**Лабораторная работа №5**

**Сервисы**

**Цель работы:** Получение практических навыков в работе с Windows-сервисами и Linux-демонами.

Перед выполнением задания прочитайте его полностью. Схематично представьте себе, что от вас требуется и только после этого выполняйте.

**Темы для предварительного изучения:**

* Функции для работы с сервисами в WinAPI ([ссылка](https://learn.microsoft.com/en-us/windows/win32/services/service-functions));
* Справочник по systemd ([ссылка](https://wiki.debian.org/systemd/Services));
* Пример скрипта для initd ([ссылка](https://gist.github.com/naholyr/4275302)).

**Функционал для сервисов:**

Сервис на любой из ОС из данной лабораторной работы должен функционально выполнять следующее:

* Считывать параметры для запуска из конфигурационного файла;

void LoadConfiguration(const char\* config\_file)

* Выполнять копирование файлов из одного каталога в другой один раз в указанный промежуток времени (все каталоги и промежуток времени указываются в конфигурации);

void SynchronizeDirectories()

* Если каталог откуда копируются файлы не существует, то сервис завершается с ошибкой!

VOID WINAPI BackupServiceMain(DWORD argc, LPTSTR\* argv)

* Принимать сигнал на повторное чтение конфигурации без перезагрузки самого сервиса;

SERVICE\_CONTROL\_RELOAD\_CONFIG

* Результаты своей работы отображать в журнале (при запуске сервиса – создаётся файл логирования с названием формата «<datetime>-service.log», где datetime – момент времени запуска сервиса);

void WriteToJournal(const char\* message, BOOL is\_error)

* Путь к каталогу в котором располагаются журнальные файлы также указывается в конфигурационном файле (если каталог не существует, то производится попытка его создания).

void EnsureDirectoryExists(const char\* directory\_path, const char\* dir\_purpose)

Сервис должен формировать в журнале сообщения следующего вида:

* При создании каталогов для файлов логирования или резервных копий, указанных в параметрах – Успех! <Имя сервиса> создал каталог <Путь к каталогу>; (для каждого каталога отдельное сообщение)
* При обнаружении что каталоги для журнальных файлов или резервных копий, указанные в параметрах, существуют – Успех! <Имя сервиса> обнаружил каталог <Путь к каталогу>; (для каждого каталога отдельное сообщение)
* При успешном запуске – Успех! Сервис <Имя сервиса> запущен с параметрами <Параметры>;
* При смене состояния – Успех! Сервис <Имя сервиса> сменил состояние с <Текущее состояние> на <Новое состояние>
* При успешном копировании файлов из одного каталога в другой – Успех! Сервис <Имя сервиса> успешно создал резервную копию файлов!
* При ошибках – Попытка выполнения операции сервисом провалена! <Текст ошибки>!

**Постановка задачи для Windows:**

**Проект DemoService:**

Должен содержать код для создания Windows-сервиса с заданным функционалом (см. выше).

**Значения по умолчанию**:

* Каталог с резервными копиями файлов – «%APPDATA%\DemoService\Reserved»;
* Каталог с журнальными файлами – «%APPDATA%\DemoService\Logs»;
* Период копирования – 60 минут.

Так же требуется реализовать обработку возможных ошибок (при открытии файла и т.д.). Если ошибка не позволяет продолжить выполнение сервиса, то он должен быть переведён в состояние «Остановлен».

**Сервис должен поддерживать принятие и обработку следующих сигналов**:

* Запуск (поддерживается по умолчанию всеми сервисами) – при запуске сервис инициализирует все необходимые ресурсы (файлы и т.д.) и начинает свою работу по отслеживанию событий;
* Остановка (SERVICE\_ACCEPT\_STOP) – при остановке сервиса происходит остановка отслеживания событий и очистка всех ресурсов;
* Пауза (SERVICE\_ACCEPT\_PAUSE\_CONTINUE) – сервис просто приостанавливает работу по отслеживанию событий без очистки ресурсов;
* Возобновление (SERVICE\_ACCEPT\_PAUSE\_CONTINUE) – возобновляет работу по отслеживанию событий;
* Пользовательский код (128 + Номер по журналу в группе) – тестовый сигнал, при котором сервис записывает сообщение «Привет, это тестовый код из сервиса <Имя сервиса>!».

