Министерство образования республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

**Институт информационных технологий**

Специальность ПОИТ

**ЛАБОРАТОРНЫЕ РАБОТЫ**

По курсу Сетевые технологии и админстрирование операционных систем

Студент-заочник 3 курса

Группы № 581072

Скоп Дмитрий Александрович

Проверил

Калитеня Иван Леонидович

Минск, 2018

**Лабораторная работа №1**

Система команд и файловая структура ОС Unix/Linux

Цель работы: изучить команды ОС для работы с файлами, каталогами, дисками, системной датой и временем; текстовый редактор Kate и файловый менеджер Midnight Commander.

Задание: Программа, подсчитывающая количество символов с одинаковыми кодами ASCII в текстовом файле. Результаты подсчета записываются в другой текстовый файл (имена файлов передаются в качестве аргументов командной строки при запуске программы). Пример вывода программы для текстового файла:

QWER REEEt

WE E EEE EER SI I IO NN

файл, полученный в результате работы программы:

1. Q код ASCII 41 =1

2. W код ASCII 42 =2

3. E код ASCII 42 =11

…

итого: 25 символов

Листинг:

#include <stdio.h>

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string.h>

using namespace std;

int main()

{

int arr[255];

for (int i = 0; i < 255; i++) {

arr[i] = 0;

}

ifstream fileInput("lab1.txt");

char buff[1000];

fileInput.getline(buff, 1000);

fileInput.close();

for (int i = 0; i<strlen(buff); i++)

{

arr[buff[i]] +=1;

}

fstream fileOutput("out.txt", ios\_base::out);

int count = 0;

for (int i =0; i< 255; i++) {

if (arr[i] != 0) {

count++;

fileOutput << count << ". " << (char)i << " РєРѕРґ ASCII " << i << " = " << arr[i] << endl;

}

}

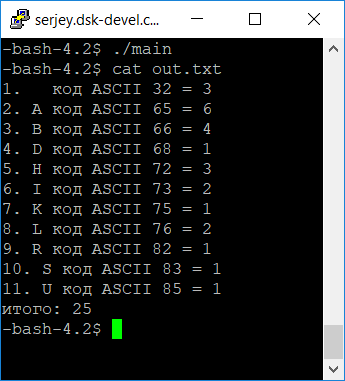
fileOutput << "РёС‚РѕРіРѕ: " << strlen(buff) << endl;

fileOutput.close();

return 0;

}

Результат выполнения:



**Лабораторная работа №2**

Управление ОС Linux с помощью интерпретатора BASH

Цель работы: исследовать основные объекты, команды, типы данных и операторы управления интерпретатора BASH; создать скрипт-файл.

Задание: Написать скрипт, находящий в заданном каталоге и его подкаталогах все файлы заданного размера (имя каталога задается пользователем в качестве первого аргумента командной строки). Диапазон (min – max) размеров файлов задается пользователем в качестве второго и третьего аргументов командной строки. Скрипт выводит результаты поиска в файл (четвертый аргумент командной строки) в следующем виде: полный путь, имя файла, его размер. На консоль выводится общее число просмотренных файлов.

Листинг:

#!/bin/bash

catal=$1

minValue=$2

maxValue=$3

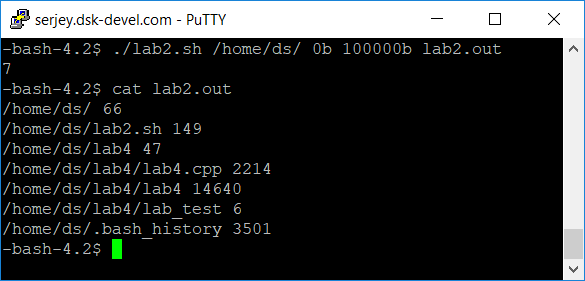
fout=$4

find $catal -size +$minValue -size -$maxValue -fprintf $fout "%p %s\n"

wc -l < $fout

exit 0

Результат выполнения:



**Лабораторная работа №3**

Структура BAT-файла, обработка аргументов, метки, операторы перехода и ветвления, организация циклов

Листинг 1:

@echo off

SETLOCAL EnableDelayedExpansion

chcp 1251 >nul

SET fileName=%1

if not exist %fileName% (

echo off >>%fileName%)

set /P studentName=Введите фамилию студента:

echo %studentName%>>%fileName%

sort %fileName% /Output %fileName%

Листинг 2:

@echo off

SETLOCAL EnableDelayedExpansion

chcp 1251 >nul

SET studentName=%1

SET fileName=%2

if not exist %fileName% (

echo Искомый файл не найден

) else (

findstr /i /v /C:"%studentName%" %fileName%>temp.txt

del %fileName%

rename temp.txt %fileName%

sort %fileName% /Output %fileName%

echo Фамилия студента %studentName% удалена из файла %fileName%

)

)

Листинг 3:

@echo off

SETLOCAL EnableDelayedExpansion

chcp 1251 >nul

SET studentName=%1

SET newStudentName=%2

SET fileName=%3

if not exist %fileName% (

echo Искомый файл не найден

) else (

findstr /i /v /C:"%studentName%" %fileName%>temp.txt

del %fileName%

rename temp.txt %fileName%

echo %newStudentName%>>%fileName%

sort %fileName% /Output %fileName%

echo Студент %studentName% теперь записан как %newStudentName% в %fileName%

)

Листинг 4:

@echo off

SETLOCAL EnableDelayedExpansion

chcp 1251 >nul

SET fileName=%1

if not exist %fileName%.txt (

echo Искомый файл не найден

) else (

"C:\Program Files\WinRAR\Rar.exe" a "%fileName%.rar" "%fileName%.txt"

)

Листинг 5:

@echo off

SETLOCAL EnableDelayedExpansion

chcp 1251 >nul

SET fileName=%1

if not exist %fileName%.rar (

echo Искомый файл не найден

) else (

"C:\Program Files\WinRAR\Rar.exe" e "%fileName%.rar"

)

Листинг 6:

@echo off

SETLOCAL EnableDelayedExpansion

chcp 1251 >nul

SET studentName=%1

SET fileName=%2

if not exist %fileName% (

echo Файл не найден

) else (

SET buf=findstr /i /C:"%studentName%" %fileName%

if [buf]==[] (

echo Студент не найден.

exit 0;

) else (

echo Студент найден.

exit /b 1

)

)

Листинг 7:

@echo off

SETLOCAL EnableDelayedExpansion

chcp 1251 >nul

:main

echo Выберите пункт меню:

echo.

echo 1 - Создание или пополнение списка группы студентов

echo 2 - Замена студента в группе

echo 3 - Исключение студента из группы

echo 4 - Помещение списка группы в одноименный архив

echo 5 - Извлечение списка группы из одноименного архива

echo 6 - Поиск студентов

echo 0 - Выход

echo.

SET /p choice="Ваш выбор: "

if not defined choice goto main

if %choice%==1 (goto job1)

if %choice%==2 (goto job3)

if %choice%==3 (goto job2)

if %choice%==4 (goto job4)

if %choice%==5 (goto job5)

if %choice%==6 (goto job6)

if %choice%==0 (goto end)

echo.

echo Неправильный выбор!

echo.

echo.

goto main

:job1

call job1.bat 581072.txt

goto main

:job2

set /p studentName=Введите фамилию студента:

call job2.bat %studentName% 581072.txt

goto main

:job3

set /p studentName=Введите фамилию заменяемого:

echo.

set /p newStudentName=Введите фамилию заменяющего:

call job3.bat %studentName% %newStudentName% 581072.txt

goto main

:job4

call job4.bat 581072

goto main

:job5

call job5.bat 581072

goto main

:job6

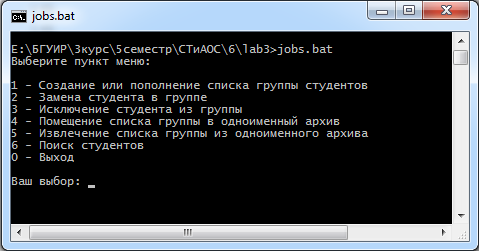
set /p studentName=Введите фамилию студента:

call job6.bat %studentName% 581072.txt

goto main

:end

Результат выполнения:



**Лабораторная работа №4**

Процессы и потоки в ОС Unix/Linux

Цель работы: исследовать методы создания процессов в ОС Linux, основные функции создания и управления процессами обмен данными между процессами.

