1. Что такое служба?

**Сервис** (**служба)** – процесс, который выполняет служебные функции.

То есть запускается или по команде, или при загрузке ОС, заканчивает работу по команде или при завершении работы ОС.

Не каждая программа запускаемая со стартом ОС является сервисом.

1. Какие бывают службы?

Обычно сервисы выполняют определенные служебные функции, необходимые для работы приложений или какого-то конкретного приложения. Примеры:

Фоновый процесс, который обеспечивает доступ к бд – **серверы**

Программы, обеспечивающие доступ к внешним устройствам, – **драйверы**

Сервис как процесс, отслеживающий работу приложения, – **мониторы**

1. Чем можно охарактеризовать службы?

**Характеристики сервисов:**

* Только фоновый режим
* Не имеют собственного управляющего интерфейса (ни GUI, ни TUI)
* Управляются специальной программой ОС – менеджером служб
* Запуск/стоп по команде, входом/выходом пользователя, стартом/выключением ОС.
* Предоставление услуг другим программам или ОС, а не пользователям

1. Что такое SCM?

**SCM** – это Service Control Manager (Менеджер Управления Службами). Это специальная программа операционной системы Windows, которая управляет работой сервисов. Её функции включают поддержку базы данных установленных сервисов, запуск/остановку сервисов, передачу им управляющих запросов и предоставление информации об их состоянии.

SCM поддерживает бд установленных сервисов в реестре.

**HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services**

содержит подраздел(название службы) для каждого сервиса

1. Где располагается информация о сервисе?

**HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\сервис**

Информация о сервисе:

* DependOnGroup – группы порядка загрузки сервисов, от которых зависит данный сервис
* DependOnService – сервисы, от которых зависит данный сервис
* Description – описание
* DisplayName – отображаемое имя
* ErrorControl – уровень управления ошибками
* FailureActions – действия выполняемые при возникновении ошибки
* Group – группа порядка загрузки сервисов, в которой состоит данный сервис
* ImagePath – путь к исполняемому файлу
* ObjectName – учетка от имени которой запускаем
* Start – тип запуска
* Tag – уникальный тег для данного сервиса в рамках группы порядка загрузки в которой он состоит
* Type – тип сервиса

**Группы порядка загрузки** – логически объеденный набор сервисов, который определяет порядок загрузки сервисов входящих в него относительно остальных групп или сервисов

В целом порядок загрузки сервисом определяется следующим образом:

1. Порядок групп в списке групп порядка загрузки  
   **HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\ServiceGroupOrder**
2. Порядок загрузки сервисов в рамках своей группы  
   **HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\GroupOrderList**

Остальные зависимости каждого из сервиса

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Сервисы должны удовлетворять соглашениям, определяющим интерфейс сервиса.  
Обычно сервис - консольное приложение, но может быть и с графическим интерфейсом.  
Должен содержать две функции обратного вызова(функция в аргументах другой функции), которые вызываются ОС.   
Одна функция - точка входа сервиса (сам сервис), вторая - реагирует на управляющие сигналы от ОС. Функции ServiceMain, ServiceCtrlHandler.

Отсюда следует, что если консольное приложение определяет один сервис, то оно должно иметь следующую структуру:

Изображение выглядит как текст, чек, Шрифт, алгебра

Содержимое, созданное искусственным интеллектом, может быть неверным.

main - запускает диспетчер сервиса, который управляет сервисом и является потоком.

Если определяется только 1 сервис, то функция называется [**ServiceMain**](https://learn.microsoft.com/en-us/windows/win32/services/service-servicemain-function). Возможны другие. Если же в приложении определяется несколько сервисов, то каждый имеет свое имя. Эта функция содержит два параметра, которые аналогичны параметрам **main**.  
  
Запуск диспетчера - StartServiceCtrlDispatcher - должна быть вызвана из main в течение 30 секунд после запуска программы, если нет - неудача. Чтобы не было этого, инициализировать сервис можно в теле ServiceMain, а инициализация глобальных переменных в отдельном потоке функции main.  
Диспетчер сервиса передает запросы ServiceCtrlHandler от менеджера сервисов, которая обрабатывает все запросы.  
Несколько сервисов - несколько диспетчеров – свои функции обработки управляющих запросов.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.  
  
Подключение обработчика к сервису- RegisterServiceCtrlHandler  
Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Точка входа - ServiceMain  
Должна выполнять:  
1. запустить RegisterServiceCtrlHandler (обработчик управляющих команд от менеджера сервисов) немедленно  
2.вызвать функцию SetServiceStatus - состояние SERVICE\_START\_PENDING  
3. локальная инициализация сервиса  
4. установить в рабочее состояние SetServiceStatus SERVICE\_RUNNING  
5. выполнять работу, реагируя на команды  
6. после SERVICE\_STOPPED освободить ресурсы и закончить работу

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Параметр ***lpServiceStatus*** функции [**SetServiceStatus**](https://learn.microsoft.com/en-us/windows/desktop/api/Winsvc/nf-winsvc-setservicestatus)должен указывать на структуру типа SERVICE\_STATUS, которая содержит информацию о состоянии сервиса. Эта структура имеет следующий формат:

