Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Системное программирование

Студент: Лопатнюк П.В.

ФИТ 3 курс 1 группа

Преподаватель: Бернацкий П.В.

Минск 2024

**Лабораторная работа №5**

**Сервисы**

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

**Функционал для сервисов:**

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

* Считывать параметры для запуска из конфигурационного файла;
* Выполнять копирование файлов из одного каталога в другой один раз в указанный промежуток времени (все каталоги и промежуток времени указываются в конфигурации);
* Если каталог откуда копируются файлы не существует, то сервис завершается с ошибкой!
* Принимать сигнал на повторное чтение конфигурации без перезагрузки самого сервиса;
* Результаты своей работы отображать в журнале (при запуске сервиса – создаётся файл логирования с названием формата «<datetime>-service.log», где datetime – момент времени запуска сервиса);
* Путь к каталогу в котором располагаются журнальные файлы также указывается в конфигурационном файле (если каталог не существует, то производится попытка его создания).

**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**:

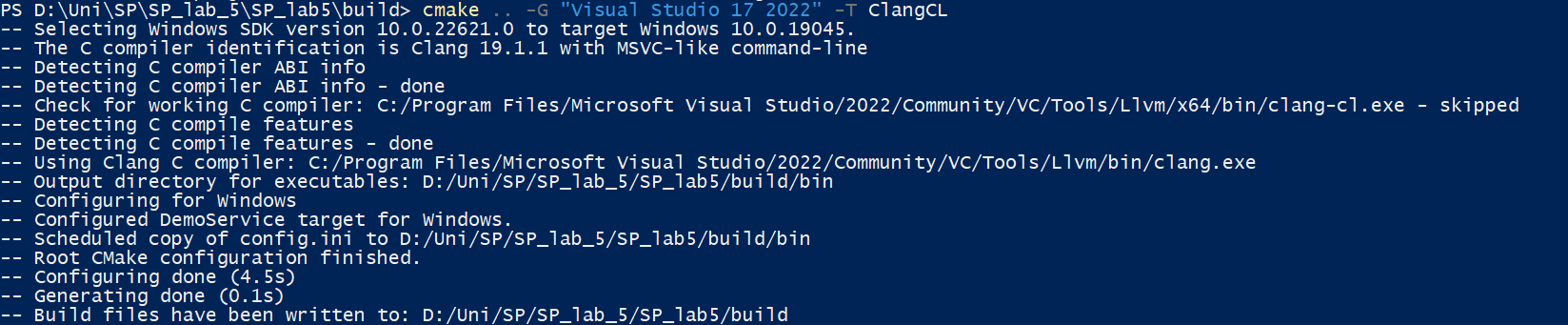
Должен содержать код для создания Windows-сервиса с заданным функционалом (см. выше).

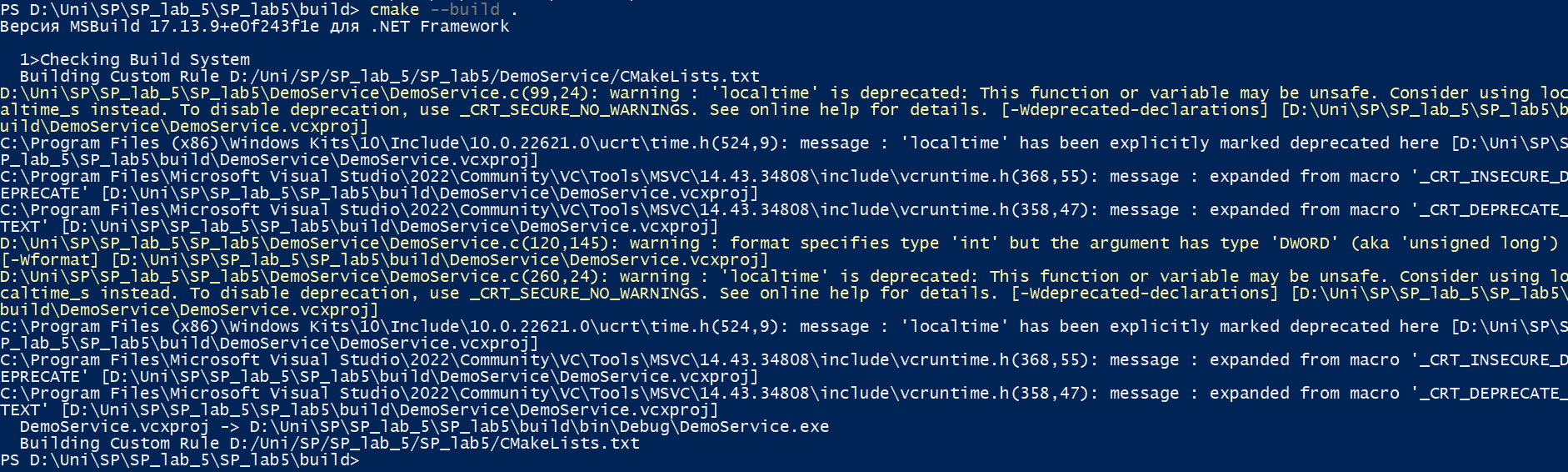
**Значения по умолчанию:**

* 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».

