

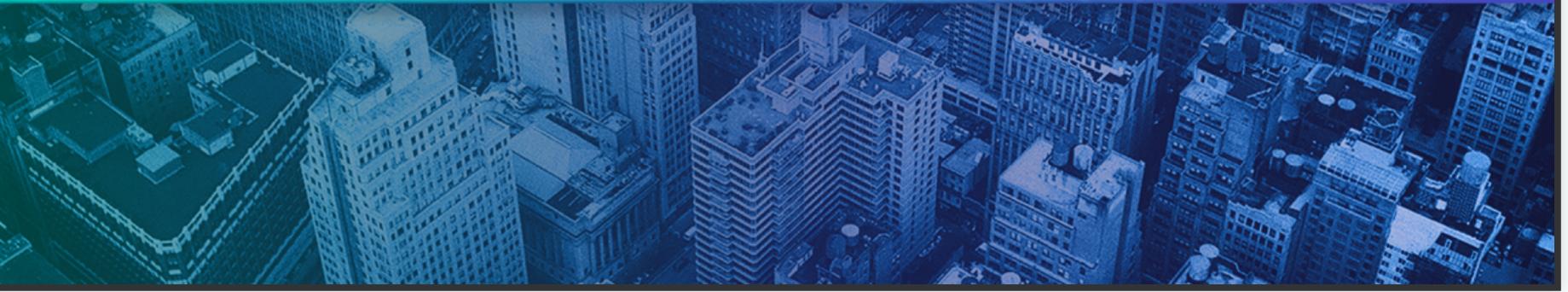


Онлайн-образование



Меня хорошо видно && слышно?

Ставьте  , если все хорошо
Напишите в чат, если есть проблемы



НЕ ЗАБЫТЬ ВКЛЮЧИТЬ
ЗАПИСЬ!!!

Инициализация системы. Systemd.

После занятия вы сможете

1. Написать и запустить Unit файл типа service, target, timer
2. Анализировать зависимости Unit файлов. Менять их при необходимости.
3. Уверенно чувствовать себя в работе с командами systemctl.

Зачем вам это уметь

ВАШ ВАРИАНТ?

Зачем вам это уметь

МОЙ ВАРИАНТ

1. Много деталей - нужно несколько подходов чтобы разобраться.
2. Диагностика ошибок.
3. Написать свои или модифицировать чужие сервисы.

План работы

1. Теория и определения
2. Пишем свой unit файл. Рассматриваем его параметры.
3. Зависимости, группы, шаблоны

Теория и определения

Вопросы

- Что такое systemd?
- Какие проблемы решает systemd?
- Сколько типов units поддерживается?
- Какие типы units знаете?

Немного истории. Упрощение администрирования

SysV	systemd
<pre>#!/bin/sh case \$1 of start) /usr/bin/mydaemon -p /var/run/mydaemon.pid ;; stop) kill \$(</pre> <p></var/run/mydaemon.pid</p> <pre>;; restart) \$0 stop; \$0 start ;; *) echo "what do you want from me, dude? esac</pre>	<pre>[Service] ExecStart=/usr/bin/mydaemon ExecStop= ExecRestart= PIDFile=/var/run/mydaemon.pid</pre>

