НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



**ЗВІТ**

**Про виконання лабораторної роботи № 9**

**з дисципліни «Операційні системи»**

**Лектор:**

Ст. викладач Грицай О.Д.

**Виконав:**

студ. групи ПЗ-23

Михалевич П.-І.В.

**Прийняла:**

Ст. викладач Грицай О.Д.

«\_\_\_» \_\_\_\_\_\_ 2021 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2021

**Тема.** Статичні та динамічні бібліотеки. WINDOWS та LINUX

**Мета.** Ознайомитися з статичними та динамічними бібліотеками в операційних системах WINDOWS та LINUX. Навчитися реалізовувати статичні та динамічні бібліотеки.

**Теоретичні відомості**

DLL (англ. Dynamic-link library — динамічно приєднувана бібліотека) — реалізовані компанією Microsoft загальні бібліотеки в ОС Windows та OS/2. Як правило бібліотеки мають розширення файлу \*.dll, \*.ocx (для бібліотек, що містять елементи керування ActiveX) або \*.drv (драйвери старих версій ОС). DLL може містити код, дані та ресурси в будь-якій комбінації. Існує два способи зв’язати dll з кодом с++: статичний та динамічний. Для статичного зв’язування потрібно створити додатковий заготовочний файл, в якому кожну функцію потрібно експортувати з dll, а у заголовочному файлі самої програми потрібно імпортувати ці функції. Для того, щоб їх використовувати також потрібно підключити \*.lib файл у проект де використовується dll. В динамічному зв’язуванні потрібно загрузити dll за допомогою фції LoadLibrary. Далі потрібно перевизначити тип для кожної функції. І присвоїти мінним типу, який ми визначили вказівник на ф-ції, який отримуємо за допомогою GetProcAddress. Практично неможливо створити додаток Windows, в якому невикористовувалися б бібліотеки DLL. У DLL містяться всі функції Win32 API і незліченну кількість інших функцій операційних систем Win32. Взагалі кажучи, DLL – це просто набори функцій, зібрані в бібліотеки. Однак, на відміну від своїх статичних родичів (файлів. Lib), бібліотеки DLL не приєднані безпосередньо до виконуваних файлів за допомогою редактора зв'язків. У виконуваний файл занесена тільки інформація про їхнє місце знаходження. У момент виконання програми завантажується вся бібліотека цілком. Завдяки цьому різні процеси можуть користуватися спільно одними і тими ж бібліотеками, знаходяться в пам'яті. Такий підхід дозволяє скоротити обсяг пам'яті, необхідний для декількох додатків, що використовують багато спільних бібліотек, а також контролювати розміри ЕХЕ-файлів. Однак, якщо бібліотека використовується тільки одним додатком, краще зробити її звичайною, статичною. Звичайно, якщо що входять до її складу функції будуть використовуватися тільки в одній програмі, можна просто вставити в неї відповідний файл з вихідним текстом. Найчастіше проект підключається до DLL статично, або неявно, на етапі компонування. Завантаженням DLL при виконанні програми управляє операційна система. Однак, DLL можна завантажити і явно, або динамічно, в ході роботи додатки.

**Хід роботи**

**Завдання**

1. Реалізувати лабораторну роботу №6 (згідно варіанту) у вигляді статичної та динамічної бібліотеки в ОС WINDOWS.

2. Запустити створену динамічну бібліотеку з командної стрічки (cmd.exe) за допомогою rundll32.exe.

3. Створити окрему програму і реалізувати статичний зв'язок між програмою та бібліотекою із п. 1.

4. Створити окрему програму і реалізувати динамічний зв'язок між програмою та бібліотекою із п. 1.

5. Експортувати головну функцію бібліотеки під іншим іменем із п. 1.

6. Реалізувати лабораторну роботу №8 у вигляді статичної та динамічної (поділюваної) бібліотеки в ОС LINUX.

7. Створити окрему програму і реалізувати статичний зв'язок між програмою та бібліотекою із п. 6

8. Створити окрему програму і реалізувати динамічний зв'язок між програмою та бібліотекою із п. 6.

9. Порівняти результати виконання програми та роботу бібліотек під ОС Windows та Linux.

10. Результати виконання роботи відобразити у звіті.

**Windows**

**Код програми**

**StaticLib:**

**Functions.h**

#pragma once

void ArraySum(int\* param);

void StaticLibFunc();

**Functions.cpp**

// StaticLib1.cpp : Defines the functions for the static library.

