НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



**ЗВІТ**

**Про виконання лабораторної роботи № 3**

**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.** Створення та керування процесами  засобами API в операційній системі Windows.

**Мета.** Ознайомитися з багатопоточністю в ОС Windows. Навчитися працювати з процесами, використовуючи WinAPI-функції.

**Теоретичні відомості**

Процес - визначають як екземпляр виконуваної програми. Кожному процесу при завантаженні виділяються ресурси операційної системи. Складові елементи процесу:

-об'єкт ядра, через який операційна система керує процесом і де зберігається статична інформація про процес

- адресний простір де зберігається код програми та дані.

При створенні процесу система створює таблицю дескрипторів, які використовуються для об’єктів ядра. Фактично це масив структур даних. Кожна структура містить вказівник на якийсь об'єкт ядра, маску доступу і деякі прапорці. Коли процес ініціалізується перший раз, таблиця пуста. Коли викликається функція що створює об'єкт ядра , ядро виділяє для цього об'єкта блок пам'яті і ініціалізує його.

Адресний простір процесу складається з набору адрес віртуальної пам'яті, які він може використати. Адресний простір процесу недоступний іншим процесам. Процес володіє системними ресурсами, такими як файли, мережеві з'єднання, пристрої введення-виведення, об'єкти синхронізації тощо. Процес містить деяку стартову інформацію для потоків, які в ньому створюватимуться. Процес має містити хоча б один потік, який система скеровує на виконання

**Постановка завдання**

1. Створити окремий процес, і здійснити в ньому розв’язок задачі згідно варіанту у відповідності до порядкового номера у журнальному списку (підгрупи).

10. Вивести для кожного стовпця кількість елементів у стовпці, що дорівнюють першому елементу стовпця в заданій матриці N\*M (N>1000 , M>10000 задається користувачем, матриця визначається випадково)

1. Реалізувати розв’язок задачі у 2-ох, 4-ох, 8-ох процесах. Виміряти час роботи процесів за допомогою функцій WinAPI. Порівняти результати роботи в одному і в багатьох процесах.
2. Для кожного процесу реалізувати можливість його запуску, зупинення, завершення та примусове завершення («вбиття»).
3. Реалізувати можливість зміни пріоритету виконання процесу.

**Хід роботи**

Код програми:

Mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "QMessageBox"

double **GetCpuTime**(PROCESS\_INFORMATION &pi){

FILETIME createTime;

FILETIME exitTime;

FILETIME kernelTime;

FILETIME userTime;

*if* ( GetProcessTimes( pi.hProcess,

&createTime, &exitTime, &kernelTime, &userTime ) != -1 )

{

SYSTEMTIME userSystemTime;

*if* ( FileTimeToSystemTime( &userTime, &userSystemTime ) != -1 )

*return* (double)userSystemTime.wHour \* 3600.0 +

(double)userSystemTime.wMinute \* 60.0 +

(double)userSystemTime.wSecond +

(double)userSystemTime.wMilliseconds / 1000.0;

*else* *return* -2;

} *else* *return* -1;

}

STARTUPINFO si[8];

PROCESS\_INFORMATION pi[8];

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

connect(ui->pushButton\_start,SIGNAL(clicked()),*this*,SLOT(MySlot()));

connect(ui->pushButton\_suspend,SIGNAL(clicked()),*this*,SLOT(MySlot()));

connect(ui->pushButton\_resume,SIGNAL(clicked()),*this*,SLOT(MySlot()));

connect(ui->pushButton\_kill,SIGNAL(clicked()),*this*,SLOT(MySlot()));

connect(ui->pushButton\_killall,SIGNAL(clicked()),*this*,SLOT(MySlot()));

*for*(int i = 0; i < 8; i++){

CloseHandle(pi[i].hThread);

CloseHandle(pi[i].hProcess);

}

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

void MainWindow::**MySlot**()

{

QPushButton\* btn = (QPushButton\*) sender();

*if*(btn->text() == "Start"){

int ProcessCount = ui->comboBox\_process\_count->currentText().toInt();

int rowCount = ui->spinBox\_row\_count->value();

int columnCount = ui->spinBox\_column\_count->value();

*for*(int i = 0; i < ui->tableWidget->rowCount(); i++)

{

*for*(int j = 0; j < ui->tableWidget->columnCount(); j++)