Компоненты systemd

- Утилиты systemd

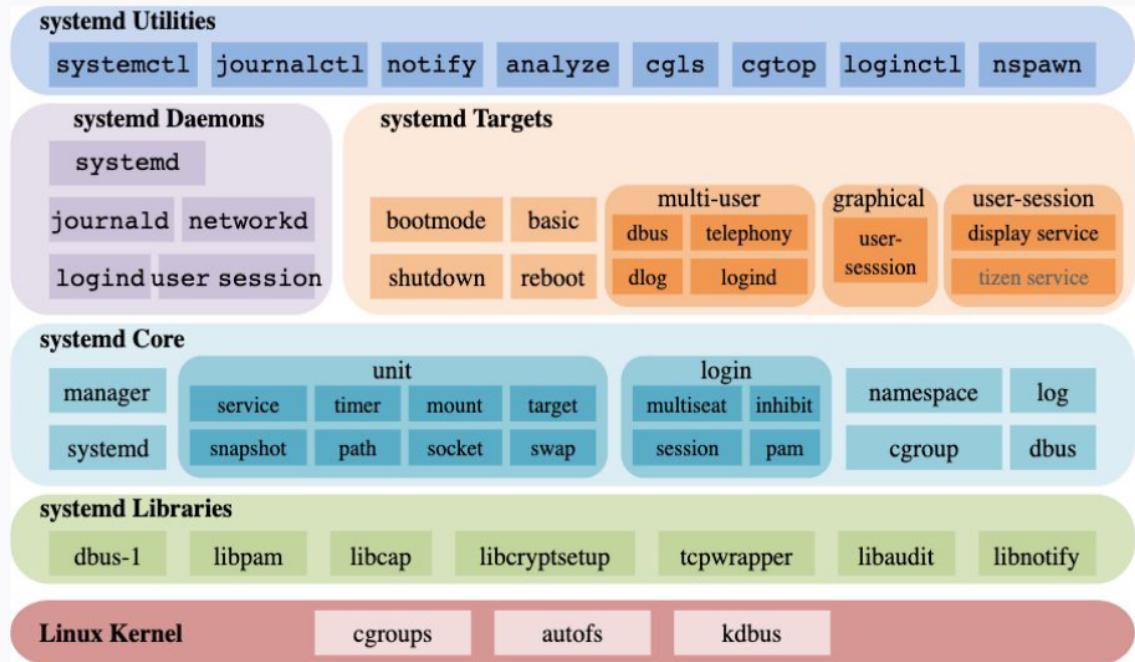
- Демоны systemd

- Таргеты systemd

- Базовые компоненты systemd

- Библиотеки systemd

- Компоненты ядра linux



Типы юнитов

- Сколько типов units поддерживается?
 - `man systemd #/CONCEPTS`
 - `systemctl -t help | wc -l`
- Units как правило имеют конфигурационные файлы и создаются во время старта системы. Но не все!
 - названия типов units используются в man pages и опциях и названиях команд, в названии конфигурационного файла
 - некоторые units генерируются автоматически (нет постоянного конфигурационного файла)

```
man systemd.timer  
man systemd.path  
man systemd.target
```

```
systemctl --type=device  
systemctl --type=target  
systemctl --type=path
```

```
systemctl list-sockets  
systemctl list-timers
```

Какие задачи решает systemd

- еще одна система инициализации
 - `systemctl status 1`
 - `man init`
 - `systemd-analyze plot > /vagrant/boot.svg`
- система зависимостей между units
 - `systemctl list-dependencies`
- контроль над процессами и разделение по группам
- контроль ресурсами системы
 - `systemctl status 721`
 - `systemctl status /dev/sda`
 - `systemctl status`

Вопросы

- Что такое `systemd`?
 - еще одна система инициализации
 - интерфейс управление системой
- Какие проблемы решает `systemd`?
 - упрощение администрирования
 - система зависимостей между units
 - эффективный контроль над процессами
- Сколько типов units поддерживается?
- Какие типы units знаете?

Пишем свой service unit файл

Где расположены unit файлы?

Системные	
/usr/lib/systemd/	основной каталог
/usr/lib/systemd/system/	юнит-файлы пакетов
/usr/lib/systemd/system/*.wants/	символические ссылки предопределенные systemd *
Админские	
/etc/systemd/	конфигурация
/etc/systemd/system	юнит файлы администратора *
/etc/systemd/system/*.wants/	символические ссылки при systemctl enable
/etc/sysconfig/	переменные

- `man systemd.unit`
- `systemd-analyze unit-paths`
- `systemd-analyze --user unit-paths`

Что такое service unit файл?

- имя файла заканчивается на .service
- в нем находится информация об управлении процессом, созданным systemd
- .ini файл с определенными секциями

```
[Unit] # общая для всех units. Документация, описание, зависимости и прочее  
[Service] # важно совпадает с типом unit. Минимально необходимая  
[Install] # используется для systemctl enable для создания символьических ссылок
```

- man systemd.unit

Создаем свой service unit

```
systemctl --user edit --full --force pyhttp
```

```
systemctl --user cat pyhttp
# /home/vagrant/.config/systemd/user/pyhttp.service
[Unit]
Description=Simple server

[Service]
ExecStart=/usr/bin/python3 -m http.server --bind 127.0.0.1 9000
```

```
systemctl --user status pyhttp
systemctl --user start pyhttp
systemctl --user status pyhttp
systemctl --user show pyhttpd # из файла на пару строчек
systemctl --user stop pyhttp
systemctl --user status pyhttp
```

Передаем переменные окружения

```
[Unit]
Description=Simple server

[Service]
Environment=IP=127.0.0.1
EnvironmentFile=%h/myservice
ExecStart=/usr/bin/python3 -m http.server --bind ${IP} ${PORT}
```

```
echo PORT=9001 > ~/myservice
```

```
systemctl --user show pyhttp | grep -i env
systemctl --user show pyhttp -p Environment
```

Конфигурация Units

Рассмотрим другие способы запуска сервиса

- Автоматическое восстановление сервиса
- Путь к файлу .path unit
- Время .time unit

Автоматическое восстановление сервиса

```
systemctl cat sshd.service
kill -9 $SSH_PID
systemctl status sshd
systemctl show sshd.service -p NRestarts
```

```
[Unit]
StartLimitIntervalSec=400 # время ожидания при перезапуске
StartLimitBurst=3 # количество попыток
[Service]
Restart=always
RestartSec=90
```

```
man systemd.service # /Restart= /Table 2. Exit causes
```