Задание: Разработать программу «интерпретатор команд», которая воспринимает команды, вводимые с клавиатуры, и осуществляет их корректное выполнение. Для этого каждая вводимая команда должна выполняться в отдельно запускаемом процессе с использованием вызова exec(). Предусмотреть контроль ошибок.

Листинг:

#include <stdio.h>

#include <unistd.h>

#include <iostream>

#include <string>

#include <string.h>

#include <sstream>

#include<vector>

#include <sys/types.h>

#include <sys/wait.h>

using namespace std;

int main() {

int success\_status = 0;

bool is\_main = true;

string command ("");

getline (cin, command);

while (command.compare("exit") != 0) {

char \*command\_name = new char[256];

char \*\*command\_args = new char\*[256];

istringstream iss(command);

string token;

int i=0;

while (getline(iss, token, ' '))

{

if (i == 0) {

strcpy(command\_name, token.c\_str());

command\_args[i] = new char[256];

strcpy(command\_args[i], "");

} else {

command\_args[i] = new char[256];

strcpy(command\_args[i], token.c\_str());

}

i++;

}

//fork process

pid\_t PID;

PID = fork();

if (PID == 0) {

//child process, do the job

if (i > 1) {

execvp(command\_name, command\_args);

} else {

execlp(command\_name, "", NULL);

}

return 0;

}

getline (cin, command);

delete[] command\_name;

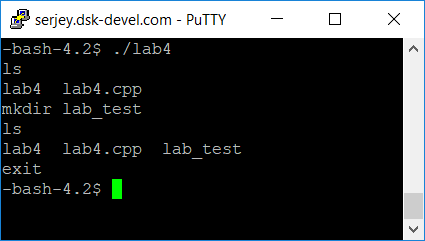
}

wait(NULL);

return 0;

}

Результат выполнения:



**Лабораторная работа №5**

Процессы и потоки в ОС Windows

Цель работы: исследовать основные объекты, команды, типы данных и операторы управления интерпретатора BASH; создать скрипт-файл.

Задание: Написать скрипт, находящий в заданном каталоге и его подкаталогах все файлы заданного размера (имя каталога задается пользователем в качестве первого аргумента командной строки). Диапазон (min – max) размеров файлов задается пользователем в качестве второго и третьего аргументов командной строки. Скрипт выводит результаты поиска в файл (четвертый аргумент командной строки) в следующем виде: полный путь, имя файла, его размер. На консоль выводится общее число просмотренных файлов.

Листинг:

#include <windows.h>

#include <tchar.h>

#include <strsafe.h>

#include <stdio.h>

#include <vector>

#include <unistd.h>

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <dirent.h>

#include <map>

#define BUF\_SIZE 255

using namespace std;

DWORD WINAPI SeeBytes( LPVOID lpParam );

void ErrorHandler(LPTSTR lpszFunction);

typedef struct MyData {

string url;

} MYDATA, \*PMYDATA;

bool ArrayEqual (char\* a, char\* b) {

for (int i = 0; i < sizeof a; i++)

{

if (a[i] != b[i]) return false;

}

return true;

}

int isFile(const char \*path)

{

struct stat path\_stat;

stat(path, &path\_stat);

return S\_ISREG(path\_stat.st\_mode);

}

DWORD WINAPI SeeBytes( LPVOID lpParam )

{

HANDLE hStdout;

PMYDATA data;

TCHAR msgBuf[BUF\_SIZE];

size\_t cchStringSize;

DWORD dwChars;

// Make sure there is a console to receive output results.

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

if( hStdout == INVALID\_HANDLE\_VALUE )

return 1;

// Cast the parameter to the correct data type.

// The pointer is known to be valid because

// it was checked for NULL before the thread was created.

data = (PMYDATA)lpParam;

ifstream inputFile(data->url.c\_str(), ifstream::binary);

char arrayKey[4] = {'r','d','c','t'};

char arrayTemp[sizeof arrayKey];

inputFile.seekg(0, inputFile.end);

int length = inputFile.tellg();

int countEquals = 0;

for (int i = 0; i < length - sizeof arrayTemp + 1; i++) {

inputFile.seekg(i, inputFile.beg);

inputFile.read((char\*)&arrayTemp, sizeof arrayTemp);

countEquals += (int) ArrayEqual(arrayKey, arrayTemp);

}

inputFile.close();

// Print the parameter values using thread-safe functions.

cout << "PID: " << GetCurrentThreadId() << "\t File: " << data->url <<"\t\t Bytes: " << length << "\t Equals: " << countEquals << "\n";

/\*StringCchPrintf(msgBuf, BUF\_SIZE, TEXT("PID: %d\t File: %s\t\t Bytes: %d\t Equals: %d\n"),

1, data->url, length, countEquals);

StringCchLength(msgBuf, BUF\_SIZE, &cchStringSize);

WriteConsole(hStdout, msgBuf, (DWORD)cchStringSize, &dwChars, NULL);\*/

return 0;

}

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

{

DIR \*dir;

struct dirent \*entry;

struct stat statbuf;

string url = "d:\\st5";

vector<string> files;

int maxThreadCount;

cout << "Enter max threads count: ";

cin >> maxThreadCount;

dir = opendir(url.c\_str());

while ( (entry = readdir(dir)) != NULL) {

if (isFile(entry->d\_name)) {

files.push\_back(entry->d\_name);

}

};

closedir(dir);

cout << "Files found: " << files.size() << endl;

int current = 0;

PMYDATA pDataArray[files.size()];

DWORD dwThreadIdArray[files.size()];

HANDLE hThreadArray[files.size()];

while(current < files.size()) {

// Allocate memory for thread data.

pDataArray[current] = (PMYDATA) HeapAlloc(GetProcessHeap(), HEAP\_ZERO\_MEMORY,

sizeof(MYDATA));

if( pDataArray[current] == NULL )

{

// If the array allocation fails, the system is out of memory

// so there is no point in trying to print an error message.

// Just terminate execution.

ExitProcess(2);

}

// Generate unique data for each thread to work with.

pDataArray[current]->url = files[current];

// Create the thread to begin execution on its own.

hThreadArray[current] = CreateThread(

NULL, // default security attributes

0, // use default stack size

SeeBytes, // thread function name

pDataArray[current], // argument to thread function

0, // use default creation flags

&dwThreadIdArray[current]); // returns the thread identifier

// Check the return value for success.

// If CreateThread fails, terminate execution.

// This will automatically clean up threads and memory.

if (hThreadArray[current] == NULL)

{

ErrorHandler(TEXT("CreateThread"));

ExitProcess(3);

}

current++;

}

// Wait until all threads have terminated.

WaitForMultipleObjects(files.size(), hThreadArray, TRUE, INFINITE);

// Close all thread handles and free memory allocations.

for(int i=0; i<files.size(); i++)

{

CloseHandle(hThreadArray[i]);

if(pDataArray[i] != NULL)

{

HeapFree(GetProcessHeap(), 0, pDataArray[i]);

pDataArray[i] = NULL; // Ensure address is not reused.

}

}

return 0;

}

void ErrorHandler(LPTSTR lpszFunction)

{

// Retrieve the system error message for the last-error code.

