 0; i < 9; i++)
{
    arr2[i] = 0;
}
struct dirent* ent;
vector<pid_t> pids;
while ((ent = readdir(dir)) != nullptr) {
    if (ent->d_type != DT_DIR) continue;
    const string pid_str = ent->d_name;
    if (pid_str.find_first_not_of("0123456789") != string::npos) continue;
    const pid_t pid = stoi(pid_str);
    char cmdline_path[64];
    sprintf(cmdline_path, "/proc/%d/cmdline", pid);
    FILE* cmdline_file = fopen(cmdline_path, "r");
    if (!cmdline_file) continue;
    char cmdline[1024];
    const size_t len = fread(cmdline, 1, sizeof(cmdline), cmdline_file);
    fclose(cmdline_file);
    if (len == 0) continue;
    cmdline[len] = '\0';
    if (strstr(cmdline, "./") == cmdline) {
        pids.push_back(pid);
        arr2[p] = pid;
```

```
        p++;
        cout << pid << " " << cmdline << endl;
    }

}
cout << "a-------------" << endl;
for (int i = 0; i < p; i++)
{
    cout << arr2[i] << endl;
}
cout << "a-------------" << endl;

for (int i = 0; i < 9; i++)
{
    high[i] = arr2[i];
}
//
            // Schedule the processes in the queues using different scheduling techniques on each level

// dis();
            // Execute the processes in the order of the queues
int counth = 1;
while (counth < 9) {



    int pid = high[counth];
    cout << pid << endl;
    string cmd = "xdotool search --pid " + to_string(pid) + " --all windowactivate";
    system(cmd.c_str());
```

```
            sleep(5);
            int ret = kill(pid, SIGKILL);

            if (ret == -1) {
                perror("Failed to send signal to process");
            }
            else {

                cout << "Terminated process " << high[counth] << " with high priority.\n";
            }
            counth++;
        }

    }

    }

  }
  return 0;
}
```

2. **Calculator.cpp:**

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <limits>
#include <unistd.h>
```

```
using namespace std;

int main() {
    double num1, num2, result;
    char op;
    cout << "calculator started (PID: "<< getpid() << " )\n";
    cout << "type operation (+, -, *, /, ^, %) or:\n ";
    cout << "m to minimize\n";
    cout << "q to quit(close)\n";
    while (true) {
        // Display the prompt and get user input
        cout << "Enter an operation: ";
        cin >> op;

        if (op == 'q' || op == 'Q') {
            cout << "calculator closing..." << endl;
            break;
        }
        if (op == 'm' || op == 'M') {
            cout << "minimizing calculator PID ( " << getpid() << "), you can resume later.\n";
            continue;
        }
        if (op == '+' || op == '-' || op == '*' || op == '/' || op == '^' || op == '%') {
            cout << "Enter two numbers: ";
            cin >> num1 >> num2;

            // Perform the requested operation
            switch (op) {
            case '+':
```

```
            result = num1 + num2;
            break;
        case '-':
            result = num1 - num2;
            break;
        case '*':
            result = num1 * num2;
            break;
        case '/':
            result = num1 / num2;
            break;
        case '^':
            result = pow(num1, num2);
            break;
        case '%':
            result = fmod(num1, num2);
            break;
        default:
            cerr << "Invalid operation." << endl;
            continue;
        }

        // Print the result
        cout << "Result: " << result << endl;
    }
    else {
        cerr << "Invalid operation." << endl;
    }
}
```

```
    return 0;
}
```

3. Calendar.cpp:

```
#include "pthread.h"
#include "iostream"
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include "bits/stdc++.h"
#include "ctime"
#include "cstdlib"

using namespace std;
int dayNumber(int day, int month, int year)
{
    static int t[] = { 0, 3, 2, 5, 0, 3, 5, 1,
                       4, 6, 2, 4 };
    year -= month < 3;
    return (year + year / 4 - year / 100 +
        year / 400 + t[month - 1] + day) % 7;
}

int numberOfDays(int monthNumber, int year)
{

    // January
    if (monthNumber == 0)
        return (31);
```

```
// February
if (monthNumber == 1)
{
    // If the year is leap then February has
    // 29 days
    if (year % 400 == 0 ||
        (year % 4 == 0 && year % 100 != 0))
        return (29);
    else
        return (28);
}

// March
if (monthNumber == 2)
    return (31);

// April
if (monthNumber == 3)
    return (30);

// May
if (monthNumber == 4)
    return (31);

// June
if (monthNumber == 5)
    return (30);
```

```
    // July
    if (monthNumber == 6)
        return (31);

    // August
    if (monthNumber == 7)
        return (31);

    // September
    if (monthNumber == 8)
        return (30);

    // October
    if (monthNumber == 9)
        return (31);

    // November
    if (monthNumber == 10)
        return (30);

    // December
    if (monthNumber == 11)
        return (31);
    return 0;
}

int main()
{
    int year;
```

```
    cout << "Enter Year:";
    cin >> year;
    cout << "\e[1;32m\tCalendar for " << year << "\n";
    int days;
    int current = dayNumber(1, 1, year);
    for (int i = 0; i < 12; i++)
    {
if(i==0)
{
cout<<"JAN"<<endl;
}
else if(i==1)
{
cout<<"FEB"<<endl;
}
else if(i==2)
{
cout<<"MAR"<<endl;
}
else if(i==3)
{
cout<<"APR"<<endl;
}
else if(i==4)
{
cout<<"MAY"<<endl;
}
else if(i==5)
{
```

```
cout<<"JUN"<<endl;
}
else if(i==6)
{
cout<<"JUL"<<endl;
}
else if(i==7)
{
cout<<"AUG"<<endl;
}
else if(i==8)
{
cout<<"SEP"<<endl;
}
else if(i==9)
{
cout<<"OCT"<<endl;
}
else if(i==10)
{
cout<<"NOV"<<endl;
}
else if(i==11)
{
cout<<"DEC"<<endl;
}
        days = numberOfDays(i, year);
        cout << "\e[1;32m  Sun  Mon  Tue  Wed  Thu  Fri  Sat\n";
        int k;
```

```
        for (k = 0; k < current; k++)
            cout << "    ";

        for (int j = 1; j <= days; j++)
        {
            printf("%4d", j);
            if (++k > 6)
            {
                k = 0;
                cout << "\n";
            }
        }

        if (k)
            cout << "\n";
        current = k;
    }

    return 0;
}
```

4. Clock.cpp:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main() {
    system("gnome-clocks");
```

```
    return 0;
}
```

createFile.cpp:

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <limits>
#include <unistd.h>
#include "fstream"
using namespace std;

int main()
{
    fstream file;
cout<<"enter name : "<<endl;
string na;
cin>>na;
    file.open(na+".txt",ios::out);
    if (!file)
    {
        cout << "Error in creating file!!!";
    }

    cout << "File created successfully.";
}
```

