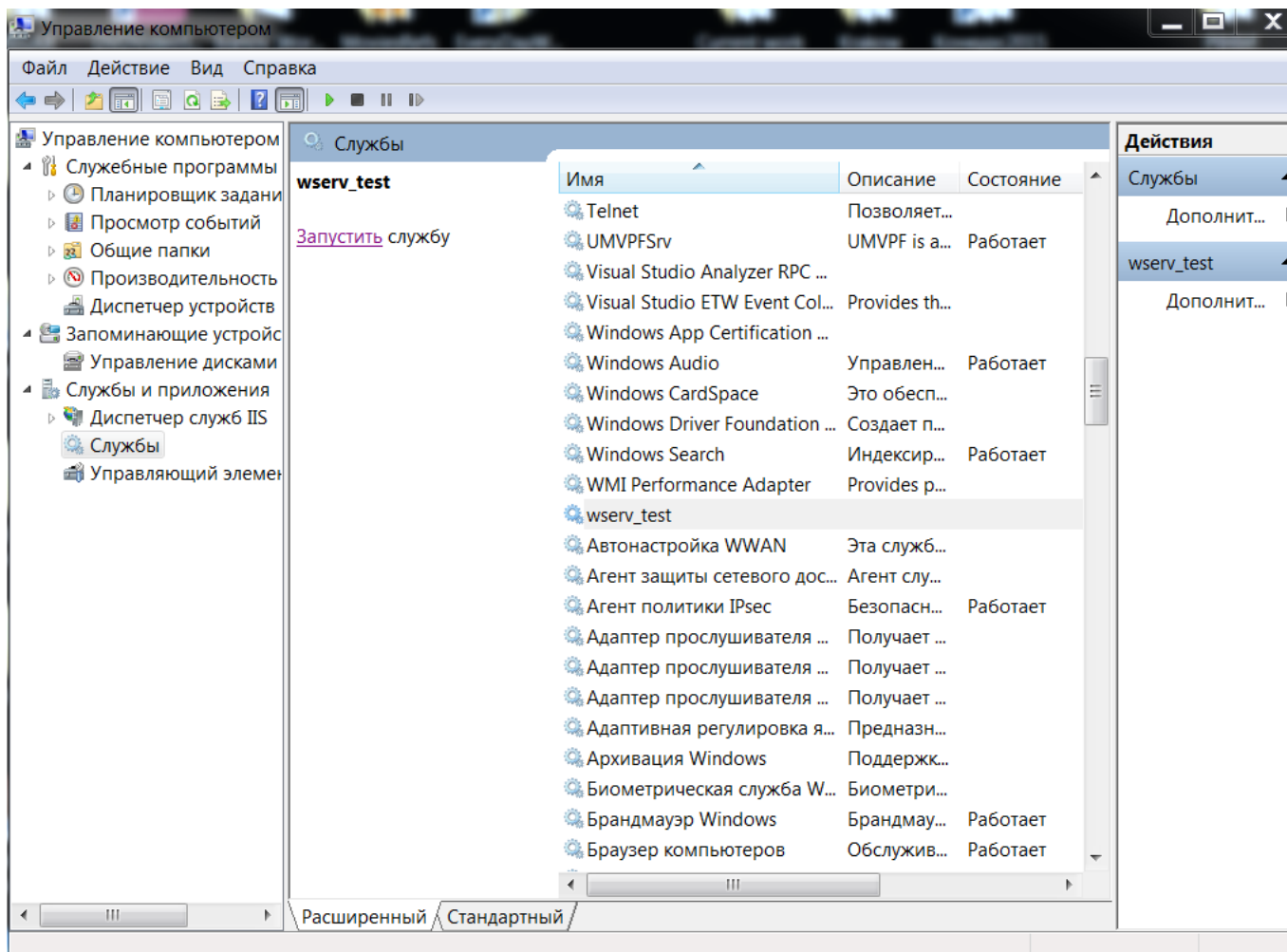


Лекция15

Windows-службы

- Определение Windows-службы.
- Управление службами с помощью оснастки MMC.
- Структура службы.
- Создание и удаление службы с помощью *Service Controller*.
- Программное создание и удаление службы.
- Программное управление службой.