**Проект** **ServiceCtl:**

Приложение предназначено для управления сервисами. Семантика вызова данного приложения следующая:

**ServiceCtl <Имя сервиса> <Операция> [Дополнительные параметры]**

Приложение должно обеспечивать следующие операции (в скобках указаны названия и дополнительные параметры для данных операций):

* Создание сервиса – (Create, [путь к исполняемому файлу сервиса]);
* Запуск сервиса – (Start, [путь к конфигурационному файлу]);
* Остановка сервиса – (Stop);
* Удаление сервиса – (Delete);
* Приостановка – (Pause);
* Возобновление – (Continue);
* Вывод информации о сервисе – (Info);
* Отправка тестового сигнала – (Test).

Продемонстрировать управление службой через данное приложение, а также через стандартную утилиту **SC**. Сделать серию скриншотов по каждой команде из списка выше выполненных обеими утилитами (**ServiceCtl** и **SC**).

**Постановка задачи для Linux:**

**Проект demoserviced:**

**Перед выполнением этой части работы проверьте какой менеджер служб у вас сейчас активен! (Если утилита systemctl выдаёт ошибку значит у вас Init, иначе Systemd)**

**В соответствии с этим требуется разработать скрипт файл для демона! Правила управления сервисом через отправку соответствующих сигналов должны быть в нём!**

Должен содержать код для создания Linux-сервиса с заданным функционалом (см. выше).

**Значения по умолчанию**:

* Каталог с резервными копиями файлов – «/srv/demoserviced/reserved»;
* Каталог с журнальными файлами – «/var/log/demoserviced»;
* Период копирования – 60 минут.

Так же требуется реализовать обработку возможных ошибок (при открытии файла и т.д.). Если ошибка не позволяет продолжить выполнение сервиса, то он должен быть переведён в состояние «Остановлен».

Конфигурационный файл расположить в каталоге, описанном в лекции.

**Демон должен поддерживать принятие и обработку следующих сигналов (путём регистрации обработчиков сигналов)**:

* Запуск (не совсем сигнал, тут описано поведение программы при запуске) – при запуске демон инициализирует все необходимые ресурсы (файлы и т.д.) и начинает свою работу по отслеживанию событий;
* Остановка (SIGTERM) – при остановке демона происходит остановка отслеживания событий и очистка всех ресурсов;
* Перезагрузка (SIGHUP) – демон повторно считывает файл конфигурации и выполняется согласно новым параметрам.

Перед любым сообщением в журнальном файле должна присутствовать метка о времени, когда было записано сообщение (без даты).

Продемонстрировать работу демона, а также управление им через утилиту systemctl (для systemd) или service (для init).

Настроить автозапуск демона вместе с операционной системой.

**Требования к выполненной работе:**

* Для генерации и сборки проекта использовать как средство CMake, так и clang напрямую;
* Для обоих случаев подготовить серию скриншотов со сборкой библиотек и приложений!

При использовании CMake должно быть выполнено следующее:

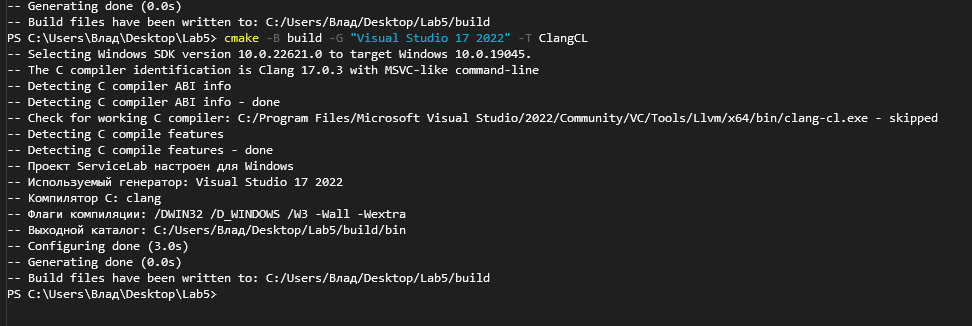
* Сборка должна быть «out-of-source»;
* Для Windows использовать генератор «Visual Studio»/«Ninja» для Linux – «Unix Makefiles», для macOS – «XCode»;
* При сборке проектов всегда должен использоваться clang;
* Все проекты для обеих ОС являются частью одного большого проекта. Добавить в конфигурационные файлы условия сборки отдельных проектов для разных ОС. Конечные файлы генерируются в общий каталог!

