

# Лабораторная работа №10

## Управление системой systemd.

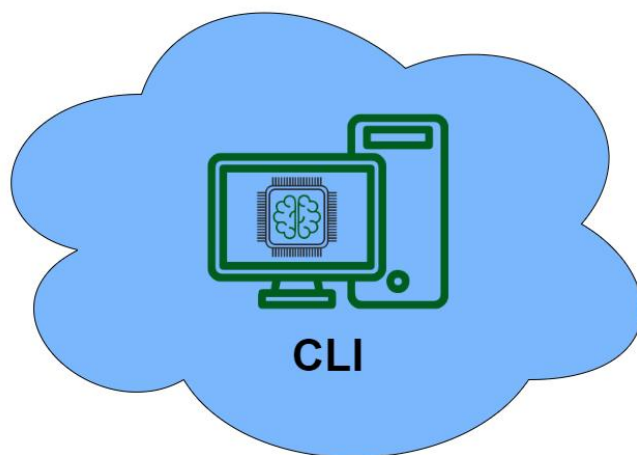
### Конфигурирование unit-файлов

#### Цель работы:

- Изучить основы управления системой с помощью systemd в операционной системе ALT Linux, понять структуру и конфигурацию unit-файлов, а также научиться создавать и редактировать собственные unit-файлы для управления сервисами, задачами и другими системными объектами.

#### Оборудование, ПО:

- ALT1
- Справочная литература или доступ в сеть интернет.



Машина	IP	Маска	Шлюз	DNS
CLI (Альт Рабочая станция 10.2)	192.168.1.1	24	-	77.88.8.8

#### Порядок работы:

1. Переименовать ПК своей фамилией
2. Вставить скриншоты результатов работы каждой команды
3. Запустить на ПК различные приложения, браузеры, и приступить к выполнению контрольных заданий.

**Контрольные вопросы:**

1. Что такое systemd, и какие задачи он решает в современных Linux-системах?
2. Какие типы unit-файлов существуют в systemd, и каковы их основные отличия?
3. Что такое таргеты (targets) в systemd, и как они используются для управления состояниями системы?
4. Как работают секции [Unit], [Service], [Install] в unit-файле? Приведите примеры использования каждой секции.
5. Как создать зависимость между сервисами, чтобы один сервис запускался только после успешного завершения другого?

**Контрольные задания:**

1. Добавьте сервис sshd в автозагрузку. Убедитесь в этом. Далее замаскируйте его. Докажите, что он замаскирован.
2. Отобразите список всех загруженных в данный момент юнитов путей.
3. Создайте собственный unit-файл для запуска скрипта, который записывает текущую дату и время в файл. Убедитесь, что скрипт запускается как сервис с помощью systemd.
4. Создайте собственный unit-файл для запуска скрипта, который будет запускаться каждые 15 минут и выводить информацию о всех процессах в системе и нагрузке на систему в целом.
5. Настройте пользовательский таргет, который включает сервисы sshd и squid. Проверьте его работу, переключаясь между вашим таргетом и стандартными таргетами systemd (например, multi-user.target).

**Используемые источники:**

- Официальная документация операционной системы Альт Рабочая Станция 10.4 <https://docs.altlinux.org/ru-RU/alt-workstation/10.4/html/alt-workstation/index.html>
- <https://www.freedesktop.org/software/systemd/man/latest/systemd.service.html> - MAN страница по синтаксису юнитов .service
- <https://www.freedesktop.org/software/systemd/man/latest/systemd.unit.html> MAN страница по синтаксису юнитов systemd

Помните, когда запускается программа - появляется соответствующий процесс. Но если посмотреть список процессов на свежезапущенной операционной системе, мы увидим более сотни процессов, хотя я всего лишь запустил эмулятор терминала, `bash`. Значит, всё остальное было запущено другими программами.

Демон `systemd` управляет запуском системы Linux, включая запуск служб и управление службами в целом. Он активирует системные ресурсы, серверные демоны и другие процессы как во время начальной загрузки, так и в работающей системе.

