# Основы Native API

# Native приложения

- Программы, предназначенные для выполнения на операционных системах Windows, способные запускаться на раннем этапе загрузки Windows, до окна входа в систему и даже до запуска каких-либо подсистем Windows.
- Экран при загрузке Windows, в котором, например, происходит проверка диска и есть тот самый режим.
- Native приложения используют только Native API, они могут использовать только функции, экспортируемые из библиотеки ntdll.dll. Для них недоступны функции WinAPI.

# Native приложения

- Native приложения запускаются на экране, который возникает до появления окна входа в систему.
- Примером native приложения является приложение chkdsk, которое запускается перед входом в Windows, если предварительно была запущена проверка системного раздела на ошибки и отложена до перезагрузки.
- Приложение работает, выводя сообщения экран, а затем происходит обычный запуск Windows.



- Преимущества использования этого режима: большая часть компонентов Windows ещё не запущена, отсутствуют многие ограничения. Этот режим, например, используется в приложениях, которые хотят что-то сделать с системным разделом Windows, но не могут, пока запущена операционная система: дефрагментаторы, конверторы файловой системы, и тому подобные утилиты.

# Что нужно знать:

- Native приложения компилируются с помощью WDK - *Windows Driver Kit* (также известный, как DDK). Есть возможность делать их и в какой-то другой среде разработки, но в WDK проще всего.
- Native приложения используют *Native API*. Оно частично документировано в MSDN для использования при написании драйверов. Но документированы не все функции. Информацию по остальным нужно брать из неофициальных источников. Например, на сайте http://undocumented.ntinternals.net/
- Функции в *ntdll.dll* имеют префиксы **Zw** и **Nt**, а также некоторые другие. Видно, что у Zw и Nt функции дублируются названия. На самом деле это одни и те же функции. Если искать в сети пример использования какой-либо функции, стоит поискать сначала с одним префиксом, потом с другим, иначе можно что-то упустить.

# Что нужно знать:

- Функции в *ntdll.dll* имеют префиксы **Zw** и **Nt**, а также некоторые другие. Видно, что у Zw и Nt функции дублируются названия. На самом деле это одни и те же функции.
- Если искать в сети пример использования какой-либо функции, стоит поискать сначала с одним префиксом, потом с другим, иначе можно что-то упустить.
- Для программирования нужны прототипы функций Native API, но в заголовочных файлах WDK присутствуют не все определения.
- Нужно использовать альтернативные заголовочные файлы, содержащие в том числе и определения недокументированных функций и типов данных.

# Что нужно знать:

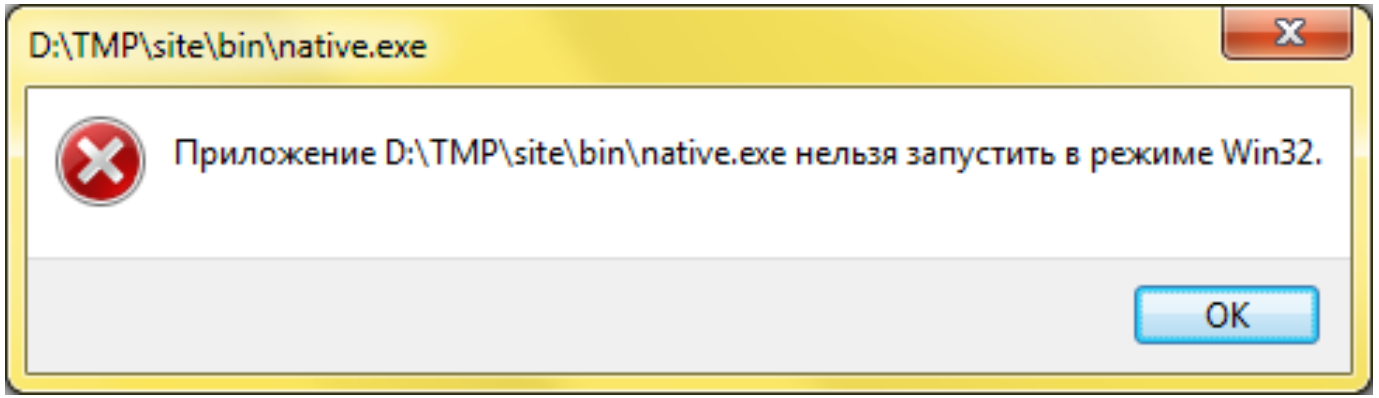
- Программировать на чистом Native API неудобно. Не обойтись без библиотеки, в которой уже реализованы некоторые рутинные действия. Существует библиотека с открытым кодом - ZenWINX, можно пользоваться ей.
- Чтобы native приложение запустилось при запуске Windows, надо положить его в каталог system32, а в ключ реестра HKLM\System\CurrentControlSet\Control\Session Manager\BootExecute прописать его имя файла, и аргументы, если они есть.
- Ключ имеет тип MULTI_SZ, может содержать несколько строк.