//

#include "pch.h"

#include "framework.h"

#include "StaticLib1.h"

#include <Windows.h>

#include <vector>

#include <iostream>

#include <mutex>

#include <cmath>

// TODO: This is an example of a library function

//void fnStaticLib1()

//{

//}

int sum = 0;

std::vector<int> array(10000, 0);

std::mutex mu;

std::vector<HANDLE> myhandle(4, 0);

std::vector<DWORD> mythreadid(4, 0);

void ArraySum(int\* param) {

mu.lock();

for (int i = param[0]; i < param[1]; i++) {

if (i == 0) {

array[i] = 2;

sum += array[i];

continue;

}

array[i] = array[i - 1] \* i + exp(i);

//std::cout << "In current thread (" << GetCurrentThreadId() << "): array["<<i<<"] = " << array[i] << std::endl;

sum+=array[i];

//sum += 1;

//std::cout << myindex++ << " / " << std::endl;

}

std::cout << "In current thread (" << GetCurrentThreadId() << "): sum =" << sum << std::endl;

mu.unlock();

}

void StaticLibFunc() {

sum = 0;

for (int i = 0; i < 4; i++) {

int\* param = new int[2];

param[0] = (i) \* (10000 / 4);

param[1] = (i + 1) \* (10000 / 4);

myhandle[i] = CreateThread(0, 0, (LPTHREAD\_START\_ROUTINE)&ArraySum, param, 0, &mythreadid[i]);

WaitForSingleObject(myhandle[i], INFINITE);

CloseHandle(myhandle[i]);

}

}

**Main.cpp**

#include "StaticLib1.h"

#include "Functions.h"

#include <iostream>

int main() {

std::cout << "Static lib:" << std::endl;

StaticLibFunc();

std::cout << "Dynamic lib:" << std::endl;

DynamicLibFunc();

return 0;

}

**DynamicLib:**

**Functions.h**

#pragma once

extern "C" \_\_declspec (dllexport) void DynamicLibFunc();

extern "C" \_\_declspec (dllexport) void DynArraySum(int\* param);

**Functions.cpp**

#include "pch.h"

#include "Functions.h"

#include <Windows.h>

#include <vector>

#include <iostream>

#include <mutex>

#include <cmath>

int sum = 0;

std::vector<int> array(10000, 0);

std::mutex mu;

std::vector<HANDLE> myhandle(4, 0);

std::vector<DWORD> mythreadid(4, 0);

void DynArraySum(int\* param) {

mu.lock();

for (size\_t i = param[0]; i < param[1]; i++) {

if (i == 0) {

array[i] = 2;

sum += array[i];

continue;

}

array[i] = array[i - 1] \* i + exp(i);

sum += array[i];

//sum += 1;

//std::cout << myindex++ << " / " << std::endl;

}

std::cout << "In current thread (" << GetCurrentThreadId() << "): sum =" << sum << std::endl;

mu.unlock();

}

void DynamicLibFunc() {

for (int i = 0; i < 4; i++) {

int\* param = new int[2];

param[0] = (i) \* (10000 / 4);

param[1] = (i + 1) \* (10000 / 4);

myhandle[i] = CreateThread(0, 0, (LPTHREAD\_START\_ROUTINE)&DynArraySum, param, 0, &mythreadid[i]);

}

for (int i = 0; i < 4; i++) {

WaitForSingleObject(myhandle[i], INFINITE);

CloseHandle(myhandle[i]);

}

}

**Main\_1.cpp**

#include "StaticLib1.h"

#include "Functions.h"

#include <iostream>

int main() {

std::cout << "Static lib:" << std::endl;

StaticLibFunc();

std::cout << "Dynamic lib:" << std::endl;

DynamicLibFunc();

return 0;

}

**Main\_2.cpp**

#include <iostream>

#include <windows.h>

HINSTANCE h;

int main() {

void (\*DllFunc)();

h = LoadLibrary(L"OS\_dll\_1.dll");

if (!h) {

std::cout << "Can't find dll.";

return 1;

}

DllFunc = (void(\*)())GetProcAddress(h, "DynamicLibFunc");

if (!DllFunc) {

return 2;

}

std::cout << "Dynamic lib:" << std::endl;