LPVOID lpMsgBuf;

LPVOID lpDisplayBuf;

DWORD dw = GetLastError();

FormatMessage(

FORMAT\_MESSAGE\_ALLOCATE\_BUFFER |

FORMAT\_MESSAGE\_FROM\_SYSTEM |

FORMAT\_MESSAGE\_IGNORE\_INSERTS,

NULL,

dw,

MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT),

(LPTSTR) &lpMsgBuf,

0, NULL );

// Display the error message.

lpDisplayBuf = (LPVOID)LocalAlloc(LMEM\_ZEROINIT,

(lstrlen((LPCTSTR) lpMsgBuf) + lstrlen((LPCTSTR) lpszFunction) + 40) \* sizeof(TCHAR));

StringCchPrintf((LPTSTR)lpDisplayBuf,

LocalSize(lpDisplayBuf) / sizeof(TCHAR),

TEXT("%s failed with error %d: %s"),

lpszFunction, dw, lpMsgBuf);

MessageBox(NULL, (LPCTSTR) lpDisplayBuf, TEXT("Error"), MB\_OK);

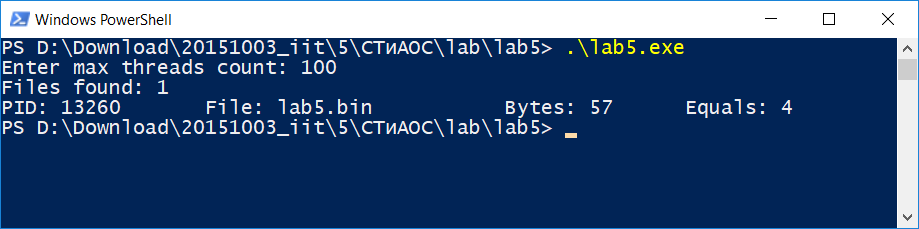
// Free error-handling buffer allocations.

LocalFree(lpMsgBuf);

LocalFree(lpDisplayBuf);

}

Результат выполнения:



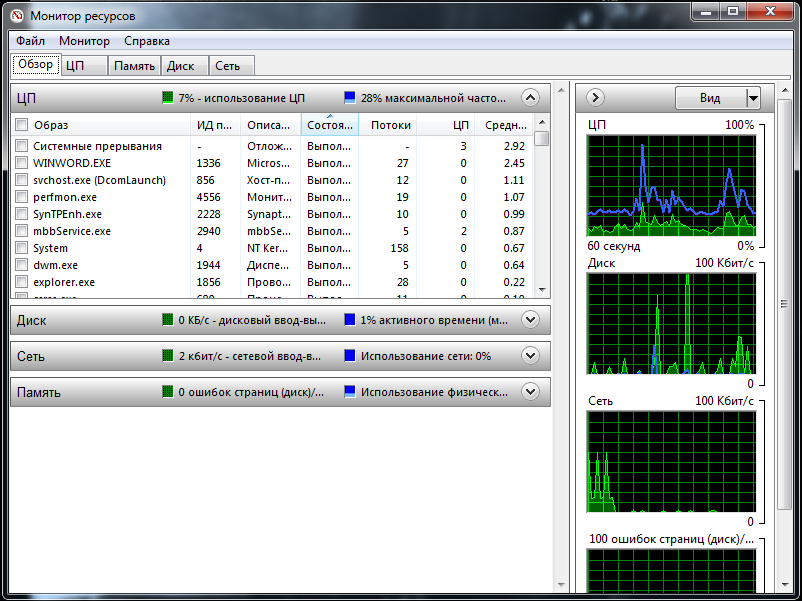
**Лабораторная работа №6**

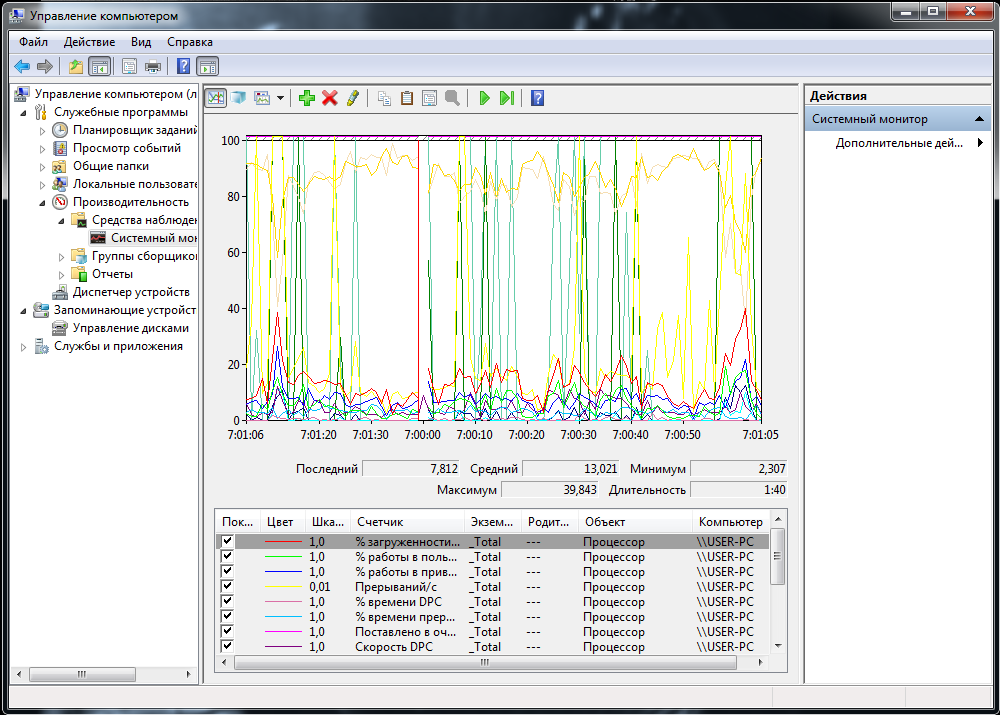
Анализ производительной системы с помощью программы «Производительность»

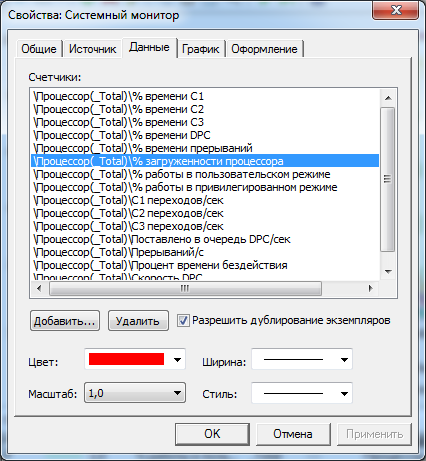
Цель: изучение средств измерения производительности Windows с помощью программы «Производительность».

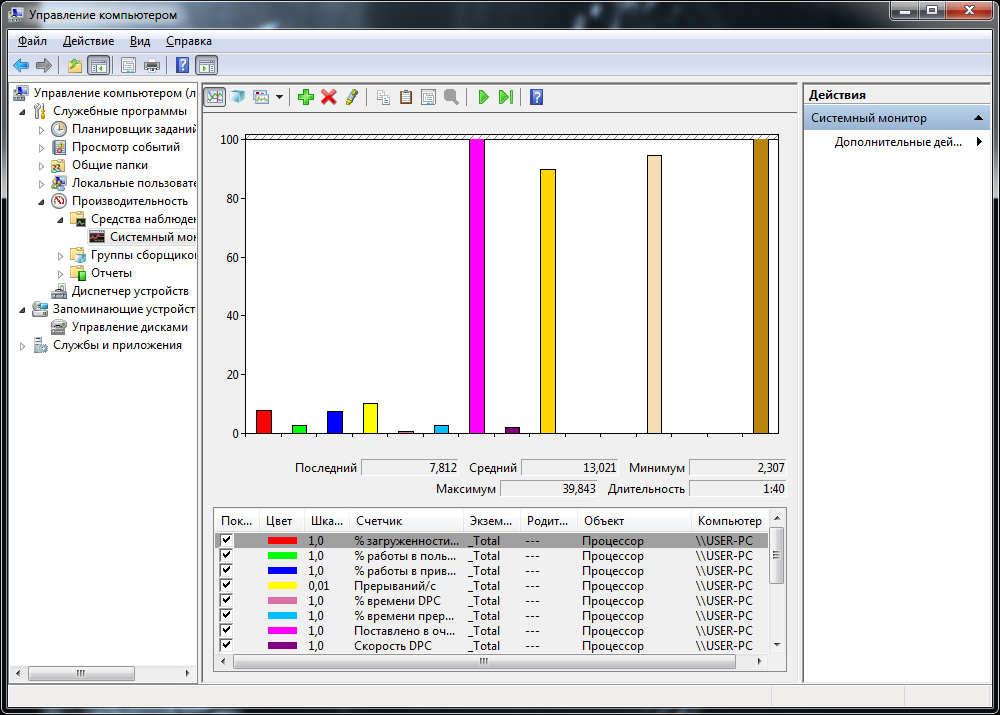
Задание: Провести исследование ОС с использованием программы «Производительность»

