Системные сервисы Systemd

Sys V init

- Управляет загрузкой при помощи скриптов
  Скрипты расположены в /etc/init.d

- Умеет разделять очередность выполнения скриптов по runlevel

Sys V init сосредоточен на инициализации системы

Считается устаревшим стандартом

Systemd

- Система инициализации и управления службами
- Распараллеливает процесс запуска, что приводит к высокой скорости загрузки

Сейчас systemd является стандартом

SysV vs Systemd

SysV

- Простой
- Легковесный
- Надежный

Systemd

- Быстрый
- Большой функционал
- Встроенное логирование

Структура Unit-файлов

[Unit]
Description=<описание юнита>
After=<после чего он должен загрузиться>
Requires=<что ему требуется для запуска>

[<тип юнита>]
<параметры юнита>

[Install]
WantedBy=<на каком этапе запускается автоматически>

Типы юнитов

- .**target** (группирует модули по уровню запуска. Эти уровни запуска показывает на каком этапе загрузки находится система)
- .**service** (юниты которые определяют сервисы, запускаются через systemd)
- .**mount** (монтирование систем)
- .**automount** (автомонтирование систем)

Типы юнитов

- **.swap** (юнит раздела подкачки. /etc/fstab, если он там указан)
- **.timer** (для запуска сервисов по расписанию)
- **.socket** (управляют сетевыми\файловыми сокетами)
- **.slice** (управление ресурсами системы через cgroups)
- **.device** (управление подключенными устройствами)
- **.path** (действия по событиям в файловой системе)
- **.scope** (создается только программно. Можно управлять процессами созданными вне system)

Типы юнита .service

```
[Unit]
Description=Entering Time Machine
After=logRuin@user.service

[Service]
Type=simple
ExecStart=/bin/bash /root/Desktop/timeMachine.sh

[Install]
WantedBy=default.target
```

[Service]
Type=<тип сервиса>

- Type=simple (тип по умолчанию, сервис запустится незамедлительно и процесс не должен создавать другие процессы)
- Type=forking (Используется когда нужно создавать дочерние процессы)

Типы юнита .service

```
[Unit]
Description=Entering Time Machine
After=logRuin@user.service

[Service]
Type=simple
ExecStart=/bin/bash /root/Desktop/timeMachine.sh

[Install]
WantedBy=default.target
```

[Service]
Type=<тип сервиса>

- Type=oneshot (Сервис завершается по выполнении задачи)
- Type=notify (такой же как и simple, но по завершению отправит сообщение)

Типы юнита .service

```
[Unit]
Description=Entering Time Machine
After=logRuin@user.service

[Service]
Type=simple
ExecStart=/bin/bash /root/Desktop/timeMachine.sh

[Install]
WantedBy=default.target
```

[Service]
Type=<тип сервиса>

- Type=dbus (сервис будет считаться запущенным когда в dbus появится переменная с именем сервиса указанная в busname)
- Type=idle (аналогичен simple, но с минимальным приоритетом)

Конфиг

- /usr/lib/systemd/system – директория для юнитов созданных пакетным менеджером
- /etc/systemd/system - директория для юнитов созданных системным администратором вручную
- ~/.config/systemd/user

управление

systemctl start, stop, restart, reload

systemctl edit

Для изменения юнита, в основном для модификации сервисов которые шли через пакетный менеджер

управление. пример

```
# systemctl edit docker.service
```

Внести изменения (например добавить окружение),

```
[Service]
Environment="TEST=123"
Environment="TEST=123=345“
```

Сохранить, вывести на экран посмотреть что получилось.

```
# systemctl cat docker.service
```

Управление временем

Systemd управляет временем при помощи сервиса systemd-timesyncd

Для работы со временем используется утилита timedatectl

timedatectl set-ntp 1 – включение автоматической синхронизации времени

Управление настройками сети

systemd управляет сетью
при помощи демона systemd-networkd

networkctl status

Практика. Создадим свой юнит

web приложение должно работать в фоне,
для отображения передаем переменную окружения
FlaskDebug, чтобы в потом, например, провести его дебаг.