# Что нужно знать:

- Программа, прописанная в этом ключе, имеет свойство запускаться даже в безопасном режиме Windows (safe mode), так что нужно быть осторожным. Ошибка в программе - и система не запустится.
- Но можно внутри приложения отслеживать факт запуска в safe mode и обрабатывать этот режим отдельно, например сделать завершение программы, если она обнаружила себя запущенной в safe mode.
- Кроме того, несмотря на то, что программа запускается и может выполнять какие-то действия, в этом режиме не работает вывод на консоль. Невозможно взаимодействие с пользователем. Это следует учитывать.

# Точка входа

- У native приложений точка входа не main и не wmain, а **NtProcessStartup**. В PE-заголовке EXE-файла есть специальное поле, означающее подсистему, в которой выполняется приложение.
- У native приложений в это поле установлено специальное значение, означающее, что EXE не требует подсистемы. У обычных приложений ставится значение, соответствующее подсистемам "Windows GUI" или "Windows console".
- Native приложения не запускаются в обычном режиме работы Windows.

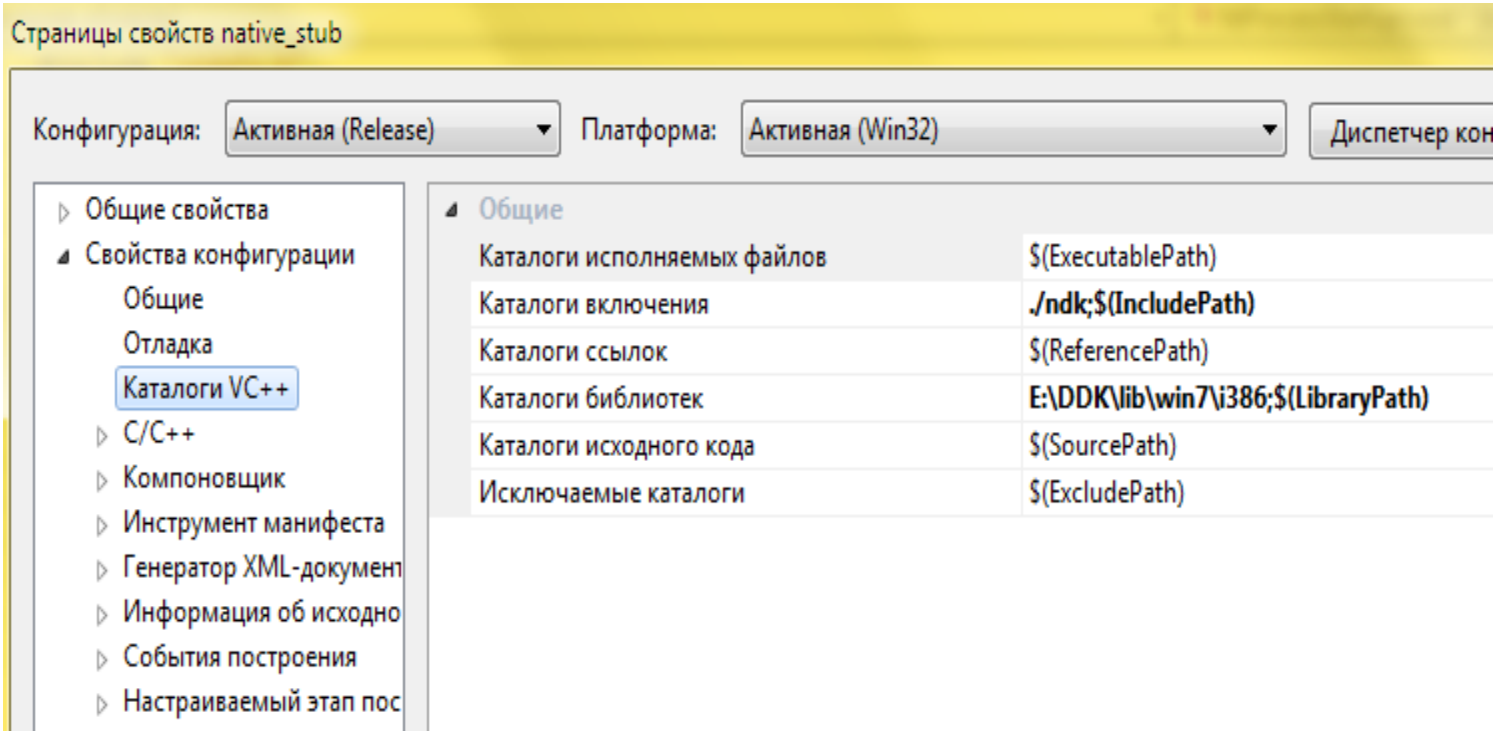


- Разработка **Native API приложений** обычно происходит с помощью компилятора, входящего в состав Windows Driver Kit (WDK). Однако, можно обойтись и без WDK. Visual Studio тоже способна создавать приложения, не требующие функционирования подсистемы Win32 для запуска.
- Для этого понадобится сама Visual Studio, заголовочные файлы и файл **ntdll.lib**, который придётся достать (из WDK или из сети ☺).