Управление службами с помощью оснастки MMC (*Microsoft Management Console*)



Управление компьютером

Файл Действие Вид Справка

Управление компьютером

- Службные программы
 - Планировщик заданий
 - Просмотр событий
 - Общие папки
 - Производительность
 - Диспетчер устройств
- Запоминающие устройства
 - Управление дисками
- Службы и приложения
 - Диспетчер служб IIS
 - Службы**
 - Управляющий элемент

Службы

wserv_test

[Остановить](#) службу
[Приостановить](#) службу
[Перезапустить](#) службу

Имя	Описание	Состояние
Telnet	Позволяет...	
UMVPFSrv	UMVPF is a...	Работает
Visual Studio Analyzer RPC ...		
Visual Studio ETW Event Col...	Provides th...	
Windows App Certification ...		
Windows Audio	Управлен...	Работает
Windows CardSpace	Это обесп...	
Windows Driver Foundation ...	Создает п...	
Windows Search	Индексир...	Работает
WMI Performance Adapter	Provides p...	
wserv_test		Работает
Автонастройка WWAN	Эта служб...	
Агент защиты сетевого дос...	Агент слу...	
Агент политики IPsec	Безопасн...	Работает
Адаптер прослушивателя ...	Получает ...	
Адаптер прослушивателя ...	Получает ...	
Адаптер прослушивателя ...	Получает ...	
Адаптивная регулировка я...	Предназн...	
Архивация Windows	Поддержк...	
Биометрическая служба W...	Биометри...	
Брандмауэр Windows	Брандмау...	Работает
Браузер компьютеров	Обслужив...	Работает

Действия

Службы

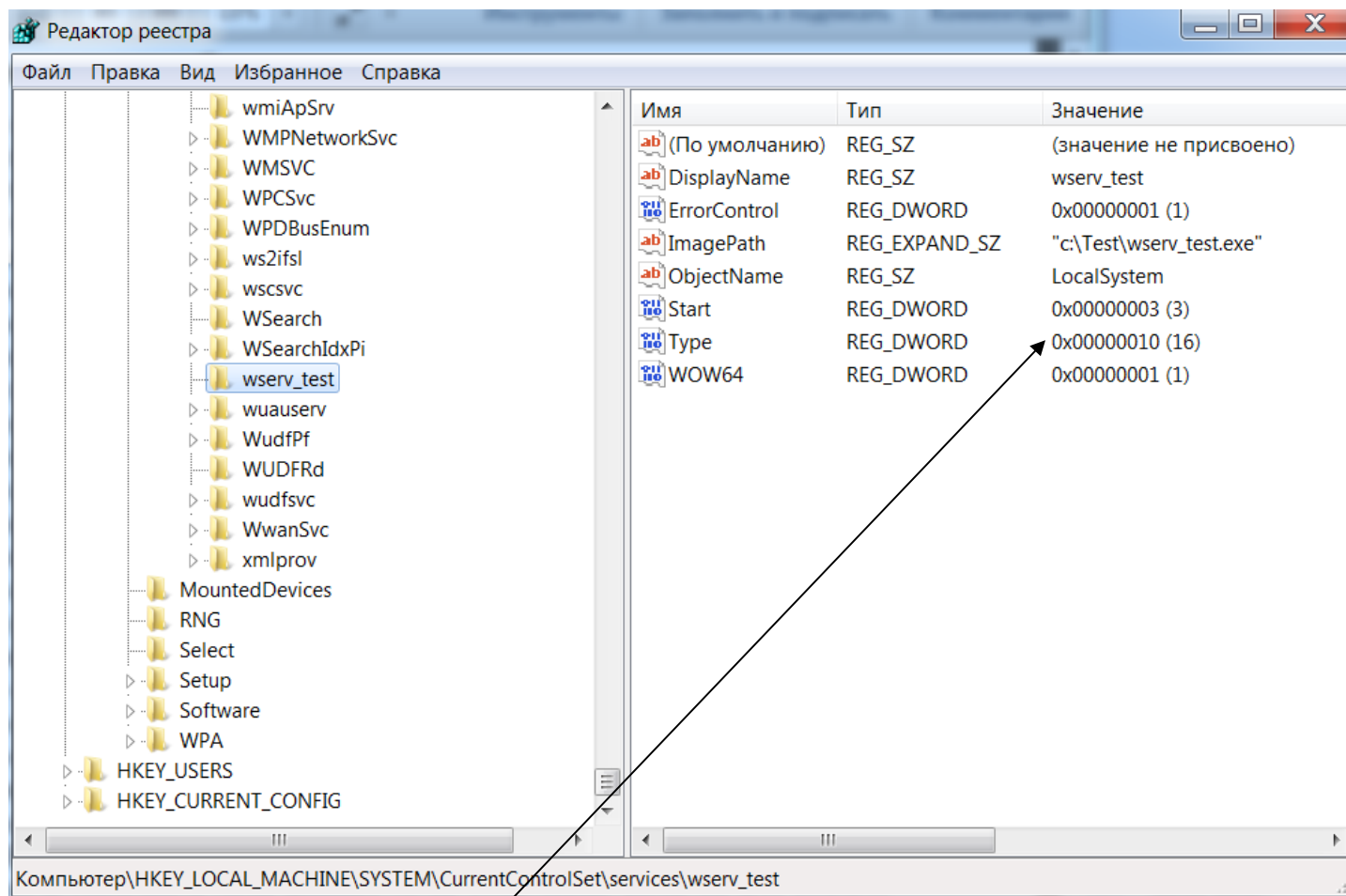
Дополнит...