Практика

1. Закинуть в /opt/web_server_rut/ следущий файл

```
#!/usr/bin/env python3

from flask import Flask
import getpass

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "hello, world!\nUser: " + getpass.getuser() + "\n"

if __name__ == '__main__':
    app.run()
```

Практика

2. Создадим сервис

#vim /etc/systemd/system/web-server-rut.service

```
[Unit]
#указываем когда будет запускаться юнит при автозапуске.
#В данном случае после того как поднимется сеть.
After=network.target

[Service]
ExecStart=/opt/web_server_rut/web_server.py
Environment="FLASK_DEBUG=1"
#по умолчанию стартует от root, что не безопасно.
User=rutuser

[Install]
#определяем на каком этапе будет запускаться сервис.
#в данном случае после загрузки базовой системы
WantedBy=multi-user.target
```

Практика

3. Перезагрузим список всех юнитов и их содержимого

```
# systemctl daemon-reload
```

4. Запустим наш сервис

```
# systemctl start web-server.rut.service
```

5. Смотрим статус

```
# systemctl status web-server.rut.service
```

6. Добавим в автозагрузку

```
# systemctl enable web-server.rut.service
# systemctl is-enabled web-server.rut.service (проверим)
```

crond

cron — классический демон, использующийся для периодического выполнения заданий в определённое время.

Основной конфигурационный файл – /etc/crontab

Структура файла:

```
# minute hour day month dayofweek       command
    30      3    *     *      mon       cat /proc/meminfo >>
                                          /tmp/meminfo
```

crond

Одна строчка – одно задание! Переносить нельзя.
Если задание большое пишите скрипт.

```
vagrant@ubuntu:/etc/cron.d$ cat e2scrub_all
30 3 * * 0 root test -e /run/systemd/system || SERVICE_MODE=1 /usr/lib/x86_64-linux-gnu/e2fsprogs/e2scrub_all_cron
10 3 * * * root test -e /run/systemd/system || SERVICE_MODE=1 /sbin/e2scrub_all -A -r
```

Минута (0 – 59)	*Целое число*
Час (0 -23)	*Список чисел*
День (1-31)	*Диапазон чисел через «-»*
Месяц (1 – 12)	*Символ * или /*
День недели (0 – 6) (Воскресенье = 0)	

crond

Дополнительно к /etc/crontab

- /etc/cron.d
- /etc/cron.daily
- /etc/cron.weekly
- /etc/cron.monthly

Ваши скрипты будут запускаться
ежедневно, еженедельно, ежемесячно.

crontab –l проверит задания cron
crontab –e добавит задание

Systemd.timer

Systemd.timer умеет всё то же самое и немного больше

Позволяет настроить автоматический запуск других юнитов по расписанию

Умеет запускать задания через промежутки времени, в отличие от crond

Задание

Напишите скрипт который будет делать бэкап директории, т.е. делать архив, например в tar.gz и складывать его в подготовленную директорию для бэкапов.

Добавьте этот скрипт в расписание на выполнение раз в n минут.

Для этого вам поможет команда tar с ключами -czvf

В файл бэкапа добавить временную метку.

- *backup_20240415164306.tar.gz*

Задание

```
vagrant@ubuntu:~/backup$ ls -al
total 4940
drwxrwxr-x 2 vagrant vagrant    4096 Apr 15 17:28 .
drwxr-x--- 7 vagrant vagrant    4096 Apr 15 17:19 ..
-rw-rw-r-- 1 vagrant vagrant    4582 Apr 15 16:43 backup_20240415164306.tar.gz
-rw-rw-r-- 1 vagrant vagrant    9668 Apr 15 17:20 backup_20240415172001.tar.gz
-rw-rw-r-- 1 vagrant vagrant   19557 Apr 15 17:21 backup_20240415172101.tar.gz
-rw-r--r-- 1 root    root      39124 Apr 15 17:22 backup_20240415172201.tar.gz
-rw-r--r-- 1 root    root      78390 Apr 15 17:23 backup_20240415172301.tar.gz
-rw-rw-r-- 1 vagrant vagrant  157226 Apr 15 17:24 backup_20240415172401.tar.gz
-rw-rw-r-- 1 vagrant vagrant  314471 Apr 15 17:25 backup_20240415172501.tar.gz
-rw-r--r-- 1 root    root     629287 Apr 15 17:26 backup_20240415172601.tar.gz
-rw-r--r-- 1 root    root    1258320 Apr 15 17:27 backup_20240415172701.tar.gz
-rw-r--r-- 1 root    root    2516737 Apr 15 17:28 backup_20240415172801.tar.gz
```