

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра Систем Штучного Інтелекту



Звіт
до лабораторної роботи № 5
з дисципліни
Операційні системи
на тему:
“Робота з динамічними бібліотеками в ОС
Windows”

Виконав: студент КН-217

Ратушняк Денис

Прийняв: доцент каф. СШІ

Кривенчук Ю. П.

Мета роботи: Ознайомитися з динамічно-зв'язувальними бібліотеками (Dynamic-link library) в ОС Windows. Навчитися реалізовувати динамічно-зв'язувальні бібліотеки.

Завдання:

- [Складність 1] Реалізувати лабораторну роботу №4 у вигляді динамічно-зв'язувальної бібліотеки. Запустити створену бібліотеку з командної стрічки (cmd.exe) за допомогою rundll32.exe
- [Складність 2] Створити окрему програму і реалізувати статичний зв'язок між програмою та бібліотекою.
- [Складність 3] Створити окрему програму і реалізувати динамічний зв'язок між програмою та бібліотекою.

Код програми

```
#include <bits/stdc++.h>
#include <windows.h>
#include <tchar.h>
// #include "lab5dllheaderv1.h"
using namespace std;
int main(){
    HMODULE h =
    LoadLibrary("A:\\T\\3_term\\Operating_Systems\\5labDLLv1\\bin\\Debug\\5labDLLv1.dll");
    if(!h){
        cout << "Could not open the library";
        return 1;
    }
    FARPROC fn = GetProcAddress(h, "runTask");
    if(!fn){
        cout << "Could not locate the function";
        return 2;
    }
    fn();
    //runTask();
    return 0;
}
```

Код хедера

```
#include <tchar.h>
#include <windows.h>
using namespace std;

#ifdef LAB5DLLHEADERV1_EXPORTS
#define LAB5DLLHEADERV1_API __declspec(dllexport)
#else
#define LAB5DLLHEADERV1_API __declspec(dllimport)
#endif

extern "C" LAB5DLLHEADERV1_API void runTask();

extern "C" LAB5DLLHEADERV1_API void calc_points();

extern "C" LAB5DLLHEADERV1_API void read(vector< vector<long long> > &a, ifstream &input);

extern "C" LAB5DLLHEADERV1_API void print(vector< vector<long long> > &a, ofstream &output);

extern "C" LAB5DLLHEADERV1_API DWORD WINAPI MyThreadFunction(LPVOID lpParam);

extern "C" LAB5DLLHEADERV1_API void solve(int sx, int sy, int step, int type, int num);
```

