Семинар #9: systemd. Практика.

Как сдавать задачи

Для сдачи ДЗ вам нужно создать репозиторий на GitLab (если он ещё не создан) под названием `devtools-homework`. Структура репозитория должна иметь вид:

```
├── seminar9_systemd/
│   ├── 01.sh
│   ├── 02.sh
│   └── ...
└── ...
```

Для каждой задачи, если не сказано иное, нужно создать один файл решения с расширением `.sh`. Если в задаче есть подзадачи, то их нужно оформлять внутри каждого из файлов в следующем формате:

```
# Subtask a
systemctl status nginx
# Subtask b
...
```

Для каждой подзадачи нужно прописать все команды, которые исполняются в ходе выполнения этой подзадачи. Если в задаче встречается вопрос, то на этот вопрос нужно ответить в комментариях (начинаются с `#`) скрипта.

Задача 1. Включить сервис

В данной задаче будет показана простейшая работа с сервисами systemd: просмотр информации о сервисе, включение/отключение сервиса. В качестве примера сервиса был выбран веб сервис nginx. Если у вас не установлен nginx в системе, то нужно будет его установить:

- Для Debian-based дистрибутивов:

  ```
  $ sudo apt install nginx
  ```

- Для RHEL-based дистрибутивов:

  ```
  $ sudo dnf install nginx
  ```

Выполните подзадачи:

(a) Проверьте статус сервиса `nginx`:

```
$ systemctl status nginx
```

Ответьте на следующие вопросы о сервисе:

- Загружен ли этот юнит-файл сервиса в systemd и что на это указывает?
- Включён ли сервис в автозагрузку и что на это указывает?
- Активен ли сервис сейчас и что на это указывает?

(b) Если сервис активен в данный момент, то остановите его командой:

```
$ sudo systemctl stop nginx
```

Проверьте теперь статус сервиса nginx. Что изменилось в статусе сервиса? Откройте любой браузер и зайдите по адресу `http://localhost`. Если сервис nginx не активен, то браузер должен выдать ошибку.

(c) Активируйте сервис командой:

```
$ sudo systemctl start nginx
```

Проверьте статус сервиса. Откройте браузер и зайдите по адресу `http://localhost`. Если сервис nginx активен, то браузер должен выдать страницу веб-сервера nginx по умолчанию (Welcome to nginx!).

(d) Используйте

```
$ sudo systemctl enable nginx
```

чтобы включить автозапуск сервиса при загрузке системы и

```
$ sudo systemctl disable nginx
```

чтобы отключить автозапуск. Проверьте статус сервиса после включения/отключения автозапуска сервиса.

Задача 2. Создаём свой простейший сервис

В данной задаче будет создан простейший сервис. Этот сервис мы назовём `alpaca`. При старте этого сервиса текущие дата и время будут записываться в файл `/var/log/alpaca.txt` .

(a) В директории `/var/log/` создайте файл `alpaca.txt` с владельцем `root` и группой владельцем `root`.

(b) В директории `/usr/local/bin` создайте скрипт `alpaca.sh`, который должен дозаписывать текущую дату и время в конец файла `/var/log/alpaca.txt`. Дайте этому скрипту права на исполнение.

(c) Перейдите в `/etc/systemd/system` и создайте там юнит-файл сервиса по имени `alpaca.service`. Содержимое юнит-файла должно быть:

```
[Unit]
Description=Simplest service, writes to the file /var/log/alpaca.txt once

[Service]
ExecStart=/usr/local/bin/alpaca.sh

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

В этом файле:

- `Description` – описание сервиса
- `ExecStart` – команда, которая будет исполнятся при старте сервиса
- `WantedBy` – эта строка говорит, к какому таргету будет подключаться данный сервис при вызове `systemctl enable`. `multi-user.target` – это таргет, который исполняется при загрузке системы. Соответственно, эта настройка означает, что сервис `alpaca` будет запускаться при запуске системы (если его автозапуск включен через `systemctl enable`).

(d) Вызовите