5. **copyFile.cpp:**

```
#include "pthread.h"
```

```
#include "iostream"
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include "bits/stdc++.h"
#include "ctime"
#include "cstdlib"
#include <fstream>
using namespace std;
int main()
{

    char ch, sourceFile[20], targetFile[20];
    FILE* fs, * ft;
    cout << "Enter the Name of Source File: ";
    cin >> sourceFile;
    fs = fopen(sourceFile, "r");
    if (fs == NULL)
    {
        cout << "\nError Occurred!";
    }
    cout << "\nEnter the Name of Target File: ";
    cin >> targetFile;
    ft = fopen(targetFile, "w");
    if (ft == NULL)
    {
        cout << "\nError Occurred!";
    }
    ch = fgetc(fs);
```

```
    while (ch != EOF)
    {
        fputc(ch, ft);
        ch = fgetc(fs);
    }
    cout << "\nFile copied successfully.";
    fclose(fs);
    fclose(ft);
    cout << endl;
}
```

6. deletefile.cpp:

```
#include "pthread.h"
#include "iostream"
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include "bits/stdc++.h"
#include "ctime"
#include "cstdlib"
#include<fstream>
using namespace std;
int main()
{
    float n = 0.05;
    int status;
    char fileName[20];
    cout << "Enter the Name of File: " << endl;
    cin >> fileName;
```

```
    status = remove(fileName);
    if (status == 0)
        cout << "\nFile Deleted Successfully!" << endl;
    else
        cout << "\nErorr Occurred!" << endl;
    cout << endl;
}
```

7. **Fileproperties.cpp:**

```
#include <iostream>
#include <sys/stat.h>
#include <unistd.h>

int main()
{
    std::string filename;
    while(true)
    {
        std::cout << "Enter File name: ";
        std::cin >> filename;

        struct stat fileStat;

        if (stat(filename.c_str(), &fileStat) < 0) {
            std::cout << "Failed to get file properties\n";
            return 1;
        }

        std::cout << "File name: " << filename << "\n";
```

```
        std::cout << "Size: " << fileStat.st_size << " bytes\n";
        std::cout << "Owner ID: " << fileStat.st_uid << "\n";
        std::cout << "Group ID: " << fileStat.st_gid << "\n";
        std::cout << "Permissions: " << fileStat.st_mode << "\n";
    }



    return 0;
}
```

8. **rename.cpp:**

```
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

int main() {
    char oldname[100], newname[100];
    cout << "Enter the name of the file to be renamed: ";
    cin >> oldname;
    cout << "Enter the new name for the file: ";
    cin >> newname;

    int status = rename(oldname, newname);

    if(status == 0) {
        cout << "File renamed successfully.\n";
    } else {
```

```
        perror("Error renaming file");
    }

    return 0;
}
```

9. notepad.cpp:

```
#include <cstdlib>
#include <iostream>
#include <unistd.h>

using namespace std;

int main() {
    char choice;
    cout << "notepad started (PID: " << getpid() << ")\n";
    while (true)
    {
        cout << "Do you want to use notepade? (q to quit,m to minimize and any other key to continue): ";
        cin >> choice;

        if (choice == 'q' || choice == 'Q') {
            break;
        }

        if (choice == 'm' || choice == 'M') {
            cout << "minimizing notepad PID ( " << getpid() << "), you can resume later.\n";
            continue;
        }
```

```
        string fileName;
        string filePath = "/home/meerab/Downloads/mini_OS_project/mini_OS/notepad.txt"; //change this to your desired directory

        cout << "Enter the file name: ";
        cin >> fileName;

        string command = "gedit " + filePath + fileName;
        system(command.c_str());
    }

    return 0;
}
```

10. song.cpp:

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main() {
    string filename;
    cout << "Enter the full name of the file (e.g., song.mp4): ";
    cin >> filename;
    // Command to play only the audio of the video file
    string command = "vlc --no-video " + filename + " --play-and-exit";
    system(command.c_str());
    return 0;
}
```

11. video.cpp:

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main() {
    string filename;
    cout << "Enter the name of the video file (without .mp4): ";
    cin >> filename;
    string command = "xdg-open " + filename + ".mp4";
    system(command.c_str());
    return 0;
}
```

12. tictactoe.cpp:

```
#include <iostream>
#include <unistd.h>
using namespace std;

// Function to print the Tic-Tac-Toe board
void printboard(int board[3][3]) {
    cout << "\nCurrent Board:\n";
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[i][j] == 0)
                cout << "_ ";
            else if (board[i][j] == 1)
                cout << "X ";
            else
```

```
            cout << "O ";
        }
        cout << endl;
    }
    cout << endl;
}

// Function to check if there's a winner
bool checkwinner(int board[3][3], int& who) {
    // Rows and columns
    for (int i = 0; i < 3; i++) {
        if (board[i][0] == board[i][1] && board[i][0] == board[i][2] && board[i][0] != 0) {
            who = board[i][0];
            return true;
        }
        if (board[0][i] == board[1][i] && board[0][i] == board[2][i] && board[0][i] != 0) {
            who = board[0][i];
            return true;
        }
    }

    // Diagonals
    if (board[0][0] == board[1][1] && board[0][0] == board[2][2] && board[0][0] != 0) {
        who = board[0][0];
        return true;
    }
    if (board[0][2] == board[1][1] && board[0][2] == board[2][0] && board[0][2] != 0) {
        who = board[0][2];
        return true;
```

```
    }

    return false;
}

int main() {
    char choice;
    cout << "tictactoe started (PID: " << getpid() << ")\n";
    while (true)
    {
        cout << "Do you want to play this game? (q to quit,m to minimize and any other key to continue): ";
        cin >> choice;

        if (choice == 'q' || choice == 'Q') {
            break;
        }

        if (choice == 'm' || choice == 'M') {
            cout << "minimizing tictactoe PID ( " << getpid() << "), you can resume later.\n";
            continue;
        }
        int board[3][3] = { 0 };  // Empty board
        int player = 1;         // Player 1 starts
        int row, col;
        int who;
        bool winner = false;

        cout << "Welcome to Tic-Tac-Toe!\n";
        printboard(board);
```

```
for (int turn = 0; turn < 9 && !winner; turn++) {
    cout << "Player " << player << "'s turn.\n";

    cout << "Enter row (0-2) and column (0-2): ";
    cin >> row >> col;

    while (row < 0 || row > 2 || col < 0 || col > 2 || board[row][col] != 0) {
        cout << "Invalid or occupied cell. Try again: ";
        cin >> row >> col;
    }

    board[row][col] = player;
    printboard(board);

    winner = checkwinner(board, who);

    if (winner) {
        cout << "The winner is Player " << who << "!\n";
        break;
    }

    // Ask if user wants to quit
    int choice;
    cout << "Press 1 to continue, 0 to quit: ";
    cin >> choice;
    if (choice == 0) {
        cout << "Game ended by user.\n";
        break;
```

```
            }

            // Switch player
            player = (player == 1) ? 2 : 1;
        }

        if (!winner) {
            cout << "Game ended in a draw.\n";
        }
    }

    return 0;
}
```