Демоны — это процессы, которые ожидают или работают в фоновом режиме, выполняя различные задачи. Как правило, демоны запускаются автоматически во время начальной загрузки системы и продолжают выполняться, пока система не будет выключена или они не будут остановлены вручную. По соглашению имена многих демонов заканчиваются на букву `d`.

Под службой в контексте `systemd` часто подразумевается один или несколько демонов, однако запуск или остановка службы может вызвать единовременное изменение состояния системы, не предполагающее оставления процесса демона в запущенном состоянии (так называемый `oneshot`).

Возможности, предоставляемые службой `systemd`:

- возможности распараллеливания (одновременный запуск нескольких служб), которые увеличивают скорость начальной загрузки системы;
- запуск демонов по запросу без использования отдельной службы;
- автоматическое управление зависимостями служб для предотвращения длительных периодов ожидания; (например, зависящая от сети служба не будет пытаться запуститься, пока сеть не будет доступна);
- совместное отслеживание связанных процессов с использованием контрольных групп (`control group`) Linux.

`systemd` использует юниты для управления различными типами объектов. Ниже приведены некоторые распространенные типы юнитов.

- Юниты служб имеют расширение **.service** и представляют системные службы. Этот тип юнита используется для запуска демонов, к которым часто происходит обращение, например веб-сервера.
- Юниты сокетов имеют расширение **.socket** и представляют сокет для межпроцессного взаимодействия (IPC), которые служба `systemd` должна отслеживать. Если клиент подключается к сокету, `systemd` запускает демон и передает ему соединение. Юниты сокетов используются для отложенного запуска служб во время начальной загрузки системы, а также для запуска редко используемых служб по запросу.
- Юниты путей имеют расширение **.path** и используются для отложенной активации служб, которая выполняется при определенном изменении в файловой системе. Такая схема обычно применяется для служб, которые используют каталоги очереди, например в системе печати.

Для управления юнитами используется команда `systemctl`. Например, с помощью команды **`systemctl -t help`** можно отобразить доступные типы юнитов.

Результат работы

Используйте команду `systemctl` для просмотра текущего состояния системы. Например, следующая команда отображает список всех загруженных в данный момент юнитов служб, разбивая вывод на страницы при помощи команды `less`.

```
$ systemctl list-units --type=service
```

Результат работы

В приведенном выше выводе опция `--type=service` ограничивает отображаемые типы юнитов юнитами служб. В выводе используются следующие столбцы.

**UNIT**

Имя юнита службы.

**LOAD**

Указывает, смогла ли служба `systemd` правильно проанализировать конфигурацию юнита и загрузить юнит в память.

**ACTIVE**

Общие сведения о состоянии активации юнита. Указывает, был ли юнит успешно запущен.

**SUB**

Дополнительные сведения о состоянии активации юнита. Здесь представлены более подробные сведения о юните, которые зависят от типа юнита, его состояния и способа выполнения.

**DESCRIPTION**

Это краткое описание юнита.

По умолчанию команда **`systemctl list-units --type=service`** отображает только юниты служб с состоянием активации `active`. Опция **`--all`** отображает все юниты служб, независимо от состояния активации.

Найдите неактивные юниты в вашей системе

Команда **systemctl** без аргументов отображает список юнитов, которые загружены и при этом активны.

Результат работы

## Просмотр состояний служб

Просмотреть состояние конкретного юнита можно с помощью команды **systemctl status name.type**. Если тип юнита не указан, команда **systemctl** отображает состояние юнита службы, если он существует.

Выведите состояния юнита сервиса печати cups

Эта команда отображает текущее состояние службы. Значения полей:

Поле Описание

Loaded—Указывает, загружен ли юнит службы в память.

Active—Указывает, запущен ли юнит службы, и если да, то как долго он работает.

Main PID—Основной идентификатор процесса службы, включая имя команды.