Код .cpp файлу з якого була створена бібліотека

```
#include <bits/stdc++.h>

#include "lab5dllheaderv1.h"

typedef long long ll;
typedef long double ld;
typedef vector< vector<ll> > matrix;

//HERE YOU CAN CHANGE INPUT
const int TOTAL_THREADS = 100;
const int MAX_SEM_COUNT = 10;
string version = "10000_10000_10000";
ll type_of_task = 1;
ll low_priority_thread = 3;
ll high_priority_thread = 7;

//HERE YOU CAN CHANGE INPUT
ll done_threads = 0;
ll needThreads;
HANDLE hMutex;
HANDLE ghSemaphore;
HANDLE hCriticalSection;
const long long mod = 1e9+7;

typedef struct MyData
{
    int startx, starty, step, type;
    int threadNum;
} MYDATA, *PMYDATA;

//PMYDATA pDataArray[TOTAL_THREADS];
MYDATA pDataArray[TOTAL_THREADS];
DWORD dwThreadIdArray[TOTAL_THREADS];
HANDLE hThreadArray[TOTAL_THREADS];
HANDLE stackTh[64];

matrix A;
ll n,m,k,max_number;
ll add;
vector < pair<ll,ll> > diag[TOTAL_THREADS];

vector< pair< pair<ll,ll>, pair<ll,ll> > > points_for_diag;

void solve(int sx, int sy, int step, int type, int num)
{
    if(type == 1)
    {
        int i = sx;
        int j = sy;
        while(step--)
        {
            diag[num][i+j].first += A[i][j];
            diag[num][i+j].second ++;
            diag[num][i-j+add].first += A[i][j];
            diag[num][i-j+add].second ++;
            //cout << i << " " << j << endl;
            //cout << num << " " << step << endl;
            j++;
            if(j == m)
            {
```

```

        j = 0;
        i++;
        if(i == n) break;
    }
}
else
{
    int now = sx * n + sy;

    while(now < n * m)
    {
        int i = now / m;
        int j = now % m;

        diag[num][i+j].first += A[i][j];
        diag[num][i+j].second++;

        diag[num][i-j+add].first += A[i][j];
        diag[num][i-j+add].second++;
        //cout << num << " " << now << " " << step << endl;
        now += step;
    }
}
ll ans = 0;
for(int i = 1 ; i < diag[num].size(); ++i)
{
    if(diag[num][i].first * diag[num][ans].second > diag[num][i].second * diag[num][ans].first) ans = i;
}
// for(int i = 0; i < diag[0].size(); ++i) cout << diag[0][i].first << " " << diag[0][i].second << endl;

ld result = (ld)diag[0][ans].first / (ld)diag[0][ans].second;
WaitForSingleObject(hMutex, INFINITE);
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
COORD pos = {0, 1};
SetConsoleCursorPosition(hConsole, pos);
cout << "Thread " << num << " MaxAverage = " << result;
ReleaseMutex(hMutex);
}

ld threadTime[TOTAL_THREADS];

DWORD WINAPI MyThreadFunction(LPVOID lpParam)
{
    if(MAX_SEM_COUNT)
    {
        DWORD dwWaitResult;
        BOOL bContinue=TRUE;

        while(bContinue)
        {
            // Try to enter the semaphore gate.
            dwWaitResult = WaitForSingleObject(
                ghSemaphore, // handle to semaphore
                0);          // zero-second time-out interval

            if(dwWaitResult == WAIT_OBJECT_0){

                // TODO: Perform task
                LARGE_INTEGER st, en, fq;
                QueryPerformanceFrequency(&fq);
            }
        }
    }
}

```

```

QueryPerformanceCounter(&st);

PMYDATA pDataVar;

pDataVar = (PMYDATA)lpParam;
MYDATA DataVar = *pDataVar;
//cout << "HERE IS " << " " << pDataVar->threadNum << " thread" << endl;
solve(DataVar.startx, DataVar.starty, DataVar.step, DataVar.type, DataVar.threadNum);
QueryPerformanceCounter(&en);

ld currtime = (en.QuadPart-st.QuadPart)*1000.0/fq.QuadPart;
threadTime[pDataVar->threadNum] = currtime;
WaitForSingleObject(hMutex, INFINITE);
done_threads ++;
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
COORD pos = {0, 0};
SetConsoleCursorPosition(hConsole, pos);
cout << fixed << setprecision(3) << 100.0*(double)done_threads/(double)needThreads << '%';
ReleaseMutex(hMutex);

//printf("Thread %d: wait succeeded\n", GetCurrentThreadId());
bContinue=FALSE;

// Release the semaphore when task is finished

if (!ReleaseSemaphore(
    ghSemaphore, // handle to semaphore
    1,           // increase count by one
    NULL))       // not interested in previous count
{
    printf("ReleaseSemaphore error: %d\n", 3);
}

// The semaphore was nonsignaled, so a time-out occurred.
else{
    //printf("Thread %d: wait timed out\n", GetCurrentThreadId());
    Sleep(3);
}
}
return 0;
}

LARGE_INTEGER st, en, fq;
QueryPerformanceFrequency(&fq);
QueryPerformanceCounter(&st);

PMYDATA pDataVar;

pDataVar = (PMYDATA)lpParam;
MYDATA DataVar = *pDataVar;
//cout << "HERE IS " << " " << pDataVar->threadNum << " thread" << endl;
solve(DataVar.startx, DataVar.starty, DataVar.step, DataVar.type, DataVar.threadNum);
QueryPerformanceCounter(&en);

ld currtime = (en.QuadPart-st.QuadPart)*1000.0/fq.QuadPart;
threadTime[pDataVar->threadNum] = currtime;
WaitForSingleObject(hMutex, INFINITE);
done_threads ++;
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
COORD pos = {0, 0};