wserv_test

Дополнит...

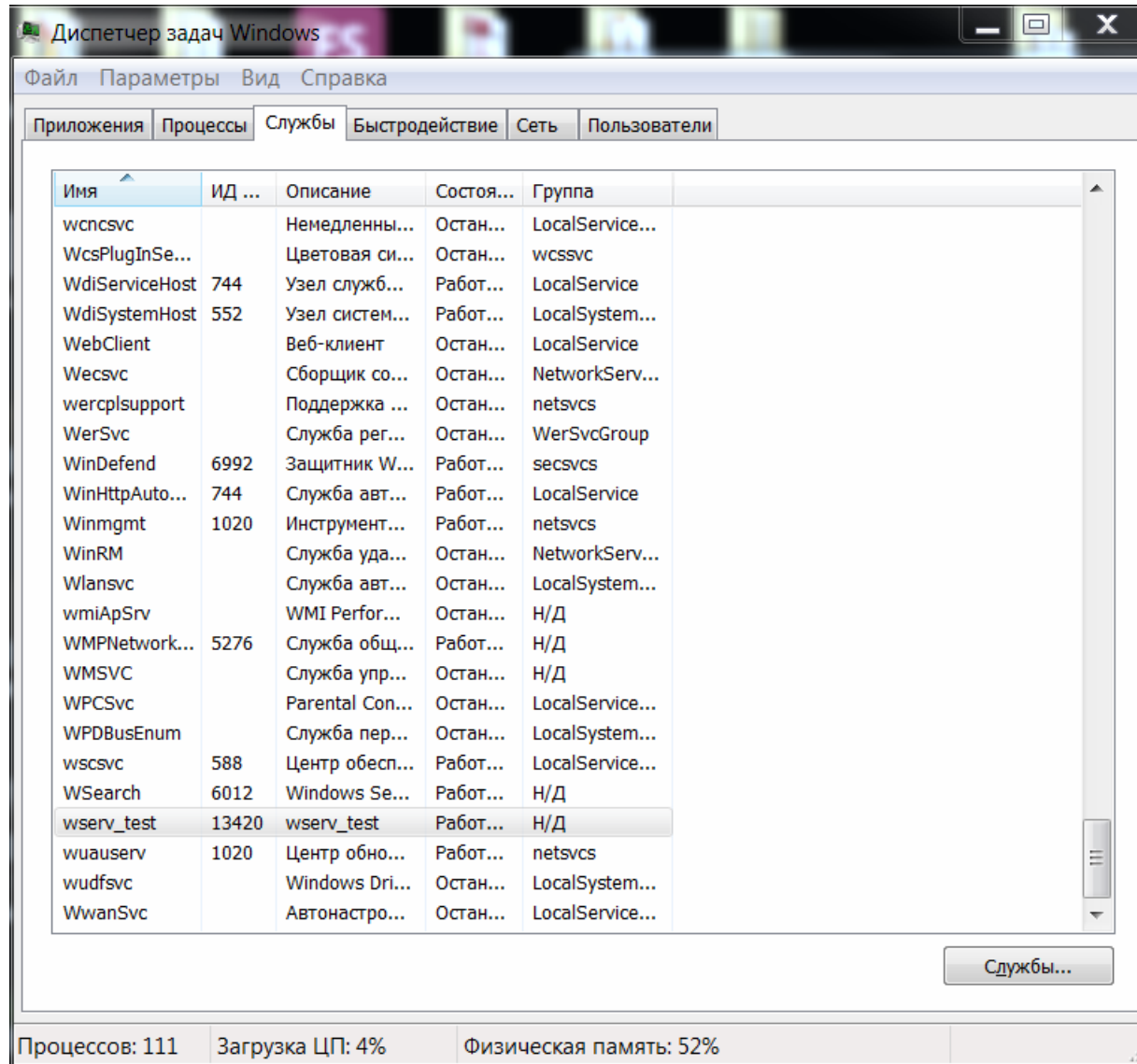
Расширенный / Стандартный

Регистрация службы в реестре



SERVICE_WIN32_OWN_PROCESS (служба выполняется в собственном процессе)
0x00000010

Отображение службы в Диспетчере задач



Диспетчер задач Windows

Файл Параметры Вид Справка

Приложения Процессы Службы Быстродействие Сеть Пользователи

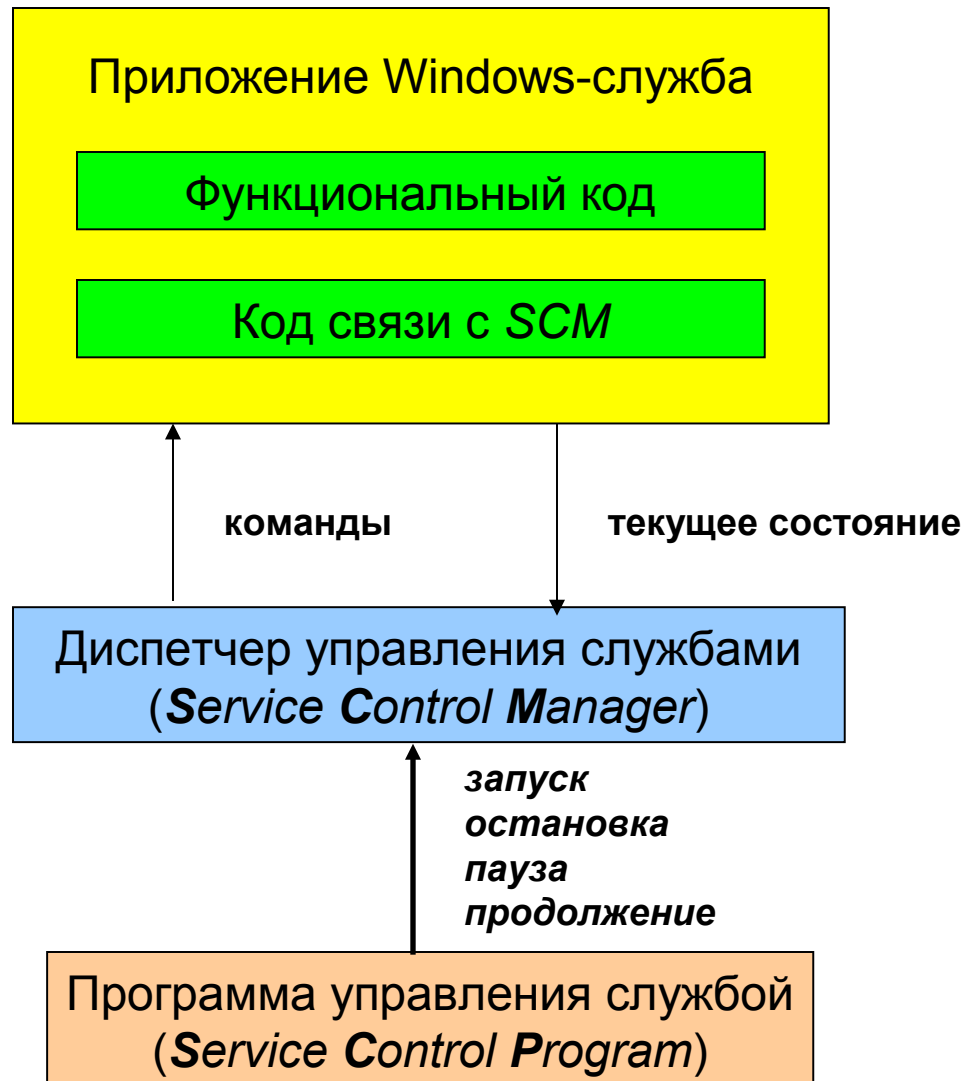
Имя образа	Пользова...	ЦП	Память (ч...	Описание
svchost.exe	LOCAL S...	00	8 700 КБ	Хост-проце...
svchost.exe	система	00	4 192 КБ	Хост-проце...
svchost.exe	NETWOR...	00	5 576 КБ	Хост-проце...
svchost.exe	система	00	29 376 КБ	Хост-проце...
svchost.exe	NETWOR...	00	8 484 КБ	Хост-проце...
svchost.exe	система	00	5 656 КБ	Хост-проце...
svchost.exe	LOCAL S...	00	10 680 КБ	Хост-проце...
svchost.exe	система	00	4 416 КБ	Хост-проце...
