Лекция 7. Службы

Архитектура ОС Windows

11 ноября 2014 г.

Введение

Виды служб

- Приложение;
- Драйвер.

Компоненты служб

- Программа служб;
- Программа управления службой (Service Control Program, SCP).
- Программа настройки (конфигурирования) служб (Service Configuration Program).
- Диспетчер управления службами (Service Control Manager, SCM).

Диспетчер служб (SCM, `Services.exe`)

Функции диспетчера служб

- Поддержка базы данных установленных служб.
- Запуск служб (при загрузке/по требованию).
- Перечисление установленных служб.
- Поддержка информации о состоянии работающих служб.
- Передача управляющих запросов к работающим службам.
- Блокировка/разблокировка базы данных служб.

База данных диспетчера служб

Элементы записи службы

- Имя (ключ).
- Тип:
  - в отдельном процесе;
  - в общем;
  - драйвер ядра;
  - драйвер файловой системы.
- Тип запуска:
  - автоматически при старте системы;
  - вручную;
  - отключена.
- Уровень контроля ошибок.

Элементы записи (окончание)

- Путь к исполняемому файлу (`.exe`, `.sys`).
- Необязательная информация о зависимостях.
- Для приложений: необязательные имя и пароль учётной записи (по умолчанию, `LocalSystem`).
- Для драйверов: необязательное имя объекта (по умолчанию на основе имени службы).

Обновление информации диспетчера служб

Способы обновления информации о работающих службах

- Для приложений: явное оповещение диспетчера (`SetServiceStatus()`).
- Для драйверов: опрос системы ввода/вывода со стороны SCM.

Структура приложения службы

Реализация службы

- Точка входа службы (`main()`, `wmain()`, `WinMain()`, `wWinMain()`).
- Функция обратного вызова `ServiceMain()`.
- Обработчик управляющих уведомлений.

Функции OpenSCManager() и CloseServiceHandle()

Определение OpenSCManager()

```
SC_HANDLE WINAPI OpenSCManager(
  _In_opt_ LPCTSTR    lpMachineName,
  _In_opt_ LPCTSTR    lpDatabaseName,
  _In_     DWORD      dwDesiredAccess
);
```

Определ. CloseServiceHandle()

```
BOOL WINAPI CloseServiceHandle(
  _In_     SC_HANDLE hSCObject
);
```

Функции OpenService() и DeleteService()

Определение OpenService()

```
SC_HANDLE WINAPI OpenService(
  _In_     SC_HANDLE hSCManager,
  _In_     LPCTSTR   lpServiceName,
  _In_     DWORD     dwDesiredAccess
);
```

Определение DeleteService()

```
BOOL WINAPI DeleteService(
  _In_     SC_HANDLE hService
);
```

Функция CreateService()

Определение функции CreateService()

```
SC_HANDLE WINAPI CreateService(
  _In_      SC_HANDLE schSCManager,
  _In_      LPCTSTR   lpctszServiceName,
  _In_opt_  LPCTSTR   lpctszDisplayName,
  _In_      DWORD     dwDesiredAccess,
  _In_      DWORD     dwServiceType,
  _In_      DWORD     dwStartType,
  _In_      DWORD     dwErrorControl,
  // ...
```

Функция CreateService() (окончание)

Определение функции CreateService() (окончание)

```
  // ...
  _In_opt_  LPCTSTR   lpctszBinaryPathName,
  _In_opt_  LPCTSTR   lpctszLoadOrderGroup,
  _Out_opt_ LPDWORD   lpdwTagId,
  _In_opt_  LPCTSTR   lpctszDependencies,
  _In_opt_  LPCTSTR   lpctszServiceStartName,
  _In_opt_  LPCTSTR   lpctszPassword
);
```

Параметры функции CreateService()

SERVICE_WIN32_OWN_PROCESS
SERVICE_WIN32_SHARE_PROCESS
SERVICE_KERNEL_DRIVER
SERVICE_FILE_SYSTEM_DRIVER

Таблица 1: значения dwServiceType

SERVICE_AUTO_START
SERVICE_DEMAND_START
SERVICE_DISABLED
SERVICE_BOOT_START
SERVICE_SYSTEM_START

Таблица 2: значения dwStartType

SERVICE_ERROR_IGNORE
SERVICE_ERROR_NORMAL
SERVICE_ERROR_SEVERE
SERVICE_ERROR_CRITICAL

Таблица 3: значения dwErrorControl

Функция StartServiceCtrlDispatcher()