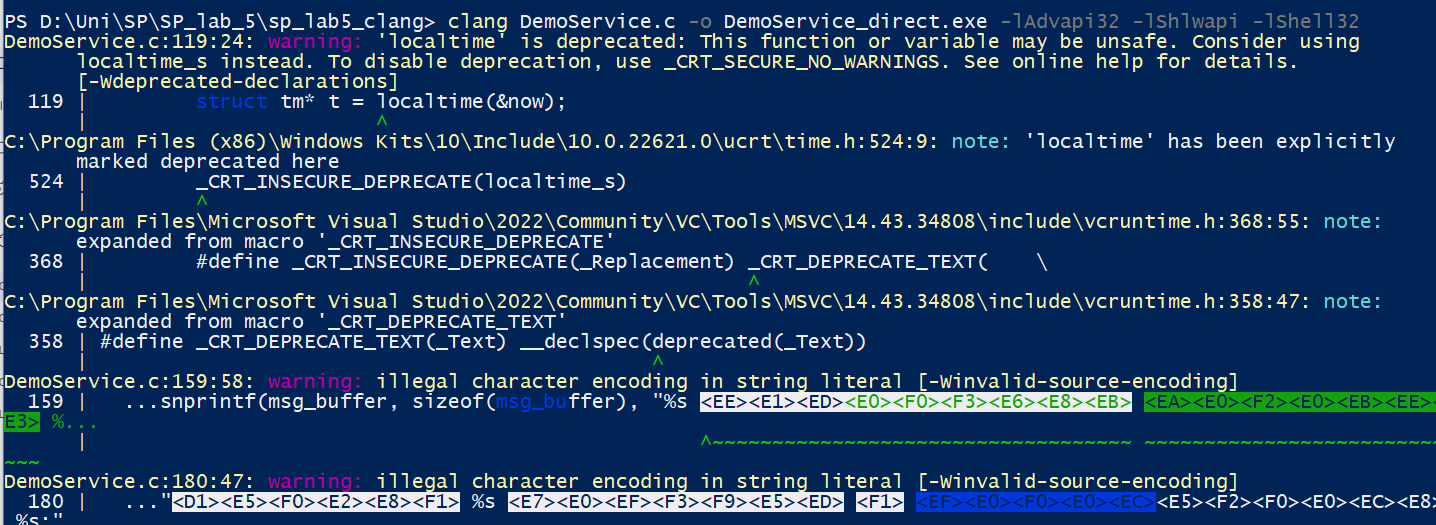
**Сервис должен поддерживать принятие и обработку следующих сигналов:**

**Сборка CMake**:

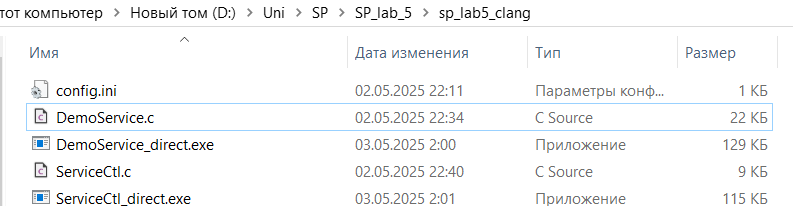




**Сборка Clang**:

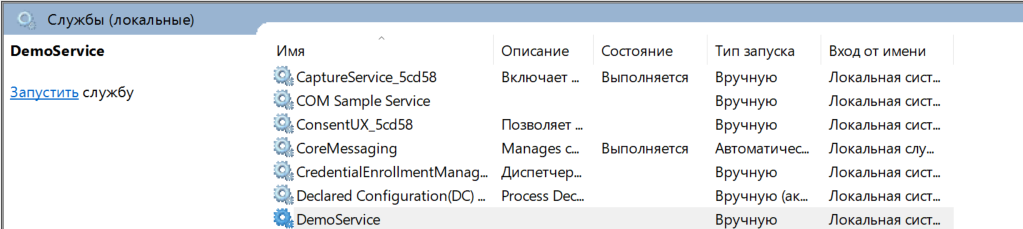






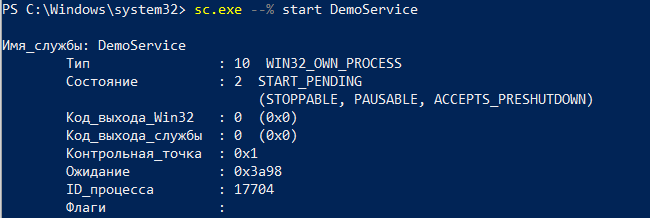
sc.exe --% create DemoService binPath= "D:\Uni\SP\SP\_lab\_5\SP\_lab5\build\bin\Debug\DemoService.exe" start= demand