svchost.exe	LOCAL S...	00	2 840 КБ	Хост-проце...
svchost.exe	система	00	4 728 КБ	Хост-проце...
svchost.exe	система	00	4 632 КБ	Хост-проце...
svchost.exe	NETWOR...	00	2 264 КБ	Хост-проце...
svchost.exe	LOCAL S...	00	6 096 КБ	Хост-проце...
svchost.exe	система	00	10 592 КБ	Хост-проце...
System	система	00	260 КБ	NT Kernel ...
taskeng.exe	ewgenij	00	2 420 КБ	Обработчи...
taskhost.exe	ewgenij	00	8 500 КБ	Хост-проце...
taskhost.exe	LOCAL S...	00	4 800 КБ	Хост-проце...
taskmgr.exe	admin	01	3 748 КБ	Диспетчер ...
TBPANEL.exe *32	ewgenij	00	2 176 КБ	Vtune : Dis...
TCPSVCS.EXE	LOCAL S...	00	1 560 КБ	TCP/IP Ser...
TimeMgmtDaemon.exe *32	система	00	1 168 КБ	Smart Tim...
TrustedInstaller.exe	система	00	4 340 КБ	Установщи...
UMVPFSrv.exe *32	система	00	1 228 КБ	Logitech Us...
wininit.exe	система	00	1 384 КБ	Автозагру...
winlogon.exe	система	00	2 800 КБ	Программа...
wmpnetwk.exe	NETWOR...	00	10 052 КБ	Служба об...
wserv_test.exe *32	система	00	992 КБ	wserv_test....
YandexDisk.exe	ewgenij	00	402 912 КБ	Яндекс.Диск
YandexDiskStarter.exe	ewgenij	00	1 680 КБ	YandexDisk...

☒ Отображать процессы всех пользователей

Завершить процесс

Процессов: 116 Загрузка ЦП: 3% Физическая память: 53%

Windows-служба



```
#include <windows.h>
#include <iostream.h>
#include <string.h>
#include <time.h>
#include <process.h>
#include <stdio.h>
```

```
SERVICE_STATUS      wserv_testStatus;
SERVICE_STATUS_HANDLE wserv_testStatusHandle;
```

```
typedef struct _SERVICE_STATUS {
    DWORD dwServiceType;
    DWORD dwCurrentState;
    DWORD dwControlsAccepted;
    DWORD dwWin32ExitCode;
    DWORD dwServiceSpecificExitCode;
    DWORD dwCheckPoint;
    DWORD dwWaitHint;
} SERVICE_STATUS, *LPSERVICE_STATUS;
```