Определение функции StartServiceCtrlDispatcher()

```
BOOL WINAPI StartServiceCtrlDispatcher(
  _In_  const SERVICE_TABLE_ENTRY *lpServiceTable
);
```

Определение структуры SERVICE_TABLE_ENTRY

```
typedef struct _SERVICE_TABLE_ENTRY
{
  LPTSTR                          lptszServiceName;
  LPSERVICE_MAIN_FUNCTION         lpfnServiceProc;
}
  SERVICE_TABLE_ENTRY, *LPSERVICE_TABLE_ENTRY;
```

Функция RegisterServiceCtrlHandler()

Определение функции RegisterServiceCtrlHandler()

```
SERVICE_STATUS_HANDLE WINAPI RegisterServiceCtrlHandler(
  _In_      LPCTSTR                 lpctszServiceName,
  _In_      LPHANDLER_FUNCTION      lpfnHandlerProc
);
```

Определение функции RegisterServiceCtrlHandlerEx()

```
SERVICE_STATUS_HANDLE WINAPI RegisterServiceCtrlHandlerEx(
  _In_      LPCTSTR                 lpctszServiceName,
  _In_      LPHANDLER_FUNCTION_EX   lpHandlerProc,
  _In_opt_  LPVOID                  lpvContext
);
```

Функция SetServiceStatus()

Определение функции SetServiceStatus()

```
BOOL WINAPI SetServiceStatus(
  _In_    SERVICE_STATUS_HANDLE hServiceStatus,
  _In_    LPSERVICE_STATUS      lpServiceStatus
);
```

Структура SERVICE_STATUS

Определение структуры SERVICE_STATUS

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

Параметры функции SetServiceStatus()

SERVICE_START_PENDING
SERVICE_RUNNING
SERVICE_PAUSE_PENDING
SERVICE_PAUSED
SERVICE_CONTINUE_PENDING
SERVICE_STOP_PENDING
SERVICE_STOPPED

Таблица 4: значения dwCurrentState

SERVICE_ACCEPT_STOP
SERVICE_ACCEPT_PAUSE_CONTINUE
SERVICE_ACCEPT_PARAMCHANGE
SERVICE_ACCEPT_SHUTDOWN
...

Таблица 5: значения dwControlsAccepted

Управляющие сообщения обработчика службы

SERVICE_CONTROL_INTERROGATE
SERVICE_CONTROL_STOP
SERVICE_CONTROL_PAUSE
SERVICE_CONTROL_CONTINUE
SERVICE_CONTROL_PARAMCHANGE
SERVICE_CONTROL_SHUTDOWN
...

Таблица 6: значения параметра `dwControl` обработчика сообщений

Функции RegisterEventSource() и ReportEvent()

Определение RegisterEventSource()

```
HANDLE RegisterEventSource(
  _In_     LPCTSTR lpUNCServerName,
  _In_     LPCTSTR lpSourceName
);
```

Определение DeregisterEventSource()

```
BOOL DeregisterEventSource(
  _Inout_  HANDLE  hEventLog
);
```

Определение ReportEvent()

```
BOOL ReportEvent(
  _In_     HANDLE     hEventLog,
  _In_     WORD       wType,
  _In_     WORD       wCategory,
  _In_     DWORD      dwEventID,
  _In_     PSID       lpUserSid,
  _In_     WORD       wNumStrings,
  _In_     DWORD      dwDataSize,
  _In_     LPCTSTR *  plpStrings,
  _In_     LPVOID     lpRawData
);
```

Точка входа в программу

Правила написания точки входа

- Служба типа `SERVICE_WIN32_OWN_PROCESS` должна немедленно вызвать `StartServiceCtrlDispatcher()`.
- Служба типа `SERVICE_WIN32_SHARE_PROCESS` может выполнить инициализацию до `StartServiceCtrlDispatcher()`, если это займёт до 30 с (иначе в другом потоке).

Поведение `StartServiceCtrlDispatcher()`

- Создание отдельного потока для каждой службы с вызовом её главной функции.
- Вызов обработчика для обработки управляющих событий.
- Функция не завершается до перехода в `SERVICE_STOPPED`.