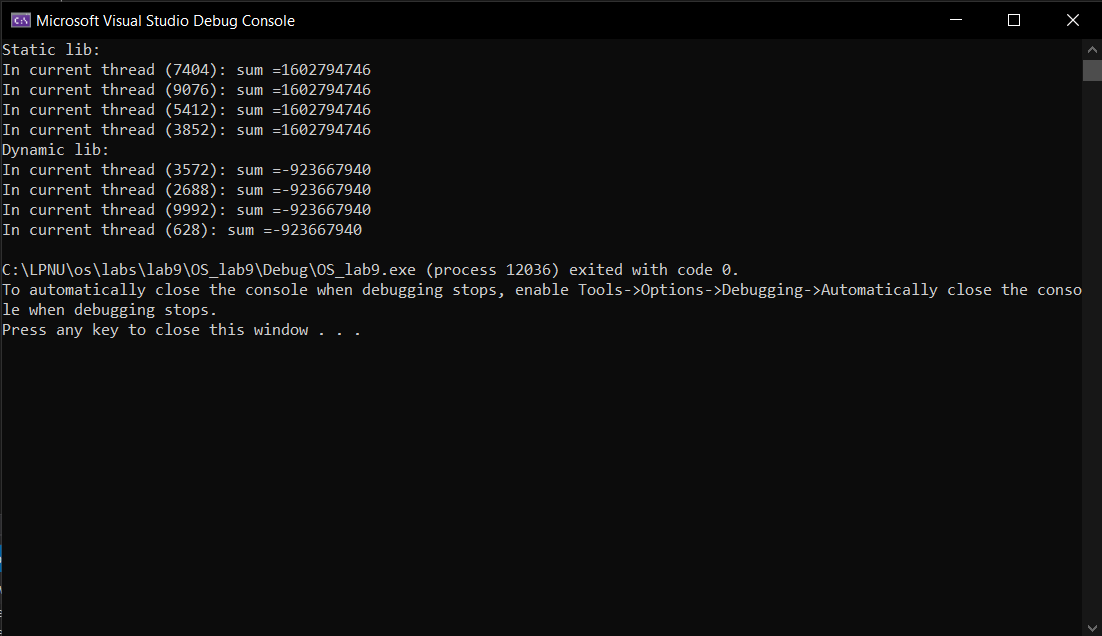
DllFunc();

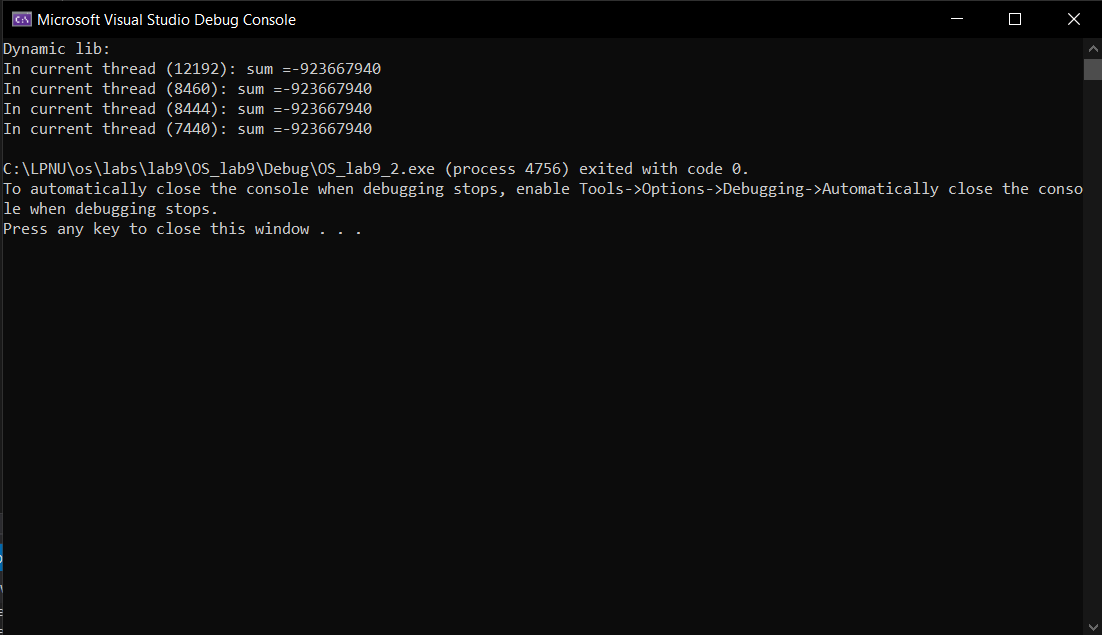
FreeLibrary(h);

return 0;