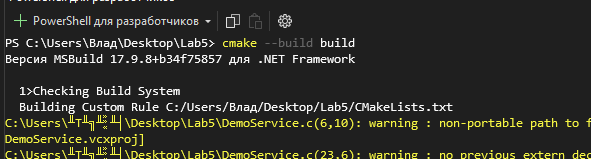
**Вопросы для контроля:**

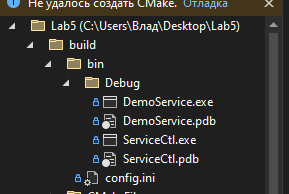
1. Что такое служба?
2. Какие бывают службы?
3. Чем можно охарактеризовать службы?
4. Что такое SCM?
5. Где располагается информация о сервисе?
6. Что такое Linux-демон?
7. Что такое Init и Systemd?

Windows:

Cmake:







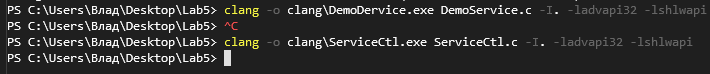
Clang:

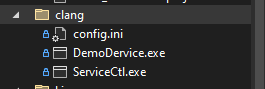
clang -o clang\DemoDervice.exe DemoService.c -I. -ladvapi32 -lshlwapi

-I. - Добавляет текущую директорию (.) в список путей для поиска заголовочных файлов

-ladvapi32 - Подключает системную библиотеку advapi32.lib (нужна для работы с Windows API, например, функций сервисов: StartServiceCtrlDispatcher, RegisterServiceCtrlHandler и т. д.).

-lshlwapi - Подключает библиотеку shlwapi.lib (используется для функций работы с путями, например, PathFileExists).

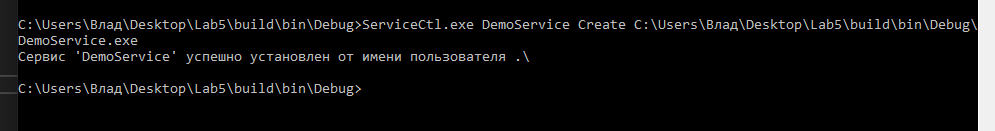




Practice:

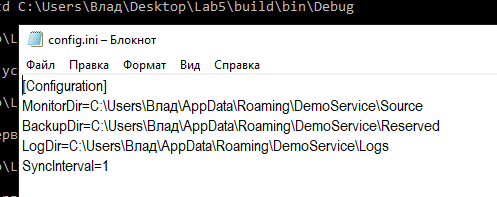
**1) ServiceCtl <Имя сервиса> <Операция> [Дополнительные параметры]**

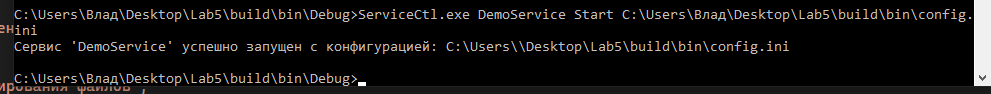
* Создание сервиса – (Create, [путь к исполняемому файлу сервиса]);





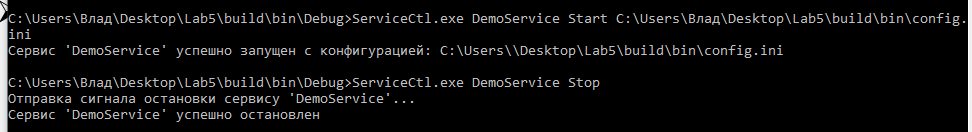
* Запуск сервиса – (Start, [путь к конфигурационному файлу]);