13. towerOfHonoi.cpp:

```
#include<iostream>
#include <unistd.h>
using namespace std;
class node
{
public:
    node* next;
    int data;

    node()
    {
        next = NULL;

    }
```

```
};
class ADT
{
public:
    node* top;
    int count = -1;
    ADT()
    {
        top = NULL;
    }

    void push(int element)
    {
        if (isfull() == true)
        {
            return;
        }
        if (top == NULL)
        {
            node* new_node = new node;
            new_node->data = element;
            top = new_node;
            new_node->next = NULL;
            count++;
        }
        else
        {
            node* new_node = new node;
            new_node->data = element;
```

```
        new_node->next = top;
        top = new_node;
        count++;
    }
}
bool isempty()
{
    if (top == NULL)
    {
        return true;
    }
    return false;
}
void display()
{
    node* current = top;
    while (current != NULL)
    {
        cout << current->data << endl;
        current = current->next;
    }
}

int pop()
{
    int var;
    if (top == NULL)
    {
        cout << "stack is underflow" << endl;
```

```
    }
    else
    {
        var = top->data;
        node* p = top;
        top = top->next;
        delete p;
        p = NULL;
        count--;
    }
    return var;
}

int Top()
{
    int var;
    if (top == NULL)
    {
        return -1;
    }
    var = top->data;
    return var;
}

bool isfull()
{
    if (count == 4)
    {
        return true;
```

```
        }
        return false;
    }



};
void diskmoves(ADT& l1, ADT& l2)
{
    if (l1.isempty() != true && l2.isfull() != true)
    {
        if (l2.isempty() == true)
        {
            l2.push(l1.Top());
            l1.pop();


        }
        else
        {
            if (l1.Top() < l2.Top() && l1.Top() != -1)
            {
                l2.push(l1.Top());
                l1.pop();
            }


        }
    }
    else
    {
        return;
```

```
    }
}
void display_if(ADT& l1, ADT& l2, ADT& l3)
{
    cout << endl;
    cout << "first cupboard:" << endl;
    l1.display();
    cout << endl;
    cout << "second cupboard:" << endl;
    l2.display();
    cout << endl;
    cout << "third cupboard:" << endl;
    l3.display();
    cout << endl;
}
bool game_solve(ADT& l1)
{
    if (l1.isfull() == true)
    {
        return true;
    }
    return false;
}

int min_moves_to_win()
{
    int count = 5;
    int n = 1;
    for (int i = 0; i < count; i++)
```

```
    {
        n = n * 2;
    }
    n = n - 1;
    return n;
}

// check that
void initialize(ADT& l1, ADT& l2, ADT& l3)
{
    for (int i = 5; i > 0; i--)
    {
        l1.push(i);
    }

}
int main()
{
    char choice;
    cout << "towerofHonoi started (PID: " << getpid() << ")\n";
    while (true)
    {
        cout << "Do you want to play this game? (q to quit,m to minimize and any other key to continue): ";
        cin >> choice;

        if (choice == 'q' || choice == 'Q') {
            break;
        }
```

```
if (choice == 'm' || choice == 'M') {
    cout << "minimizing towerOfHonoi PID ( " << getpid() << "), you can resume later.\n";
    continue;
}
ADT l1;
int count1 = 0;
ADT l2, l3;
int choice = 0;
int mn = 0;
cout << "-----------------------------" << endl;
cout << "|||||    THE TOWER OF HENOI    |||||" << endl;
cout << "-----------------------------" << endl;
cout << "1.playing game" << endl;
cout << "2.exit the game" << endl;
cin >> choice;
if (choice == 1)
{
    initialize(l1, l2, l3);
    while (true)
    {
        display_if(l1, l2, l3);
        cout << "1. move from 1 to 2" << endl;
        cout << "2. move from 1 to 3" << endl;
        cout << "3. move from 2 to 1" << endl;
        cout << "4. move from 2 to 3" << endl;
        cout << "5. move from 3 to 1" << endl;
        cout << "6. move from 3 to 2" << endl;
        cin >> mn;
        if (mn == 1)
```

```
{
    diskmoves(l1, l2);
    count1++;
}
else if (mn == 2)
{
    diskmoves(l1, l3);
    count1++;
}
else if (mn == 3)
{
    diskmoves(l2, l1);
    count1++;
}
else if (mn == 4)
{
    diskmoves(l2, l3);
    count1++;
}
else if (mn == 5)
{
    diskmoves(l3, l1);
    count1++;
}
else if (mn == 6)
{
    diskmoves(l3, l2);
    count1++;
}
```

```
        else
        {
            cout << "you enter the wrong number" << endl;
        }
        if (game_solve(l3) == true)
        {
            cout << "solve game successfully" << endl;
            display_if(l1, l2, l3);
            int mov = min_moves_to_win();
            cout << endl;
            cout << "minimum moves to win:" << mov << endl;
            if (count1 == mov)
            {
                cout << "your moves:" << count1 << endl;
                cout << "you complete the game in minimum number of moves" << endl;
            }
            else
            {
                cout << "your moves:" << count1 << endl;
                cout << "you cannot win the game in using minimum number of moves" << endl;
            }
            exit(1);
        }

    }
}
else
{
    return 0;
```

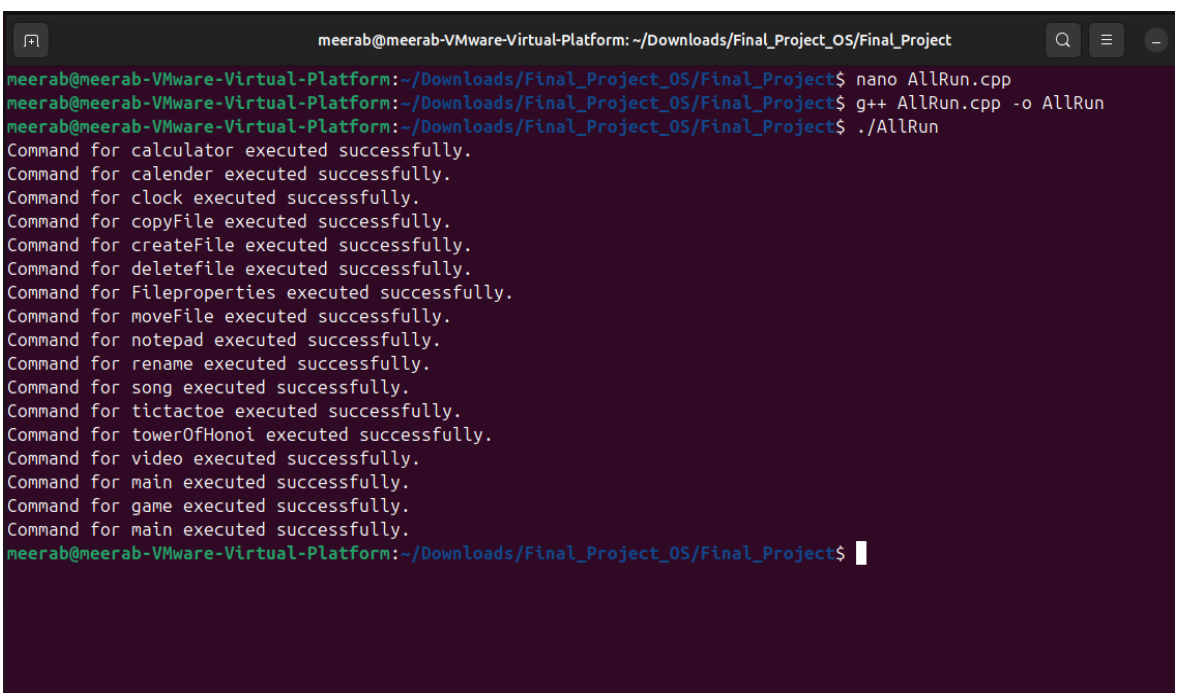
```
        }
    }
    return 0;
}
```

14. game.cpp:

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main() {
    srand(time(0));
    int number = rand() % 100 + 1; // Random number between 1 and 100
    int guess;
    int attempts = 0;

    cout << "====== Guess the Number Game ======\n";
    cout << "I'm thinking of a number between 1 and 100.\n";
    cout << "Enter -1 at any time to quit.\n";

    do {
        cout << "Enter your guess: ";
        cin >> guess;

        if (guess == -1) {
            cout << "You chose to quit the game. The number was: " << number << endl;
            break;
```

```
        }

        attempts++;

        if (guess < number) {
            cout << "Too low!\n";
        } else if (guess > number) {
            cout << "Too high!\n";
        } else {
            cout << "Congratulations! You guessed it in " << attempts << " attempts.\n";
        }

    } while (guess != number);

    return 0;
}
```

Outputs:



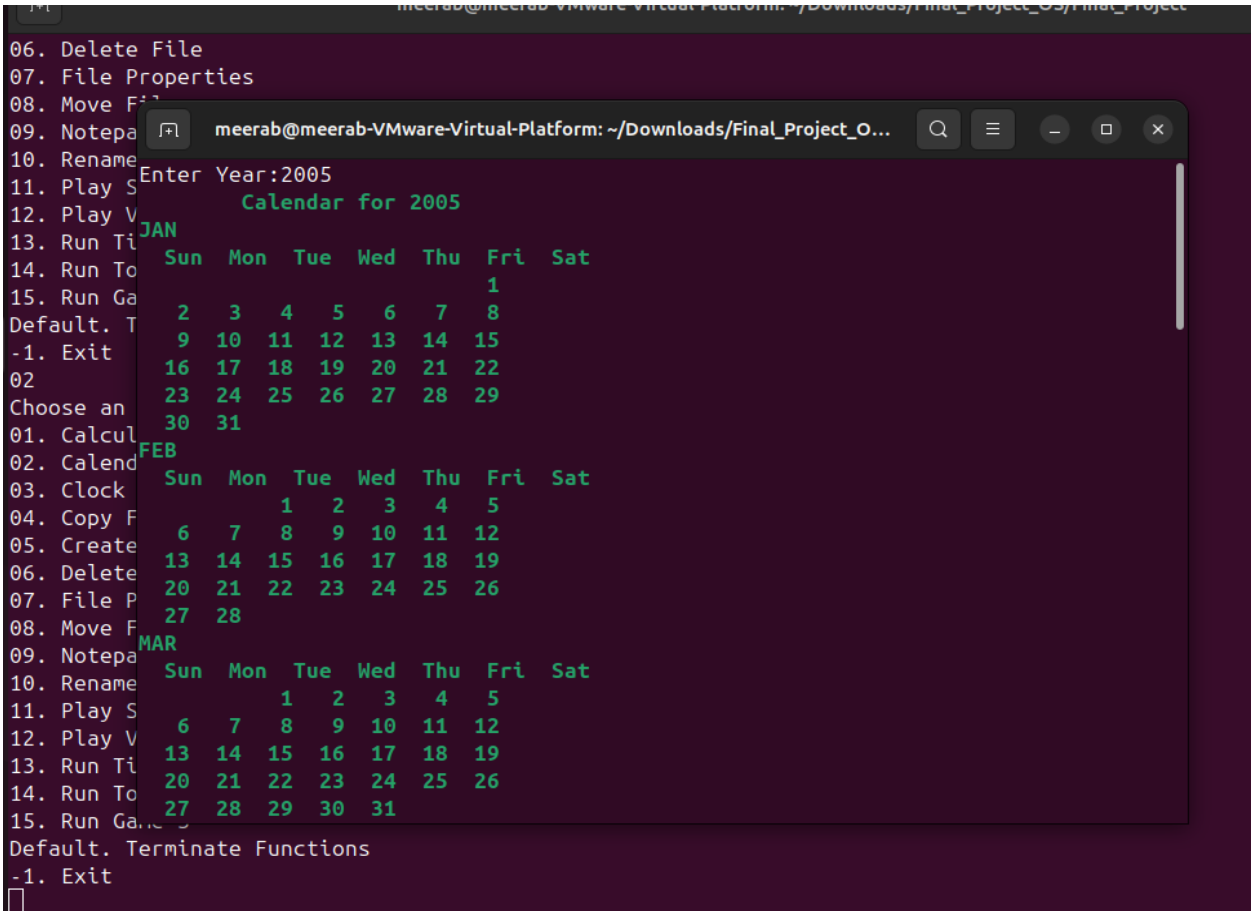
```
Command for tictactoe executed successfully.
Command for towerOfHonoi executed successfully.
Command for video executed successfully.
Command for main executed successfully.
Command for game executed successfully.
Command for main executed successfully.
meerab@meerab-VMware-Virtual-Platform:~/Downloads/Final_Project_OS/Final_Project$ nano main.cpp
meerab@meerab-VMware-Virtual-Platform:~/Downloads/Final_Project_OS/Final_Project$ g++ main.cpp -o main
meerab@meerab-VMware-Virtual-Platform:~/Downloads/Final_Project_OS/Final_Project$ ./main
Welcome to fmOS!!
getting started.....
Enter RAM: 2
Enter storage: 256
Choose an option:
01. Calculator
02. Calendar
03. Clock
04. Copy File
05. Create File
06. Delete File
07. File Properties
08. Move File
09. Notepad
10. Rename File
11. Play Song
12. Play Video
13. Run Tic Tac Toe
14. Run Tower of Honoi
15. Run Game 3
Default. Terminate Functions
-1. Exit
```

Calculator:

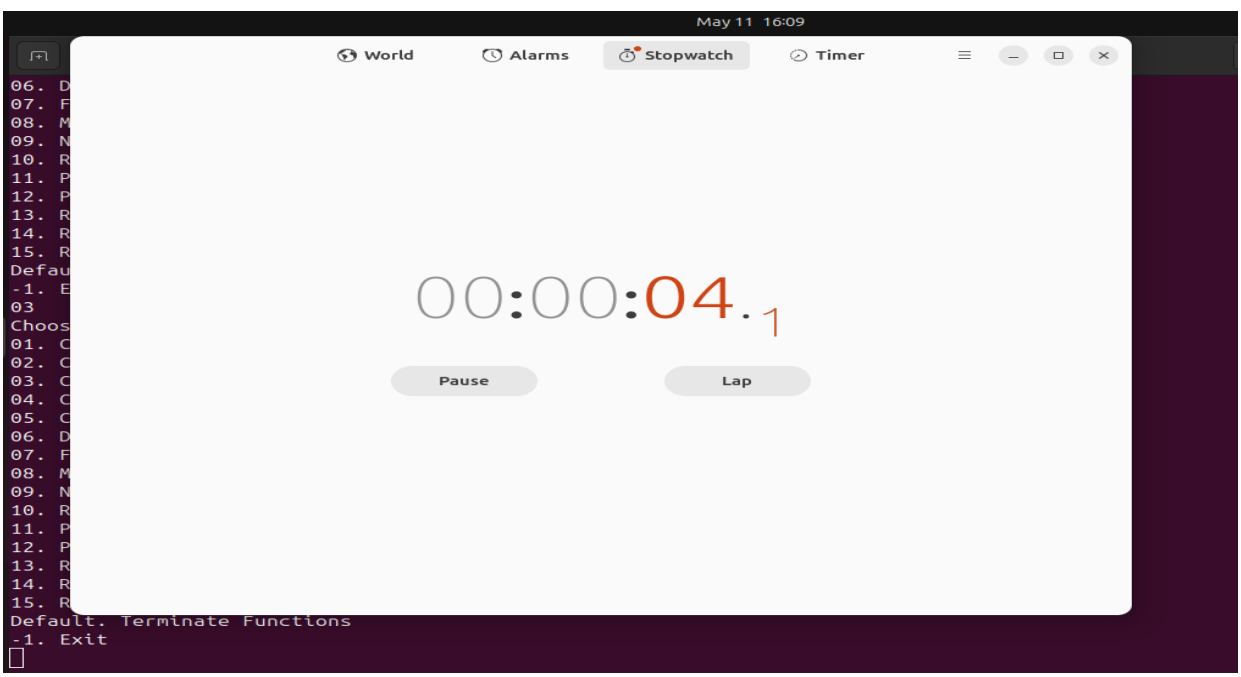
```
08. Move File
09. Notepad
10. Rename
11. Play S
12. Play V
13. Run Ti
14. Run To
15. Run Ga
Default. T
-1. Exit
01
RAM : 1
Storage :
Choose an
01. Calcul
02. Calend
03. Clock
04. Copy F
05. Create
06. Delete
07. File P
08. Move F
09. Notepa
10. Rename
11. Play S
12. Play V
13. Run Ti
14. Run To
15. Run Ga
Default. Terminate Functions
-1. Exit
```