```

```

        SetConsoleCursorPosition(hConsole, pos);
        cout << fixed << setprecision(3) << 100.0*(double)done_threads/(double)needThreads << '%';
        ReleaseMutex(hMutex);
        return 0;
    }

void print(matrix &a)
{
    for(int i = 0; i < a.size(); ++ i)
    {
        for(int j = 0; j < a[i].size(); ++ j)
        {
            cout << a[i][j] << " ";
        }
        cout << "\n";
    }
    cout << "\n";
}

void print(matrix &a, ofstream &output)
{
    for(int i = 0; i < a.size(); ++ i)
    {
        for(int j = 0; j < a[i].size(); ++ j)
        {
            output << a[i][j] << " ";
        }
        output << "\n";
    }
    output << "\n";
}

void read(matrix &a, ifstream &input)
{
    for(int i = 0; i < a.size(); ++ i)
    {
        for(int j = 0; j < a[i].size(); ++ j)
        {
            input >> a[i][j];
        }
    }
}

void calc_points()
{
    points_for_diag.resize(2 * m + 2 * n - 2);
    ll x1,y1,x2,y2;
    x1 = y1 = x2 = y2 = 0;

    for(int i = 0; i < (2 * m + 2 * n - 2)/2; ++i)
    {
        points_for_diag[i] = {{x1, y1}, {x2, y2}};
        if(x1 == n - 1) y1++;
        else x1++;

        if(y2 == m - 1) x2++;
        else y2++;
    }

    x1 = x2 = 0;
    y1 = y2 = m-1;
}

```

```

for(int i = (2 * m + 2 * n - 2)/2; i < (2 * m + 2 * n - 2); ++i)
{
    points_for_diag[i] = {{x1, y1}, {x2, y2}};
    if(y1 == 0) x1++;
    else y1--;

    if(x2 == n - 1) y2--;
    else x2++;
}
}

void runTask()
{
    string test_path = "A:\\T\\3_term\\Operating_Systems\\5lab\\tests\\" + version + "_in.txt";

    low_priority_thread %= TOTAL_THREADS;
    high_priority_thread %= TOTAL_THREADS;
    if(low_priority_thread == high_priority_thread) low_priority_thread--;
    if(low_priority_thread == -1) low_priority_thread = 1;
    if(low_priority_thread == TOTAL_THREADS) low_priority_thread = 0;

    if(MAX_SEM_COUNT) ghSemaphore = CreateSemaphore(
        NULL, // default security attributes
        MAX_SEM_COUNT, // initial count
        MAX_SEM_COUNT, // maximum count
        NULL); // unnamed semaphore

    if(MAX_SEM_COUNT && ghSemaphore == NULL)
    {
        printf("CreateSemaphore error: %d\n", 1);
        return;
    }

    ifstream input(test_path);
    ld t0 = clock();
    input >> n >> m >> max_number;
    add = n + 2*m - 2;
    calc_points();
    A.resize(n);
    for(int i = 0; i < n; ++i) A[i].resize(m,0);

    read(A, input);
    //print(A);

    ll step1 = (n * m / TOTAL_THREADS);
    if(!step1) step1++;

    LARGE_INTEGER st, en, fq;
    QueryPerformanceFrequency(&fq);
    QueryPerformanceCounter(&st);

    ld t1 = clock();
    ll now1 = 0;
    ll now2 = 0;

    needThreads = min(ll(n * m), ll(TOTAL_THREADS));
    ll step2 = needThreads;

    hMutex = CreateMutex(NULL, FALSE, NULL);
    for(int i = 0; i < needThreads; ++i)