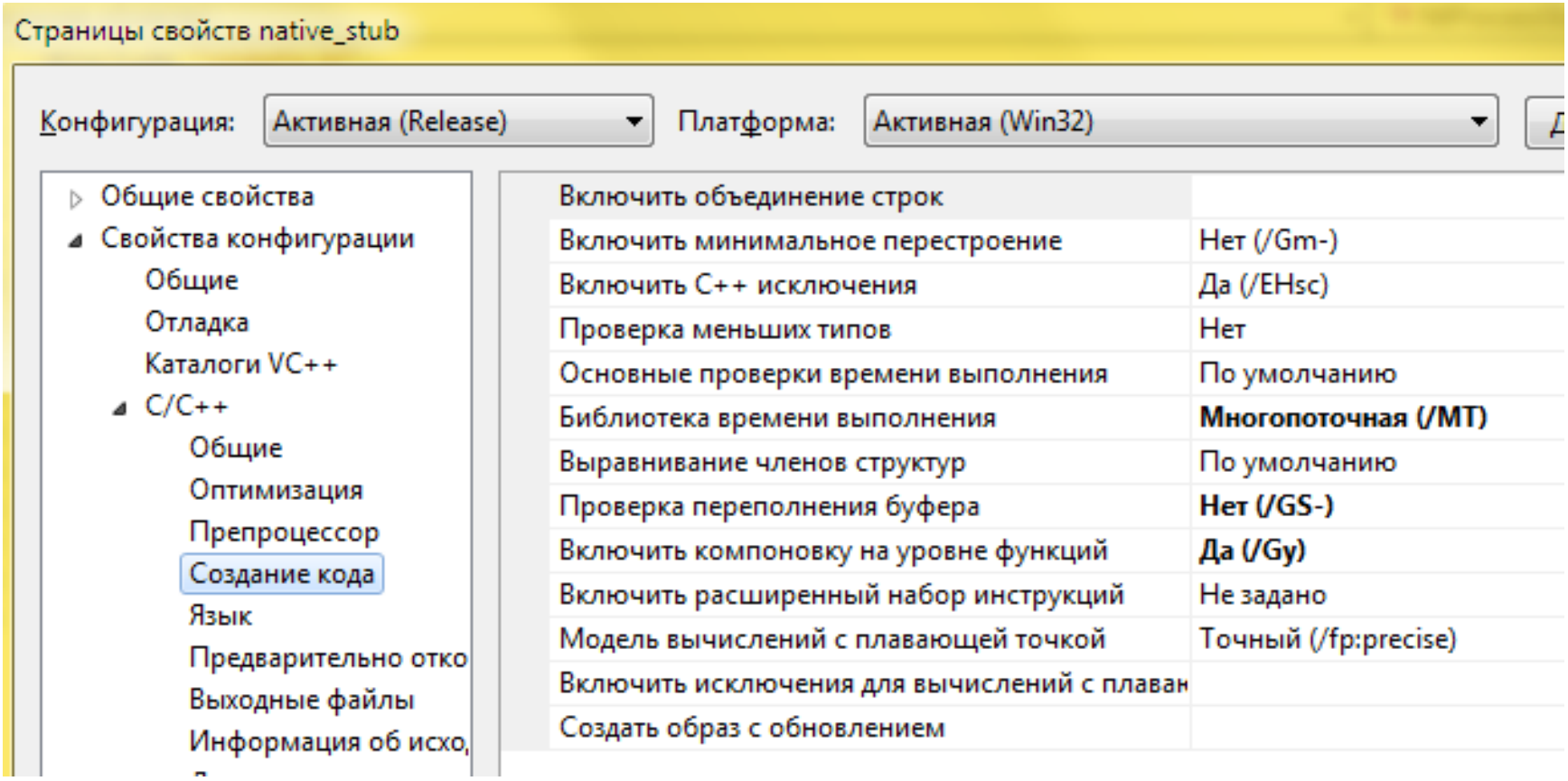
# Настройка проекта Visual Studio

- В Visual Studio нужно создать новый проект — консольное приложение.
- Это оптимальный тип проекта для переделки в Native-приложение.
- В каталог с проектом или в иное место нужно положить заголовочные файлы .

