МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Дисциплина: Операционные системы

Лабораторная работа № 4

**Архитектура памяти Windows**

|  |  |
| --- | --- |
|  | Выполнила: ст. группы ПВ-31  Зановская А.И.  Проверил: Михелев В.М. |

Белгород

2018

**Цель работы: Получение практических навыков по использованию Win32 API для исследования памяти Windows**

Задание:

Разработать программное обеспечение для приложения, которое:

* выдает информацию, получаемую при использовании API *GetSystemInfo.*
* выдает информацию, получаемую при использовании API *GlobalMemoryStatus.* При выводе информации использовать диаграммы.
* Составляет карту виртуальной памяти для любого процесса.

Текст программы

mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <windows.h>

#include <tlhelp32.h>

#include <psapi.h>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

private slots:

void on\_comboBox\_FullProc\_currentTextChanged(const QString &arg1);

private:

Ui::MainWindow \*ui;

//1 задание

QString procArhToString(int n);

void procRevision(short r, int &mod, int &step);

void showSystemInformation();

//3 задание

void settingsOfTable();

HANDLE getProcessHandleByName(QString& pName);

bool getModuleFileNameEx(HANDLE hProcess, HMODULE hModule, QString &fname, QString &name);

void procFullList();

void showInfoOfSelectedProcess(QString nameProc);

void showInfoWithoutModules();

QString stateToQstring(int state);

QString typeToQString(int type);

QString protectToString(DWORD protect);

};

#endif // MAINWINDOW\_H

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QComboBox>

#include <QMessageBox>

#include <QTableWidget>

#include <QWidget>

#include <QPainter>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

showSystemInformation();

procFullList();

settingsOfTable();

}

MainWindow::~*MainWindow*()

{

delete ui;

}

QString MainWindow::procArhToString(int n)

{

switch (n) {

case PROCESSOR\_ARCHITECTURE\_INTEL:

return "x86";

case PROCESSOR\_ARCHITECTURE\_MIPS:

return "MIPSx64";

case PROCESSOR\_ARCHITECTURE\_ARM:

return "ARM";

case PROCESSOR\_ARCHITECTURE\_IA64:

return "Intel Itanium-based";

case PROCESSOR\_ARCHITECTURE\_AMD64:

return "AMDx64";

case PROCESSOR\_ARCHITECTURE\_UNKNOWN:

return "Неизвестно";

default:

return "Ошибка!";

}

}

void MainWindow::procRevision(short r, int &mod, int &step)

{

mod = (r >> 8);

step = (r & 0x00FF);

}

bool MainWindow::getModuleFileNameEx(HANDLE hProcess, HMODULE hModule, QString &fname, QString &name)

{

WCHAR buffer[MAX\_PATH + 1];

DWORD result = GetModuleFileNameEx(hProcess, hModule, buffer, MAX\_PATH + 1);

if (result != 0) {

name = fname = QString::fromWCharArray(buffer);

name.remove(0, name.lastIndexOf('\\') + 1);

return true;

}

return false;

}

void MainWindow::showSystemInformation()

{

int mod, step;

\_SYSTEM\_INFO sysinf;

GetSystemInfo(&sysinf);

procRevision(sysinf.wProcessorRevision, mod, step);

ui->lineEdit\_ArhProc->setText(procArhToString(sysinf.wProcessorArchitecture));

ui->lineEdit\_LvlProc->setText(QString::number(sysinf.wProcessorLevel));

ui->lineEdit\_KolProc->setText(QString::number(sysinf.dwNumberOfProcessors));

ui->lineEdit\_RedProc->setText(QString("Модель %1, шаг: %2").arg(mod).arg(step));

ui->lineEdit\_SizePage->setText(QString::number(sysinf.dwPageSize));

ui->lineEdit\_DownAdr->setText(QString("%1").arg((long)sysinf.lpMinimumApplicationAddress, 0, 16).toUpper());

ui->lineEdit\_UpAdr->setText(QString("%1").arg((long)sysinf.lpMaximumApplicationAddress, 0, 16).toUpper());

ui->lineEdit\_MaskProc->setText(QString("%1").arg(sysinf.dwActiveProcessorMask, 32, 2, QChar('0')));

ui->lineEdit\_Granul->setText(QString::number(sysinf.dwAllocationGranularity));

}

void MainWindow::procFullList()

{

ui->comboBox\_FullProc->addItem("");

QString name, fname;

DWORD\* processesId = new DWORD[255];

DWORD bytes, size = 255 \* sizeof(DWORD);

if (EnumProcesses(processesId, size, &bytes)) {

uint count = bytes / sizeof(DWORD);

for (uint i = 0; i < count; i++) {

HANDLE h = OpenProcess(PROCESS\_QUERY\_INFORMATION | PROCESS\_VM\_READ | SYNCHRONIZE, false, (int) processesId[i]);

if (getModuleFileNameEx(h, NULL, fname, name)) {

ui->comboBox\_FullProc->addItem(name);