```

```

{
    diag[i].resize(2 * n + 2 * m - 2, {0, 0});
    if(i%64 == 0) WaitForMultipleObjects(64, stackTh, TRUE, INFINITE);

    //pdataArray[i] = (PMYDATA) HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
sizeof(MYDATA));

    //if(pdataArray[i] == NULL) ExitProcess(2);

    pdataArray[i].threadNum = i;
    pdataArray[i].type = type_of_task;

    if(type_of_task == 1)
    {
        pdataArray[i].startx = now1 / m;
        pdataArray[i].starty = now1 % m;
        pdataArray[i].step = step1;
        now1 += step1;
    }
    else
    {
        pdataArray[i].startx = now2 / m;
        pdataArray[i].starty = now2 % m;
        pdataArray[i].step = step2;
        now2 ++;
    }

    hThreadArray[i] = CreateThread(
        NULL,                // default security attributes
        0,                   // use default stack size
        MyThreadFunction,    // thread function name
        &pdataArray[i],      // argument to thread function
        0,                   // use default creation flags
        &dwThreadIdArray[i]); // returns the thread identifier

    stackTh[i%64] = hThreadArray[i];

    if(hThreadArray[i] == NULL )
    {
        printf("CreateThread error: %d\n", 1);
        return;
    }

    if(i == high_priority_thread)
        SetThreadPriority(hThreadArray[i],THREAD_PRIORITY_HIGHEST);

    if(i == low_priority_thread)
        SetThreadPriority(hThreadArray[i],THREAD_PRIORITY_LOWEST);

}

WaitForMultipleObjects(min(64,(int)needThreads), stackTh, TRUE, INFINITE);

for(int i = 0; i < needThreads; i++)
    if(hThreadArray[i] != INVALID_HANDLE_VALUE) CloseHandle(hThreadArray[i]);

CloseHandle(hMutex);
if(MAX_SEM_COUNT) CloseHandle(ghSemaphore);

```



```

for(int i = 1; i < needThreads; ++i)
{
    for(int j = 0; j < diag[0].size(); ++j)
    {
        diag[0][j].first += diag[i][j].first;
        diag[0][j].second += diag[i][j].second;
        //cout << i << " " << j << " " << diag[i][j].first << " " << diag[i][j].second << endl;
    }
}

ll ans = 0;
for(int i = 1 ; i < diag[0].size(); ++i)
{
    if(diag[0][i].first * diag[0][ans].second > diag[0][i].second * diag[0][ans].first) ans = i;
}
// for(int i = 0; i < diag[0].size(); ++i) cout << diag[0][i].first << " " << diag[0][i].second << endl;

ld result = (ld)diag[0][ans].first / (ld)diag[0][ans].second;

QueryPerformanceCounter(&en);
ld t2 = clock();

ld process_time = (en.QuadPart-st.QuadPart)*1000.0/fq.QuadPart;
long long inttime = round(process_time * 100);
long long dectime = inttime % 100;
inttime /= 100;
string result_path = "A:\\T\\3_term\\Operating_Systems\\5lab\\results_prob_right\\" +
to_string(TOTAL_THREADS);
result_path += "threads_" + to_string(type_of_task) + "typeOfTask_" + to_string(MAX_SEM_COUNT)+
"maxSemCount_" + to_string(inttime) + "." + to_string(dectime) + "ms_" + version + "_out.txt";
ofstream output(result_path);

//cout << process_time << " ms" << endl;

output << "Number of diagonal where average element is the biggest = " << ans << " \n";
output << fixed << setprecision(5) << "Average number = " << result << "\n";
output << "2 points of this diagonal\n";
output << points_for_diag[ans].first.first+1 << " " << points_for_diag[ans].first.second+1 << " " <<
points_for_diag[ans].second.first+1 << " " << points_for_diag[ans].second.second+1 << "\n";

ld d1 = t1 - t0;
ld d2 = t2 - t1;

HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
COORD pos = {0, 2};
SetConsoleCursorPosition(hConsole, pos);

cout << "\nVALUE1 is time consumed for reading VALUE2 is time consumed for calculating" << endl;
cout << fixed << setprecision(5) << d1/CLOCKS_PER_SEC << " s " << d2/CLOCKS_PER_SEC << " s"
<< endl;
cout << "TIME OF HIGH PRIORITY THREAD: " << threadTime[high_priority_thread] << " ms" << endl;
cout << "TIME OF LOW PRIORITY THREAD: " << threadTime[low_priority_thread] << " ms" << endl;
cout << "TIME OF ALL THREADS" << endl;
for(int i = 0; i < TOTAL_THREADS; ++ i)
{
    cout << threadTime[i] << " ";
}
int a;
cin >> a;
return;
}