```
meerab@meerab-VMware-Virtual-Platform: ~/Downloads/Final_Project_O...
calculator started (PID: 31947 )
type operation (+, -, *, /, ^, %) or:
 m to minimize
q to quit(close)
Enter an operation: +
Enter two numbers: 4 7
Result: 11
Enter an operation: ^
Enter two numbers: 2 4
Result: 16
Enter an operation: q
calculator closing...
meerab@meerab-VMware-Virtual-Platform:~/Downloads/Final_Project_OS/Final_Project
$
```

Calendar:



Clock:



Create File:

```
06. Delete File
07. File Properties
08. Move Fil
09. Notepa
10. Rename
11. Play S
12. Play V
13. Run Ti
14. Run To
15. Run Ga
Default. T
-1. Exit
05
Choose an
01. Calcul
02. Calend
03. Clock
04. Copy F
05. Create
06. Delete
07. File P
08. Move F
09. Notepa
10. Rename
11. Play S
12. Play V
13. Run Ti
14. Run To
15. Run Ga
Default. Terminate Functions
-1. Exit
```

```
meerab@meerab-VMware-Virtual-Platform: ~/Downloads/Final_Project_O...
enter name :
meerab.txt
File created successfully.meerab@meerab-VMware-Virtual-Platform:~/Downloads/Fina
l_Project_OS/Final_Project$
```

Copy file:

```
File Properties
Move Fil
Notepa
Rename
Play S
Play V
Run Ti
Run To
Run Ga
ault. T
Exit

ose an
Calcul
Calend
Clock
Copy F
Create
Delete
File P
Move F
Notepa
Rename
Play S
Play V
Run Ti
Run To
Run Ga
```

```
meerab@meerab-VMware-Virtual-Platform: ~/Downloads/Final_Project_O...
Enter the Name of Source File: meerab.txt

Enter the Name of Target File: fatima

File copied successfully.
meerab@meerab-VMware-Virtual-Platform:~/Downloads/Final_Project_OS/Final_Project
$
```

Delete file:

```
meerab@meerab-VMware-Virtual-Platform: ~/Downloads/Final_Project_OS/Final_Project
06. Delete File
07. File Properties
08. Move Fi
09. Notepa
10. Rename
11. Play S
12. Play V
13. Run Ti
14. Run To
15. Run Ga
Default. T
-1. Exit
06
Choose an
01. Calcul
02. Calend
03. Clock
04. Copy F
05. Create
06. Delete
07. File P
08. Move F
09. Notepa
10. Rename
11. Play S
12. Play V
13. Run Ti
14. Run To
15. Run Ga
Default. Terminate Functions
-1. Exit
```

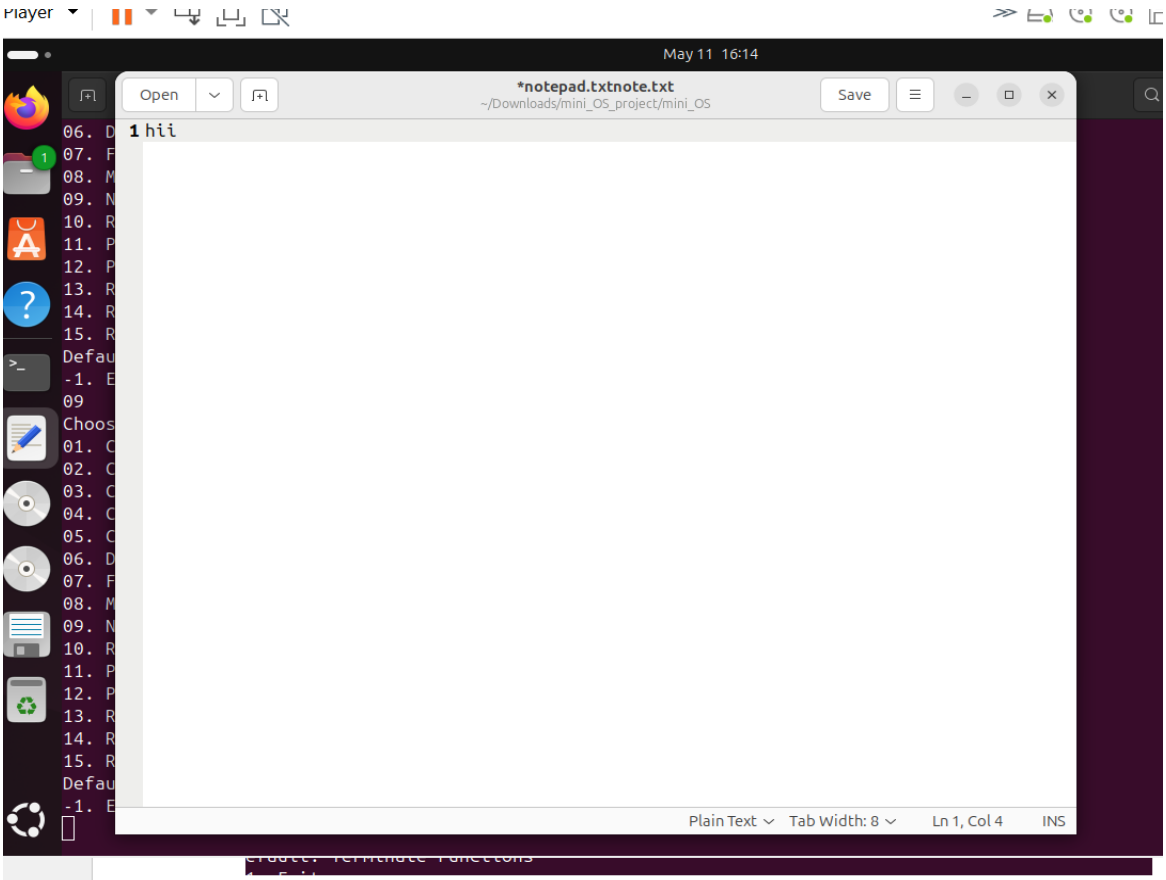
```
meerab@meerab-VMware-Virtual-Platform: ~/Downloads/Final_Project_O...
Enter the Name of File:
fatima