}

**Результат програми:**



****

**Linux**

**Код програми:**

**Staticlib:**

**Functions.cpp:**

#include <algorithm>

#include <pthread.h>

#include <math.h>

#include <vector>

#include <iostream>

#include <unistd.h>

pthread\_mutex\_t mu;

int sum = 0;

int myindex = 0;

std::vector<int> array(10000,0);

struct MyParam{

int first;

int second;

};

extern "C" void \*MutexArraySum(void \* data)

{

auto param = \*(MyParam\*)(data);

int sum = param.first;

for(int i = param.first; i< param.second; i++){

if(i == 0){

array[i] = 2;

sum+=array[i];

continue;

}

pthread\_mutex\_lock(&mu);

//array[i] = array[i-1]\*i + exp(i);

//sum+=array[i];

sum+=1;

//std::cout<<++myindex<< " / " << std::endl;

pthread\_mutex\_unlock(&mu);

}

std::cout << "In current thread (" << gettid() << "): sum =" << sum << std::endl;

return nullptr;

}

**Program:**

**Main.cpp:**

#include <iostream>

#include <algorithm>

#include <pthread.h>

#include <math.h>

#include <vector>

#include <unistd.h>

extern "C" void\* MutexArraySum(void \*);

struct MyParam{

int first;

int second;

};

int main(){

std::vector<pthread\_t> pthread(4,0);

for(int i = 0; i<4; i++){

MyParam\* param = new MyParam;

param->first = (i)\*(2500);

param->second = (i+1)\*(2500);

pthread\_create(&pthread[i], nullptr, MutexArraySum, (void\*)param);

}

for(int i = 0; i<4 ; i++){

pthread\_join(pthread[i], nullptr);

}

//std::cout << "Static lib:" << std::endl;

//StaticFunc();

return 0;

}

**Dynamiclib:**

**Functions:**

#include <algorithm>

#include <pthread.h>

#include <math.h>

#include <vector>

#include <iostream>

#include <unistd.h>

pthread\_mutex\_t mu;

int sum = 0;

int myindex = 0;

std::vector<int> array(10000,0);

struct MyParam{

int first;

int second;

};

extern "C" void \*DynMutexArraySum(void \* data)

{

auto param = \*(MyParam\*)(data);

int sum = param.first;

for(int i = param.first; i< param.second; i++){

if(i == 0){

array[i] = 2;

sum+=array[i];

continue;

}

pthread\_mutex\_lock(&mu);

//array[i] = array[i-1]\*i + exp(i);

//sum+=array[i];

sum+=1;

//std::cout<<++myindex<< " / " << std::endl;

pthread\_mutex\_unlock(&mu);

}

std::cout << "In current thread (" << gettid() << "): sum =" << sum << std::endl;

return nullptr;

}

**Main\_1:**

#include <iostream>

#include <algorithm>

#include <pthread.h>

#include <math.h>

#include <vector>

#include <unistd.h>

//extern void StaticFunc();

extern "C" void\* DynMutexArraySum(void \*);

struct MyParam{

int first;

int second;

};

int main(){

std::vector<pthread\_t> pthread(4,0);

for(int i = 0; i<4; i++){

MyParam\* param = new MyParam;

param->first = (i)\*(2500);

param->second = (i+1)\*(2500);

pthread\_create(&pthread[i], nullptr, DynMutexArraySum, (void\*)param);

}

for(int i = 0; i<4 ; i++){

pthread\_join(pthread[i], nullptr);

}

//std::cout << "Static lib:" << std::endl;

//StaticFunc();

return 0;

}

**Main\_2:**

#include <iostream>

#include <algorithm>

#include <pthread.h>

#include <math.h>

#include <vector>

#include <unistd.h>

#include <dlfcn.h>

//extern "C" void\* DynMutexArraySum(void \*);

void\* lib\_handler;

void\* (\*DynMutexArraySum)(void \* data);

struct MyParam{

int first;

int second;

};

int main(int argc, char\* argv[]){

lib\_handler = dlopen("/home/pavlo/OS\_lab\_9\_dll/libdynamic.so",RTLD\_LAZY);

if(!lib\_handler){

std::cout << dlerror() << std::endl;

exit(1);

}

DynMutexArraySum = (void\* (\*)(void \* data))dlsym(lib\_handler,argv[1]);

std::vector<pthread\_t> pthread(4,0);

for(int i = 0; i<4; i++){

MyParam\* param = new MyParam;

param->first = (i)\*(2500);

param->second = (i+1)\*(2500);

pthread\_create(&pthread[i], nullptr, DynMutexArraySum, (void\*)param);