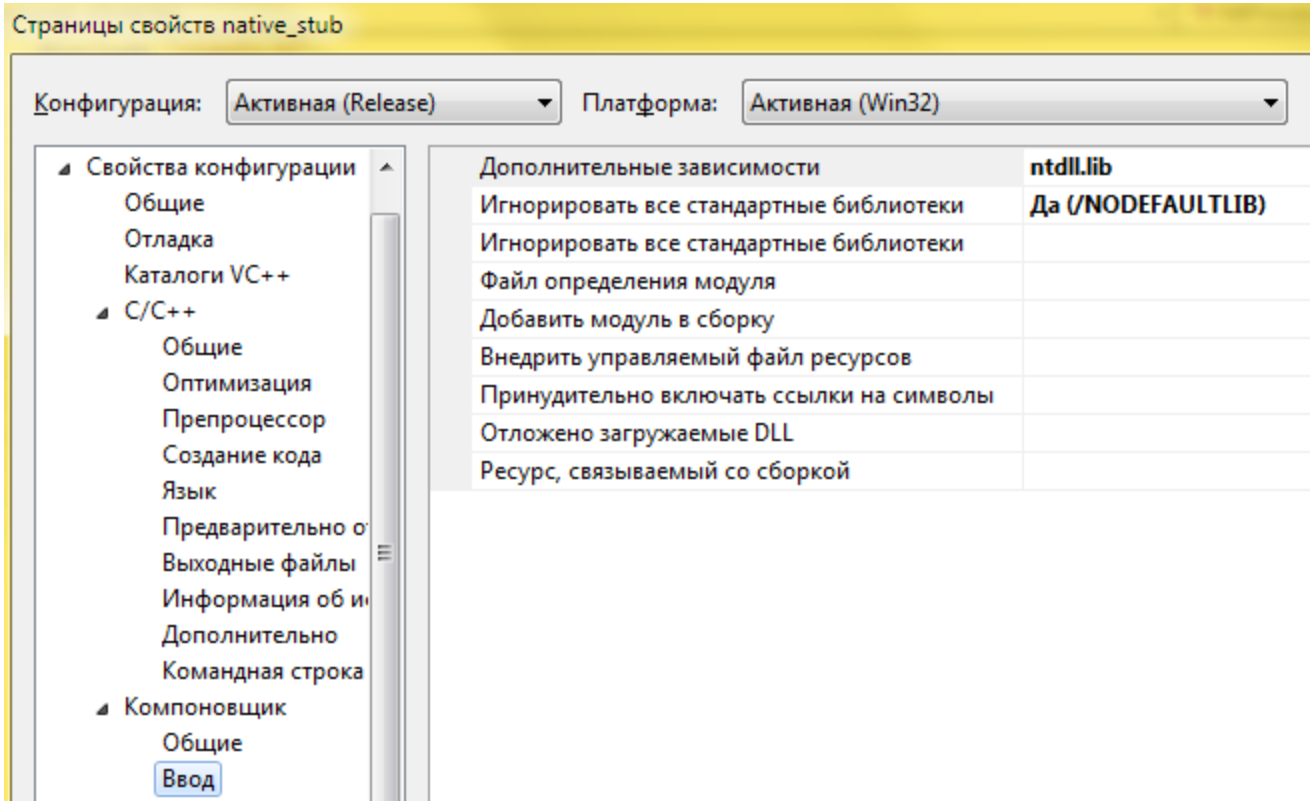
В настройках проекта во вкладке «Свойства конфигурации › Каталоги VC++» нужно добавить путь к NDK в пункт «Каталоги включения», и путь к ntdll.lib в пункте «Каталоги библиотек».