Изображение выглядит как текст, чек, алгебра

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Технические типы** служб Windows (определяемые в dwServiceType):

* SERVICE\_WIN32\_OWN\_PROCESS (самостоятельный процесс)
* SERVICE\_WIN32\_SHARE\_PROCESS (разделяемый процесс)
* SERVICE\_KERNEL\_DRIVER (драйвер ядра)
* SERVICE\_FILE\_SYSTEM\_DRIVER (драйвер файловой системы)
* SERVICE\_USER\_OWN\_PROCESS / SERVICE\_USER\_SHARE\_PROCESS (пользовательские процессы)
* Могут быть интерактивными (SERVICE\_INTERACTIVE\_PROCESS)

Кроме того, первые два флага могут быть установлены совместно с флагом SERVICE\_INTERACTIVE\_PROCESS – сервис может взаимодействовать с рабочим столом

Поле ***dwCurrentState*** содержит текущее состояние сервиса. Это поле может принимать одно из следующих значений:

* SERVICE\_STOPPED – сервис остановлен
* SERVICE\_START\_PENDING – сервис стартует
* SERVICE\_STOP\_PENDING – сервис останавливается
* SERVICE\_RUNNING – сервис работает
* SERVICE\_CONTINUE\_PENDING – сервис переходит в рабочее состояние
* SERVICE\_PAUSE\_PENDING – сервис переходит в состояние ожидания
* SERVICE\_PAUSED – сервис находится в состоянии ожидания

Поле ***dwControlsAccepted*** содержит коды управляющих команд, которые могут быть переданы обработчику этих команд, определенному в приложении. В этом поле может быть установлена любая комбинация следующих флагов:

* SERVICE\_ACCEPT\_STOP – можно остановить сервис
* SERVICE\_ACCEPT\_PAUSE\_CONTINUE – можно приостановить и возобновить сервис
* SERVICE\_ACCEPT\_SHUTDOWN – сервис информируется о выключении системы
* SERVICE\_ACCEPT\_PRESHUTDOWN – сервис может выполнить действия перед выключением системы
* SERVICE\_ACCEPT\_PARAMCHANGE - сервис может вновь прочитать свои стартовые параметры без перезагрузки
* SERVICE\_ACCEPT\_NETBINDCHANGE - сервис является сетевой компонентой

Поле ***dwServiceSpecificExitCode*** содержит код возврата из сервиса, этот код действителен только в том случае, если в поле ***dwWin32ExitCode*** установлено значение ERROR\_SERVICE\_SPECIFIC\_ERROR

Поле ***dwCheckPoint*** содержит значение, которое сервис должен периодически увеличивать на единицу, сообщая о продвижении своей работы во время инициализации и длительных переходов из состояния в состояние

Это значение может использоваться программой пользователя, которая отслеживает работу сервиса. Если это значение не используется пользовательской программой и переход из состояния в состояние занимает менее 30 секунд, то оно может быть установлено в 0

Поле ***dwWaitHint*** содержит интервал времени в миллисекундах, в течение которого сервис переходит из состояния в состояние перед вызовом функции установки состояния [**SetServiceStatus**](https://learn.microsoft.com/en-us/windows/desktop/api/Winsvc/nf-winsvc-setservicestatus)

Если в течение этого интервала не произошло изменение состояния сервиса в поле [**dwServiceState**](https://learn.microsoft.com/en-us/windows/desktop/api/Winsvc/nf-winsvc-setservicestatus) или не изменилось значение поля ***dwCheckPoint***, то менеджер сервисов считает, что в сервисе произошла ошибка

Если определяется только один сервис, то название [**ServiceCtrlHandler**](https://learn.microsoft.com/en-us/windows/win32/services/service-control-handler-function). Если в приложении несколько сервисов, то для каждого свое имя. Эта функция содержит только один параметр, который содержит код управляющего сигнала Так как обработчик запросов вызывается диспетчером сервиса, то и коды управляющих сигналов он получает от своего диспетчера

Возможны следующие основные управляющие коды:

* SERVICE\_CONTROL\_STOP – остановить сервис
* SERVICE\_CONTROL\_PAUSE – приостановить сервис
* SERVICE\_CONTROL\_CONTINUE – возобновить сервис
* SERVICE\_CONTROL\_INTERROGATE – обновить состояние сервиса
* SERVICE\_CONTROL\_SHUTDOWN – закончить работу сервиса

Но обработчик должен обрабатывать только те команды, которые допускаются в поле ***dwControlsAccepted*** структуры типа **SERVICE\_STATUS**. По соглашению обработчик всегда получает сигнал с кодом **SERVICE\_CONTROL\_INTERROGATE**, по которому он должен немедленно обновить состояние сервиса. Для обновления состояния сервиса используется функция [**SetServiceStatus**](https://learn.microsoft.com/en-us/windows/desktop/api/Winsvc/nf-winsvc-setservicestatus). Кроме того, обработчик может получать коды, определенные пользователем. Для кодов пользователя зарезервирован диапазон от 128 до 255.

