Федеральное агентство по образованию и науке РФ

Пензенский государственный университет

Кафедра «Информационная безопасность

систем и технологий»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

«Системное программирование в операционных системах»

ПГУ 2.090105.001 ПЗ

Дисциплина: Безопасность операционных систем

Группа: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Разработал: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проект принят с оценкой: \_\_\_\_\_\_\_\_\_

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_

Пенза 2006

# Реферат

Пояснительная записка содержит 18 с., 1 рис., 2 приложения.

ГРАФИЧЕСКИЙ ИНТЕРФЕЙС, ПРОГРАММА, Алгоритм, РЕЕСТР

Объектом исследования является графическая операционная система.

Целью работы является создание программы, осуществляющей запуск программы по расписанию, расписание хранится в реестре. Программа должна нормально функционировать в среде графической операционной системы.

В процессе работы была разработана программа, осуществляющая запуск программы по расписанию, расписание хранится в реестре, написанная на языке, предназначенном для программирования графической операционной системы. Разработанная программа обладает графическим интерфейсом.

В результате выполнения работы были получены навыки программирования для графической операционной системы.

**Содержание**

Введение

1. Интерфейс пользователя
2. Руководство пользователю
3. Проверка работоспособности программы

Заключение

Приложение А. Алгоритм работы программы

Приложение Б. Листинг программы

**Введение**

Операционная система в наибольшей степени определяет облик всей вычислительной системы в целом.

ОС ограждает программистов от аппаратуры дискового накопителя и предоставляет ему простой файловый интерфейс, операционная система берет на себя все малоприятные дела, связанные с обработкой прерываний, управлением таймерами и оперативной памятью, а также другие низкоуровневые проблемы. В каждом случае та абстрактная, воображаемая машина, с которой, благодаря операционной системе, теперь может иметь дело пользователь, гораздо проще и удобнее в обращении, чем реальная аппаратура, лежащая в основе этой абстрактной машины. Также операционная система предоставляется пользователю некоторой расширенной или виртуальной машины, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальную машину.

**1. Интерфейс пользователя**

Методы и средства взаимодействия с аппаратными и программными средствами называют пользовательским интерфейсом. Интерфейс взаимодействия с пользователем – важный элемент любой программы. Любая программа должна обладать пользовательским интерфейсом, иначе, кроме своего разработчика, она будет недоступна в понимании лицу с ней не знакомому. Кроме того, интерфейс должен быть дружественным. Как правило, популярность программных продуктов напрямую зависит от интерфейса. Программы, имеющие наиболее удобный интерфейс, всегда пользуются спросом.

Интерфейсы бывают различных типов. В данном курсовом проекте разработанная программа имеет графический интерфейс. Написанная программа внешне представляет из себя графическое приложение, графической операционной системы. При старте программы на дисплее появляется графическое окно. Никакой функциональной нагрузки это окно не несет, через него не предусмотрен ввод команд и параметров в программу. Это окно лишь обозначает работу программы. Разработанный интерфейс в полной мере соответствует функциональному назначению программы.

В первом блоке производится открытие раздела реестра, если он успешно открывается, то программа получает текущее время, если нет, то на экран выводиться сообщение “Cannot open registry key!”, это означает, что раздел реестра не может открыться. После этого проверяется счётчик строк. В данной программе предусмотрена возможность создания расписания для 30 программ. Если проверка выявит, что счётчик равен 0, то программа завершит своё выполнение. В противном случае произойдёт считывание строки. Если строка считана, и считанное время равно текущему, то производится запуск заданной расписанием программы

Алгоритм работы программы изображен на рисунке А.1 в приложении А.

**2. Руководство пользователю**

Перед тем как осуществлять запуск программы по расписанию необходимо создать список запускаемых программ с указанием времени запуска каждой из них. Эту информацию необходимо вводить в специальный файл с расширением reg. Ввод информации необходимо осуществлять в определенном формате. Пример строки из файла schedule.reg, отвечающей за запуск программы калькулятора в 14:09.

**"1"="14:09 c:\\windows\\system32\\calc.exe"**

Далее необходимо импортировать информацию из файла в реестр. Теперь можно запускать данную программу. Для этого необходимо запустить исполняемый файл list.exe в среде графической операционной системы любым из возможных способов.

3. Проверка работоспособности программы

Для наглядной проверки работоспособности программы сделаем расписание запуска следующих программ:

- calc.exe

- cmd.exe

- notepad.exe

Для этого необходимо занести данные в файл schedule:

**Windows Registry Editor Version 5.00**

**[HKEY\_CURRENT\_USER\shedule]**

**"1"="14:09 c:\\windows\\system32\\calc.exe"**

**"2"="14:10 c:\\windows\\system32\\cmd.exe"**

**"3"="13:15 c:\\windows\\system32\\notepad.exe"**

Затем необходимо импортировать эти данные в реестр.

После этого запускаем программу. На экране появляется окно с надписью “*выполнение”*, что свидетельствует о работе программы.

В период выполнения, программа проверяет расписание, которое хранится в реестре, и если находит там время совпадающее с текущим, то запускается соответствующая программа.