{

QTableWidgetItem\* item = *new* QTableWidgetItem;

item->setText("");

item->setTextAlignment(Qt::*AlignCenter*);

ui->tableWidget->setItem(i,j,*item*);

}

}

std::wstring cmd = L"C:\\politeh 2 kurs\\os\\labs\\lab3\\OS\_lab\_3\\x64\\Debug\\OS\_lab\_3 "

+ std::to\_wstring(rowCount/ ProcessCount) + L" " + std::to\_wstring(columnCount/ ProcessCount);

*for*(int i = 0; i < ProcessCount; i++){

ZeroMemory(&si[i], *sizeof*(si[i]));

si[i].cb = *sizeof*(si[i]);

ZeroMemory(&pi[i], *sizeof*(pi[i]));

CreateProcess(NULL,&(cmd[0]),NULL,NULL,*true*, REALTIME\_PRIORITY\_CLASS |

CREATE\_NEW\_CONSOLE,NULL,NULL,&si[i],&pi[i]);

ui->tableWidget->item(i,0)->setText(QString::number(pi[i].dwProcessId));

ui->tableWidget->item(i,1)->setText("Suspended");

ui->tableWidget->item(i,2)->setText("High");

ui->tableWidget->item(i,3)->setText("0");

}

Sleep(1500);

*for*(int i = 0; i < ProcessCount; i++)

{

SuspendThread(pi[i].hThread);

ui->tableWidget->item(i,3)->setText(QString::number(GetCpuTime(*pi[i]*)));

}

}*else* *if*(btn->text() == "Suspend"){

DWORD dwCode;

GetExitCodeProcess(pi[ui->tableWidget->currentRow()].hProcess,&dwCode);

*if*(ui->tableWidget->item(ui->tableWidget->currentRow(),1)->text() != "Suspended" &&

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->text() != "Killed" &&

dwCode == STATUS\_PENDING){

SuspendThread(pi[ui->tableWidget->currentRow()].hThread);

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->setText("Suspended");

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

}*else*{

*if*(dwCode != STATUS\_PENDING){

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->setText("Killed");

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

}

QMessageBox::information(*this*,"Info","You can't suspend the process!");

}

}*else* *if*(btn->text() == "Resume"){

DWORD dwCode;

GetExitCodeProcess(pi[ui->tableWidget->currentRow()].hProcess,&dwCode);

*if*(ui->tableWidget->item(ui->tableWidget->currentRow(),1)->text() != "Running" &&

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->text() != "Killed" &&

dwCode == STATUS\_PENDING){

ResumeThread(pi[ui->tableWidget->currentRow()].hThread);

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->setText("Running");

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

}*else*{

*if*(dwCode != STATUS\_PENDING){

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->setText("Killed");

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

}

QMessageBox::information(*this*,"Info","You can't resume the process!");

}

}*else* *if*(btn->text() == "Kill"){

DWORD dwCode;

GetExitCodeProcess(pi[ui->tableWidget->currentRow()].hProcess,&dwCode);

*if*(ui->tableWidget->item(ui->tableWidget->currentRow(),1)->text() != "Killed" &&

dwCode == STATUS\_PENDING){

TerminateThread(pi[ui->tableWidget->currentRow()].hThread, 0);

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->setText("Killed");

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

}*else*{

*if*(dwCode != STATUS\_PENDING){

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->setText("Killed");

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

}

QMessageBox::information(*this*,"Info","The process has been killed yet!");

}

}*else* *if*(btn->text() == "Kill all"){

*for*(int i = 0; i < ui->tableWidget->rowCount(); i++){

DWORD dwCode;

GetExitCodeProcess(pi[i].hProcess,&dwCode);

*if*(ui->tableWidget->item(i,1)->text() != "Killed" &&

dwCode == STATUS\_PENDING){

TerminateThread(pi[i].hThread, 0);

ui->tableWidget->item(i,1)->setText("Killed");

ui->tableWidget->item(i,3)->setText(QString::number(GetCpuTime(*pi[i]*)));

}*else*{

*if*(dwCode != STATUS\_PENDING){

ui->tableWidget->item(i,1)->setText("Killed");

ui->tableWidget->item(i,3)->setText(QString::number(GetCpuTime(*pi[i]*)));

}

}

}

}

}

void MainWindow::**on\_comboBox\_process\_count\_currentTextChanged**(*const* QString &arg1)

{

ui->tableWidget->setRowCount(arg1.toInt());