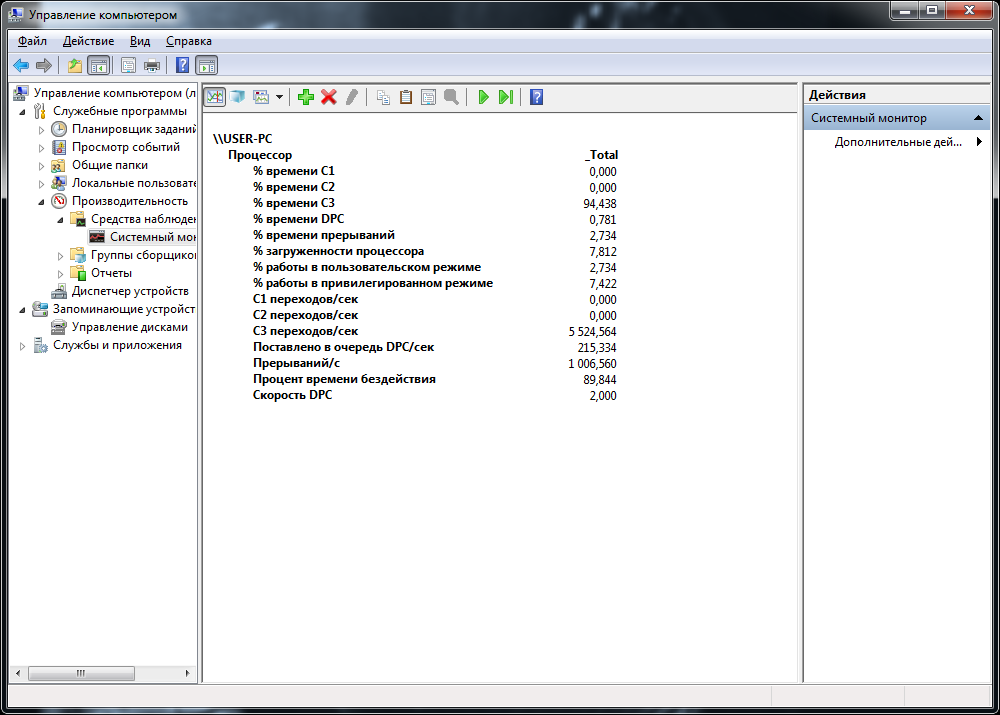
1-9)