}

CloseHandle(h);

}

}

}

HANDLE MainWindow::getProcessHandleByName(QString &pName)

{

QString name, fname;

DWORD\* processesId = new DWORD[255];

DWORD bytes, size = 255 \* sizeof(DWORD);

if (EnumProcesses(processesId, size, &bytes)) {

uint count = bytes / sizeof(DWORD);

for (uint i = 0; i < count; i++) {

HANDLE h = OpenProcess(PROCESS\_QUERY\_INFORMATION | PROCESS\_VM\_READ | SYNCHRONIZE, false, (int) processesId[i]);

if (getModuleFileNameEx(h, NULL, fname, name)) {

if (name == pName)

return h;

}

CloseHandle(h);

}

}

return NULL;

}

void MainWindow::showInfoOfSelectedProcess(QString nameProc)

{

ui->tableWidget->clear();

ui->tableWidget->clearContents();

ui->tableWidget->setRowCount(0);

ui->tableWidget->*reset*();

settingsOfTable();

ui->tableWidget->showColumn(0);

int numRow = ui->tableWidget->rowCount();

ui->tableWidget->insertRow(numRow);

int pageSize = 4096;

HANDLE hProcess = getProcessHandleByName(nameProc);

if (hProcess != 0) {

int initAddress = 0;

int numRow = 0;

MEMORY\_BASIC\_INFORMATION memInfo;

while (VirtualQueryEx(hProcess, (void\*) initAddress, &memInfo, sizeof(memInfo))) {

ui->tableWidget->insertRow(numRow);

QTableWidgetItem\* baseAddressItem = new QTableWidgetItem(QString("%1").arg((long) memInfo.BaseAddress, 0, 16));

ui->tableWidget->setItem(numRow, 0, baseAddressItem);

initAddress += memInfo.RegionSize;

QTableWidgetItem\* finalAddressItem = new QTableWidgetItem(QString("%1").arg(initAddress - 1, 0, 16));

ui->tableWidget->setItem(numRow, 1, finalAddressItem);

ui->tableWidget->setItem(numRow, 2, new QTableWidgetItem(QString::number(memInfo.RegionSize/pageSize))); //количество страниц

ui->tableWidget->setItem(numRow, 3, new QTableWidgetItem(stateToQstring(memInfo.State)));

if (memInfo.State != MEM\_FREE) {

ui->tableWidget->setItem(numRow, 5, new QTableWidgetItem(protectToString(memInfo.Protect)));

ui->tableWidget->setItem(numRow, 6, new QTableWidgetItem(typeToQString(memInfo.Type)));

}

numRow++;

}

ui->tableWidget->resizeColumnsToContents();

CloseHandle(hProcess);

}

}

void MainWindow::settingsOfTable()

{

ui->tableWidget->setColumnCount(7);

ui->tableWidget->setHorizontalHeaderLabels(QStringList()

<< "Базовый адрес" << "Конечный адрес"

<< "Количество страниц" << "Размер области" << "Состояние памяти"

<< "Тип защиты" << "Тип области");

ui->tableWidget->setSelectionMode(QAbstractItemView::SingleSelection);

ui->tableWidget->setEditTriggers(QAbstractItemView::NoEditTriggers);

ui->tableWidget->setSelectionBehavior(QAbstractItemView::SelectRows);

ui->tableWidget->hideColumn(4);

ui->tableWidget->resizeColumnsToContents();

ui->tableWidget->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

ui->tableWidget->verticalHeader()->setSectionResizeMode(QHeaderView::Stretch);

}

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

{

if(arg1 != "")

showInfoOfSelectedProcess(arg1);

}

QString MainWindow::stateToQstring(int state)

{

switch (state) {

case MEM\_COMMIT: return "Передано";

case MEM\_RESERVE: return "Зарезервировано";

case MEM\_FREE: return "Свободно";

}

}

QString MainWindow::typeToQString(int type)

{

switch (type) {

case MEM\_IMAGE: return "Загрузочный";

case MEM\_MAPPED: return "Файл данных";

case MEM\_PRIVATE: return "Приватный";

}

}

QString MainWindow::protectToString(DWORD protect)