Status—Дополнительная информация о службе.

loaded	Файл конфигурации юнита был обработан.
--------	--

active (running)	Служба запущена с одним или несколькими выполняющимися процессами.
active (exited)	Единоновременная настройка успешно завершена.
active (waiting)	Служба запущена, но ожидает события.
inactive	Служба не запущена.
enabled	Служба запускается во время начальной загрузки системы.
disabled	Служба не настроена для запуска во время начальной загрузки системы.
статического	Служба не может быть включена, но может быть автоматически запущена включенным юнитом.

Команда `systemctl` позволяет проверить определенные состояния службы. Используйте следующие команды, чтобы убедиться, что юнит службы в данный момент активен, загружен или произошел ли в нем сбой.

<code>systemctl is-active cups.service</code>

<code>systemctl is-enabled cups.service</code>

systemctl is-failed cups.service

## Запуск и остановка служб

Есть много причин, почему может потребоваться вручную остановить или запустить службу. Среди них — обновление службы, изменение файла конфигурации и удаление службы. Кроме того, администратор может вручную запустить редко используемую службу.

Для остановки службы используется команда

```
$ systemctl stop [имя.службы]
```

systemctl stop cups.service

Для запуска службы используется команда

```
$ systemctl start[имя.службы]
```

systemctl start cups.service

## Перезапуск и перезагрузка служб

Во время перезапуска работающей службы она останавливается, а затем запускается. При этом идентификатор процесса изменяется, и при запуске служба получает новый идентификатор процесса. Чтобы перезапустить работающую службу, используйте аргумент **restart** с командой `systemctl`. Перезапустите службу `cups.service`.

systemctl	restart	cups.service.
Посмотрите статус до и после перезапуска. Какие изменения вы видите?		

Некоторые службы могут перезагружать собственные файлы конфигурации без перезапуска. Этот процесс называется перезагрузкой службы. Перезагрузка службы не изменяет идентификатор процесса, связанный с различными процессами службы. Чтобы перезагрузить работающую службу, используйте аргумент **reload** с командой systemctl. Перезагрузите службу cups.service.

systemctl reload cups.service.

Не получается? Подумайте и ответьте выше - почему?

systemctl reload sshd.service.
Посмотрите статус до и после перезапуска. Какие изменения вы видите?

## Отображение списка зависимостей юнитов

Некоторые службы требуют, чтобы сначала запускались другие службы, создавая зависимости от других служб. Другие службы могут не запускаться во время начальной загрузки, а запускаться только по требованию. В обоих случаях systemd и systemctl запускают службы по мере необходимости для разрешения зависимостей и запуска редко используемых служб. Например, если служба печати CUPS не запущена и в каталог очереди печати помещен файл, система запустит демоны или команды, относящиеся к службе CUPS, чтобы выполнить запрос на печать.



Чтобы полностью остановить службы печати в системе, необходимо остановить все три юнита. Отключение службы приводит к отключению ее зависимостей.

Команда **systemctl list-dependencies UNIT** отображает иерархическое сопоставление зависимостей для запуска юнита службы. Для отображения обратных зависимостей (юнитов, зависящих от указанного юнита) используйте с командой опцию **--reverse**.

```
systemctl list-dependencies cups.service
```

Поясните вывод

```
systemctl list-dependencies --reverse cups.service
```

Поясните вывод

## Маскировка и демаскировка служб

Иногда в системе могут оказываться конфликтующие службы. Например, существует несколько способов управления почтовыми серверами (среди них — postfix и sendmail). Маскировка службы предотвращает случайный запуск администратором службы, конфликтующей с другими службами. Маскировка создает в каталогах конфигурации ссылку на файл /dev/null, предотвращающий запуск службы.

Для маскировки используется команда

```
$ systemctl mask [имя_юнита]
```

```
systemctl mask cups.service
```

Остановите и заново запустите службу cups. Поясните почему вам удалось/не

удалось ее запустить.

Используйте команду **systemctl unmask** для демаскировки юнита службы.

systemctl unmask cups.service

Отключенная служба может быть запущена вручную или другими юнитами, но она не запускается автоматически во время начальной загрузки. Замаскированная служба не может быть запущена ни вручную, ни автоматически.