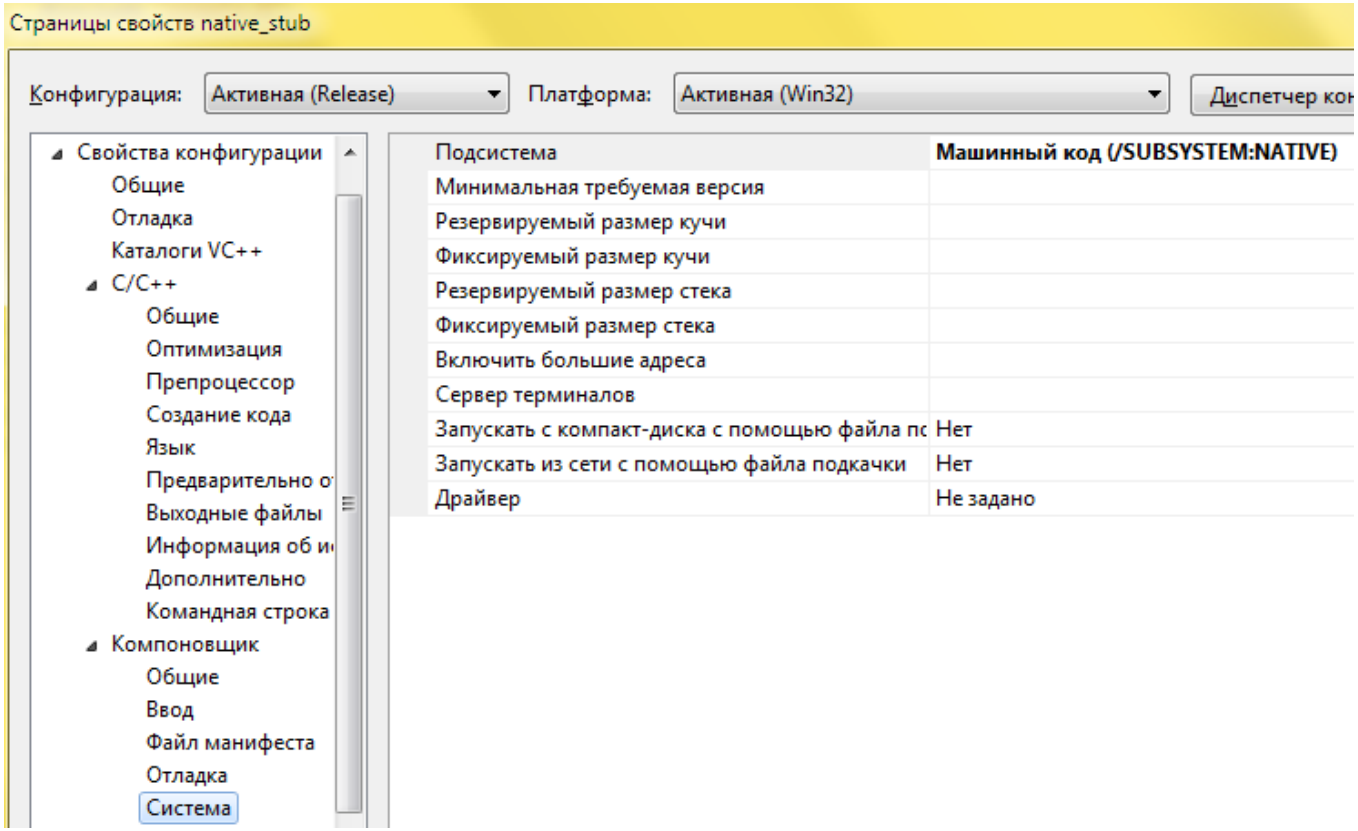
- Запуск Native-приложения в отладчике Visual Studio, да и вообще, при работающей системе невозможен. Так что сразу можно выбирать тип сборки проекта «Release».
- На вкладке «C/C++ › Создание кода» нужно отключить проверку переполнения буфера (опция должна быть выставлена в /GS-). Если этого не сделать, то при сборке Native-приложения возникнет ошибка:



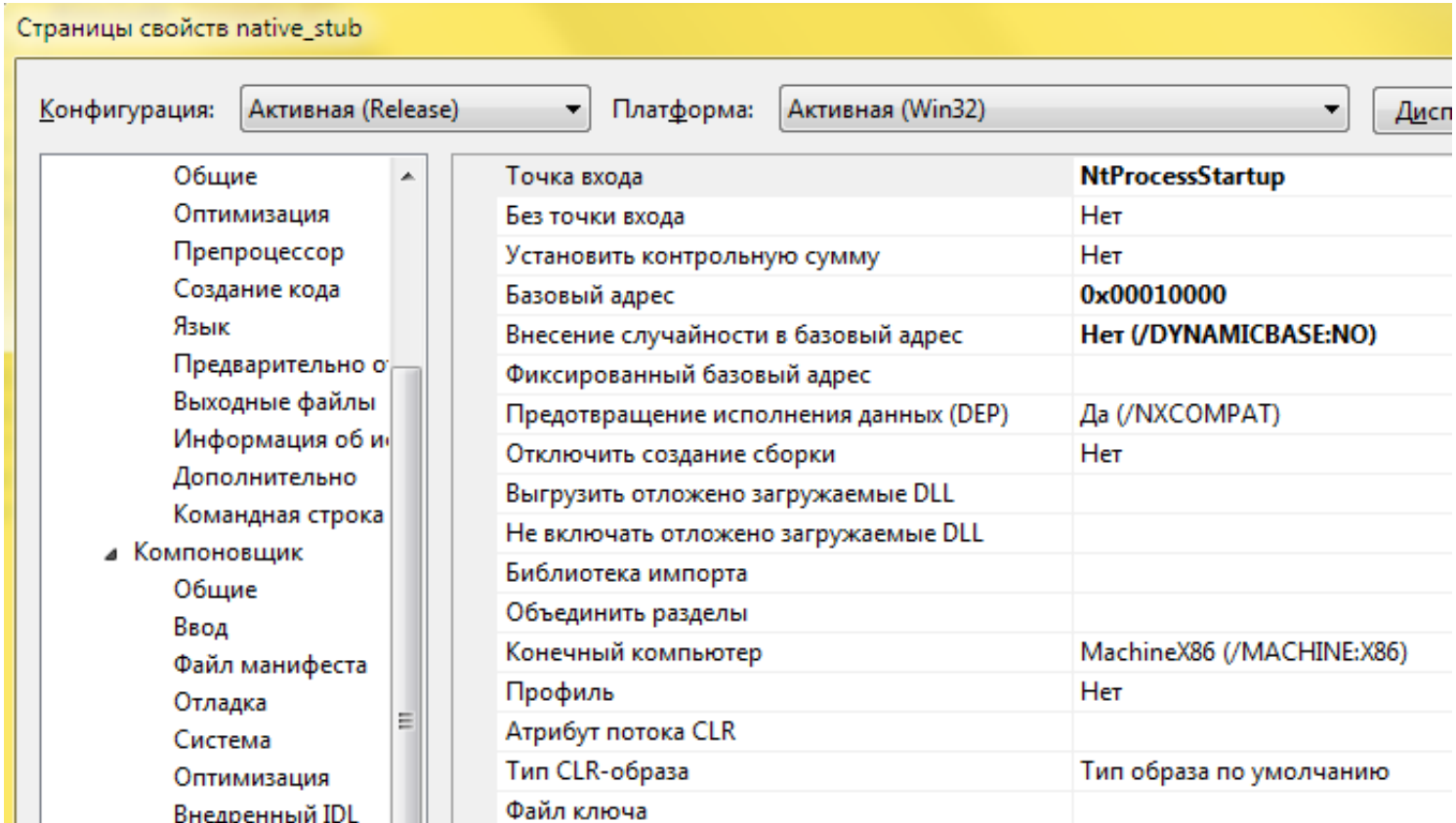
- На вкладке «Компоновщик › Ввод» в пункте «Дополнительные зависимости» нужно убрать всё, что содержится в нём и добавить только одну зависимость: ntdll.lib.
- В пункте «Игнорировать все стандартные библиотеки» нужно выбрать «Да». Native-приложение не может позволить себе иметь зависимость от других библиотек, кроме ntdll, по причине того, что все остальные библиотеки рассчитаны на работу в рамках подсистемы Win32.
- Native-приложение работает вне этой системы.