Задание точки/точек входа в службу и запуск бесконечного цикла в **потоке** **службы**:

```
void main(int argc, char *argv[]){
    SERVICE_TABLE_ENTRY DispatchTable[] =
    {
        { TEXT("wserv_test"), wserv_testStart },
        { NULL, NULL }
    };
    //.....//
    StartServiceCtrlDispatcher( DispatchTable));
}
```

```
typedef struct _SERVICE_TABLE_ENTRY {
    LPTSTR IpServiceName;
    LPSERVICE_MAIN_FUNCTION IpServiceProc;
} SERVICE_TABLE_ENTRY, *LPSERVICE_TABLE_ENTRY;
```

Создание службы с помощью **Service Controller (SC)**:

```
c:\>c:\windows\syswow64\sc create wserv_test binPath= C:\Test\service1.exe
[SC] CreateService: успех
```

Программное создание службы

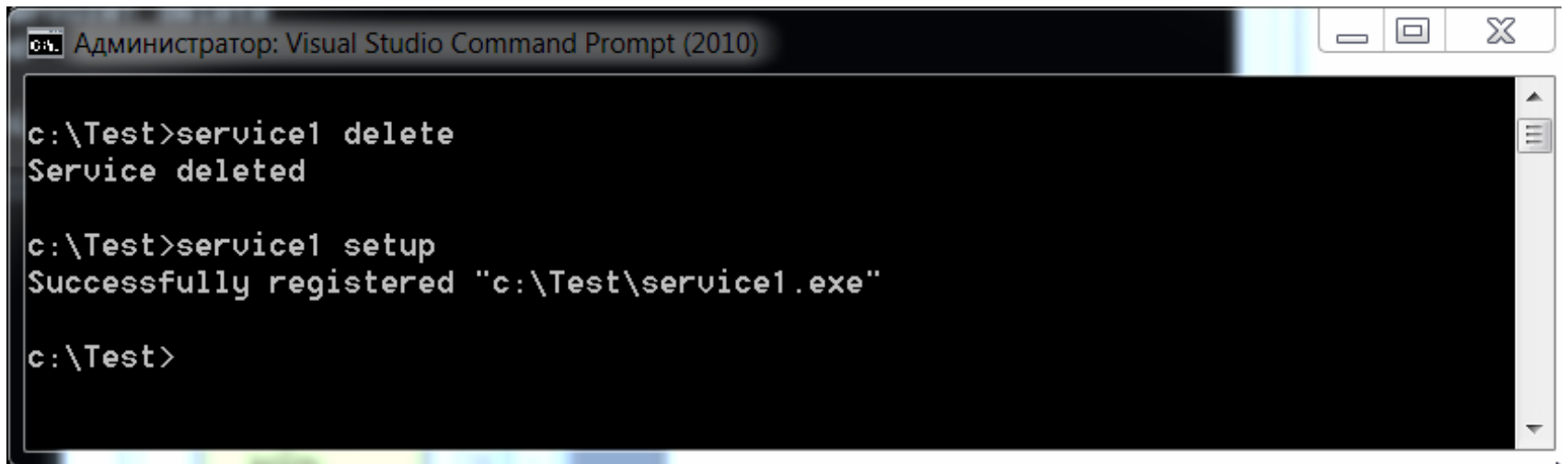
```
if (argc>1 && !strcmp(argv[1],"setup")){
    char pname[1024];
    pname[0]="";
    GetModuleFileName(NULL,pname+1,1023);
    strcat(pname,"\\");
    SC_HANDLE scm= OpenSCManager(NULL,NULL,
        SC_MANAGER_CREATE_SERVICE),svc;
    if (!scm) {
        cout<<"Can't open SCM\n";
        exit(1);
    }
    if (!(svc=CreateService(scm,"wserv_test","wserv_test",
        SERVICE_ALL_ACCESS, SERVICE_WIN32_OWN_PROCESS,
        SERVICE_DEMAND_START, SERVICE_ERROR_NORMAL, pname,
        NULL,NULL,NULL,NULL,NULL))) {
        cout<<"Registration error!\n";
        exit(2);
    }
    cout<<"Successfully registered "<<pname<<"\n";
    CloseServiceHandle(svc); CloseServiceHandle(scm);
    exit(0);
}
```