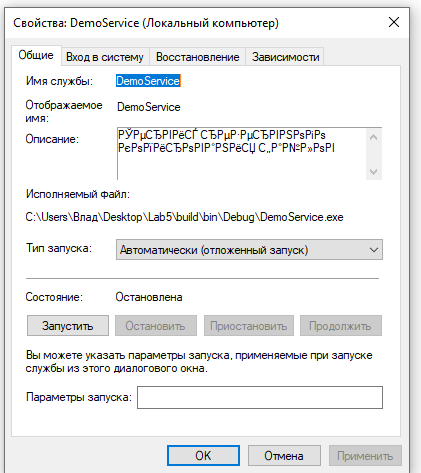




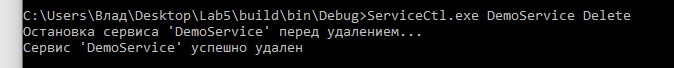


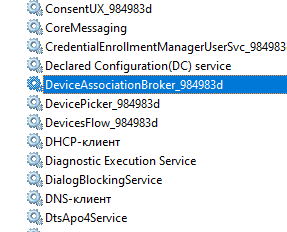
* Остановка сервиса – (Stop);





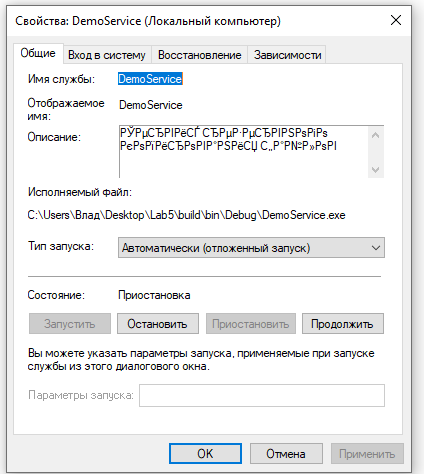
* Удаление сервиса – (Delete);





* Приостановка – (Pause);



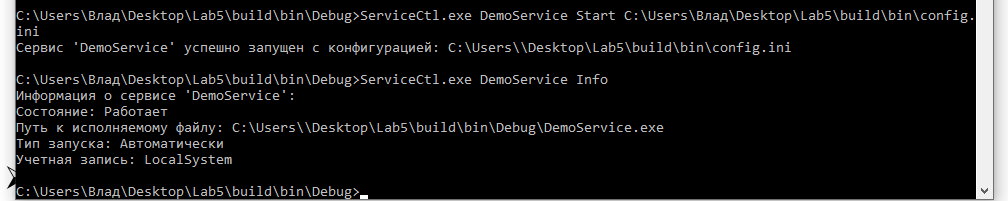


* Возобновление – (Continue);

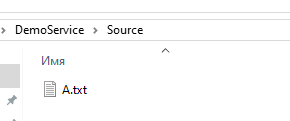


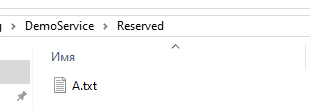


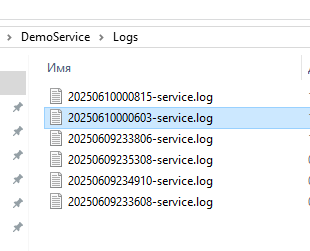
* Вывод информации о сервисе – (Info);

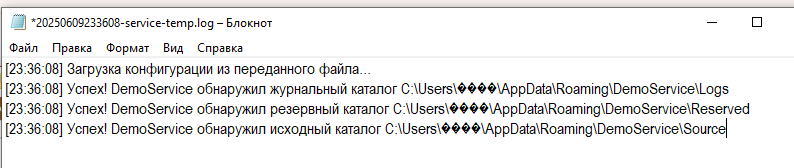


* Отправка тестового сигнала – (Test).