**Заключение**

В результате курсового проекта была разработана программа, которая позволяет осуществлять запуск программы по расписанию, которое хранится в реестре. Средством написания данной программы являлась интегрированная среда программирования, специально предназначенная для программирования графической операционной системы. Разработанная программа обладает графическим интерфейсом.

В итоге, задание на курсовой проект было выполнено в полном объеме.

**Приложение А**

(ОБЯЗАТЕЛЬНОЕ)

**Алгоритм работы программы**



Рисунок 1- Алгоритм работы программы

**Приложение Б**

(ОБЯЗАТЕЛЬНОЕ)

**Листинг программы**

// list.cpp : Defines the entry point for the application.

//

#include "stdafx.h"

#include "windows.h"

#include "stdlib.h"

#include "winioctl.h"

#include "shellapi.h"

// *функция проверки расписания в реестре*

void CheckShedule(void);

char shed[30][3];

// *оконная функция*

LRESULT CALLBACK WindowProc(HWND hWnd,

UINT message,

WPARAM wParam,

LPARAM lParam);

// *основная функция*

int APIENTRY WinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPSTR lpCmdLine,

int nCmdShow)

{

// *создание и заполнение структуры окна*

WNDCLASS wcl;

wcl.style=CS\_HREDRAW|CS\_VREDRAW;

wcl.lpfnWndProc=WindowProc;

wcl.cbClsExtra=0;

wcl.cbWndExtra=DLGWINDOWEXTRA;

wcl.hInstance=hInstance;

wcl.hIcon=LoadIcon(NULL, IDI\_APPLICATION);

wcl.hCursor=LoadCursor(NULL, IDC\_ARROW);

wcl.hbrBackground=(HBRUSH)(COLOR\_BTNFACE+1);

wcl.lpszMenuName=NULL;

wcl.lpszClassName="WWW";

strcpy(shed[0], "1");

strcpy(shed[1], "2");

strcpy(shed[2], "3");

strcpy(shed[3], "4");

strcpy(shed[4], "5");

strcpy(shed[5], "6");

strcpy(shed[6], "7");

strcpy(shed[7], "8");

strcpy(shed[8], "9");

strcpy(shed[9], "10");

strcpy(shed[10], "11");

strcpy(shed[11], "12");

strcpy(shed[12], "13");

strcpy(shed[13], "14");

strcpy(shed[14], "15");

strcpy(shed[15], "16");

strcpy(shed[16], "17");

strcpy(shed[17], "18");

strcpy(shed[18], "19");

strcpy(shed[19], "20");

strcpy(shed[20], "21");

strcpy(shed[21], "22");

strcpy(shed[22], "23");

strcpy(shed[23], "24");

strcpy(shed[24], "25");

strcpy(shed[25], "26");

strcpy(shed[26], "27");

strcpy(shed[27], "28");

strcpy(shed[28], "29");

strcpy(shed[29], "30");

// *регистрация класса окна*

if (!RegisterClass(&wcl)) return 0;

*// создание диалогового окна*

HWND hWnd=CreateDialogParam(hInstance,"TESTWIN",NULL,NULL,NULL);

*// прорисовка окна*

ShowWindow(hWnd, nCmdShow);

*// обновление окна*

UpdateWindow(hWnd);

*// установка таймера*

SetTimer(hWnd, 1, 60000, NULL);

*// цикл обработки сообщений*

MSG lpMsg;

while (GetMessage(&lpMsg, NULL, 0 ,0))

{

TranslateMessage(&lpMsg);

DispatchMessage(&lpMsg);

}

return lpMsg.wParam;

}

LRESULT CALLBACK WindowProc(HWND hWnd, UINT message,

WPARAM wParam, LPARAM lParam)

{

switch(message)

{

case WM\_DESTROY: PostQuitMessage(0); break;

*// если пришло сообщение от таймера, проверка расписания в реестре*

case WM\_TIMER:

CheckShedule();

break;

default: return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

void CheckShedule(void)

{

*// объявление переменных*

SYSTEMTIME time;

HKEY hOpen;

DWORD datatype, Buffsize;

long opRes;

char\* szAlloc = "shedule\\";

char hour[3], minute[3], buff[1024];

hour[2] = minute[2] = 0;

*// открытие раздела реестра*

opRes = RegOpenKey(HKEY\_CURRENT\_USER,szAlloc,&hOpen);

*// если не открылся, то сообщение и выход*

if(opRes)

{

MessageBox(0, "Cannot open registry key!", "Error", MB\_ICONSTOP);

return;

}

*// получение текущего времени*

GetLocalTime(&time);

for(int i = 0; i < 30; i++)

{

*// чтение строки из реестра с временем и строкой запуска*

if (RegQueryValueEx(hOpen,shed[i],NULL,&datatype,(BYTE\*)buff,&Buffsize))

*// если не прочитано, то переход к следующей*

continue;

*// считывание времени запуска*

hour[0] = buff[0];

hour[1] = buff[1];

minute[0] = buff[3];

minute[1] = buff[4];

*// если текущее время равно считанному из реестра*

if((atoi(hour) == time.wHour) && (atoi(minute) == time.wMinute))

{

*// запуск программы*

ShellExecute(0,"open", &buff[6], NULL, NULL, SW\_SHOWNORMAL);

}

*// обнуление буфера*

memset(buff, 0, sizeof(buff));

}

}