```

```
int main(){
    runTask();
    return 0;
}
```

```
Command Prompt - 5labDLLv1.exe
100.000% Windows [Version 10.0.22621.608]
Thread 76 MaxAverage = 1391.000rights reserved.

VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
8.39900 s 0.25400 s
TIME OF HIGH PRIORITY THREAD: 21.04190 ms
TIME OF LOW PRIORITY THREAD: 16.43290 ms
TIME OF ALL THREADS
12.24900 17.89820 11.49780 16.43290 11.45080 11.66830 17.85840 21.04190 12.91580 17.11980 12.71390 22.81820 11.66100 19
.36330 22.56740 22.80070 23.42930 22.62010 12.49540 17.26170 21.98480 19.21180 22.68870 16.61450 23.01220 12.23690 22.4
9610 11.95500 21.60630 11.62430 14.72740 22.37160 22.55070 22.68550 14.03390 22.48170 12.33120 18.36660 11.67500 11.851
50 16.20750 12.22350 16.17460 11.96970 22.79100 11.53020 11.71890 23.02250 15.60130 22.47400 12.05160 17.02100 23.20560
13.58110 11.59080 19.13980 23.46250 12.47110 11.82310 12.34290 18.21870 11.88650 12.94770 13.43860 18.80760 13.68350 1
4.84140 16.87570 15.46650 18.66960 13.78730 20.08000 18.00560 16.13510 22.14700 20.74970 12.07580 11.60970 18.81480 11.
91320 11.97840 13.64750 19.49440 12.11960 19.54150 22.11890 21.92160 14.67610 22.03150 12.73390 22.20780 17.93150 11.93
330 22.00600 21.50260 14.42880 12.20930 15.53910 11.71870 11.79240

A:\T\3_term\Operating_Systems\5labDLLv1\bin\Debug> 5labDLLv1.exe
```

```
main.cpp X lab5dllheaderv1.h X main.cpp X
1 #include <bits/stdc++.h>
2 #include <windows.h>
3 #include <tchar.h>
4 #include "lab5dllheaderv1.h"
5 using namespace std;
6 int main()
7 {
8     /*HMODULE h = LoadLibrary
9     if(!h){
10         cout << "Could not op
11         return 1;
12     }
13     FARPROC fn = GetProcAddress
14     if(!fn){
15         cout << "Could not lo
16         return 2;
17     }
18     fn();*/
19     runTask();
20     return 0;
21 }
22

A:\T\3_term\Operating_Systems\5labDLLv1\bin\Debug\5labDLLv1.exe
100.000%
Thread 90 MaxAverage = 1391.000

VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
8.34400 s 0.28300 s
TIME OF HIGH PRIORITY THREAD: 11.86500 ms
TIME OF LOW PRIORITY THREAD: 22.12700 ms
TIME OF ALL THREADS
19.97590 19.33940 21.98900 22.12700 11.52590 11.56480 11.77130 11.86500 20.39030 22.67030 11.57920 18.18960 11.86540 12.
79210 20.27540 11.75700 22.36090 11.49580 16.18160 15.69760 14.74090 22.05940 22.78830 23.15810 22.53560 18.96910 22.301
80 22.55700 11.84830 22.94330 12.08950 22.98630 19.60140 18.91240 11.87060 15.00230 11.78350 23.29470 22.60890 22.68760
23.21950 11.27770 21.08290 15.67820 21.10460 16.17330 12.57190 22.95630 20.83060 11.96560 14.88600 23.38180 19.48440 14.