Программное удаление службы

```
if (argc>1 && !strcmp(argv[1],"delete")){
    SC_HANDLE scm=OpenSCManager(NULL,NULL,
                                SC_MANAGER_CREATE_SERVICE);

    if (!scm) {
        cout<<"Can't open SCM\n";
        exit(1);
    }
    SC_HANDLE svc=OpenService(scm,"wserv_test",DELETE);
    if (!svc){
        cout<<"Can't open service\n";
        exit(2);
    }
    if (!DeleteService(svc)){
        cout<<"Can't delete service\n";
        exit(3);
    }
    cout<<"Service deleted\n";
    CloseServiceHandle(svc);
    CloseServiceHandle(scm);

    exit(0);
}
```



```
c:\Test>service1 delete
Service deleted

c:\Test>service1 setup
Successfully registered "c:\Test\service1.exe"

c:\Test>
```

Код точки входа в службу: инициализация полей структуры SERVICE_STATUS, регистрация обработчика управления

```
void __stdcall wserv_testStart (DWORD argc, LPTSTR *argv) {
    DWORD status;

    wserv_testStatus.dwServiceType      = SERVICE_WIN32;
    wserv_testStatus.dwCurrentState     = SERVICE_START_PENDING;
    wserv_testStatus.dwControlsAccepted = SERVICE_ACCEPT_STOP |
                                           SERVICE_ACCEPT_PAUSE_CONTINUE;
    wserv_testStatus.dwWin32ExitCode    = 0;
    wserv_testStatus.dwServiceSpecificExitCode = 0;
    wserv_testStatus.dwCheckPoint       = 0;
    wserv_testStatus.dwWaitHint         = 0;

    wserv_testStatusHandle = RegisterServiceCtrlHandler(TEXT("wserv_test"),
                                                         CtrlHandler);
    if (wserv_testStatusHandle == (SERVICE_STATUS_HANDLE)0)
        return;
```

Код точки входа в службу: обновление полей структуры SERVICE_STATUS, посылка сообщения SCM, выполнение кода службы.

```
wserv_testStatus.dwCurrentState    = SERVICE_RUNNING;
wserv_testStatus.dwCheckPoint      = 0;
wserv_testStatus.dwWaitHint        = 0;
if (!SetServiceStatus (wserv_testStatusHandle, &wserv_testStatus)){
    status = GetLastError();
}
/////////////////////// Полезный код службы /////////////////////////
FILE* fp;
SYSTEMTIME stSystemTime;
while (wserv_testStatus.dwCurrentState!=SERVICE_STOPPED){
    if (wserv_testStatus.dwCurrentState!=SERVICE_PAUSED){
        GetSystemTime(&stSystemTime);
        fp=fopen("c:\\Test\\serv_log.txt", "a");
        fprintf(fp,"%d:%d:%d\\n",stSystemTime.wHour,
                stSystemTime.wMinute, stSystemTime.wSecond);
        fclose(fp);
    }Sleep(5000);
}
///////////////////////
return;
}
```

Функция обработки управления службой.

```
VOID __stdcall CtrlHandler (DWORD Opcode) {
    DWORD status;
    switch(Opcode) {
        case SERVICE_CONTROL_PAUSE:
            wserv_testStatus.dwCurrentState = SERVICE_PAUSED;
            break;
        case SERVICE_CONTROL_CONTINUE:
            wserv_testStatus.dwCurrentState = SERVICE_RUNNING;
            break;
        case SERVICE_CONTROL_STOP:
            wserv_testStatus.dwWin32ExitCode = 0;
            wserv_testStatus.dwCurrentState = SERVICE_STOPPED;
            wserv_testStatus.dwCheckPoint = 0;
            wserv_testStatus.dwWaitHint = 0;
            if (!SetServiceStatus (wserv_testStatusHandle, &wserv_testStatus)){
                status = GetLastError();
            }
            return;
        default:
            break;
    }
}
```

```
if (!SetServiceStatus (wserv_testStatusHandle, &wserv_testStatus)) {  
    status = GetLastError();  
}  
return;  
}
```