

В пользовательском режиме работают следующие виды процессов:

* системные процессы (system processes) – компоненты Windows, отвечающие за решение критически важных системных задач (т. е. аварийное завершение одного из этих процессов вызывает крах или нестабильную работу всей системы), но выполняемые в пользовательском режиме. Основные системные процессы:
  + Winlogon.exe – процесс входа в систему и выхода из неё;
  + Smss.exe (Session Manager – диспетчер сеансов) – процесс выполняет важные операции при инициализации системы (загрузка необходимых DLL, запуск процессов Winlogon и Csrss и др.), а затем контролирует работу Winlogon и Csrss;
  + Lsass.exe (Local Security Authentication Subsystem Server – сервер подсистемы локальной аутентификации) – процесс проверяет правильность введенных имени пользователя и пароля;
  + Wininit.exe – процесс инициализации системы (например, запускает процессы Lsass и Services);
  + Userinit.exe – процесс инициализации пользовательской среды (например, запускает системную оболочку – по умолчанию, Explorer.exe);
  + Services.exe (SCM, Service Control Manager – диспетчер управления службами) – процесс, отвечающий за выполнение служб – см. ниже;
* службы (сервисы, services) – приложения, работающие в фоновом режиме и не требующие взаимодействия с пользователем. Службы могут быть как частью операционной системы (например, Windows Audio – служба для работы со звуком, или Print Spooler – диспетчер печати), так и частью пользовательского приложения (например, служба СУБД SQL Server). За службы отвечает системный процесс Services.exe;
* пользовательские приложения (user applications) ¬– прикладные программы, запускаемые пользователем;
* подсистемы окружения (environment subsystems) – компоненты, предоставляющие доступ приложениям к некоторому подмножеству системных функций. Windows поддерживает две подсистемы окружения:
  + собственно Windows – при помощи данной подсистемы выполняются 32 разрядные приложения Windows (Win32), а также 16 разрядные приложения Windows (Win16), приложения MS DOS и консольные приложения (Console). За подсистему Windows отвечает системный процесс Csrss.exe и драйвер режима ядра Win32k.sys;
  + POSIX (Portable Operating System Interface for UNIX – переносимый интерфейс операционных систем UNIX) – подсистема для UNIX-приложений. Начиная с Windows Server 2003 R2 компонент, реализующий эту подсистему, называется SUA (Subsystem for UNIX-based Applications). Компонент не устанавливается в Windows по умолчанию.

Все перечисленные процессы пользовательского режима (кроме подсистемы POSIX1) для взаимодействия с модулями режима ядра используют библиотеки *Windows* *DLL* (*Dynamic* Link *Library* – динамически подключаемая библиотека). Каждая *DLL* экспортирует набор *Windows* *API* функций, которые может вызывать процесс.

*Windows* *API* (*Windows* *Application* *Programming* *Interface*, WinAPI) – это способ взаимодействия процессов пользовательского режима с модулями режима ядра. WinAPI включает тысячи функций и хорошо документирован [[10](https://intuit.ru/studies/courses/10471/1078/literature#literature.10)].

Основные *Windows* *DLL* следующие:

* Kernel32.dll – базовые функции, в том числе работа с процессами и потоками, управление памятью и вводом выводом;
* Advapi32.dll – функции, в основном связанные с управлением безопасностью и доступом к реестру;
* User32.dll – функции, отвечающие за управление окнами и их элементами в GUI приложениях (Graphical User Interface – графический интерфейс пользователя);
* Gdi32.dll – функции графического пользовательского интерфейса (Graphics Device Interface, GDI), обеспечивающие рисование на дисплее и принтере графических примитивов и вывод текста.

Библиотека Ntdll.*dll* экспортирует в большинстве своем недокументированные системные функции, реализованные, в основном, в Ntoskrnl.exe. Набор таких функций называется *Native* *API* ("родной" *API*).

Библиотеки *Windows* *DLL* преобразуют вызовы документированных WinAPI функций в вызовы функций *Native* *API* и переключают *процессор* на режим ядра.

**Компоненты режима ядра**

*Диспетчер* системных сервисов (*System* *Service* *Dispatcher*) работает в режиме ядра, перехватывает вызовы функций от Ntdll.*dll*, проверяет их параметры и вызывает соответствующие функции из Ntoskrnl.exe.

Исполнительная система и *ядро* содержатся в Ntoskrnl.exe (NT Operating *System* *Kernel* – *ядро* операционной системы NT) (по поводу использования термина "*ядро*" в *Windows* см. лекцию 1 "Введение в операционные системы").

Исполнительная система (*Executive*) представляет собой совокупность компонентов (называемых диспетчерами – *manager*), которые реализуют основные задачи операционной системы:

* диспетчер процессов (process manager) – управление процессами и потоками (см. лекцию 6 "Процессы и потоки");
* диспетчер памяти (memory manager) – управление виртуальной памятью и отображение её на физическую (см. лекцию 8 "Управление памятью");
* монитор контроля безопасности (security reference monitor) – управление безопасностью (см. лекцию 9 "Безопасность");
* диспетчер ввода вывода (I/O manager), диспетчер кэша (cache Manager), диспетчер Plug and Play (PnP Manager) – управление внешними устройствами и файловыми системами (см. лекцию 10 "Управление устройствами" и лекцию 11 "Файловая система NTFS");
* диспетчер электропитания (power manager) – управление электропитанием и энергопотреблением;
* диспетчер объектов (object manager), диспетчер конфигурации (configuration manager), механизм вызова локальных процедур (local procedure call) – управление служебными процедурами и структурами данных, которые необходимы остальным компонентам.

*Ядро* (*Kernel*) содержит функции, обеспечивающие поддержку компонентам исполнительной системы и осуществляющие планирование потоков (см. лекцию 7 "Планирование потоков"), *механизмы* синхронизации, обработку прерываний.

*Компонент* *Windows* *USER* и *GDI* отвечает за пользовательский графический *интерфейс* (окна, *элементы управления* в окнах – *меню*, кнопки и т. п., рисование), является частью подсистемы *Windows* и реализован в драйвере Win32k.sys.

Взаимодействие диспетчера ввода вывода с устройствами обеспечивают драйверы (drivers) – программные модули, работающие в режиме ядра, обладающие максимально полной информацией о конкретном устройстве (драйверы подробнее рассматриваются в лекции 10 "Управление устройствами").

Однако, и драйверы, и *ядро* не взаимодействуют с физическими устройствами напрямую – посредником между программными компонентами режима ядра и аппаратурой является *HAL* (*Hardware* *Abstraction* *Layer*) – уровень абстрагирования от оборудования, реализованный в *Hal*.*dll*. *HAL* позволяет скрыть от всех программных компонентов особенности аппаратной платформы (например, различия между материнскими платами), на которой установлена *операционная система*.