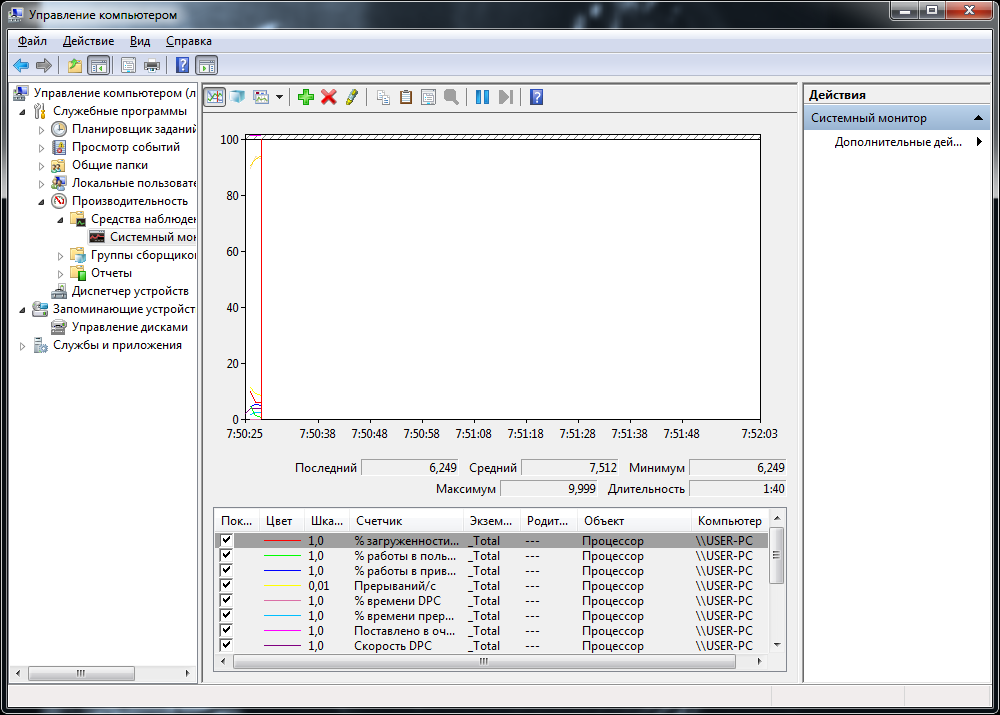


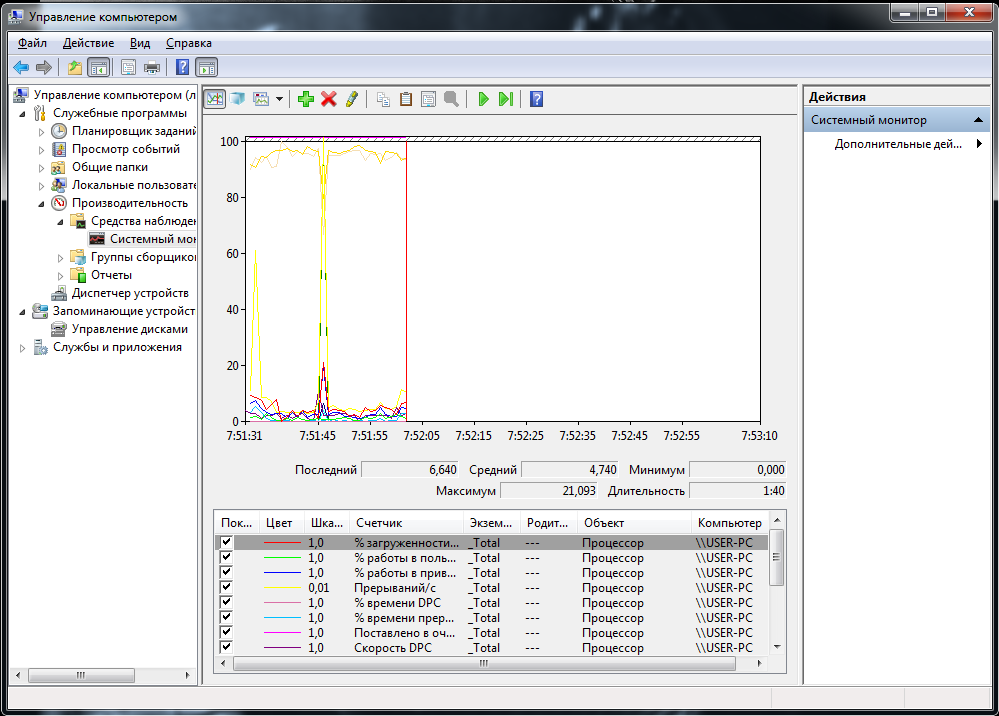








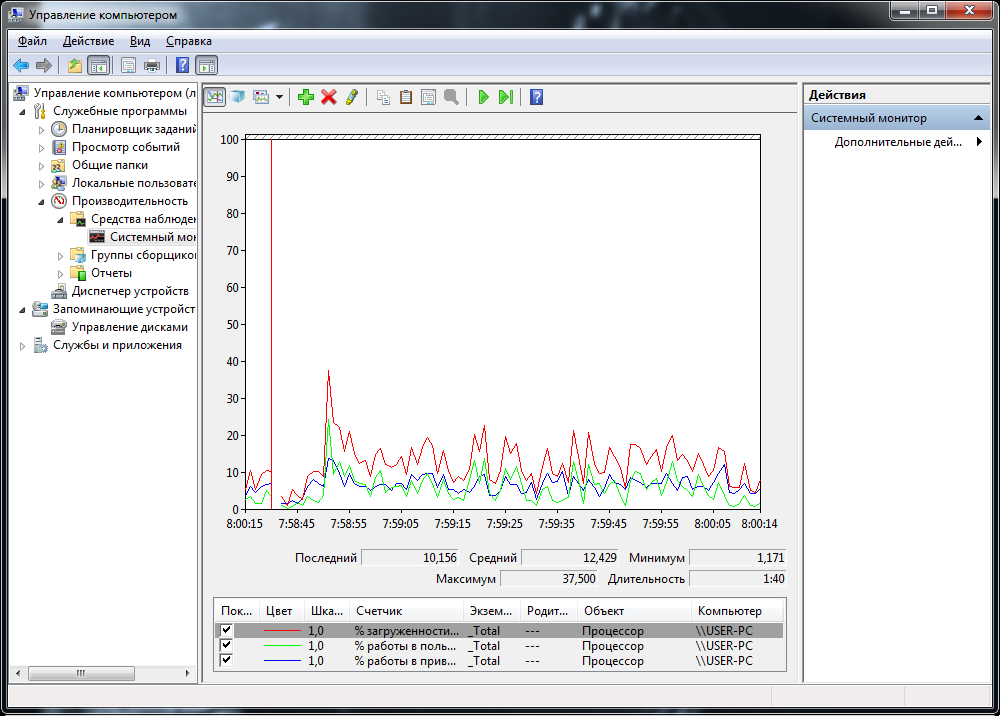




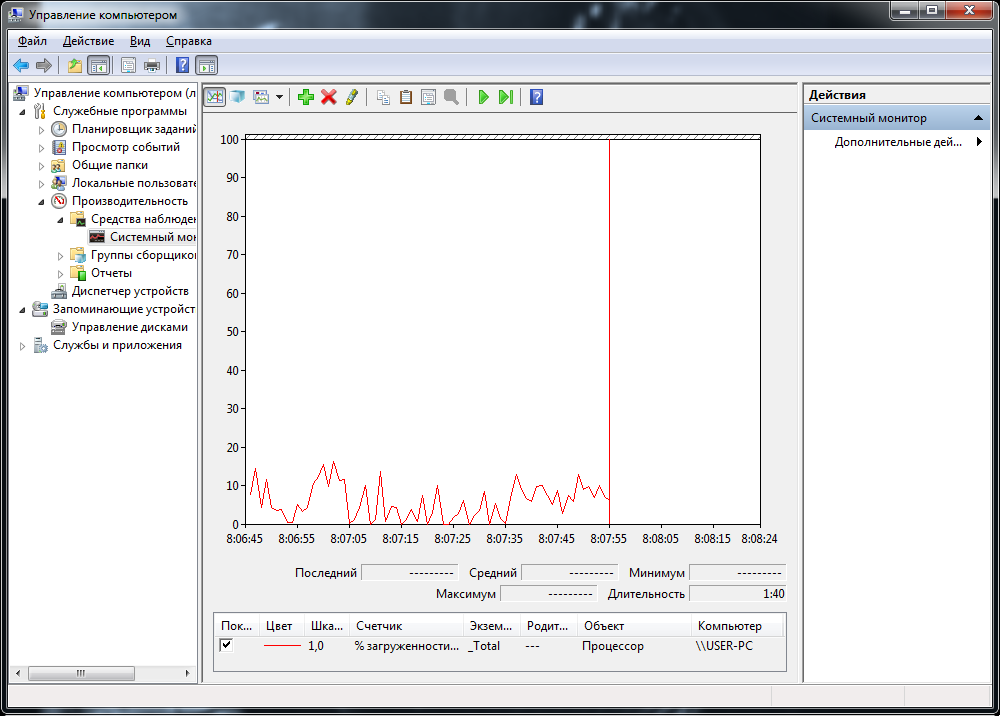
10) Таблица показаний:

|  |  |  |  |
| --- | --- | --- | --- |
| Действие | Процессор | Оперативная память | Внешняя память |
| Запуск программы | + | – | + |
| Редактирование документа | + | + | + |
| Сохранение документа | + | + | + |
| Завершение работы программы | + | – | – |

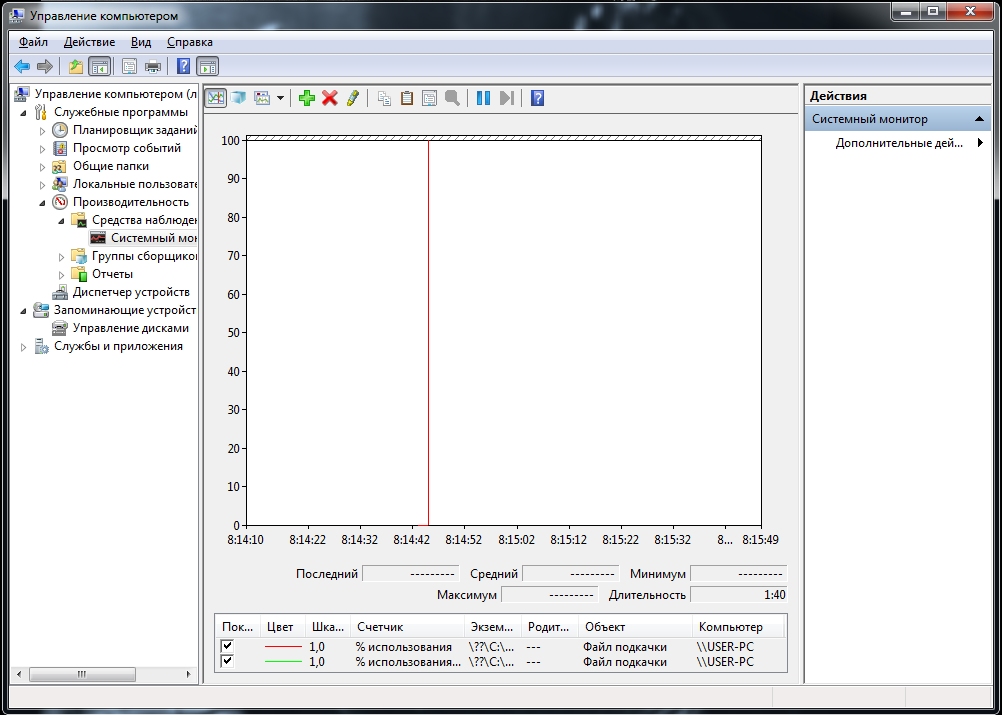
11)



12-13)



14)