}

for(int i = 0; i<4 ; i++){

pthread\_join(pthread[i], nullptr);

}

//std::cout << "Static lib:" << std::endl;

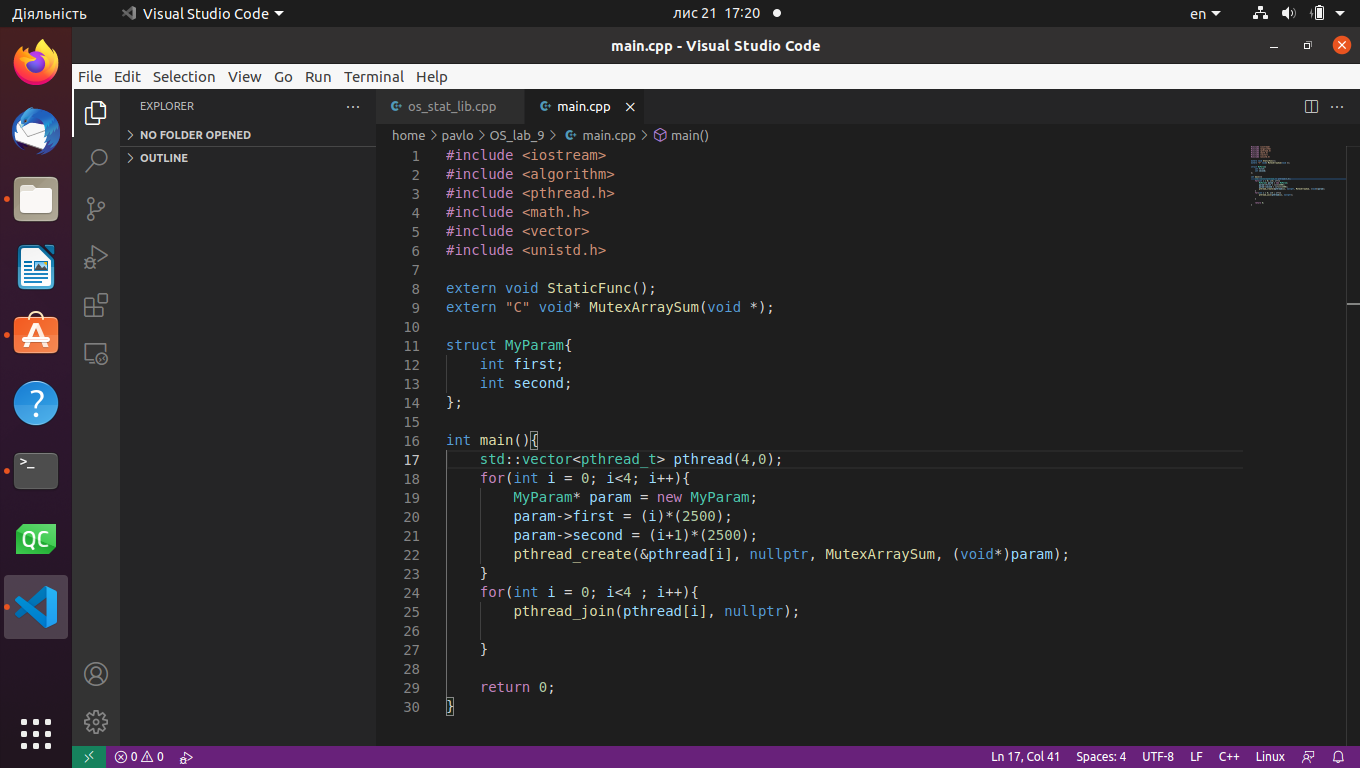
//StaticFunc();