Другие способы запуска services

- .path unit

```
systemctl --user start example.path  
systemctl --user status example.path
```

- .time unit

```
systemctl --user start periodic-work.timer  
systemctl --user status periodic-work.timer  
systemctl list-timers  
systemd-analyze calendar "*-*-* *:*:01"
```

- .socket unit

Шаблоны

- Создаем **один** файл шаблона из которого генерируются юниты в момент вызова
- имя файла шаблона <service_name>@.service
 - systemctl start <service_name>@<argument>.service
 - argument - instance string можно получить внутри сервиса %i либо %l

```
# /home/vagrant/.config/systemd/user/otus@.service
[Unit]
Description=Otus demo service %i

[Service]
ExecStart=/usr/bin/echo hello from %i
```

```
systemctl --user start otus@testtesttest.service
systemctl --user status otus@testtesttest.service
journalctl -e
```

Вопросы

- Какие секции в .ini файле?
- Какие параметры запомнили?
- Какие способы запуска сервисов?

Зависимости. Группы сервисов.

Как создать зависимости units

- Создадим 3 сервиса
 - otus_basic
 - otus_prof
 - otus_adv
 - `systemctl --user status otus_adv otus_prof otus_basic`
- Порядок запуска не определен. Стартуют параллельно.
- Добавим к сервисам
 - зависимости состояния
 - зависимости порядка запуска
- Создадим target (группировка сервисов)

Requirement dependencies

- **Requirement dependencies** do not influence the order in which services are started or stopped
 - порядок старта сервисов не меняется
 - запускаются параллельно
- Другое название - state dependencies
 - Requires= важно состояние процесса, если процесс остановился с ошибкой, то unit остановится
- Wants= (A weaker version of Requires=), Requisite=, BindTo=
- man systemd.unit

Ordering dependencies

- Определяют порядок старта сервисов
 - Before=
 - After= sudo systemctl cat sshd

Изменим конфигурационный файл

```
.config/systemd/user/otus_basic.service
[Unit]
Description=Otus demo service 1
Before=otus_prof.service
```

```
systemd-analyze --user dot --from-pattern='linux*.target' \
| dot -Tsvg > /vagrant/target.svg
```

Targets

- Объединяем юниты в группы

```
cat > ~/.config/systemd/user/linux_course.target <<EOF
[Unit]
Description=OTUS target
Wants=otus_basic.service
Conflicts=rescue.service rescue.target
AllowIsolate=yes
After=default.target
[Install]
WantedBy=linux_course.target
EOF
```

```
systemctl --user list-dependencies linux_course.target
systemctl --user enable otus_basic.service
```

```
Wants=otus_basic.service otus_prof.service
systemctl --user daemon-reload
systemctl --user add-wants linux_course.target otus_adv
```

Стандартные targets

```
systemctl isolate  
systemctl default-target  
systemctl list-dependencies  
man systemd.special
```

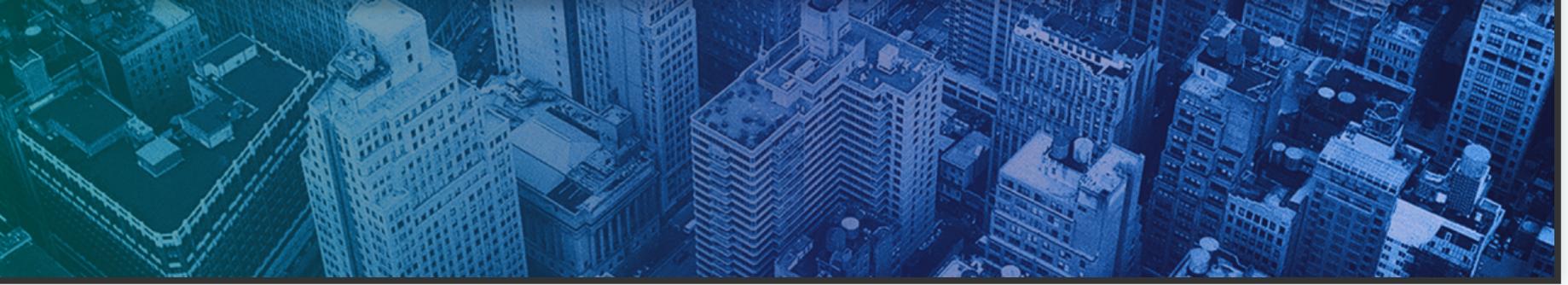
Рефлексия



Отметьте 3 пункта, которые вам запомнились с вебинара



Что вы будете применять в работе из сегодняшнего вебинара?



Заполните, пожалуйста,
опрос о занятии по ссылке в чате