Запуск службы в работающей системе не гарантирует, что эта служба будет автоматически запущена после перезагрузки системы. Аналогичным образом остановка службы в работающей системе не предотвращает ее запуска после перезагрузки системы. Чтобы служба могла запускаться во время начальной загрузки, необходимо создать соответствующие ссылки в каталогах конфигурации systemd. Эти ссылки создаются и удаляются командой **systemctl**.

Чтобы служба могла запускаться во время начальной загрузки, используйте команду **systemctl enable**, тем самым добавив ее в автозагрузку.

systemctl enable sshd.service
Докажите, что ваша служба добавлена в автозагрузку.

Приведенная выше команда создает символическую ссылку из файла юнита службы (обычно в каталоге `/usr/lib/systemd/system`) на то место на диске, где команда `systemd` ищет файлы (в каталоге `/etc/systemd/system/TARGETNAME.target.wants`). Включение службы не

запускает службу в текущем сеансе. Чтобы запустить службу и включить ее автоматический запуск во время начальной загрузки, выполните команды `systemctl start` и `systemctl enable`.

Чтобы отключить автоматический запуск службы, используйте команду **`systemctl disable`**. Она удаляет символическую ссылку, созданную при включении службы. Обратите внимание, что отключение службы не приводит к ее остановке.

```
systemctl disable sshd.service
```

Задача	Команда
Просмотр подробной информации о состоянии юнита.	<b><code>systemctl status UNIT</code></b>
Остановка службы в работающей системе.	<b><code>systemctl stop UNIT</code></b>
Запуск службы в работающей системе.	<b><code>systemctl start UNIT</code></b>
Перезапуск службы в работающей системе.	<b><code>systemctl restart UNIT</code></b>
Перезагрузка файла конфигурации работающей службы.	<b><code>systemctl reload UNIT</code></b>
Полное отключение запуска службы (как вручную, так и во время начальной загрузки).	<b><code>systemctl mask UNIT</code></b>
Включение доступа к замаскированной службе.	<b><code>systemctl unmask UNIT</code></b>

Настройка службы на запуск во время начальной загрузки.	<b>systemctl enable <i>UNIT</i></b>
Отключение запуска службы во время начальной загрузки.	<b>systemctl disable <i>UNIT</i></b>
Отображение списка юнитов, необходимых для указанного юнита.	<b>systemctl list-dependencies <i>UNIT</i></b>

## Создание простого systemd unit

unitd работает с сервисами описанными в своей конфигурации. Основой для создания сервиса служит юнит (unit) – это текстовый файл с описанием на подобии ini файла в системе windows. Конфигурационный файл состоит из секций. Внутри каждой секции указываются необходимые параметры. Обязательными во всех юнитах являются две секции, остальные используются в зависимости от типа юнита.

Каталоги хранения юнитов

**/lib/systemd/system** – юниты поставляемые вместе с системой и устанавливаемыми приложениями

**/run/systemd/system** – юниты созданные динамически (в рантайме)

**/etc/systemd/system** – юниты системного администратора (тут и будем хранить наши)

Всего в системе существуют следующие типы юнитов:

- **target** — группирует модули
- **service** — отвечает за запуск сервисов (служб) и поддерживает вызов интерпретаторов для исполнения пользовательских скриптов
- **mount** — занимается монтированием файловых систем
- **automount** — автомонтирование файловых систем, используется при обращении к точке монтирования;
- **swap** — отвечает за подключение файла подкачки
- **timer** — запускает модули по расписанию, аналог cron
- **socket** — запуск модуля при подключению к сокету
- **slice** — группировка других модулей в контейнер (дерево) cgroups
- **device** — использует реакцию на подключение какого-либо устройства
- **path** — запуск модуля по событию доступа по конкретному пути в файловой системе

Разберем синтаксис юнита на примере сервиса sshd

Откроем файл по пути **/lib/systemd/system/sshd.service**

```
root@alt: /usr/lib/systemd
Файл Правка Вид Поиск Терминал Помощь
[Unit]
Description= ssh rabotaet
After=syslog.target network.target

[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStartPre=/usr/bin/ssh-keygen -A
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $EXTRAOPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=always

[Install]
WantedBy=multi-user.target
```

Видим 3 секции.