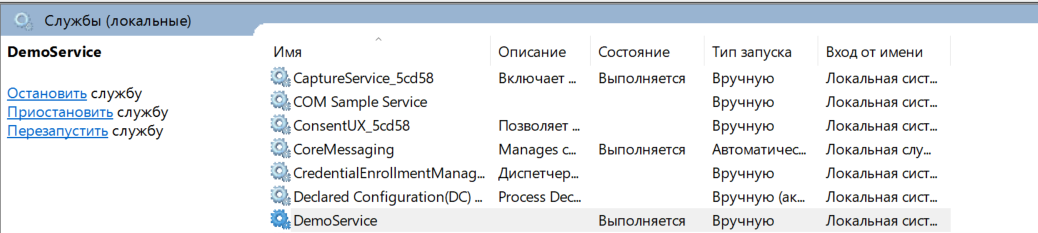


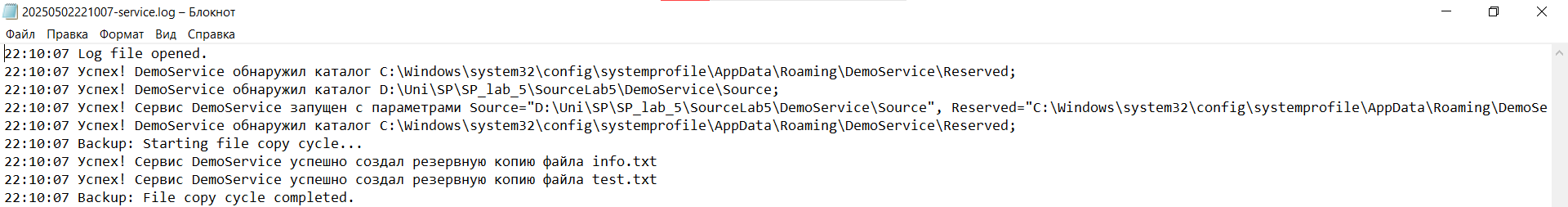


* **Запуск** (поддерживается по умолчанию всеми сервисами) – при запуске сервис инициализирует все необходимые ресурсы (файлы и т.д.) и начинает свою работу по отслеживанию событий;

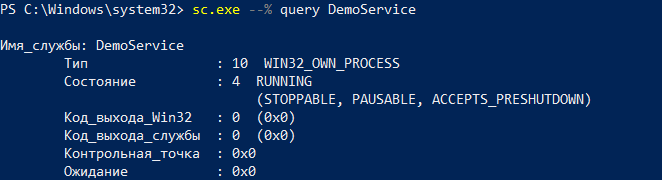
sc.exe --% start DemoService



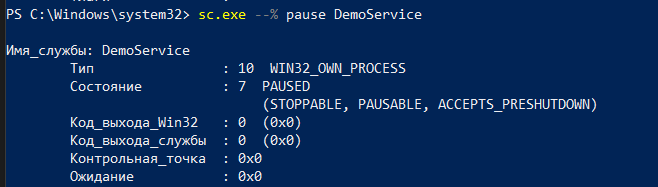




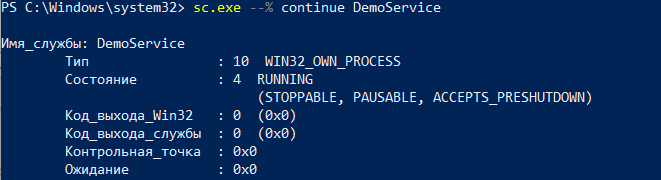
sc.exe --% query DemoService



* **Пауза** (SERVICE\_ACCEPT\_PAUSE\_CONTINUE) – сервис просто приостанавливает работу по отслеживанию событий без очистки ресурсов;



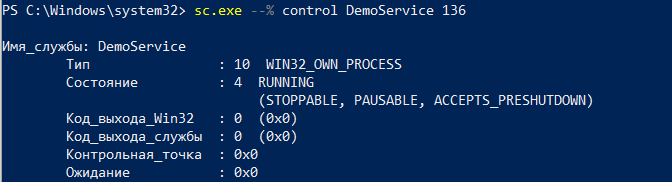
* **Возобновление** (SERVICE\_ACCEPT\_PAUSE\_CONTINUE) – возобновляет работу по отслеживанию событий;



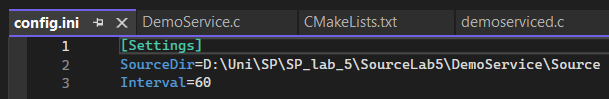
* **Пользовательский код** (128 + Номер по журналу в группе) – тестовый сигнал, при котором сервис записывает сообщение «Привет, это тестовый код из сервиса <Имя сервиса>!».