}

void MainWindow::**on\_comboBox\_priority\_currentTextChanged**(*const* QString &arg1)

{

*switch*(ui->comboBox\_priority->currentIndex()){

*case* 0:

SetPriorityClass(pi[ui->tableWidget->currentRow()].hProcess,REALTIME\_PRIORITY\_CLASS);

ui->tableWidget->item(ui->tableWidget->currentRow(),2)->setText(ui->comboBox\_priority->currentText());

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

*break*;

*case* 1:

SetPriorityClass(pi[ui->tableWidget->currentRow()].hProcess,HIGH\_PRIORITY\_CLASS);

ui->tableWidget->item(ui->tableWidget->currentRow(),2)->setText(ui->comboBox\_priority->currentText());

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

*break*;

*case* 2:

SetPriorityClass(pi[ui->tableWidget->currentRow()].hProcess,ABOVE\_NORMAL\_PRIORITY\_CLASS);

ui->tableWidget->item(ui->tableWidget->currentRow(),2)->setText(ui->comboBox\_priority->currentText());

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

*break*;

*case* 3:

SetPriorityClass(pi[ui->tableWidget->currentRow()].hProcess,NORMAL\_PRIORITY\_CLASS);

ui->tableWidget->item(ui->tableWidget->currentRow(),2)->setText(ui->comboBox\_priority->currentText());

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

*break*;

*case* 4:

SetPriorityClass(pi[ui->tableWidget->currentRow()].hProcess,BELOW\_NORMAL\_PRIORITY\_CLASS);

ui->tableWidget->item(ui->tableWidget->currentRow(),2)->setText(ui->comboBox\_priority->currentText());

ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

*break*;

*case* 5:

SetPriorityClass(pi[ui->tableWidget->currentRow()].hProcess,IDLE\_PRIORITY\_CLASS);

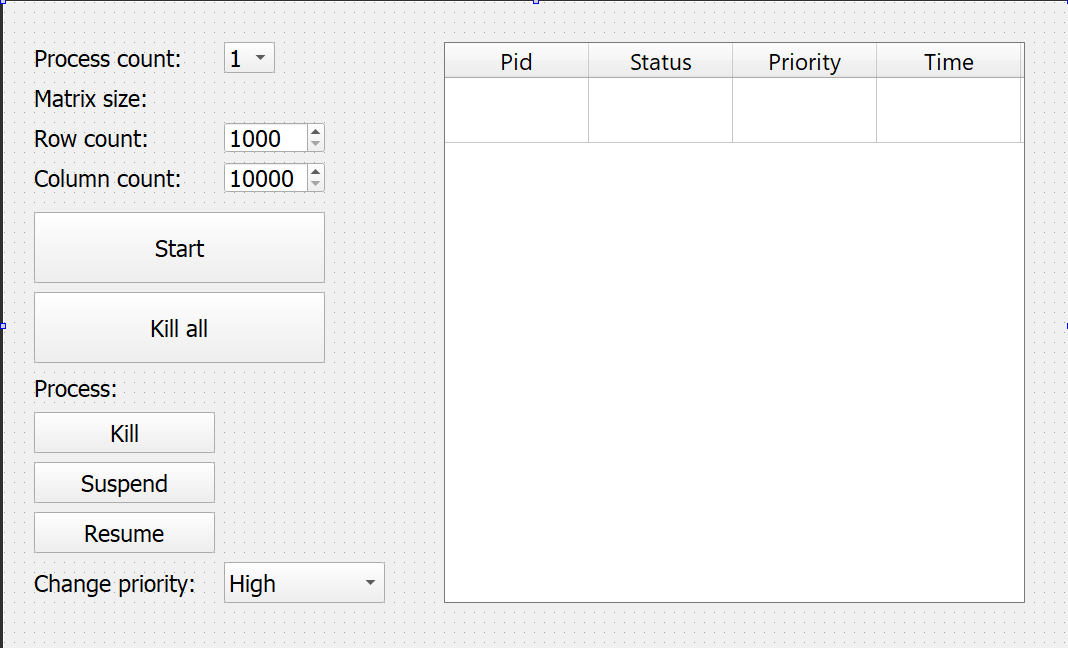
ui->tableWidget->item(ui->tableWidget->currentRow(),2)->setText(ui->comboBox\_priority->currentText());