26330 14.17020 11.83330 11.60490 22.04170 23.34110 23.34580 22.97980 13.59020 11.63460 13.47130 20.18550 19.02850 16.734
90 16.34000 18.41680 16.59530 16.49120 19.83740 19.84490 17.50450 19.38500 15.11510 15.34220 22.82590 23.07780 22.46570
14.89440 12.08560 14.96250 21.98290 11.67130 23.28460 22.65140 22.54250 11.34110 11.66110 22.24410 19.46410 23.50890 22.
27800 22.16560 12.46600 14.29300 22.17260 12.03780 22.24310
```

```
main.cpp X lab5dllheaderv1.h X main.cpp X
1 #include <bits/stdc++.h>
2 #include <windows.h>
3 #include <tchar.h>
4 #include "lab5dllheaderv1.h"
5 using namespace std;
6 int main()
7 {
8     HMODULE h = LoadLibrary("A:\\T\\
9     if(!h){
10         cout << "Could not open the
11         return 1;
12     }
13     FARPROC fn = GetProcAddress(h,
14     if(!fn){
15         cout << "Could not locate t
16         return 2;
17     }
18     fn();
19     //runTask();
20     return 0;
21 }
22

A:\T\3_term\Operating_Systems\5labDLLv1\bin\Debug\5labDLLv1.exe
100.000%
Thread 82 MaxAverage = 1391.000

VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
8.57300 s 0.27500 s
TIME OF HIGH PRIORITY THREAD: 21.70300 ms
TIME OF LOW PRIORITY THREAD: 16.60220 ms
TIME OF ALL THREADS
19.89970 16.24880 15.61550 16.60220 12.22850 18.32120 13.39550 21.70300 20.01010 18.05650 11.50600 19.24960 11.19470 11.
30980 11.87440 22.62730 12.16920 19.28760 14.60900 12.68880 11.44450 11.67430 11.64300 22.13060 22.39480 14.32650 23.307
10 18.93270 21.50810 16.89840 11.67950 19.41010 13.20990 12.88400 20.94910 19.92730 11.68330 15.90760 22.35650 22.16430
14.84620 18.55590 20.38200 22.69960 13.45990 12.41750 22.26850 13.22460 11.86290 14.83980 12.48590 22.62300 14.28700 22.
51650 14.82450 22.30390 11.62410 12.27500 12.92910 22.48380 11.52040 11.50730 15.36850 22.48260 19.51550 12.38700 19.610
00 14.15750 19.57610 14.17420 18.62230 18.45100 15.63510 22.36440 22.10240 22.24950 11.65620 22.87530 22.23210 22.09240
19.73760 11.85700 20.97000 21.98830 22.18280 19.29950 14.33130 22.42020 23.25690 11.97770 11.74570 22.13080 13.24360 22.
01940 22.20000 17.20090 23.19190 19.71130 14.39200 13.92830
```

Висновок:

Я закріпив вміння та навички роботи з динамічно-зв'язувальними бібліотеками (Dynamic-link library) в ОС Windows. Навчитися реалізовувати динамічно-зв'язувальні бібліотеки. Створив свою динамічну бібліотеку, запустив головну функцію через консоль, та створив програму, у якій підключив новостворену бібліотеку статично, а потім динамічно.