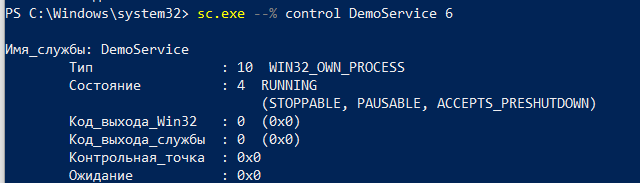
sc.exe --% control DemoService 136

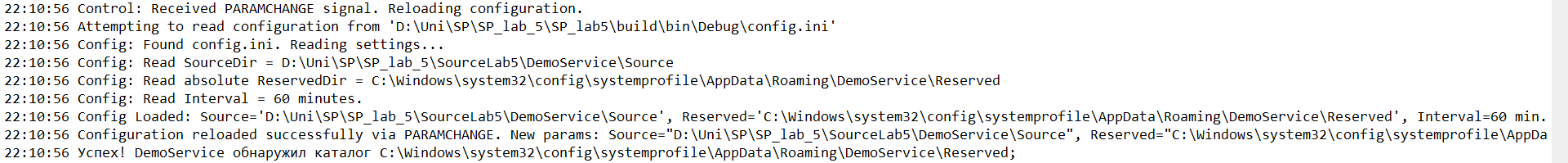


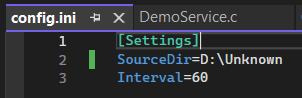


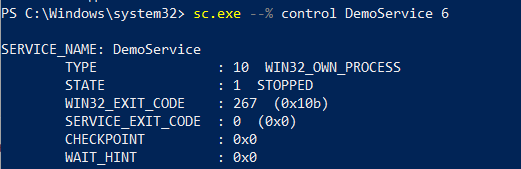
sc.exe --% control DemoService 6







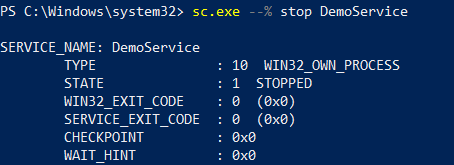






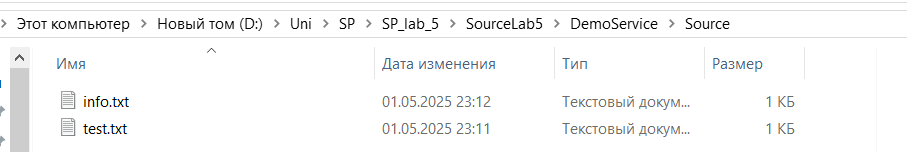
* **Остановка** (SERVICE\_ACCEPT\_STOP) – при остановке сервиса происходит остановка отслеживания событий и очистка всех ресурсов;

sc.exe --% stop DemoService

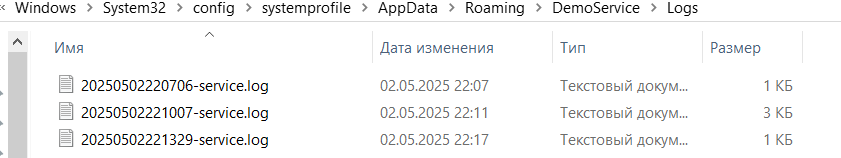


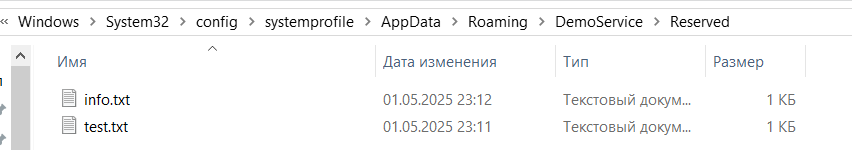
sc.exe --% delete DemoService





C:\Windows\System32\config\systemprofile\AppData\Roaming





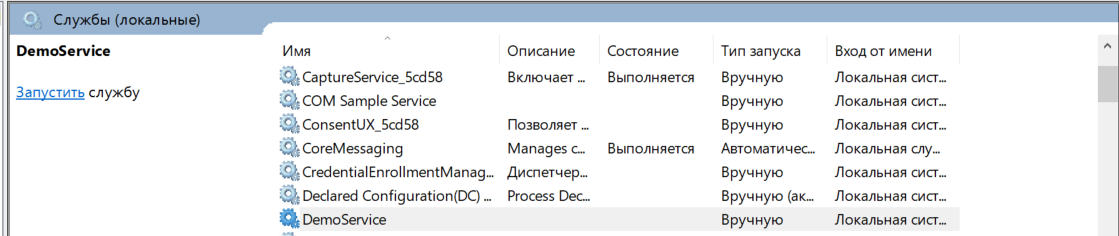
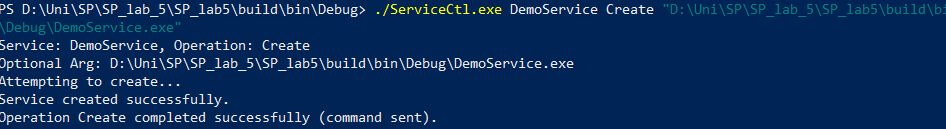
Проект **ServiceCtl**:

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

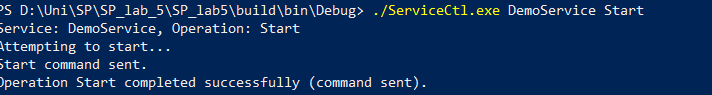
ServiceCtl <Имя сервиса> <Операция> [Дополнительные параметры]

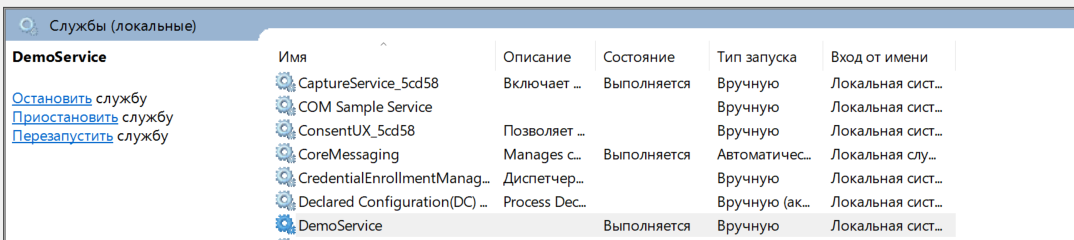
Приложение должно обеспечивать следующие операции (в скобках указаны названия и дополнительные параметры для данных операций):