## [Unit]

Первый раздел, который есть в большинстве юнит-файлов. Обычно его используют для определения метаданных устройства и настройки отношения юнита к другим службам/юнитам.

В нашем примере есть 2 поля:

**Description** – описание юнита для большего понимания пользователя. Это поле отображается в статусе

**After** – зависимость, т.е. в данном случае запускать юнит только после запуска **network.target** и **sshd-keygen.target**

Но глобально тут может быть объявлены следующие параметры:

- **Documentation=:** Эта директива предоставляет местоположения документации. Это могут быть внутренние доступные справочные страницы, либо доступные в интернете (URL-адреса).
- **Requires=:** В этой директиве перечислены все юниты, от которых зависит этот юнит (служба или устройство). Если текущий блок активирован, перечисленные здесь юниты также должны успешно активироваться, иначе он потерпит неудачу. Эти блоки запускаются параллельно с текущим устройством по умолчанию.
- **Wants=:** Эта директива похожа на **Requires=**, но менее строгая. **Systemd** будет пытаться запустить любые юниты, перечисленные здесь, когда это устройство активировано. Если эти устройства не обнаружены или не запущены,

текущий блок будет продолжать функционировать. Это рекомендуемый способ настройки большинства зависимостей. Опять же, это подразумевает параллельную активацию, если не изменено другими директивами.

- **BindsTo=:** Эта директива аналогична **Requires =**, но также приводит к остановке текущего устройства, когда соответствующий узел завершается.
- **Before=:** Юниты, перечисленные в этой директиве, не будут запущены до тех пор, пока текущий блок не будет отмечен как запущенный, если они будут активированы одновременно.
- **After=:** Юниты, перечисленные в этой директиве, будут запущены до запуска текущего устройства (юнита).
- **Conflicts=:** Данный юнит можно использовать для отображения юнитов, которые нельзя запускать одновременно с текущим устройством. Запуск устройства с этой связью приведет к остановке других устройств.

## **[Service ]**

**EnvironmentFile** – файлы переменного окружения

**ExecStartPre**–пути к файлам, которые должны быть выполнены до старта юнита.

**ExecStart** – полный путь к исполняемому файлу программы с параметрами запуска

**ExecReload** – полный путь к исполняемому файлу программы с параметрами перезапуска программы

**KillMode** – указывается как будет завершен процесс. В данном случае параметр **process** говорит о том что будет закрыт только главный процесс

**Restart** – перезагрузка процесса, параметр **on-failure** указывает на автоматическую перезагрузку в случае отказа процесса

Также тут может быть указан тип запуска юнита:

### **Type –**

**simple** (по умолчанию) – происходит незамедлительный запуск этой службы, с учетом того что процесс не разветвляется (**fork**). Не используйте **simple** если пользуетесь очередностью запуска.

**forking** – служба считается запущенной после того, после разветвления процесса с завершением родительского процесса. Используется для запуска классических демонов исключая случаи, когда в таком поведении процесса нет необходимости. Также желательно указать **PIDFile=**, чтобы **systemd** мог отслеживать основной процесс.

**oneshot** – удобен для скриптов, которые выполняют одно задание и завершаются.

### **[Install]**

**WantedBy** – указывает на каком уровне запуска стартует сервис, параметр **multi-user.target** указывает на запуск в многопользовательском режиме без графики, параметр **graphical.target** указывает на запуск только в графическом режиме.

Параметров в синтаксисе юнитов множество, при написании собственного юнита обратите внимание на справочные материалы

Проанализируйте юнит файл сервиса <b>sshd</b> . Напишите текстом логику его работы, что и в какой последовательности выполняется. Какие есть особенности у <b>sshd</b> исходя из синтаксиса юнита?