{

QString str;

if (protect & PAGE\_EXECUTE\_READWRITE)

str += "Исполнение/чтение/запись";

else if (protect & PAGE\_READONLY)

str += "Только чтение";

else if (protect & PAGE\_READWRITE)

str += "Чтение/запись";

else if (protect & PAGE\_EXECUTE\_WRITECOPY)

str += "Исполнение/запись копирование";

else if (protect & PAGE\_EXECUTE\_READ)

str += "Исполнение/чтение";

else if (protect & PAGE\_EXECUTE)

str += "Исполнение";

else if (protect & PAGE\_NOACCESS)

str += "Нет доступа";

else if (protect & PAGE\_WRITECOPY)

str += "Запись копированием";

if (protect & PAGE\_GUARD)

str += "|Защищенный";

if (protect & PAGE\_NOCACHE)

str += "|Запрет кэширования";

if (str.length() == 0)

return QString::number(protect);

return str;

}

Задание 2.

mainwindow.cpp

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QMainWindow>

#include <windows.h>

#include <tlhelp32.h>

#include <psapi.h>

#include <QComboBox>

#include <QMessageBox>

#include <QTableWidget>

#include <QWidget>

#include <QPainter>

#include <QTimer>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

private:

Ui::MainWindow \*ui;

QTimer \*timer;

double sec = 0, sec1, sec2;

void paintEvent(QPaintEvent\* event);

void timerEvent(QTimerEvent\*);

void showDiag();

void timerUpd();

};

#endif // MAINWINDOW\_H

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QDebug>

#include <QTime>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

timer = new QTimer();

connect(timer,SIGNAL (timeout()), this, SLOT(timerUpd()));

timer->start(1000\*60);

}

MainWindow::~*MainWindow*()

{

delete ui;

}

void MainWindow::timerUpd()

{

sec++; ui->sec->setText(QString::number(sec));

repaint();

}

void MainWindow::showDiag()

{

MEMORYSTATUSEX memstat;

memstat.dwLength = sizeof(memstat);

GlobalMemoryStatusEx(&memstat);

ui->proc->setText(QString::number(memstat.dwMemoryLoad));

ui->phys\_all->setText(QString::number(memstat.ullTotalPhys/ 1024 / 1024) + " МБ");

ui->phys\_awail->setText(QString::number(memstat.ullAvailPhys/ 1024 / 1024) + " МБ");

ui->v\_all->setText(QString::number(memstat.ullTotalVirtual/ 1024 / 1024) + " МБ");

ui->v\_awail->setText(QString::number(memstat.ullAvailVirtual/ 1024 / 1024) + " МБ");

ui->page\_all->setText(QString::number(memstat.ullTotalPageFile/ 1024 / 1024) + " МБ");

ui->page\_awail->setText(QString::number(memstat.ullAvailPageFile/ 1024 / 1024) + " МБ");

}

void MainWindow::*paintEvent*(QPaintEvent \*event)

{

MEMORYSTATUSEX memstat;

memstat.dwLength = sizeof(memstat);

GlobalMemoryStatusEx(&memstat);

ui->textEdit->setText(QString::number(memstat.dwMemoryLoad));

ui->phys\_all->setText(QString::number(memstat.ullTotalPhys/ 1024 / 1024) + " МБ");

ui->phys\_awail->setText(QString::number(memstat.ullAvailPhys/ 1024 / 1024) + " МБ");

ui->v\_all->setText(QString::number(memstat.ullTotalVirtual/ 1024 / 1024) + " МБ");

ui->v\_awail->setText(QString::number(memstat.ullAvailVirtual/ 1024 / 1024) + " МБ");

ui->page\_all->setText(QString::number(memstat.ullTotalPageFile/ 1024 / 1024) + " МБ");

ui->page\_awail->setText(QString::number(memstat.ullAvailPageFile/ 1024 / 1024) + " МБ");

QPoint lt(width()/7,height()/7); //top

QPainter painter(this);

QRect rect[6];

int t = memstat.ullAvailPhys / (double)memstat.ullTotalPhys \* 100;

rect[0] = QRect(lt.x()+100, lt.y()+90, -20, -t);

rect[1] = QRect(lt.x()+120, lt.y()+90, -20, -100);

t = memstat.ullAvailVirtual/(double)memstat.ullTotalVirtual \* 100;

rect[2] = QRect(lt.x()+100, lt.y()+220, -20, -t);

rect[3] = QRect(lt.x()+120, lt.y()+220, -20, -100);

t = memstat.ullAvailPageFile /(double)memstat.ullTotalPageFile \* 100;

rect[4] = QRect(lt.x()+100, lt.y()+350, -20, -t);

rect[5] = QRect(lt.x()+120, lt.y()+350, -20, -100);

painter.setBrush(Qt::darkGray);

for (int i = 0; i < 6; i++)

if(i%2==0)

painter.drawRect(rect[i]);

painter.setBrush(Qt::gray);

for (int i = 0; i < 6; i++)

if(i%2==1)

painter.drawRect(rect[i]);

}



 

