

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 в самой задаче не сказано иное, нужно создать 1 скрипт с расширением `.sh` и шебангом в начале скрипта. Если задача делится на подзадачи нужно, если в самой задаче не сказано иное, создать скрипт для каждой подзадачи. Названия файлов решений для всех задач/подзадач должны начинаться с номера задачи, например `01.sh` или `04b.sh`, даже если в условии задачи используется другое имя для скрипта.

Задача 1. Включить сервис

- (a) Проверьте, установлен ли на вашей системе http-сервер `nginx`, выполнив команду

```
$ apt list --installed nginx
```

Если пакет не установлен, то программа ничего не напечатает.

- (b) Проверьте, есть ли в системе сервис `nginx`:

```
$ systemctl status nginx
```

- (c) В Linux откройте браузер и зайдите по адресу `http://localhost`. Если страница не открывается, то либо `nginx` не установлен, либо сервис не запущен.

- (d) Если `nginx` не был установлен, то установите его:

```
$ sudo apt install nginx
```

Используйте вместо `apt` пакетный менеджер вашей системы.

- (e) Проверьте теперь статус сервиса `nginx`. Затем запустите сервис с помощью команды:

```
$ sudo systemctl start nginx
```

- (f) В браузер и зайдите по адресу `http://localhost`. Теперь сервис должен работать и в браузере для вас (и для всех, кто зайдёт на ваш сервер по `http`) должно отображаться сообщение от `nginx`.

- (g) Остановите сервис

```
$ sudo systemctl stop nginx
```

и снова зайдите в браузере на локальный адрес.

- (h) Используйте

```
$ sudo systemctl enable nginx
```

чтобы сервис включался при запуске системы и

```
$ sudo systemctl disable nginx
```

чтобы не включался. Проверьте статус сервиса после включения/отключения автозапуска сервиса.

Задача 2. Создаём свой простейший сервис

- (a) В вашей домашней директории создайте файл `a.txt` и скрипт `a.sh`, который должен дозаписывать текущую дату и время в конец файла `a.txt`. Добавьте скрипту права на исполнение для всех пользователей. В скрипте нужно использовать полный путь до файла `a.txt`, так как предполагается, что это файл может запускать `root`, у которого домашняя директория будет отличаться от вашей.
- (b) Перейдите в `/etc/systemd/system` и создайте там юнит-файл сервиса по имени `alpaca.service`. Нужно будет использовать `sudo`. Содержимое должно быть:

```
[Unit]
Description=Simplest server, writes to the file a.txt once

[Service]
ExecStart=/home/<user>/a.sh
User=<user>

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

Где вместо `<user>` подставьте имя пользователя, которого вы используете. В этом файле:

- `Description` – описание сервиса
- `ExecStart` – `bash`-скрипт, который будет исполняться при старте сервиса
- `User` – от имени этого пользователя будет запускаться скрипт (по умолчанию будет `root`).
- `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 скрипт `a.sh` будет исполняться при запуске системы.

- (c) Вызовите

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

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

- (d) Проверьте, что наш сервис добавился, используя:

```
$ systemctl status alpaca
```

- (e) Запустите сервис `alpaca`

```
$ systemctl start alpaca
```

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

- (f) Включите сервис в автозапуск с помощью `systemctl enable`. Перезагрузите виртуальную машину и убедитесь, что сервис запустился при запуске программы, посмотрев содержимое файла `a.txt`.
- (g) Если посмотреть статус нашего сервиса, то будет отображаться, что он неактивен. Сервис неактивен, так как скрипт `a.sh` быстро отрабатывает и завершается. Измените скрипт `a.sh` так, чтобы он

- Сначала дозаписывал в конец файла `a.txt` строку
`Started дата< и время>`
- Затем ожидал 30 секунд
- Наконец, дозаписывал в конец файла `a.txt` строку
`Finished дата< и время>`

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

- (h) 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/mylogs` 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. Используйте `Restart=always` в юнит файле.

Задача 4. Сервис логов по таймеру

Создайте сервис который бы записывал в файл `/var/log/mylogs` текущую дату и время, загрузку процессора и количество занятой оперативной памяти каждый день ровно в 12:00 дня. В этой задаче не используйте `sleep`. Вместо этого используйте `systemd timers`.

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) С использованием одного сервиса, который будет запускать демона. Этот демон должен будет проверять состояние файла каждые пять секунд и восстанавливать его.
- (b) С использованием дополнительного path юнита.