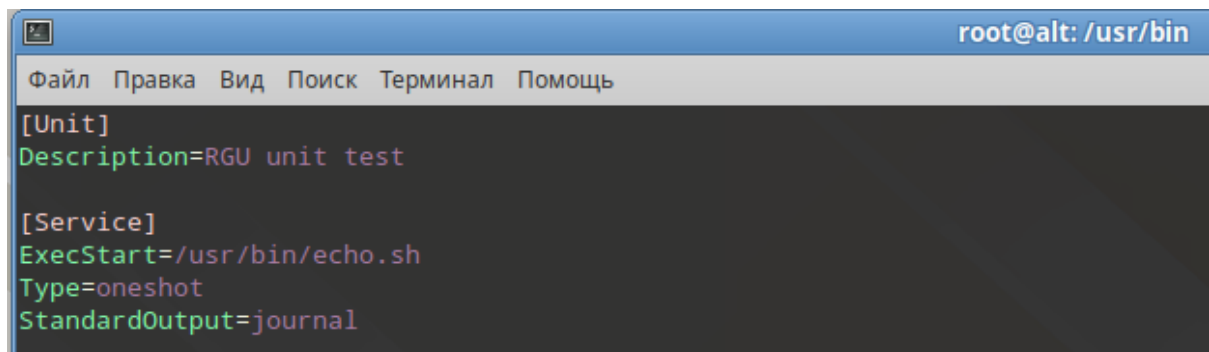
**Как пример, создадим простейший юнит, который будет отправлять всем залогиненным пользователем в систему сообщения.**

Перейдем в директорию хранения юнита и с помощью команды **touch** создадим его.

```
[root@alt systemd]# cd /etc/systemd/system/  
[root@alt system]# touch echotext.service
```

Самое содержание будет выглядеть следующим образом:





```
root@alt: /usr/bin
Файл Правка Вид Поиск Терминал Помощь
[Unit]
Description=RGU unit test

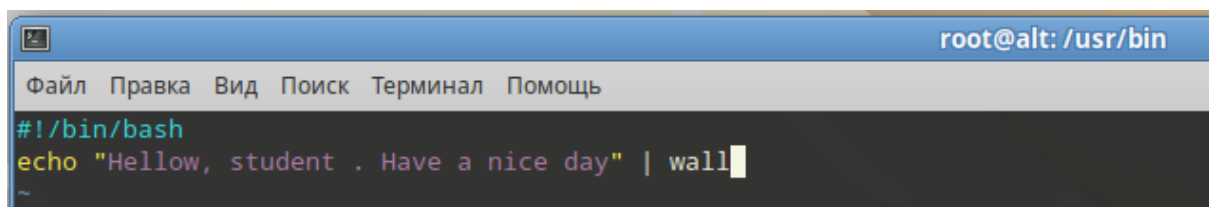
[Service]
ExecStart=/usr/bin/echo.sh
Type=oneshot
StandardOutput=journal
```

Добавим описание юнита. В поле ExecStart укажем расположение нашего будущего скрипта. И стандартный вывод укажем в журнал только.

Далее перейдем в нужную директорию и создадим исполняемый файл.

```
[root@alt system]# cd /usr/bin/
[root@alt bin]# touch echo.sh
[root@alt bin]# vim echo.sh
[root@alt bin]# chmod +x echo.sh
```

В теле скрипта будем использовать командный интерпретатор bash. С помощью команды wall будем отправлять сообщение всем пользователям в системе.



```
root@alt: /usr/bin
Файл Правка Вид Поиск Терминал Помощь
#!/bin/bash
echo "Hellow, student . Have a nice day" | wall
~
```

Для того чтобы юнит был опознан системой нам необходимо выполнить следующую команду.

```
[root@alt bin]# systemctl daemon-reload
```

Данная команда перечитывает все файлы юнитов в вашей системе.

Юнит готов. Можем его запускать.

```
[root@alt bin]# systemctl start echotext.service
```

Результат работы
------------------

--

Далее посмотрим его статус

```
[root@alt bin]# systemctl status echotext.service
```

Результат работы