- На вкладке «Компоновщик › Система» в пункте «Подсистема» нужно выбрать «Машинный код (**/SUBSYSTEM:NATIVE**)».



- На вкладке «Компоновщик › Дополнительно» в пункте «Точка входа» нужно написать **NtProcessStartup**. Именно так обычно называется точка входа в Native-приложение, а не main и не WinMain.
- В исходном тексте, который создала Visual Studio для нашего проекта нужно убрать сгенерированную ей точку входа и написать свою.
- В пункте «Базовый адрес» нужно написать «0x00010000».



# Точка входа Native-приложения

- В файле native_stub.cpp, являющимся главным файлом проекта, должна содержаться точка входа, прописанная ранее в настройках проекта. Нужно удалить сгенерированную точку входа main и написать свою, под именем NtProcessStartup:

```
void NtProcessStartup(void* StartupArgument)
{
  UNICODE_STRING str;
  PPEB pPeb = (PPEB)StartupArgument;
  RtlNormalizeProcessParams(pPeb->ProcessParameters);
  RtlInitUnicodeString(&str, L"Hello, world!\nCommand line is: ");
  NtDisplayString(&str);
  RtlInitUnicodeString(&str, pPeb->ProcessParameters-
    >CommandLine.Buffer);
  NtDisplayString(&str);
  NtTerminateProcess(NtCurrentProcess(), 0);
}
```

- Программа выводит «Hello, world!», показывает содержимое своей командной строки и завершает своё выполнение вызовом **NtTerminateProcess**.

# Заголовочный файл проекта

- В stdafx.h нужно написать следующее:

```
#pragma once
#define WIN32_NO_STATUS
#include <windows.h>
#include <ntndk.h>
```

# Запуск

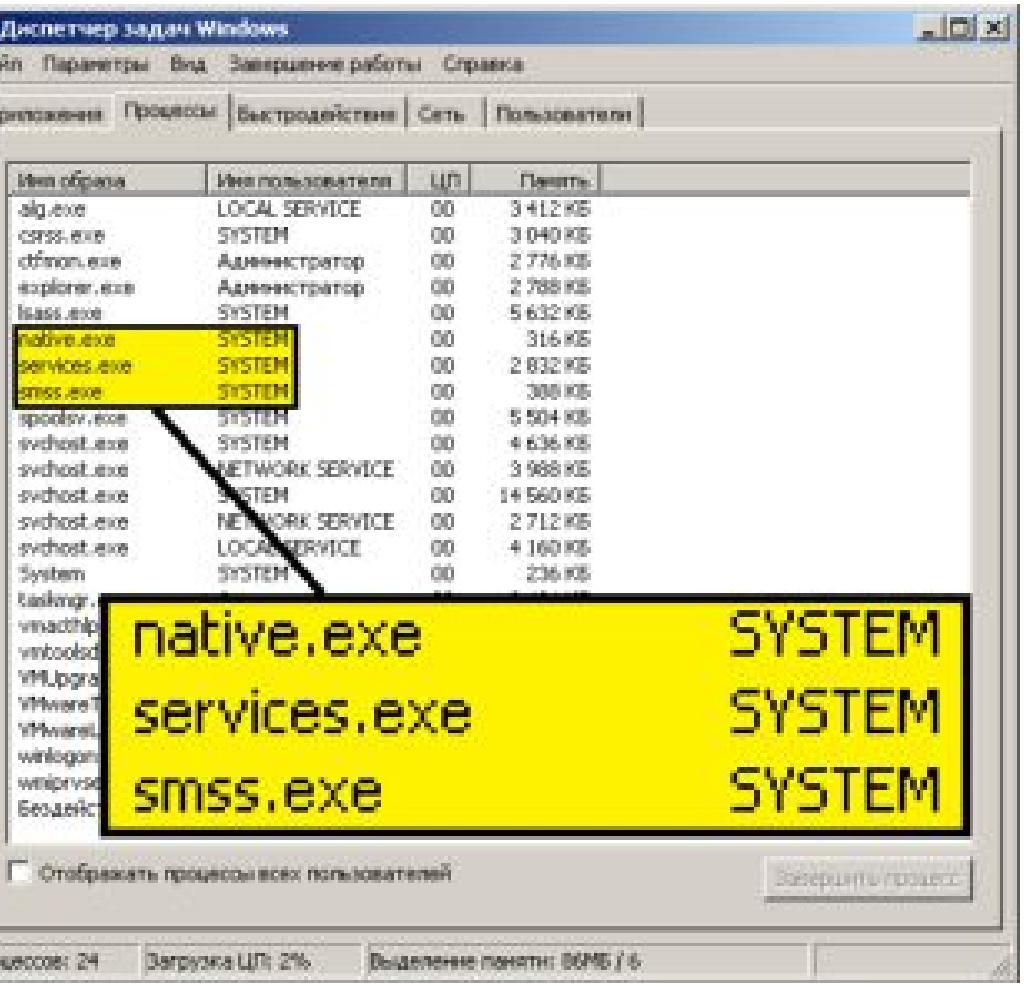
- В результате сборки проекта получается файл *.exe размером 2 Кб.
- Антивирус может ошибочно считать его руткитом (TR/Rootkit.Gen), хотя файл не содержит ничего, кроме вывода своей командной строки на экран.
- Запуск файла возможен стандартным для Native-приложений способом: через ключ реестра BootExecute.