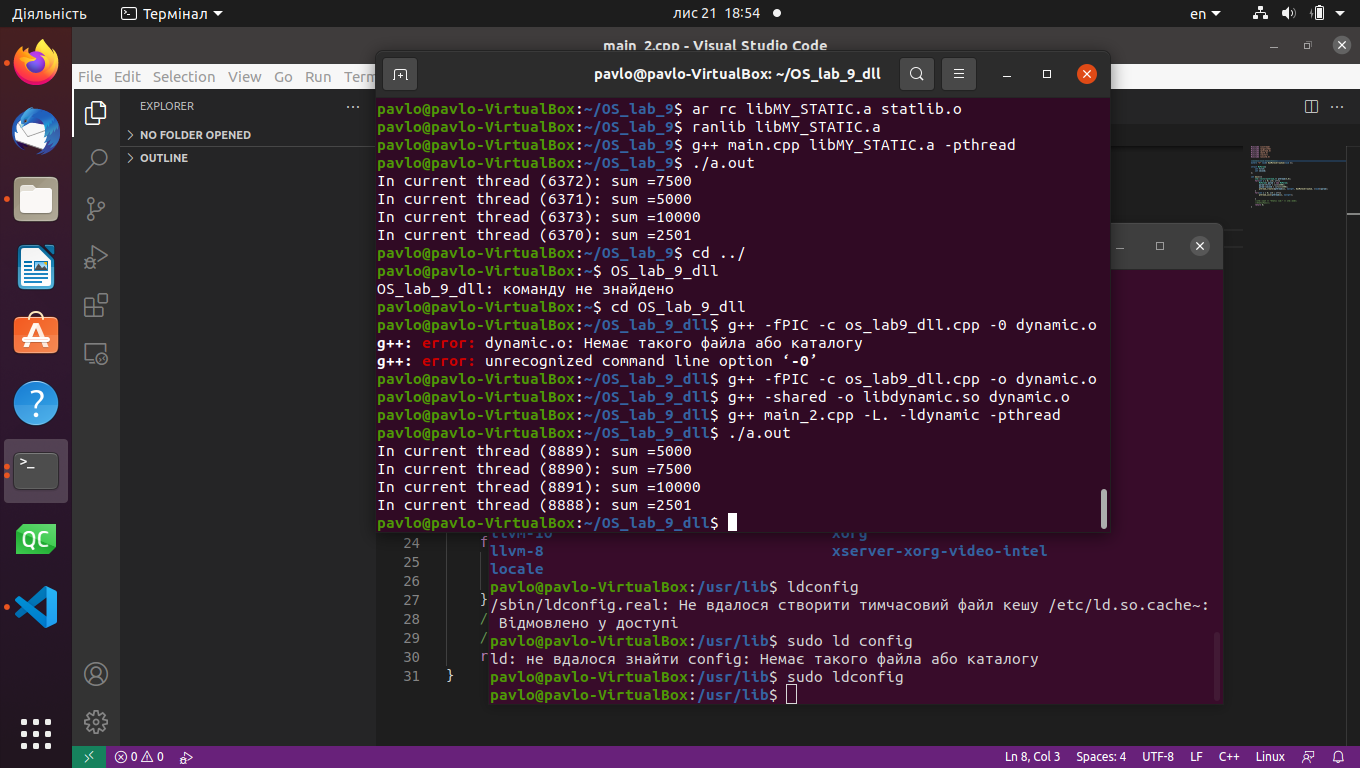
dlclose(lib\_handler);