Основная функция службы

Алгоритм работы функции службы

1. Проинициализировать глобальные переменные
2. Вызвать `RegisterServiceCtrlHandler()`. Выполнить инициализацию, если это займёт до 30 с (иначе в другом потоке).
3. Выполнить инициализацию менее, чем за 1 с. Иначе одно из:
   - Перейти в `SERVICE_RUNNING` с `dwControlsAccepted == 0`.
   - Перейти в `SERVICE_START_PENDING` с `dwControlsAccepted == 0`, установить `dwWaitHint` и вызывать периодически `SetServiceStatus()` (при прогрессе).
4. Перейти в `SERVICE_RUNNING` с $\sim$ флагами в `dwControlsAccepted`.
5. Выполнить нужные действия и завершиться, при необходимости меняя состояния.
6. В случае ошибки перейти в `SERVICE_STOP_PENDING`, если освобождение ресурсов будет длительным. Затем — в `SERVICE_STOPPED` из последнего потока, установив `dwServiceSpecificExitCode` и `dwWin32ExitCode`.

Пример приложения службы

Пример (srv_program.cpp)

```
#include <windows.h>
#include <tchar.h>

#include <iostream>
#include <cstdio>
#include <cstdlib>

using namespace std;

TCHAR g_tszServiceName[] = _T("My Dummy Server");
SERVICE_STATUS g_SeviceStatus;
SERVICE_STATUS_HANDLE g_hServiceStatusHandle;
HANDLE g_hServiceStopEvent = NULL;
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
void MySvcError(LPCTSTR lpctszFunction)
{
  HANDLE hEventSource = RegisterEventSource(
    NULL, g_tszServiceName);
  if (NULL == hEventSource)
    return;
  //
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
static TCHAR s_tszBuffer[1000];
_sntprintf(
  s_tszBuffer,
  sizeof (s_tszBuffer),
  _T("%s() failed with %d"),
  lpctszFunction,
  GetLastError());
LPCTSTR alpszStrings[] =
{
  g_tszServiceName,
  s_tszBuffer
};
//
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
  ReportEvent(
    hEventSource,         // дескриптор журнала
    EVENTLOG_ERROR_TYPE, // тип события
    0,                    // категория события
    999,                  // идентификатор события
    NULL,                 // идентификатор безопасности
    2,                    // размер alpszStrings
    0,                    // нет двоичных данных
    alpszStrings,         // массив строк
    NULL);                // нет двоичных данных
  //
  DeregisterEventSource(hEventSource);
}    // MySvcError()
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
SC_HANDLE MyOpenSCManager()
{
  SC_HANDLE schManager = OpenSCManager(
    NULL,                    // локальная система
    NULL,                    // база ServicesActive
    SC_MANAGER_ALL_ACCESS);  // полные права
  if (NULL == schManager)
  {
    cerr << "OpenSCManager() failed: " << GetLastError() << endl;
    exit(1);
  }
  return schManager;
}    // MyOpenSCManager()
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
void MySvcInstall()
{
  static TCHAR s_tszModulePath[65536];
  DWORD dwResult = GetModuleFileName(
    NULL, s_tszModulePath,
    sizeof (s_tszModulePath) / sizeof (TCHAR));
  if (0 == dwResult)
  {
    cerr << "Cannot get the module path: " << GetLastError() << endl;
    return;
  }
  //
  SC_HANDLE schManager = MyOpenSCManager();
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
SC_HANDLE schService = CreateService(
  schManager,                  // база SCM
  g_tszServiceName,            // имя службы
  g_tszServiceName,            // отображаемое имя
  SERVICE_ALL_ACCESS,          // права доступа
  SERVICE_WIN32_OWN_PROCESS,   // тип службы
  SERVICE_DEMAND_START,        // тип запуска
  SERVICE_ERROR_NORMAL,        // тип обработки ошибок
  s_tszModulePath,             // путь к файлу
  NULL,                        // группа порядка загрузки
  NULL,                        // тег в группе
  NULL,                        // зависимости
  NULL,                        // учётная запись LocalSystem
  NULL);                       // пароль
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
  //
  if (NULL == schService)
  {
    cerr << "CreateService() failed: " << GetLastError() << endl;
    CloseServiceHandle(schManager);
    return;
  }