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

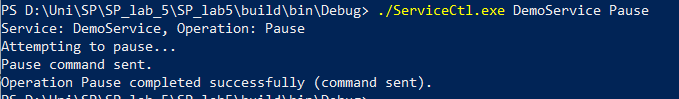


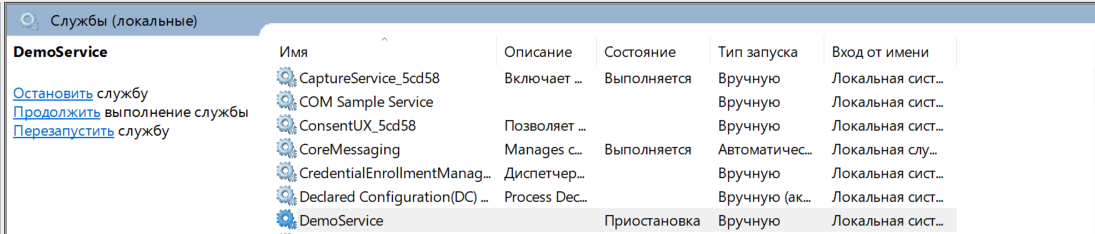
* Запуск сервиса – (Start, [путь к конфигурационному файлу]);



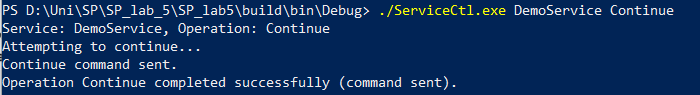


* Приостановка – (Pause);

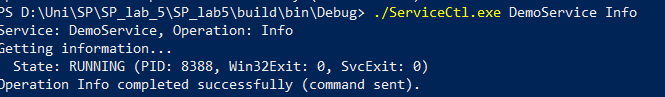


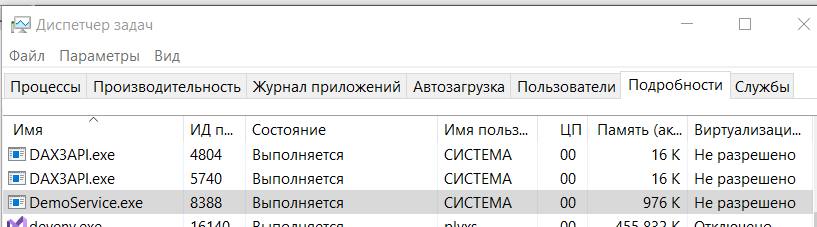


* Возобновление – (Continue);

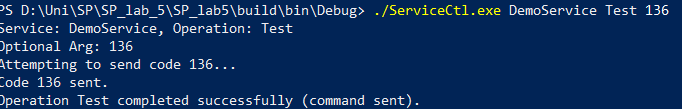


* Вывод информации о сервисе – (Info);

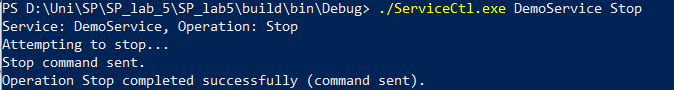


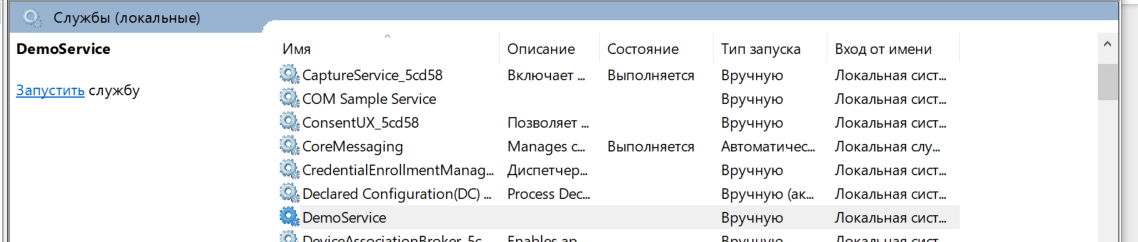


* Отправка тестового сигнала – (Test);

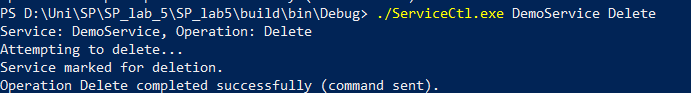


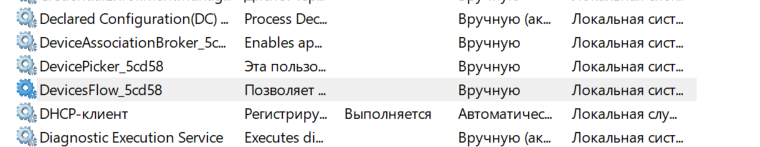
* Остановка сервиса – (Stop);