```
$ sudo systemctl daemon-reload
```

для того, чтобы systemd перечитал конфигурационные файлы юнитов и добавил наш юнит в систему.

(e) Проверьте, что наш сервис добавился, используя:

```
$ systemctl status alpaca
```

(f) Запустите сервис `alpaca`

```
$ sudo systemctl start alpaca
```

В результате этого должен запуститься сервис `alpaca`, а это значит, что запустится скрипт `alpaca.sh`. Проверьте что в файл `/var/log/alpaca.txt` будет дозаписываться строка при каждом запуске сервиса `alpaca`.

(g) Включите сервис в автозапуск с помощью `systemctl enable`. Перезагрузите виртуальную машину и убедитесь, что сервис запустился при запуске программы, посмотрев содержимое файла `/var/log/alpaca.txt`.

(h) Если посмотреть статус нашего сервиса, то будет отображаться, что он неактивен. Сервис неактивен, так как `alpaca.sh` быстро отрабатывает и завершается. По умолчанию (если тип сервиса `simple`) в этом случае он считается неактивным. Измените скрипт `/usr/local/bin/alpaca.sh` так, чтобы он:

- Сначала дозаписывал в конец файла `/var/log/alpaca.txt` строку

  ```
  Started < дата и время >
  ```

- Затем ожидал 30 секунд
- Наконец, дозаписывал в конец файла `/var/log/alpaca.txt` строку

  ```
  Finished < дата и время >
  ```

После этого запустите сервис и посмотрите его статус. В течении 30 секунд этот статус должен быть `active`.

(i) Посмотрите на все запущенные сервисы:

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

Если ваш сервис активен, то он отобразится в этом списке.

Задача 3. Сервис логов

Создайте сервис который бы записывал в файл `/var/log/mylog.log` текущую дату и время, загрузку процессора и количество занятой оперативной памяти каждые 30 секунд (используйте `sleep 30`). Сервис должен быть активен пока его не остановят.

Для решения этой задачи создайте файл `03.sh` в котором бы содержались все команды нужные для решения данной задачи. Также создайте файл `u03.service`, который бы представлял собой юнит-файл данного сервиса. Также создайте файл `s03.sh`, который бы представлял собой скрипт, запускаемый данным сервисом.

Задача 4. Сервис логов по таймеру

Создайте сервис который бы записывал в файл `/var/log/mylog2.log` текущую дату и время, загрузку процессора и и количество занятой оперативной памяти каждые две минуты. В этой задаче не используйте `sleep`. Вместо этого используйте systemd timers.

Для решения этой задачи создайте файл `04.sh` в котором бы содержались все команды нужные для решения данной задачи. Также создайте файл `u04.service`, который бы представлял собой юнит-файл данного сервиса. Также создайте файл `s04.sh`, который бы представлял собой скрипт, запускаемый данным сервисом.

Задача 5. Бессмертный файл

Создайте файл `immortal.txt` с содержимым `"I am immortal"`. Создайте сервис, который будет следить за этим файлом. В случае удаления этого файла, сервис должен сразу же восстанавливать его с тем же содержимым. В случае изменения содержимого файла, его содержимое должно сразу меняться на изначальное `"I am immortal"`. Решите эту задачу двумя способами:

(a) Неэффективный подход с использованием одного сервиса, который будет запускать демона. Этот демон должен будет проверять состояние файла каждые пять секунд и восстанавливать его.

Для решения этой подзадачи создайте файл `05a.sh` в котором бы содержались все команды нужные для решения. Также создайте файл `u05a.service`, который бы представлял собой юнит-файл данного сервиса. Также создайте файл `s05a.sh`, который бы представлял собой скрипт, запускаемый данным сервисом.

(b) Правильный подход с использованием дополнительного path-юнита.

Для решения этой подзадачи создайте файл `05b.sh` в котором бы содержались все команды нужные для решения задачи. Также создайте файл `u05b.service`, который бы представлял собой юнит-файл данного сервиса и файл `u05b.path`, который бы представлял собой файл path-юнита. Также создайте файл `s05b.sh`, который бы представлял собой скрипт, запускаемый данным сервисом.