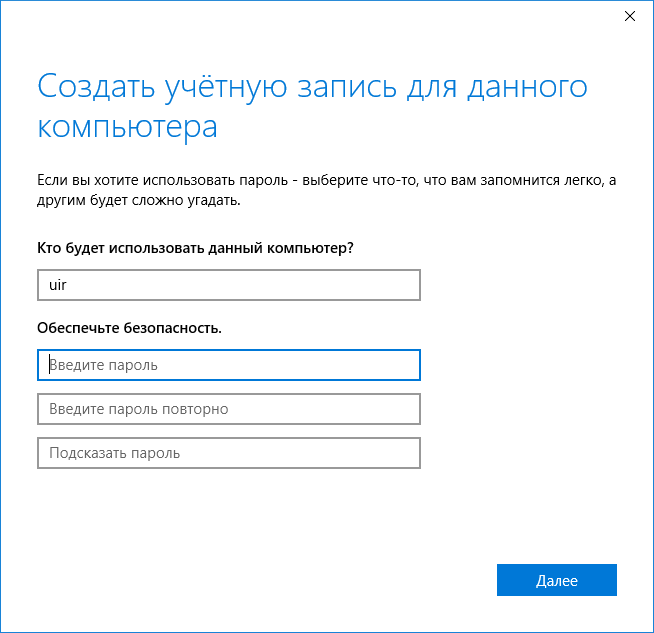


**Лабораторная работа №7**

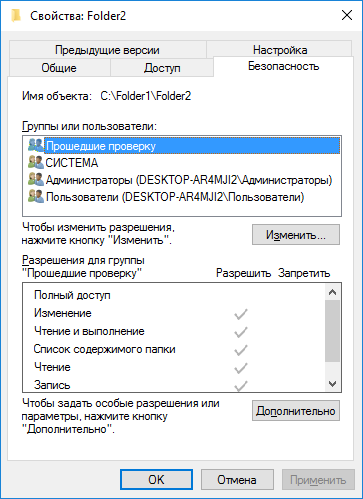
Использование приёмов работы с файловой системой NTFS. Назначение разрешений доступа к файлам и папкам

Цель работы: Научиться устанавливать разрешения NTFS для файлов и для папок для отдельных пользователей и групп в операционной системы Windows ХР, а также устранять проблемы доступа к ресурсам.

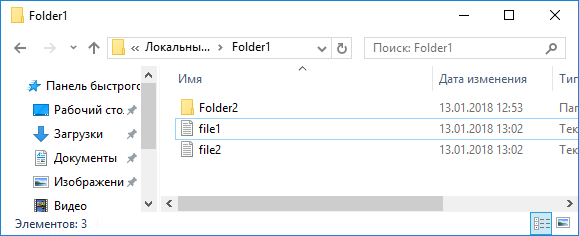
1-3)

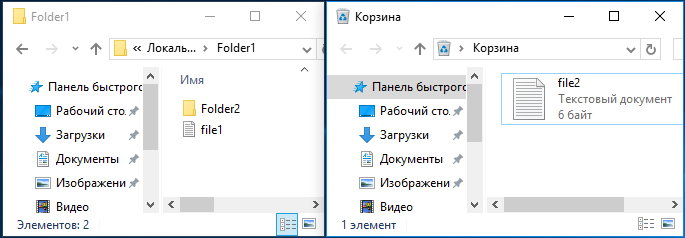


4)

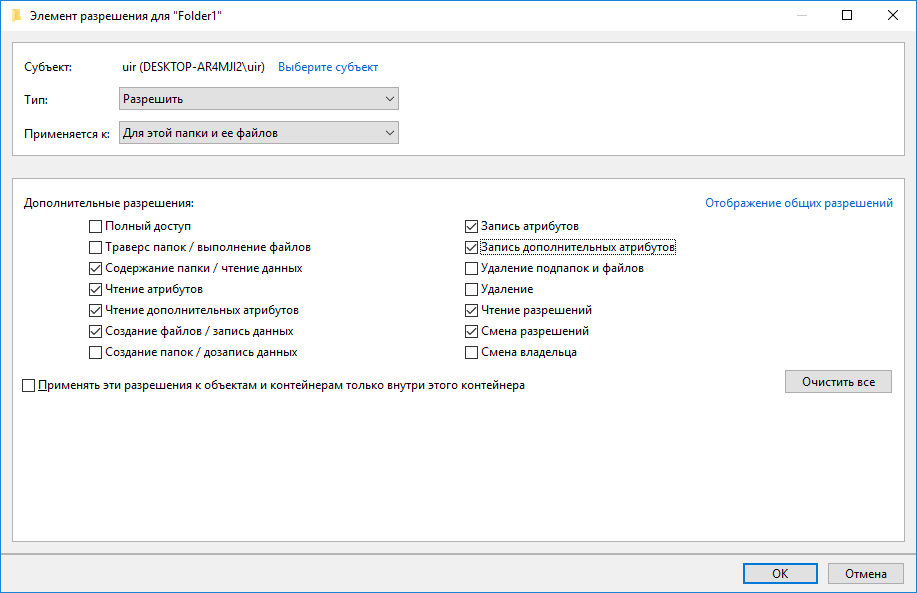


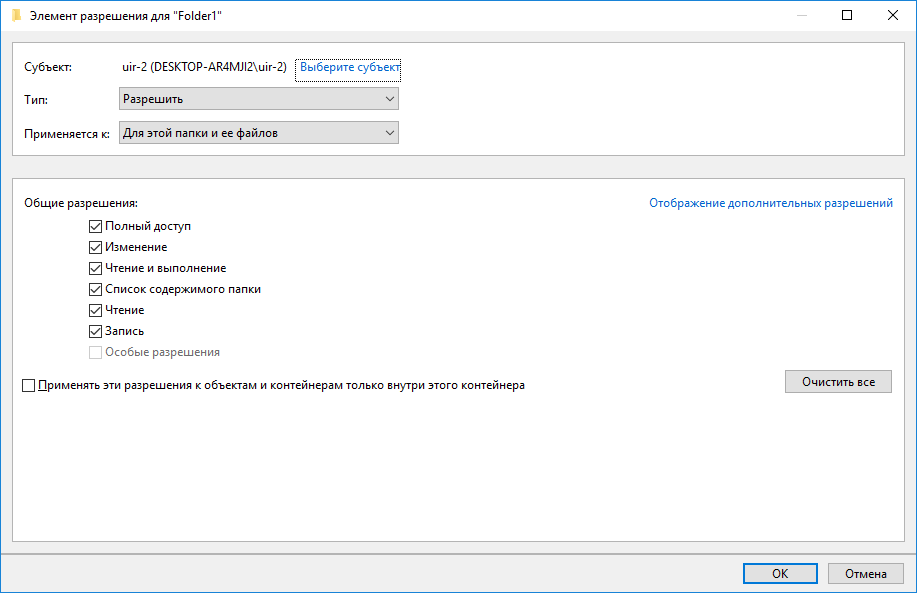
5-6)

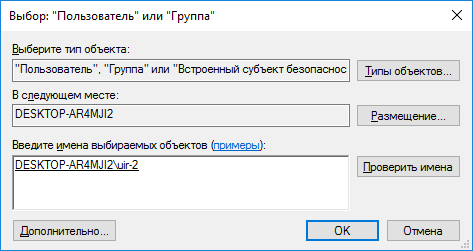


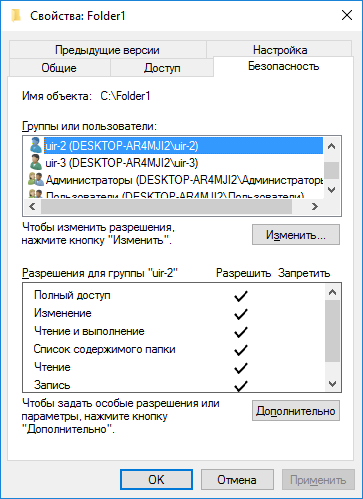


7)

