**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий, механики и оптики**

**Кафедра вычислительной техники**

Лабораторная работа 4

Управление процессами

Выполнил: Гхази Даниэль

Группа P3218

Преподаватель: Зыков А.Г.

2017 г.

**Задание**

Разработать алгоритм и программу, реализующую следующие действия:

Сгенерировать (или использовать из предыдущих заданий) массив данных, который будет использоваться для сортировки и поиска в нём элементов.

Создать три отдельных процесса, в каждом из которых:

a) Отсортировать исходный массив по заданному в командной строке шаблону.

b) Осуществить поиск элемента, указанного также в командной строке, в отсортированном массиве.

Вывести временные характеристики и ID процессов.

Определить количество открытых дескрипторов.

**Листинг программы**

#define \_WIN32\_WINNT 0x0501

#pragma comment(lib, "Kernel32.lib")

#include<SDKDDKVer.h>

#include<windows.h>

#include <time.h>

#include <string.h>

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <cstdlib>

using namespace std;

int showTime(const char\* overture, LPSYSTEMTIME systemTime){

cout << overture << systemTime->wHour << ":" << systemTime->wMinute << ":" << systemTime->wSecond << ":" << systemTime->wMilliseconds;

return 0;

}

int showTime2(const char\* overture, LPSYSTEMTIME systemTime){

cout << overture << systemTime->wHour + 3 << ":" << systemTime->wMinute << ":" << systemTime->wSecond << ":" << systemTime->wMilliseconds;

return 0;

}

int showDelemiter(int height, int width){

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

for (int j = 0; j < width; j++){

cout << "-";

}

cout << endl;

}

return 0;

}

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

setlocale(LC\_ALL,"Russian");

HANDLE\* processHandles;

DWORD\* processIds;

DWORD numOfOpenedHandles;

processHandles = (HANDLE\*)malloc(3\*sizeof(HANDLE));

processIds = (DWORD\*)malloc(3\*sizeof(DWORD));

STARTUPINFO startUpInfo;

FILETIME creationFileTime;

SYSTEMTIME creationSystemTime;

FILETIME exitFileTime;

SYSTEMTIME exitSystemTime;

FILETIME userFileTime;

SYSTEMTIME userSystemTime;

FILETIME kernelFileTime;

SYSTEMTIME kernelSystemTime;

SECURITY\_ATTRIBUTES StdOutSA ={sizeof(SECURITY\_ATTRIBUTES), NULL, TRUE};

HANDLE hFileLog = CreateFile("log.out",GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, &StdOutSA, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

PROCESS\_INFORMATION processInformation0;

PROCESS\_INFORMATION processInformation1;

PROCESS\_INFORMATION processInformation2;

const char\* commandLine0 = "sort up data1";

const char\* commandLine1 = "sort down data2";

const char\* commandLine2 = "sort up data3";

GetStartupInfo(&startUpInfo);

startUpInfo.dwFlags = STARTF\_USESTDHANDLES;

startUpInfo.hStdOutput = hFileLog;

CreateProcess(NULL, (LPSTR)commandLine0, NULL, NULL, TRUE, 0, NULL, NULL, &startUpInfo, &processInformation0);

CreateProcess(NULL, (LPSTR)commandLine1, NULL, NULL, TRUE, 0, NULL, NULL, &startUpInfo, &processInformation1);

CreateProcess(NULL, (LPSTR)commandLine2, NULL, NULL, TRUE, 0, NULL, NULL, &startUpInfo, &processInformation2);

processHandles[0] = processInformation0.hProcess;

processHandles[1] = processInformation1.hProcess;

processHandles[2] = processInformation2.hProcess;

processIds[0] = processInformation0.dwProcessId;

processIds[1] = processInformation1.dwProcessId;

processIds[2] = processInformation2.dwProcessId;

WaitForMultipleObjects(3,processHandles,TRUE,INFINITE);

for (int i = 0; i < 3; i++){

cout << "Process ID: " << processIds[i] << endl;

GetProcessTimes(processHandles[i],&creationFileTime,&exitFileTime,&kernelFileTime,&userFileTime);

FileTimeToSystemTime(&creationFileTime,&creationSystemTime);

FileTimeToSystemTime(&exitFileTime,&exitSystemTime);

FileTimeToSystemTime(&userFileTime,&userSystemTime);

FileTimeToSystemTime(&kernelFileTime,&kernelSystemTime);

cout << showTime("Kernel system time\t",&kernelSystemTime)<<"\n";

cout << showTime("User time\t\t",&userSystemTime);

cout << showTime2("\nCreation time\t\t",&creationSystemTime) <<"\n";

cout << showTime2("Exit time\t\t",&exitSystemTime) <<"\n";

GetProcessHandleCount(processHandles[i], &numOfOpenedHandles);

cout << "In the process " << numOfOpenedHandles << " descriptors were opened" << endl << endl;

}

CloseHandle(hFileLog);

getchar();

return 0;

}

**Вывод**

В ходе выполнения работы было выяснено, что механизм управления процессами в среде Windows не представляет особенной сложности: процесс создается при помощи функции CreateProcess, множество параметров которой обеспечивают исключительную гибкость, например, можно произвольно установить поток ввода или вывода, а также настроить параметры командной строки. Функция GetProcessTimes позволяет получить временные характеристики процесса, однако неудобство составляет необходимость ручной конвертации возвращаемых данных в более удобный для манипуляции формат SYSTEMTIME.