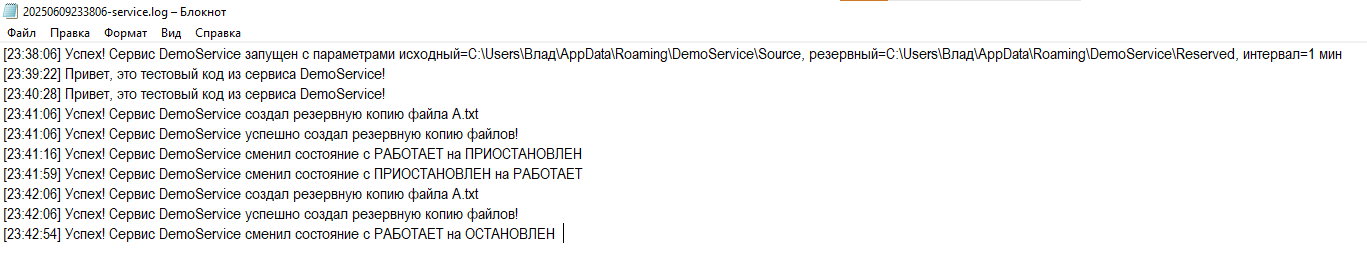


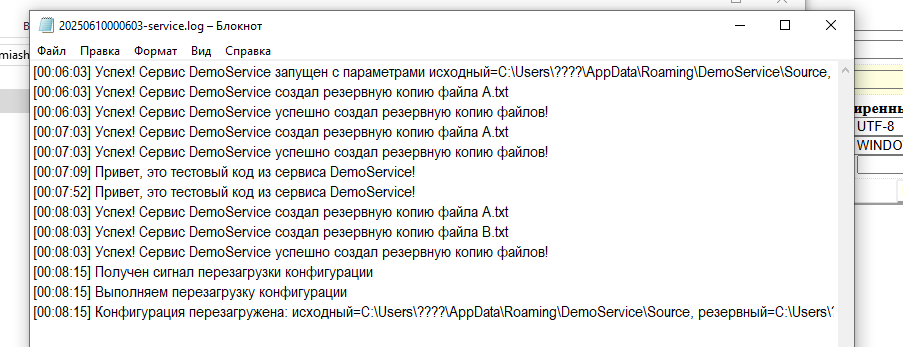








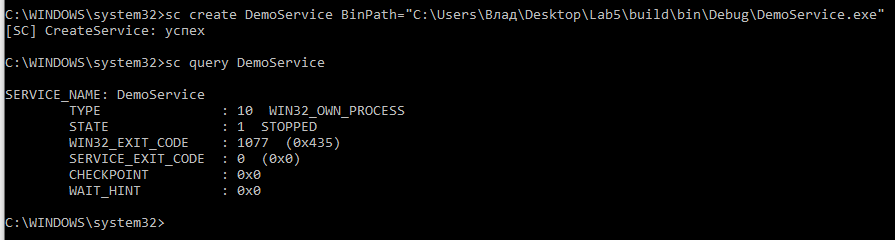




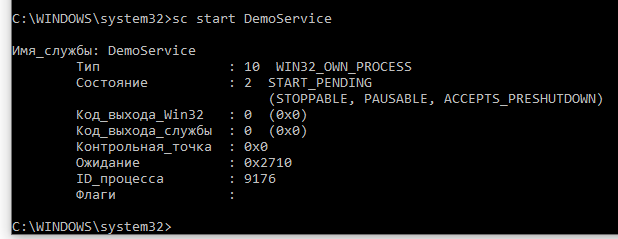
**Sc.exe:**

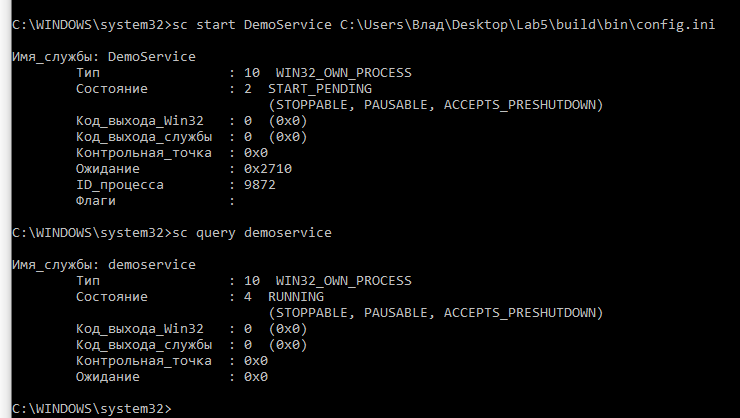
* Создание сервиса – (Create, [путь к исполняемому файлу сервиса]);