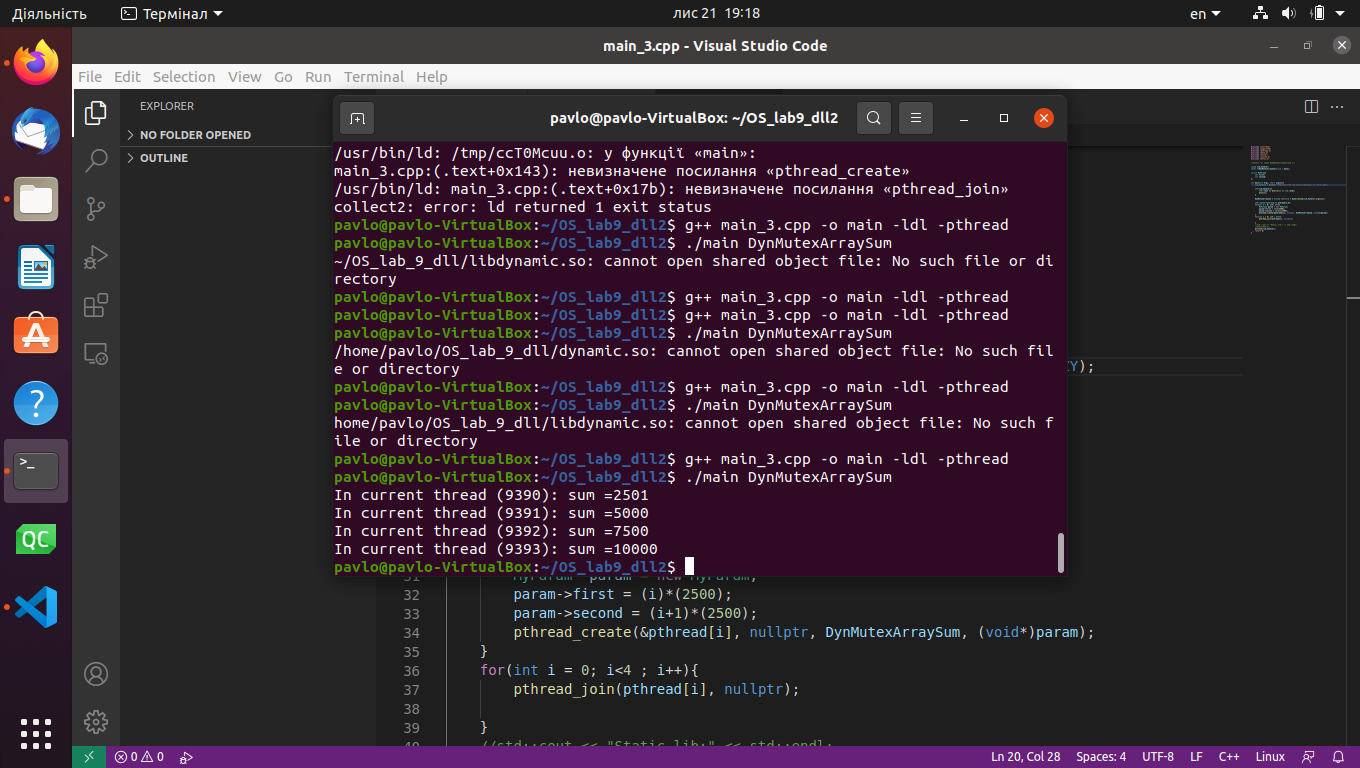
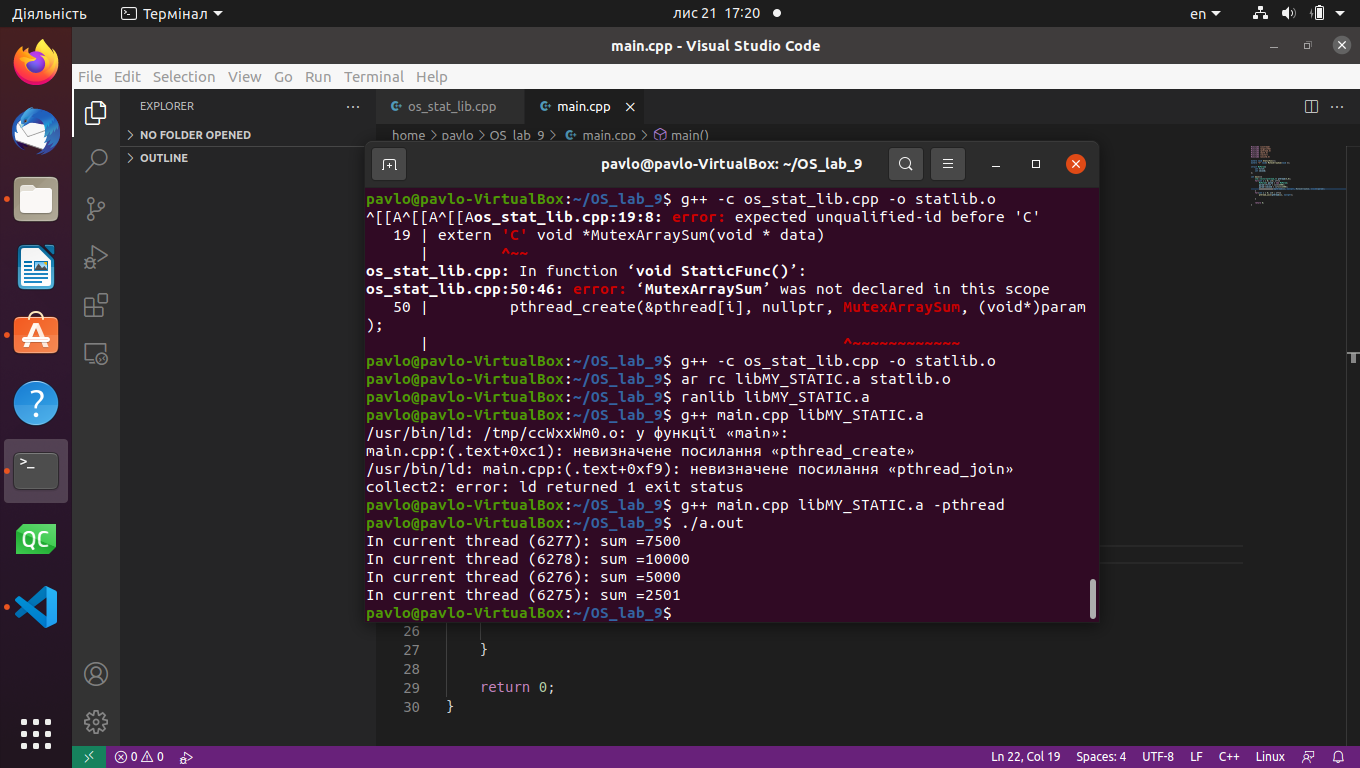
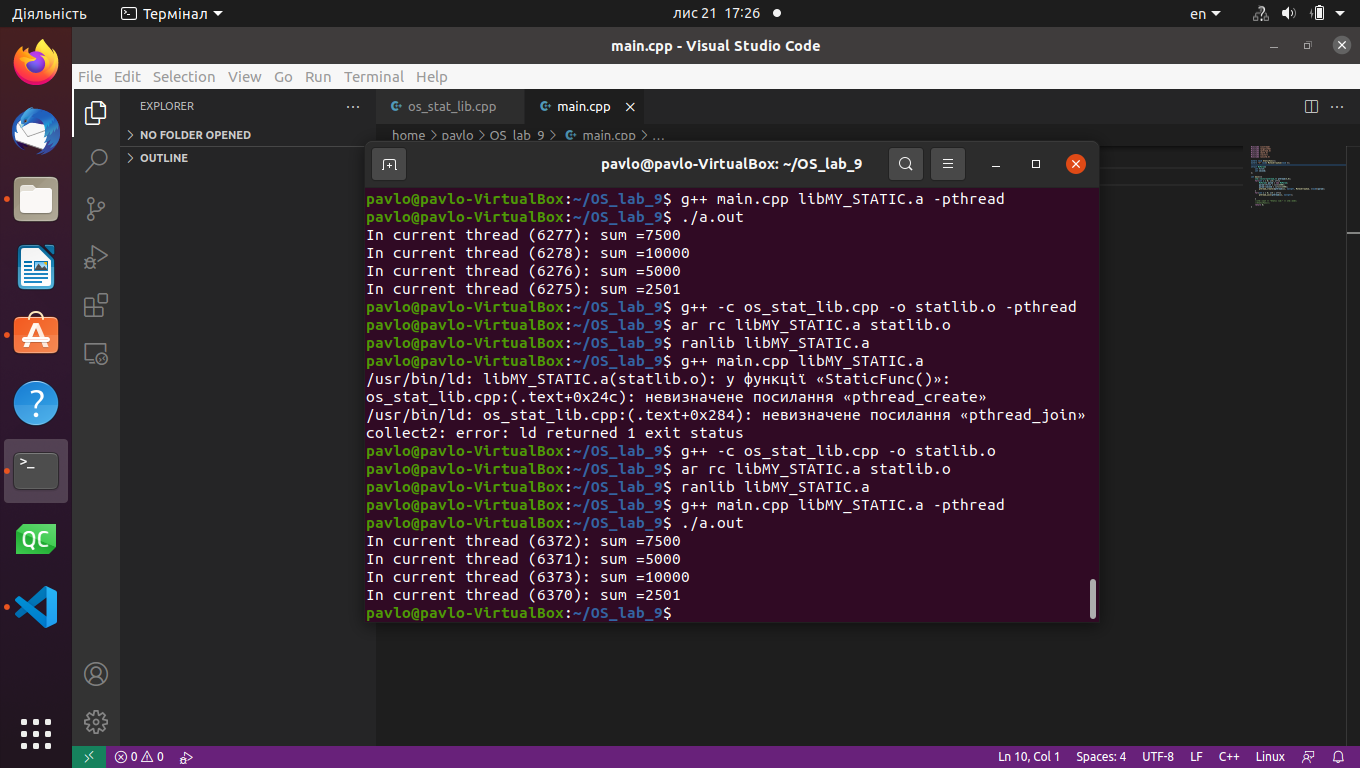
return 0;

}

**Результат програми:**

****

****

****

##### Висновок

На даній лабораторній роботі, я ознайомився з статичними та динамічними бібліотеками в операційних системах WINDOWS та LINUX та реалізував статичні та динамічні бібліотеки.