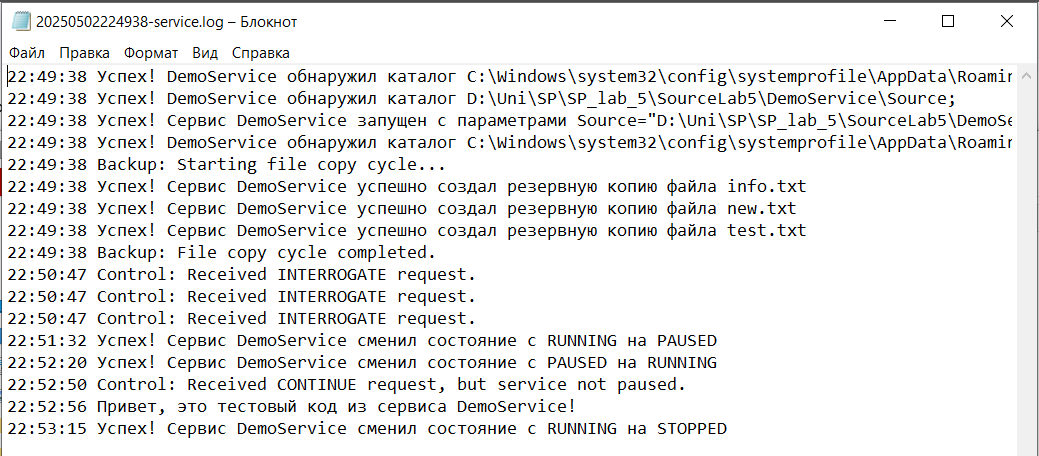




* Удаление сервиса – (Delete).







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)

ps -p 1 -o comm=



sudo apt install libinih-dev pkg-config -y



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.

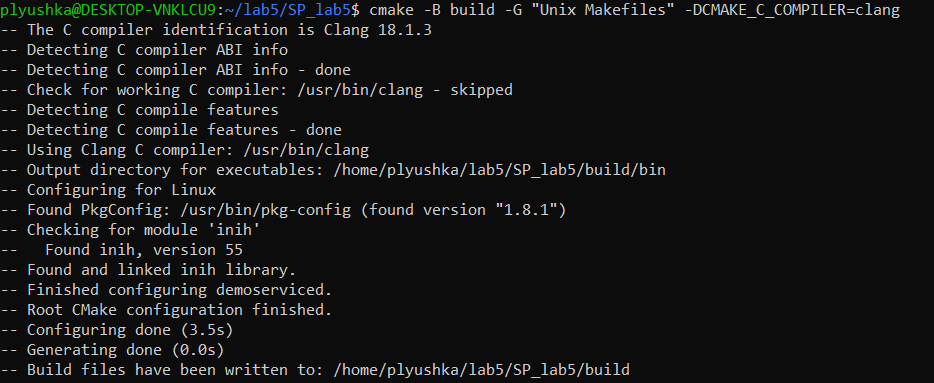
**Демон должен поддерживать принятие и обработку следующих сигналов (путём регистрации обработчиков сигналов):**

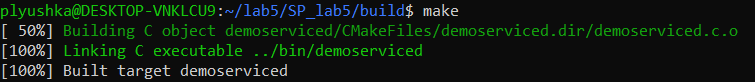
* Запуск (не совсем сигнал, тут описано поведение программы при запуске) – при запуске демон инициализирует все необходимые ресурсы (файлы и т.д.) и начинает свою работу по отслеживанию событий;
* Остановка (SIGTERM) – при остановке демона происходит остановка отслеживания событий и очистка всех ресурсов;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

**Сборка CMake**:





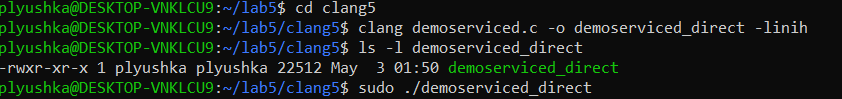
**Сборка Clang:**

Перейдитём в каталог, где лежит ваш файл demoserviced.c.

**Выполним Команду Компиляции clang:**  
 clang demoserviced.c -o demoserviced\_direct -linih

**Проверим, появился ли новый исполняемый файл в текущем каталоге:**

ls -l demoserviced\_direct



**Настроить автозапуск демона вместе с операционной системой.**

1. **Копирование исполняемого файла демона и установка прав на исполнение для бинарника**

sudo cp bin/demoserviced /usr/local/sbin/demoserviced

sudo chmod +x /usr/local/sbin/demoserviced

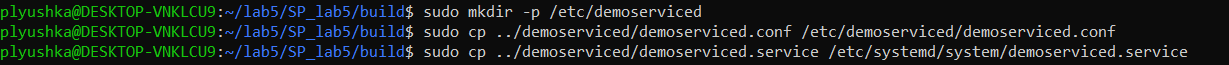


1. **Создание каталога для файла конфигурации, копирование файла конфигурации и файла службы systemd из исходников проекта**

sudo mkdir -p /etc/demoserviced

sudo cp ../demoserviced/demoserviced.conf /etc/demoserviced/demoserviced.conf

sudo cp ../demoserviced/demoserviced.service /etc/systemd/system/demoserviced.service

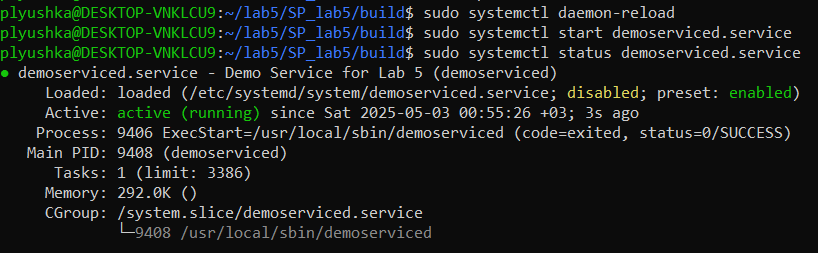


1. **Перезагрузить конфигурацию systemd, чтобы он увидел новый .service файл, запустить службу и проверить статус.**