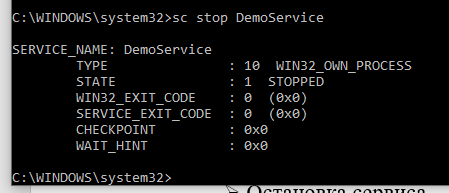
* Запуск сервиса – (Start, [путь к конфигурационному файлу]);

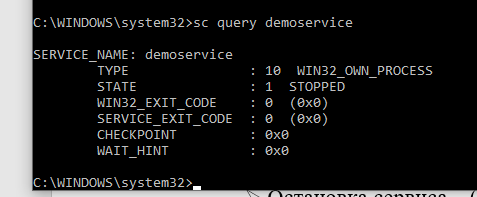






* Остановка сервиса – (Stop);

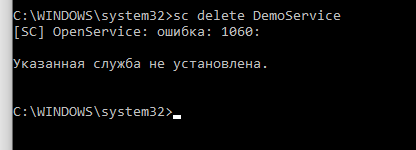




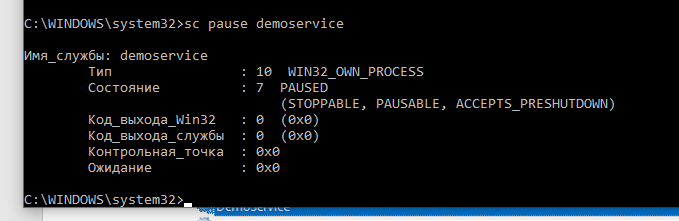


* Удаление сервиса – (Delete);



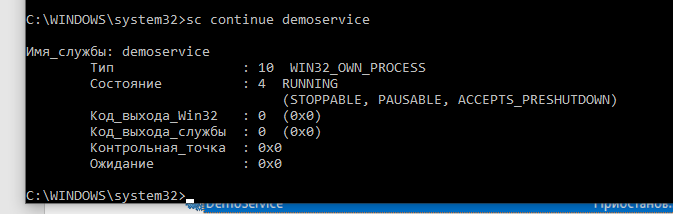


* Приостановка – (Pause);



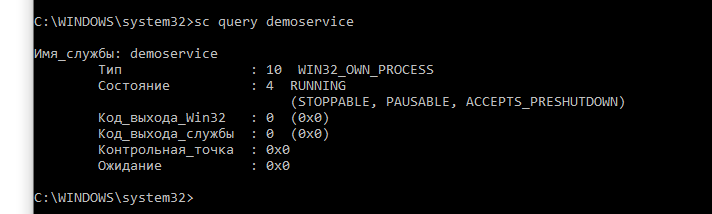


* Возобновление – (Continue);

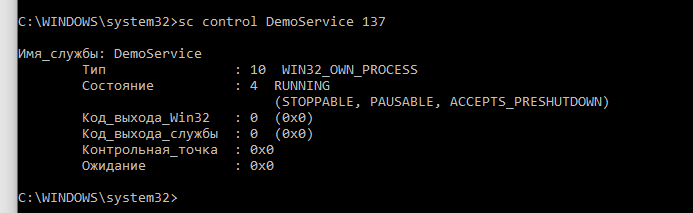


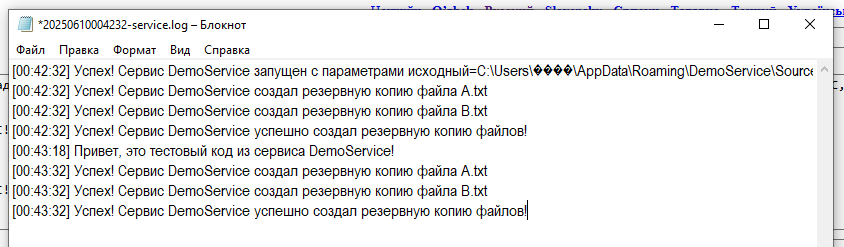


* Вывод информации о сервисе – (Info);