File Deleted Successfully!

meerab@meerab-VMware-Virtual-Platform:~/Downloads/Final_Project_OS/Final_Project
$
```

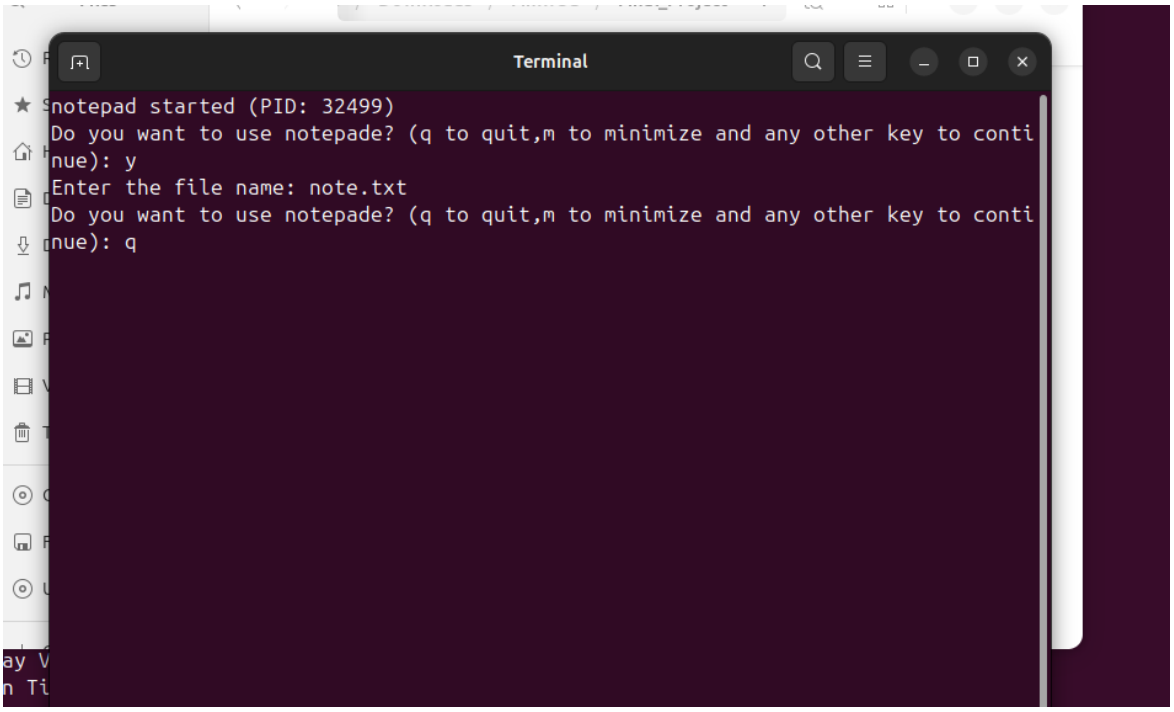
File properties:

```
6. Delete File
7. File Properties
8. Move Fi
9. Notepa
0. Rename
1. Play S
2. Play V
3. Run Ti
4. Run To
5. Run Ga
efault. T
1. Exit
7
hoose an
1. Calcul
2. Calend
3. Clock
4. Copy F
5. Create
6. Delete
7. File P
8. Move F
9. Notepa
0. Rename
1. Play S
2. Play V
3. Run Ti
4. Run To
5. Run Ga
efault. Terminate Functions
```

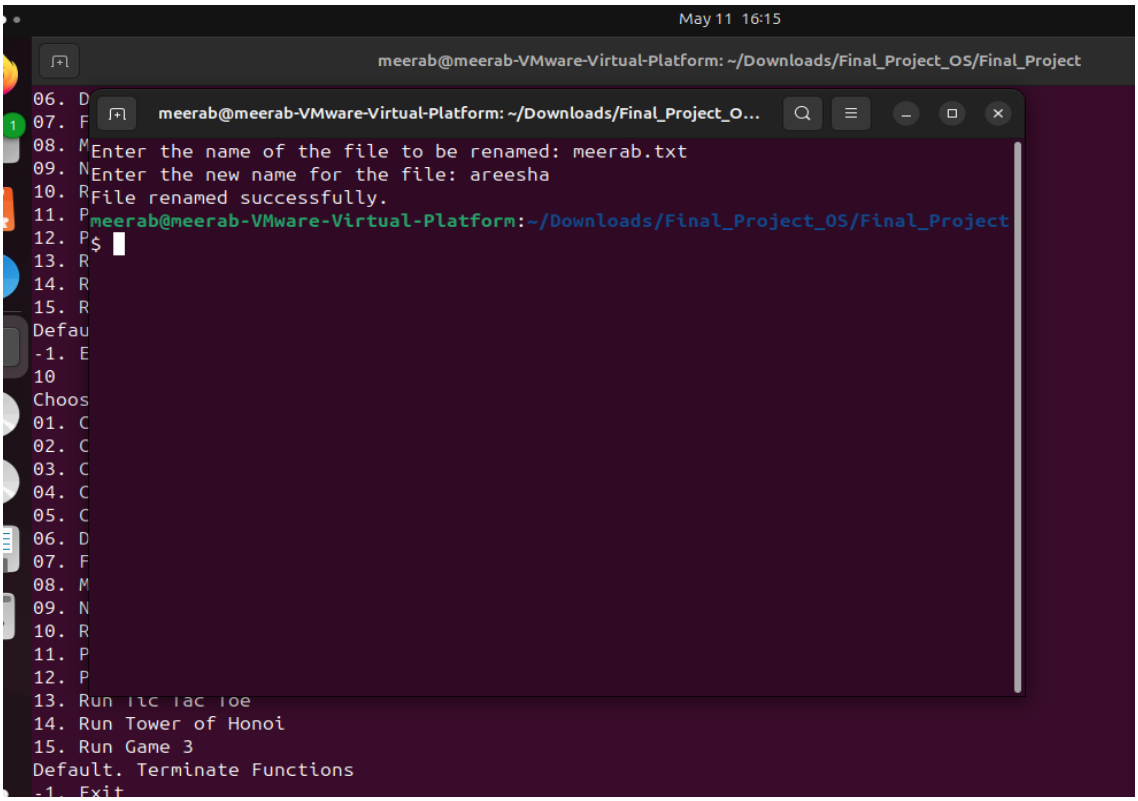
```
Terminal
Enter File name: meerab.txt
File name: meerab.txt
Size: 0 bytes
Owner ID: 1000
Group ID: 1000
Permissions: 33204
Enter File name:
```

Notepad:

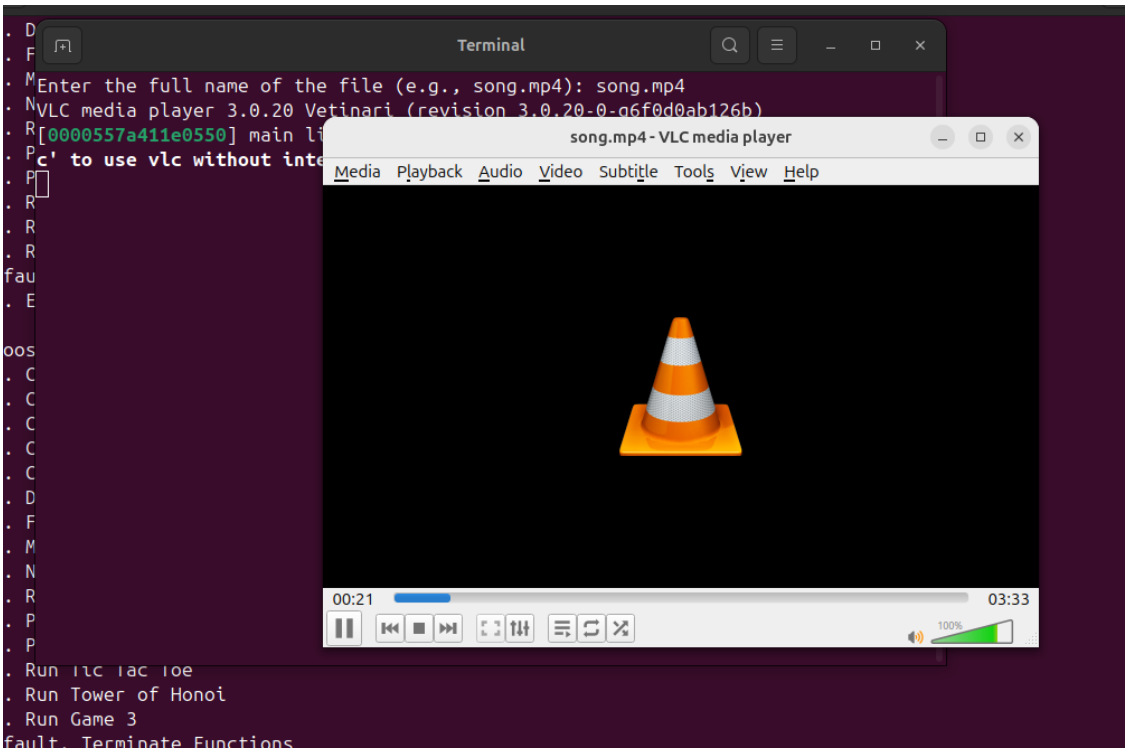

May 11 16:14
Open
*notepad.txtnote.txt
~/Downloads/mini_OS_project/mini_OS
Save
1 hii
Plain Text
Tab Width: 8
Ln 1, Col 4
INS


Terminal
notepad started (PID: 32499)
Do you want to use notepade? (q to quit,m to minimize and any other key to continue): y
Enter the file name: note.txt
Do you want to use notepade? (q to quit,m to minimize and any other key to continue): q

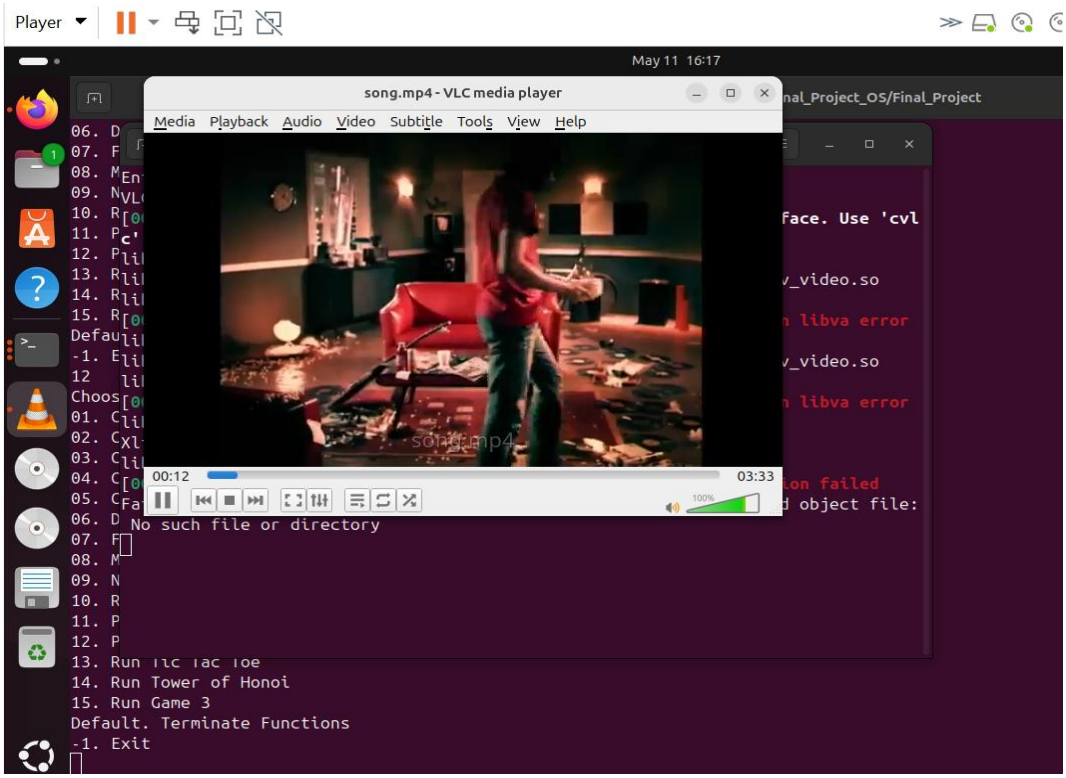
File rename:



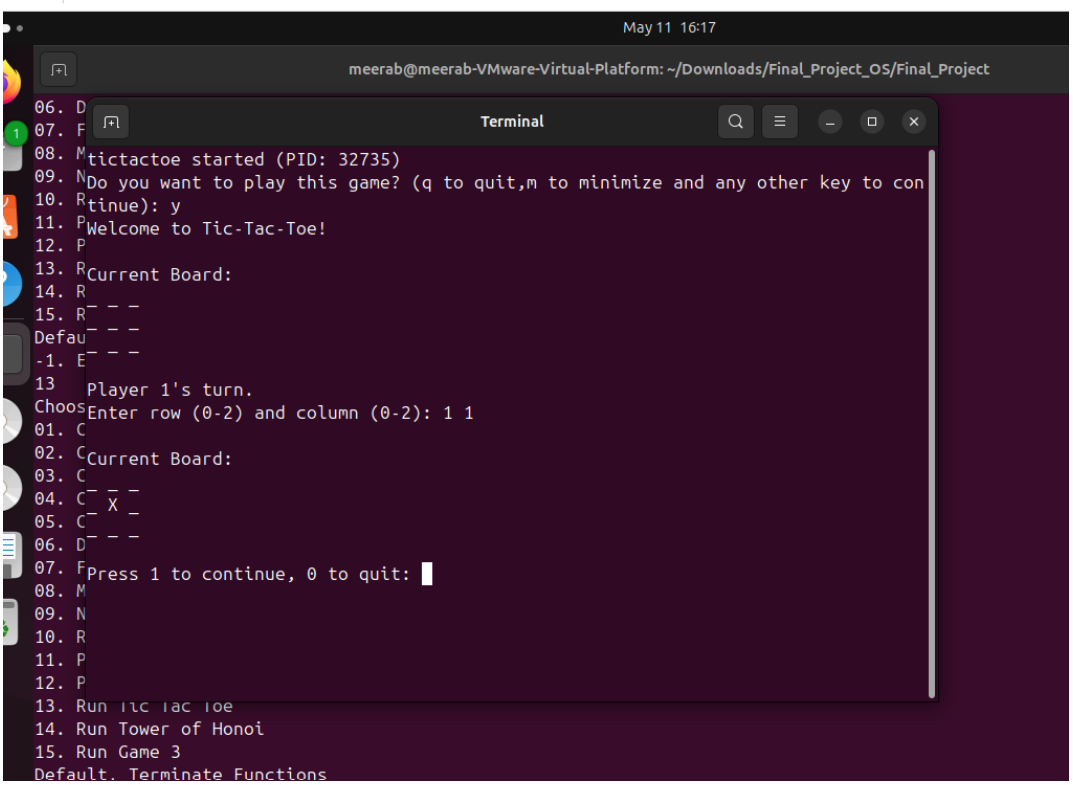
Song:



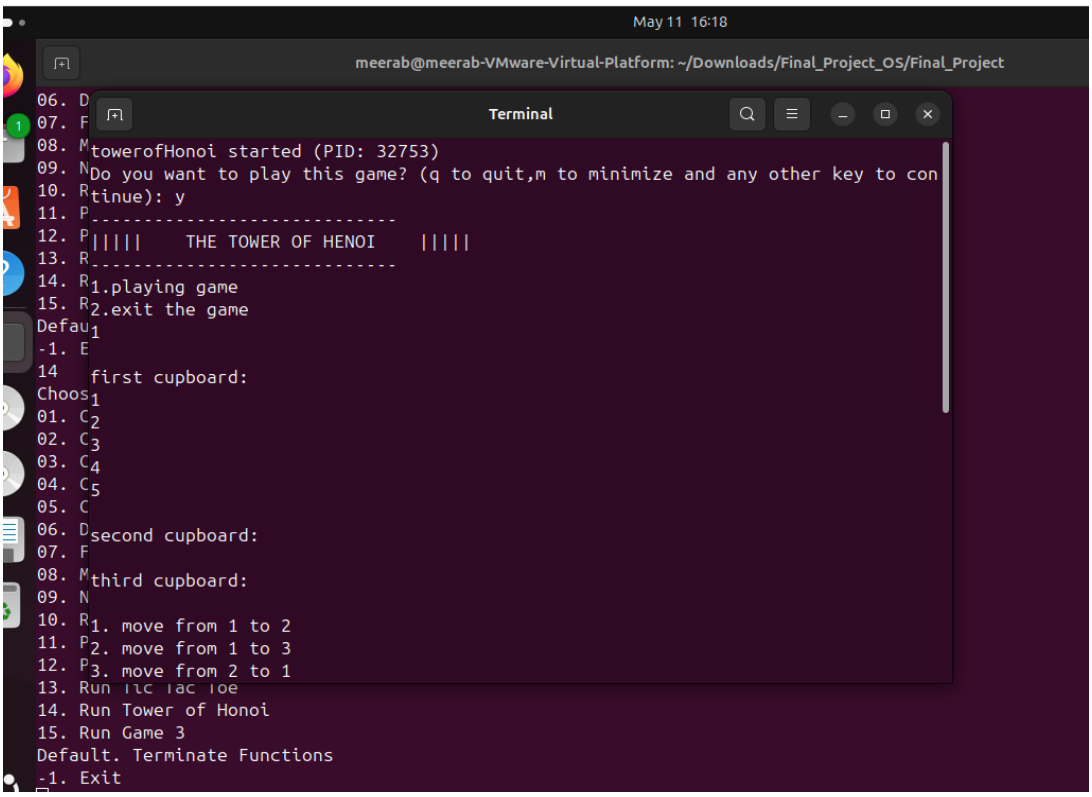
Video:



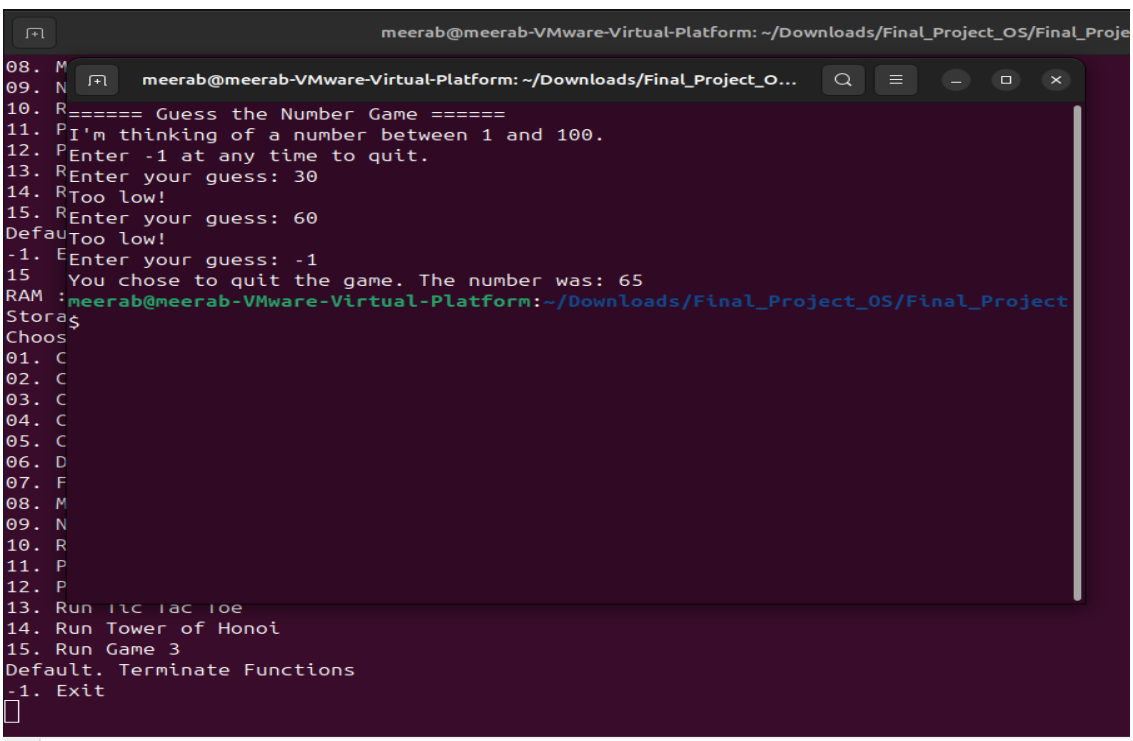
Tic Tac Toe:



Tower Of Honoi:



Guessing Game:



```
meerab@meerab-VMware-Virtual-Platform: ~/Do
Default. Terminate Functions
-1. Exit
15
RAM : 1
Storage : 255.8
Choose an option:
01. Calculator
02. Calendar
03. Clock
04. Copy File
05. Create File
06. Delete File
07. File Properties
08. Move File
09. Notepad
10. Rename File
11. Play Song
12. Play Video
13. Run Tic Tac Toe
14. Run Tower of Honoi
15. Run Game 3
Default. Terminate Functions
-1. Exit
-1
31930 ./main
a------------
31930
a------------
0
sh: 1: xdotool: not found
Killed
meerab@meerab-VMware-Virtual-Platform:~/Downloads/Final_Project_
```

Conclusion:

This project helped build a small OS in C++ that can run different tasks and manage resources. It was a great way to learn how operating systems handle multitasking, memory, and process control. The project made OS concepts easier to understand in a practical way.