  //
  cout << g_tszServiceName << " installed successfully" << endl;
  CloseServiceHandle(schService);
  CloseServiceHandle(schManager);
}    // MySvcInstall()
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
void MySvcUninstall()
{
  SC_HANDLE schManager = MyOpenSCManager();
  SC_HANDLE schService = OpenService(
    schManager,
    g_tszServiceName,
    SC_MANAGER_ALL_ACCESS);
  if (NULL == schService)
  {
    cerr << "OpenService() failed: " << GetLastError() << endl;
    CloseServiceHandle(schManager);
    return;
  }
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
  //
  BOOL bSuccess = DeleteService(schService);
  if (bSuccess)
    cout << g_tszServiceName << " uninstalled successfully" << endl;
  else
    cerr << "DeleteService() failed: " << GetLastError() << endl;
  //
  CloseServiceHandle(schService);
  CloseServiceHandle(schManager);
}    // MySvcUninstall()
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
void MySvcReport(
  DWORD dwCurrentState, DWORD dwWin32ExitCode, DWORD dwWaitHint)
{
  static DWORD dwCheckPoint = 1;
  //
  g_SeviceStatus.dwCurrentState = dwCurrentState;
  g_SeviceStatus.dwWin32ExitCode = dwWin32ExitCode;
  g_SeviceStatus.dwWaitHint = dwWaitHint;
  g_SeviceStatus.dwControlsAccepted =
    dwCurrentState == SERVICE_START_PENDING ?
    0 : SERVICE_ACCEPT_STOP;
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
  g_SeviceStatus.dwCheckPoint =
    dwCurrentState == SERVICE_RUNNING ||
    dwCurrentState == SERVICE_STOPPED ?
    0 : dwCheckPoint ++;
  //
  SetServiceStatus(
    g_hServiceStatusHandle,
    &g_SeviceStatus);
}    // MySvcReport()
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
DWORD WINAPI MySvcCtrlHandlerEx(
  DWORD dwControl, DWORD dwEventType,
  LPVOID lpvEventData, LPVOID lpvContext)
{
  switch (dwControl)
  {
    case SERVICE_CONTROL_STOP:
      //
      MySvcReport(SERVICE_STOP_PENDING, NO_ERROR, 0);
      SetEvent(g_hServiceStopEvent);
      MySvcReport(g_SeviceStatus.dwCurrentState, NO_ERROR, 0);
      return NO_ERROR;
      //
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
    case SERVICE_CONTROL_INTERROGATE:
      //
      return NO_ERROR;
  }    // switch (dwControl)
  //
  return ERROR_CALL_NOT_IMPLEMENTED;
}    // MySvcCtrlHandler()
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
VOID WINAPI MySvcMain(DWORD dwArgc, LPTSTR *plpszArgv)
{
  g_hServiceStatusHandle = RegisterServiceCtrlHandlerEx(
    g_tszServiceName,
    &MySvcCtrlHandlerEx,
    NULL);
  //
  if (0 == g_hServiceStatusHandle)
  {
    MySvcError(_T("RegisterServiceCtrlHandlerEx"));
    return;
  }
  //
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
g_SeviceStatus.dwServiceType = SERVICE_WIN32_OWN_PROCESS;
g_SeviceStatus.dwServiceSpecificExitCode = 0;
//
MySvcReport(SERVICE_START_PENDING, NO_ERROR, 3000);
//
// Инициализация и работа службы
//
g_hServiceStopEvent = CreateEvent(
  NULL,     // атрибуты безопасности по умолчанию
  TRUE,     // ручной сброс
  FALSE,    // не сигнализировано
  NULL);    // без имени
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
  //
  if (NULL == g_hServiceStopEvent)
  {
    MySvcReport(SERVICE_STOPPED, NO_ERROR, 0);
    return;
  }
  //
  MySvcReport(SERVICE_RUNNING, NO_ERROR, 0);
  WaitForSingleObject(g_hServiceStopEvent, INFINITE);
  MySvcReport(SERVICE_STOPPED, NO_ERROR, 0);
}    // MySvcMain()
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
int main(int nArgC, char *apszArgV[])
{
  if (nArgC > 1)
  {
    if (lstrcmpiA(apszArgV[1], "install") == 0)
    {
      MySvcInstall();
      return 0;
    }
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
    if (lstrcmpiA(apszArgV[1], "uninstall") == 0)
    {
      MySvcUninstall();
      return 0;
    }
  }    // if (nArgC > 1)
  //
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
SERVICE_TABLE_ENTRY aDispatchTable[] =
{
  {
    g_tszServiceName,
    &MySvcMain
  },
  {
    NULL,
    NULL
  }
};
//
BOOL bSuccess = StartServiceCtrlDispatcher(aDispatchTable);
```