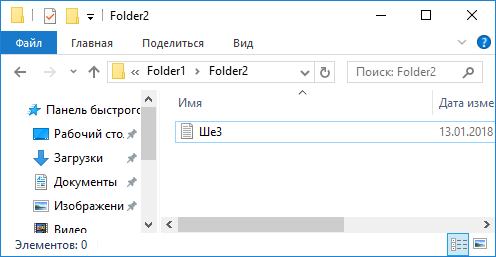




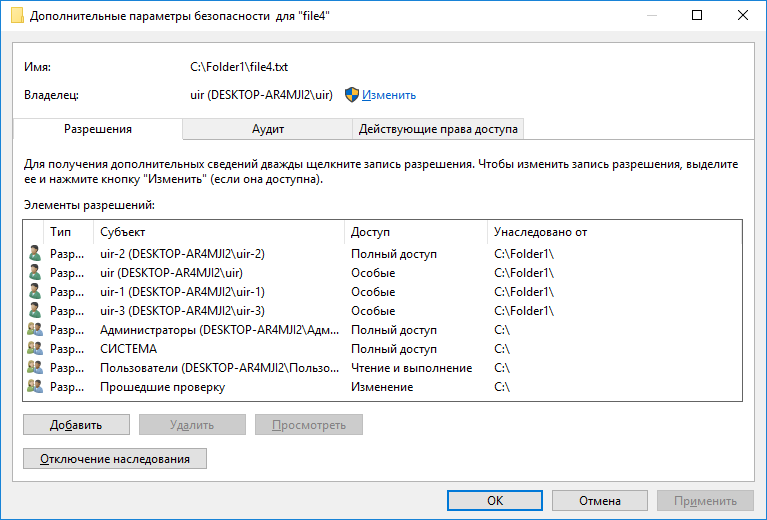


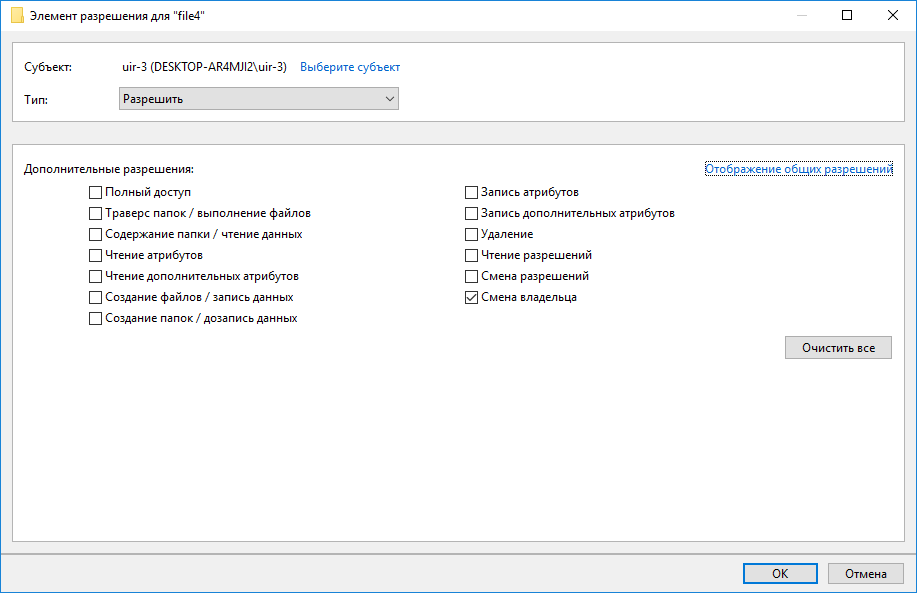


8)

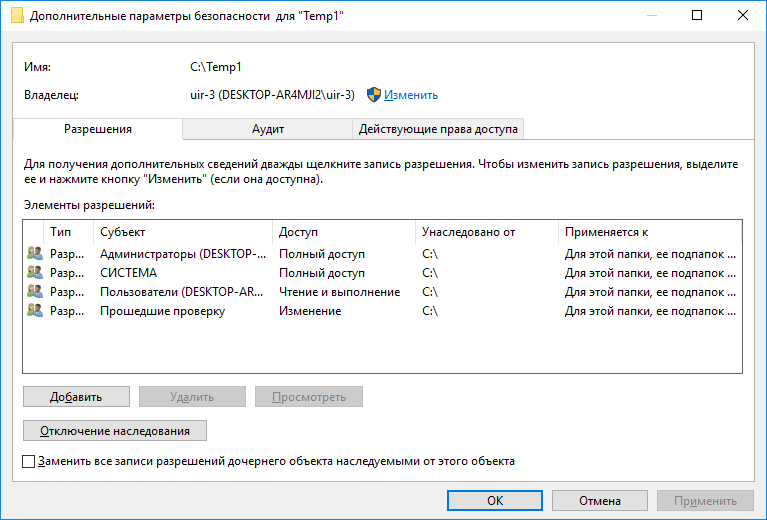


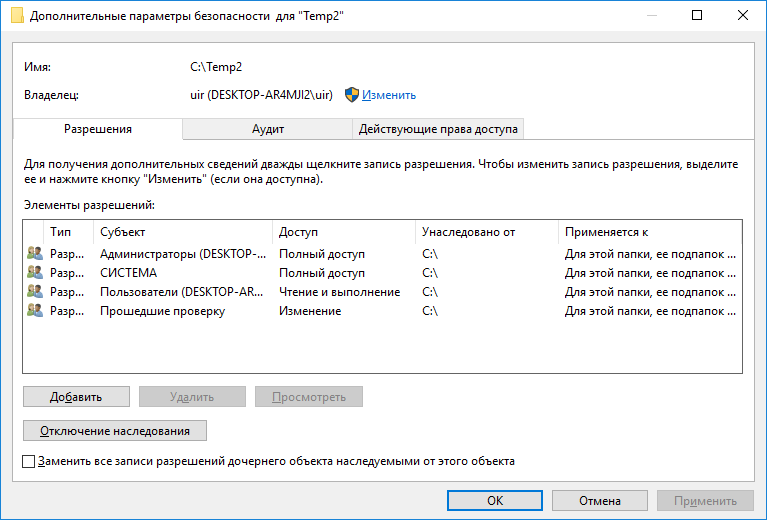
9)