Вызывается обработчик управляющих запросов диспетчером сервиса и, следовательно, выполняется в его контексте

Обработчик должен изменить состояние сервиса в течение 30 секунд, в противном случае диспетчер сервисов считает, что произошла ошибка

Если для изменения состояния сервиса требуется более продолжительный интервал времени, то для этой цели нужно запустить отдельный поток

Для обработки кода **SERVICE\_CONTROL\_SHUTDOWN** сервису отводится 20 секунд, в течение которых он должен освободить захваченные им ресурсы

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Для того чтобы менеджер сервисов знал о существовании определенного сервиса – его нужно установить

Сервис может запускаться как операционной системой при загрузке, так и **программно** – из приложения

Если сервис больше не нужен, то его нужно удалить из базы данных операционной системы

Для работы с сервисами в операционных системах Windows предназначены специальные функции из [**WinAPI Service**](https://learn.microsoft.com/en-us/windows/win32/services/service-reference)

* **WinAPI функции** (библиотека Advapi32.dll):
  + OpenSCManager: Доступ к SCM.
  + CreateService, DeleteService: Установка/удаление.
  + OpenService, StartService, ControlService: Управление.
  + QueryServiceStatusEx: Состояние (лекция: "WinAPI функции").
* **Утилита sc.exe**:
  + sc create , sc start, sc stop, sc delete, sc query, sc control.

1. Что такое Linux-демон?

**Демон - процесс со свойствами: длинный жц, фоновый режим без контролирующего терминала.**

* **Особенности**:
  + Запускаются при старте системы, завершаются при выключении.
  + Не имеют терминала (не получают сигналы SIGINT, SIGHUP).
  + Название обычно заканчивается на d (например, httpd).

Чтобы стать демоном:  
1. fork, родитель рип, потомок продолжает (становится потомком init).  
2. setsid, новая сессия (отвязка от терминала)  
3. fork, чтобы не стать лидером сессии  
4. Очистить атрибут unmask чтобы были права на файлы и каталоги созданные демоном  
5. смена рабочего каталога процесса на корневой (система не сможет быть отключена, если будет не корневой)  
6. закрыть файловые дескрипторы, которые демон унаследовал от родителя  
(Демон закрывает дескрипторы 0, 1 и 2, чтобы освободить ресурсы, так как они больше не необходимы в фоновом режиме и могут ссылаться на терминал, который уже потерян.)

1. Закрывая дескрипторы 0, 1 и 2 и перенаправляя их в /dev/null, демон предотвращает ошибки ввода/вывода и исключает возможность повторного открытия этих дескрипторов для записи в стандартные потоки.

**Обработка сигнала SIGTERM**: Демон должен корректно обрабатывать сигнал SIGTERM, освободив все ресурсы перед завершением, чтобы избежать утечек памяти и других проблем. При выключении компьютера отправляется процессом **init** всем своим потомкам. По умолчанию этот сигнал приводит к завершению процесса. Если демону перед этим необходимо освободить какие-либо ресурсы, он должен делать это в обработчике данного сигнала.

Демон должен быстро освобождать ресурсы, так как через 5 секунд после SIGTERM будет отправлен сигнал SIGKILL, который принудительно завершит процесс.

**Закрытие файловых дескрипторов**: Демон должен внимательно следить за закрытием всех файловых дескрипторов, чтобы избежать утечек и ошибок, особенно с длинным жизненным циклом.

**Единственный экземпляр**: Для предотвращения запуска нескольких экземпляров демона, демон должен использовать блокировку файла, удостоверяясь, что только один его экземпляр работает в системе.

1. Что такое Init и Systemd?

Init (SysV Init) и Systemd это менеджеры сервисов (системы инициализации) в Linux, которые отвечают за запуск, остановку и управление демонами (службами).

**Init**: Более старая система, использует скрипты в /etc/init.d/. Управляется в основном командой service. Считается устаревшей.

**Systemd**: Более новая и современная система, использует конфигурационные файлы (unit-файлы, например, .service) обычно в /etc/systemd/system/. Управляется командой systemctl.

Обе системы могут выполнять процесс «демонизации» (отсоединение от терминала и т.д.) для запускаемых программ и позволяют настраивать автозапуск служб при загрузке ОС.

1. Скомпилировать демон.
2. Поместить бинарный файл /usr/sbin/<demoserviced>.
3. Создать файл описания < demoserviced >.service в /etc/systemd/system. **Изображение выглядит как текст, снимок экрана, Шрифт

   Содержимое, созданное искусственным интеллектом, может быть неверным.**
4. Создать /etc/demoserviced/< demoserviced >.config для конфигураций. (необязательно)
5. Логи хранить в /var/log.
6. Проверить распознавание: systemctl status < demoserviced >.
7. Управлять: systemctl start/stop/restart < demoserviced >.
8. Для автозапуска: systemctl enable < demoserviced >