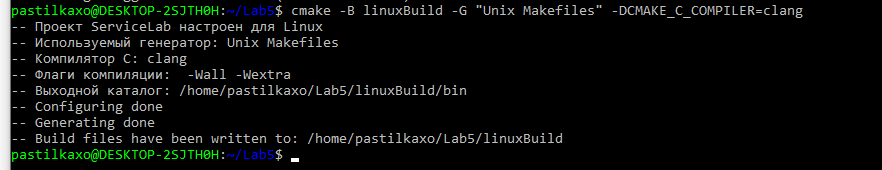
* Отправка тестового сигнала – (Test).

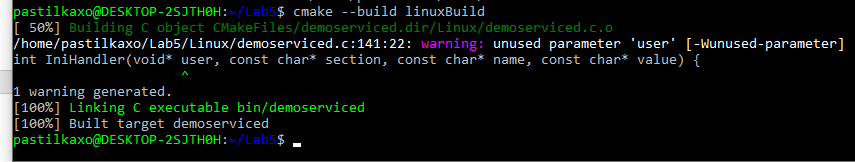


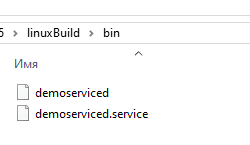


Linux:

Cmake:







Clang:

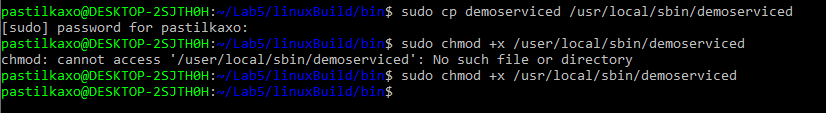




Автозапуск:



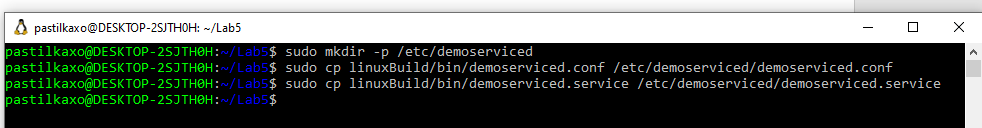
Располагаем получившийся бинарный файл в одной из папок **/sbin** или **/usr/sbin**

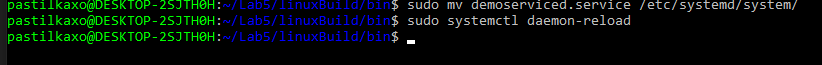


Располагаем скрипт запуска **<имя демона>** в каталоге **/etc/init.d/** (меняем ему права через **chmod** на **755**)



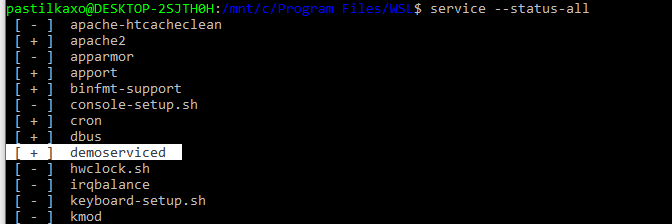
Создаём каталог **/etc/<имя демона>** в котором располагаем конфигурационные файлы которые будут использоваться самим сервисом