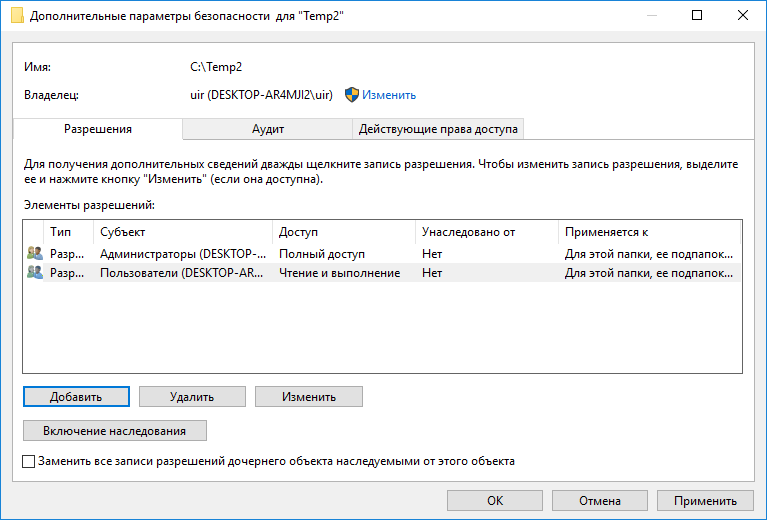


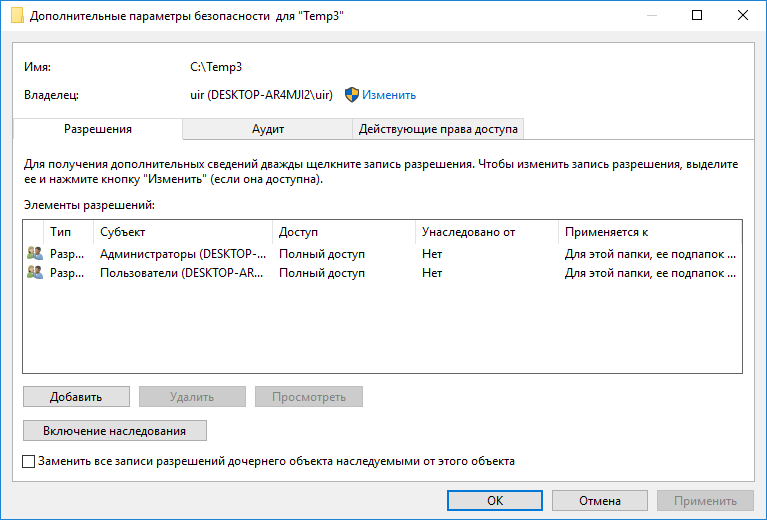


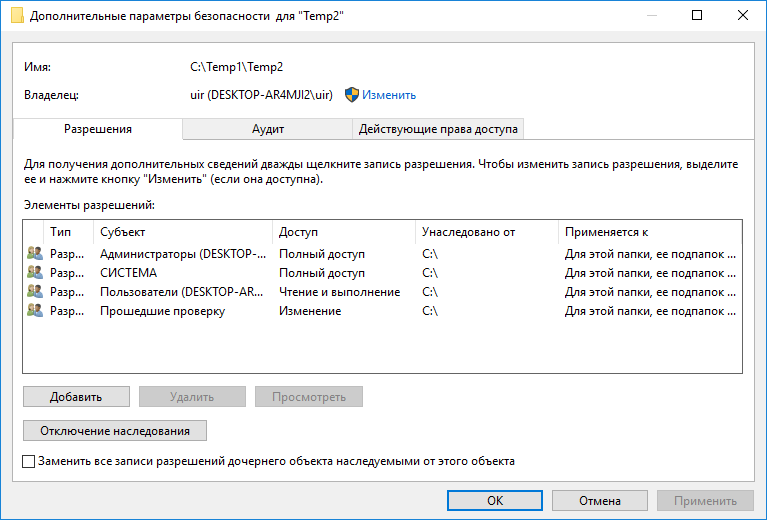
10)

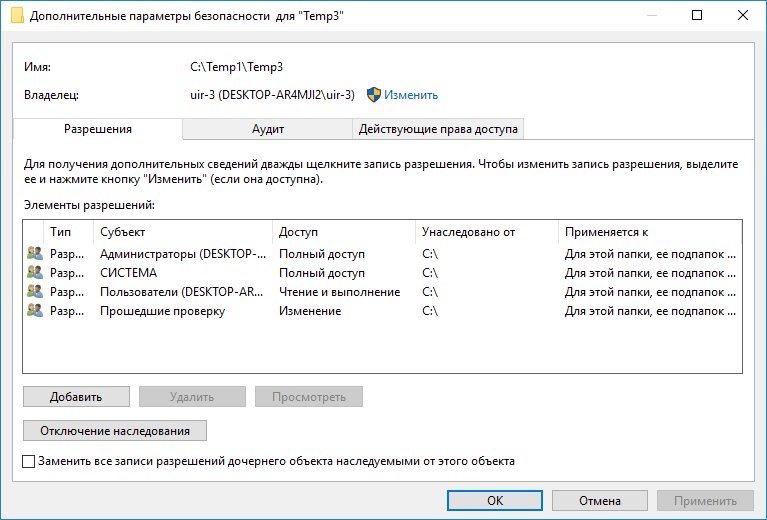








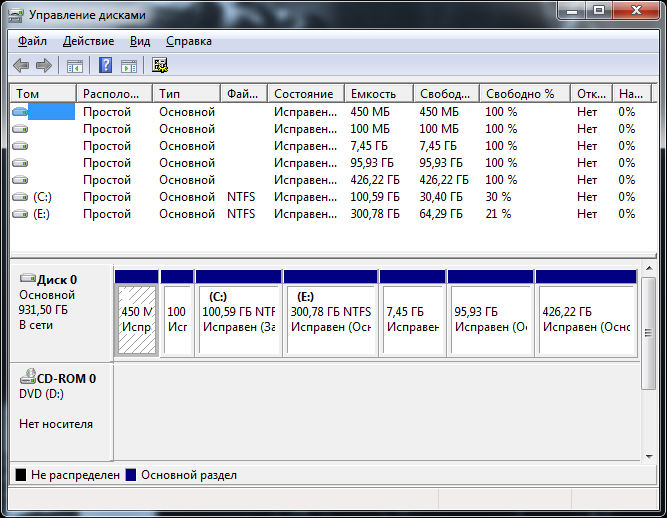




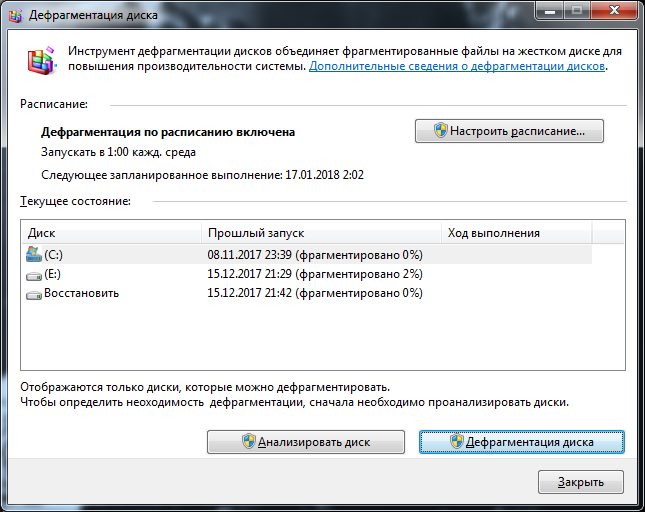
**Лабораторная работа № 8**

Операционная система Windows

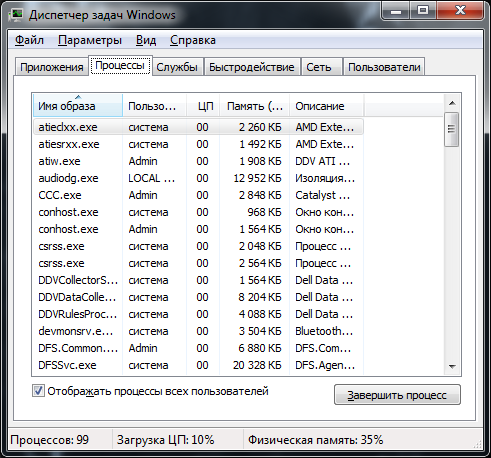
1. Управлениями дисками



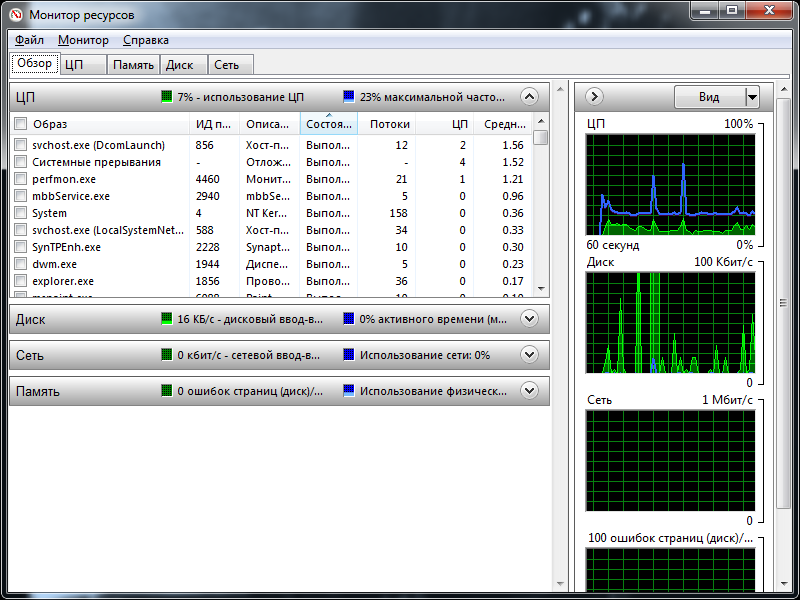
1. Дефрагментация дисков



1. Диспетчер задач



1. Монитор ресурсов



1. Свойства системы