Пример приложения службы (продолжение)

Пример (srv_program.cpp, продолжение)

```
  //
  // Возврат по завершении службы
  //
  if (!bSuccess)
    if (GetLastError() ==
      ERROR_FAILED_SERVICE_CONTROLLER_CONNECT)
      cerr << "Running as an application" << endl;
    else
      MySvcError(_T("StartServiceCtrlDispatcher"));
}    // main()

// Конец файла
```

Управление службой

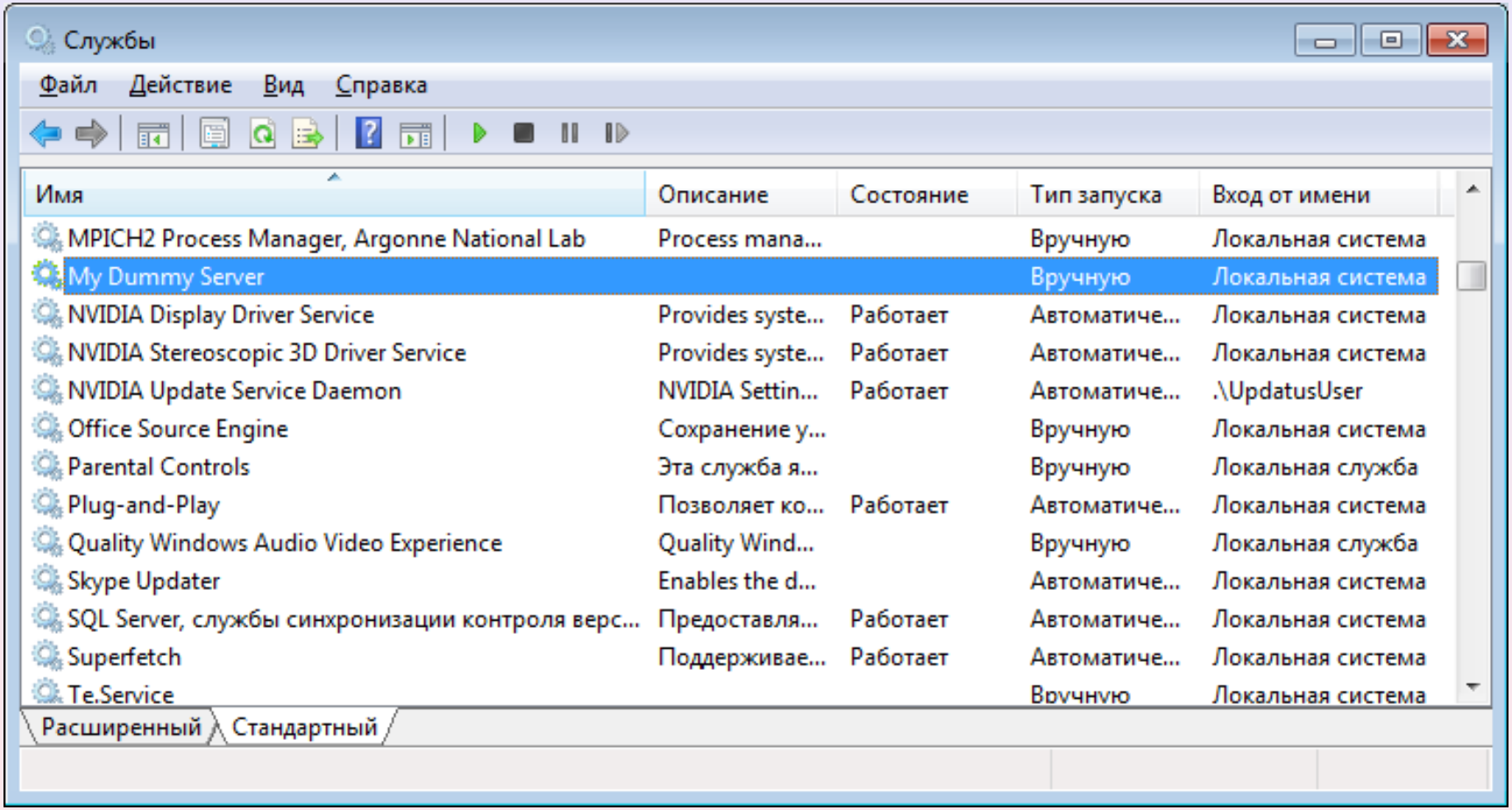

Рис. 1: окно управления службами Windows

События службы

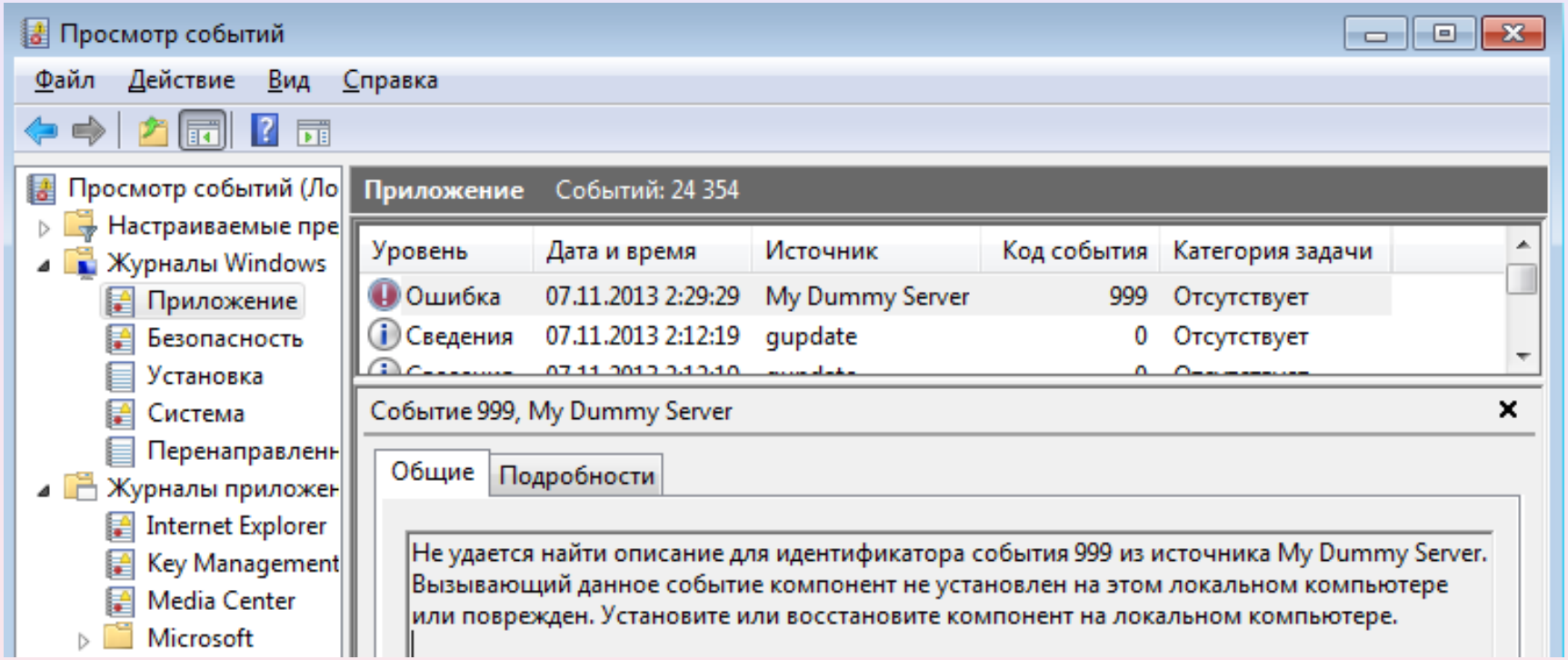

Рис. 2: окно просмотра событий Windows

Реализация управления и конфигурации служб

Управление службой

- StartService();
- EnumDependentServices();
- ControlService();
- QueryServiceStatusEx();

Конфигурация службы

- QueryServiceConfig();
- QueryServiceConfig2();
- ChangeServiceConfig();
- ChangeServiceConfig2();

Использование утилиты управления службами

Пример

```
d:\>sc query "My Dummy Server"

SERVICE_NAME: My Dummy Server
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 1077  (0x435)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```