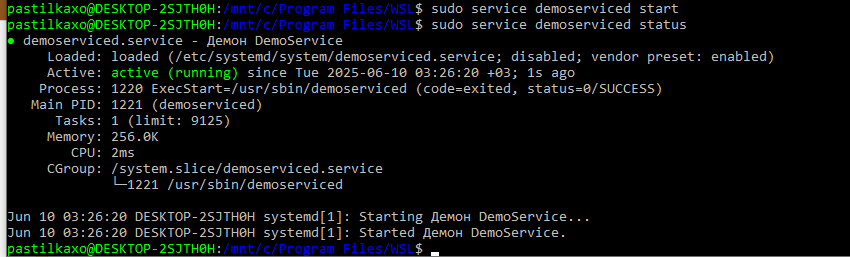


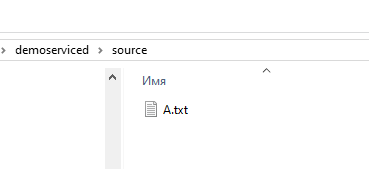


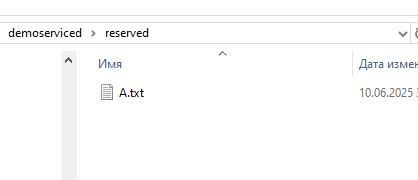
Practice:

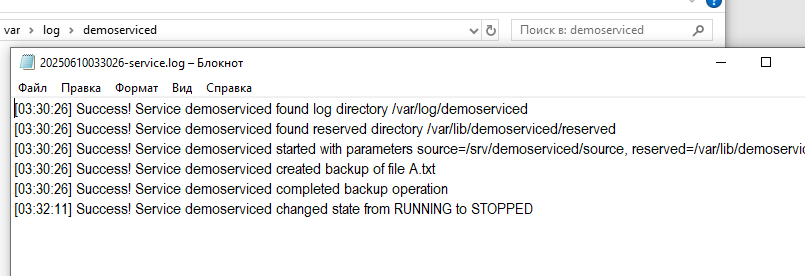
**Демон должен поддерживать принятие и обработку следующих сигналов (путём регистрации обработчиков сигналов)**:

* Запуск (не совсем сигнал, тут описано поведение программы при запуске) – при запуске демон инициализирует все необходимые ресурсы (файлы и т.д.) и начинает свою работу по отслеживанию событий;

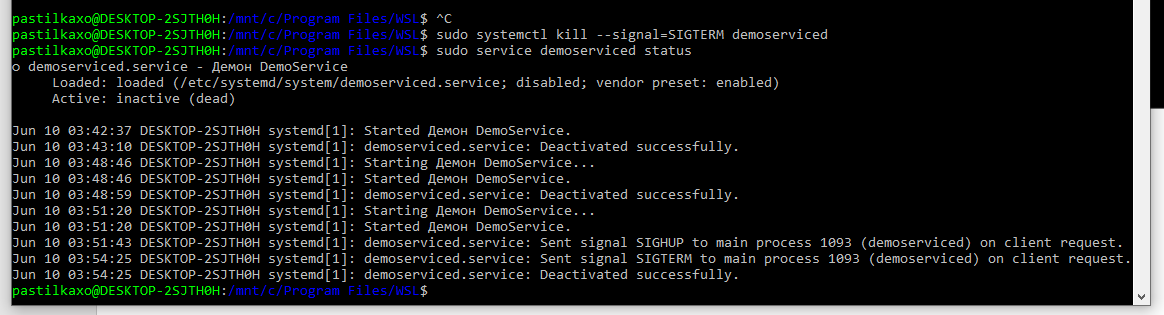


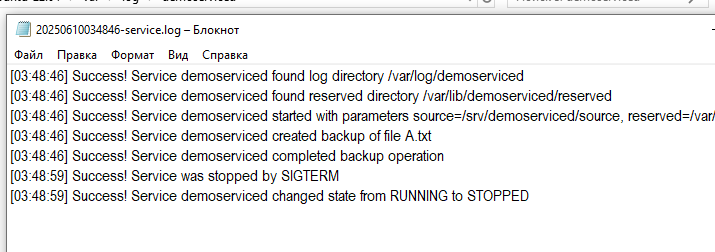


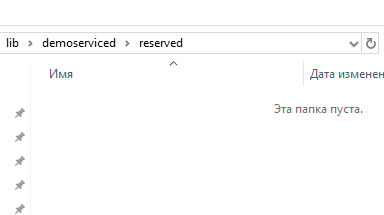




* Остановка (SIGTERM) – при остановке демона происходит остановка отслеживания событий и очистка всех ресурсов;







* Перезагрузка (SIGHUP) – демон повторно считывает файл конфигурации и выполняется согласно новым параметрам.