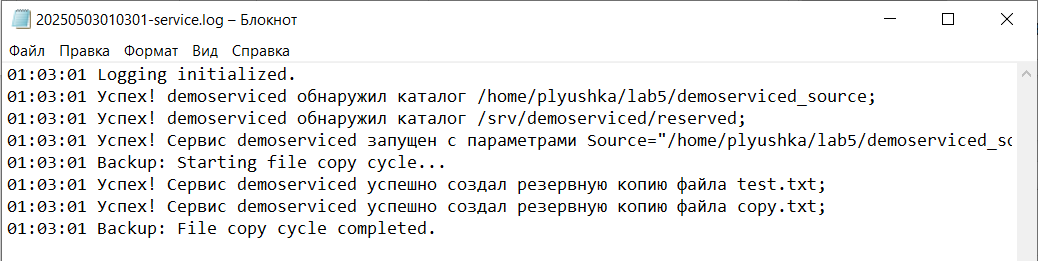
sudo systemctl daemon-reload

sudo systemctl start demoserviced.service

sudo systemctl status demoserviced.service

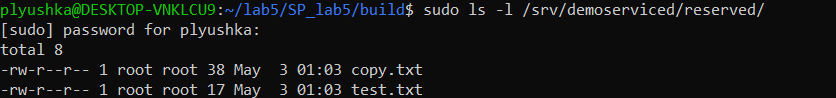


1. **Посмотреть последние логи демона**



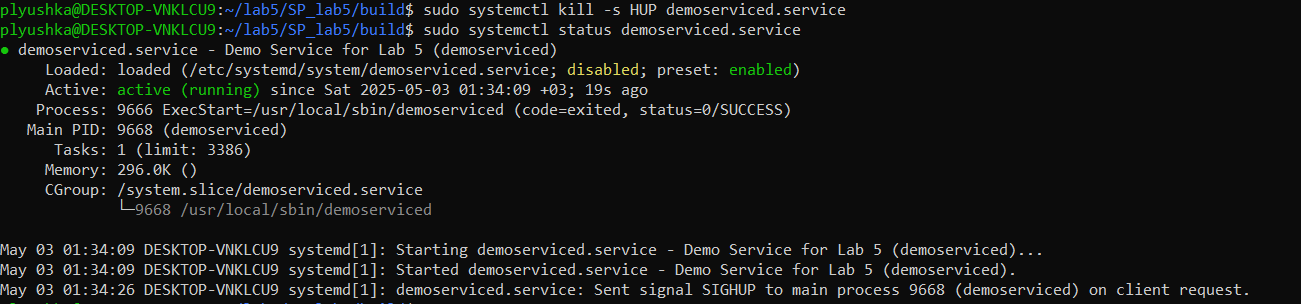
1. **Проверить, скопировались ли файлы (после ожидания Interval минут)**

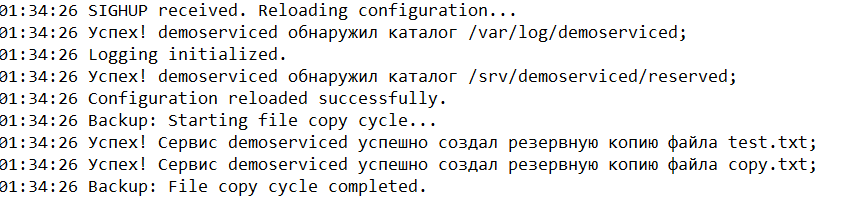
sudo ls -l /srv/demoserviced/reserved/

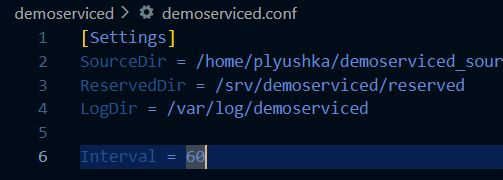


1. **Отправить сигнал SIGHUP для перезагрузки конфигурации демона**

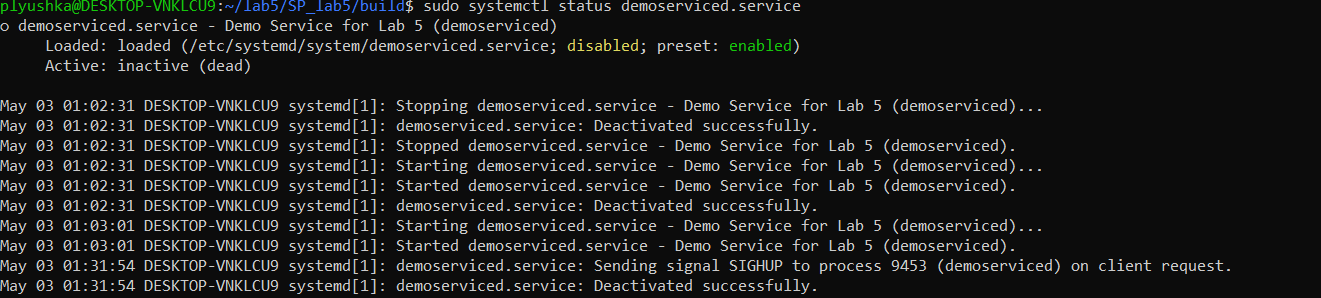
sudo systemctl kill -s HUP demoserviced.service