# Запуск приложений через ключ реестра BootExecute

- **Автозапуск приложений** режима native (задаётся в ветке реестра **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager**.
- Там есть ключ, позволяющий запустить приложение на этапе загрузки системы - **BootExecute**.
- Это мультистроковый параметр, содержащий строку «autocheck autochk *». После неё можно добавить свою команду запуска. Например, можно поместить native.exe в папку %systemroot%\system32, а в BootExecute прописать строку «native». В результате native.exe запустится сразу после autochk.exe при запуске системы. Здесь же можно указать командную строку процесса, например «native -some -command». Чтобы запустить программу из любого каталога системы, нужно указать полный путь к исполняемому файлу, в обычном формате (например, C:\tmp\native.exe).

# Модификаторы режима запуска

- При указании имени native-приложения, кроме идентификатора autocheck (используется при указании autochk в этом списке) возможны идентификаторы async и debug.
- Идентификатор debug приводит к установке ProcessParameters->DebugFlags = TRUE. Идентификатор async приводит к тому, что система не ожидает завершения запускаемого процесса, и оно продолжает работать, а система в это время продолжает загрузку.
- В результате получается приложение, работающее в живой системе, отображающееся в диспетчере задач как запущенное от имени пользователя SYSTEM.




- Второй ключ реестра, через который возможен запуск, носит название **SetupExecute** и полностью аналогичен BootExecute.
- Разница между ними в том, что запуск из этих ключей происходит на разных этапах инициализации системы. На этапе запуска из SetupExecute в системе уже создан файл подкачки и инициализированы переменные среды, а на этапе BootExecute еще нет.

# Запуск процесса из приложения, использующего Native API

- При разработке приложений Native API может возникнуть необходимость запустить процесс. В native режиме невозможен запуск Win32-приложений, так как процессы подсистемы Win32 при создании требуют уведомления CSRSS о новом процессе (а он ещё неактивен).
- В native режиме возможен запуск других native приложений с помощью функции RtlCreateUserProcess.
- В параметрах функции нужно в соответствующих структурах передать полный путь к исполняемому файлу, причём в NT-формате (то есть с префиксом \??\), имя файла процесса для отображения в списке процессов и командную строку с параметрами (это строка, которой был запущен процесс, содержащая его ключи запуска).

# Запуск процесса

- Например, запускаем процесс **autochk.exe** с параметрами, находясь в каталоге C:\windows\system32.
- Тогда в RtlCreateUserProcess нужно будет передать следующие строки:
- Имя файла для отображения в списке процессов: **autochk.exe**
- Командная строка: **autochk.exe /p \??\C:**
- Полный путь: \??\C:\windows\system32\autochk.exe

- При попытке запустить не Native, а Win32 приложение произойдёт ошибка и вы увидите синий экран.
- Это происходит потому, что функция CreateNativeProcess в Native shell не проверяет подсистему запускаемого исполняемого файла.
- Запускать следует либо собственные native приложения, либо native приложения, идущие в комплекте Windows, такие как autocheck.exe, autoconv.exe, autofmt.exe, autolfn.exe.

# Информационные ссылки

1. Начальные сведения о Native API
2. Недокументированные функции NTDLL
3. WDK https://developer.microsoft.com/ru-ru/windows/hardware/windows-driver-kit
4. NTAPI Undocumented Functions
5. ZenWINX Library http://ultradefrag.sourceforge.net/doc/lib/zenwinx/