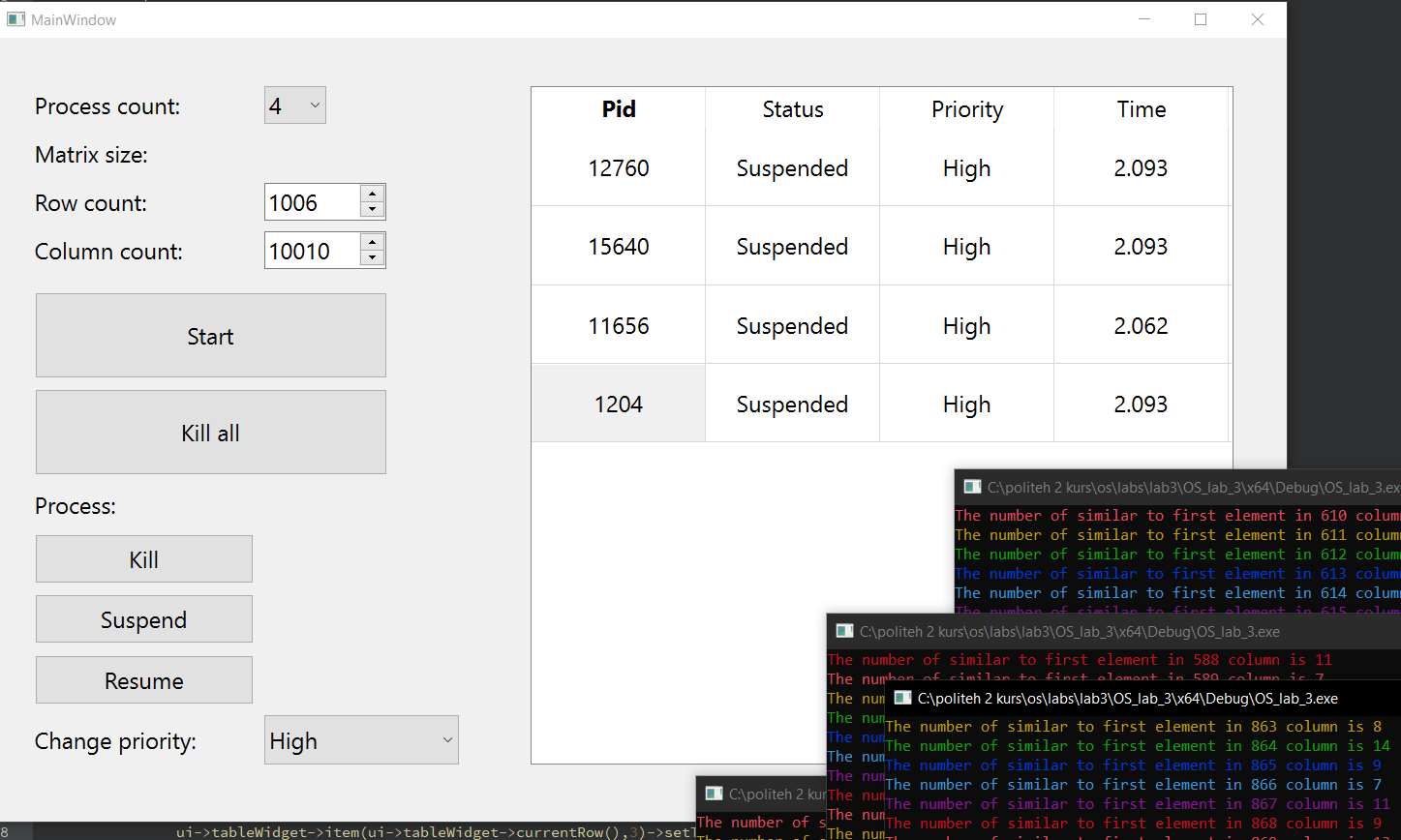
ui->tableWidget->item(ui->tableWidget->currentRow(),3)->setText(QString::number(GetCpuTime(*pi[ui->tableWidget->currentRow()]*)));

*break*;

}

}

****

****

##### Висновок

На даній лабораторній роботі я навчився працювати з процесами в ОС Windows з використанням функцій WinAPI.

За допомогою функції CreateProcess я реалізував індивідуальне завдання у 1, 2, 4 і 8 процесах. Також навчився використовувати такі функції, як SetPriorityClass , TerminateThread, SuspendThread, ResumeThread.