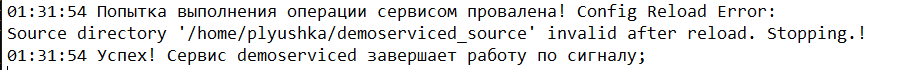








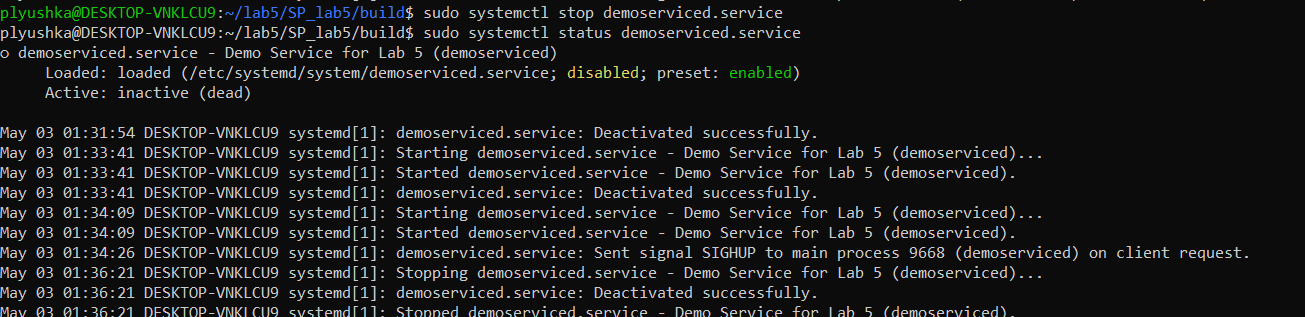




1. **Остановить службу и проверить статус**

sudo systemctl stop demoserviced.service

sudo systemctl status demoserviced.service



1. **Включить автозапуск службы при загрузке системы**

sudo systemctl enable demoserviced.service

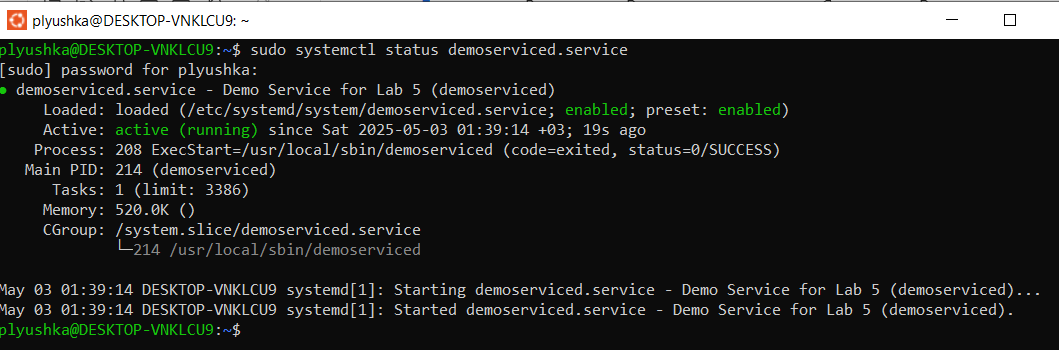


**Перезагрузить систему**

sudo reboot

**После перезагрузки проверить статус**

sudo systemctl status demoserviced.service



**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?**

Служба (или сервис) – это процесс, который выполняет служебные (вспомогательные) функции. Они являются аналогами резидентных программ и часто запускаются при загрузке ОС или по команде, работая до завершения работы ОС или до получения команды на остановку. Важно отметить, что не любая программа, стартующая с ОС, является службой.

1. **Какие бывают службы?**

Выделяются службы по их **функциям**:

* Серверы: Обеспечивают доступ к ресурсам, например, к базам данных.
* Драйверы: Обеспечивают доступ к внешним устройствам.
* Мониторы: Отслеживают работу других приложений.

**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)

1. **Чем можно охарактеризовать службы?**

**Характеристики**:

* Работают только в фоновом режиме.
* Не имеют собственного интерфейса управления (ни GUI, ни TUI).
* Управляются специальной программой ОС – менеджером служб (SCM).
* Запускаются/останавливаются при старте/выключении ОС, входе/выходе пользователя или по команде от менеджера служб.
* Предназначены для предоставления услуг другим программам или ОС, а не напрямую 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.

1. **Где располагается информация о сервисе?**

Информация об установленных сервисах Windows хранится в реестре Windows, в разделе:  
**HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services**

Внутри этого раздела для каждой службы (и драйвера) существует свой подраздел, содержащий параметры конфигурации, такие как ImagePath (путь к файлу), Type (тип службы), Start (тип запуска), DisplayName, DependOnService, DependOnGroup и другие.

1. **Что такое Linux-демон?**

Демон (daemon) как процесс в Linux, обладающий следующими свойствами:

* Имеет длинный жизненный цикл (часто работает от загрузки до выключения системы).
* Выполняется в фоновом режиме.
* Не имеет контролирующего терминала.

Лабораторная работа добавляет, что демоны обычно обрабатывают сигналы, такие как SIGTERM (для остановки) и SIGHUP (для перезагрузки конфигурации).

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.