Сервисы. Знакомство с демонами

Программы которые работают в фоне называют сервисами (даемонами, демонами). Для управления сервисами используется systemctl (не путать с sysclt)

$ man sysclt

NAME

sysctl - configure kernel parameters at runtime

SYNOPSIS

sysctl [options] [variable[=value]] [...]

sysctl -p [file or regexp] [...]

DESCRIPTION

sysctl is used to modify kernel parameters at runtime. The parameters available are those

listed under /proc/sys/. Procfs is required for sysctl support in Linux. You can use

sysctl to both read and write sysctl data.

$ man systemctl

NAME

systemctl - Control the systemd system and service manager

SYNOPSIS

systemctl [OPTIONS...] COMMAND [NAME...]

DESCRIPTION

systemctl may be used to introspect and control the state of the "systemd" system and

service manager. Please refer to systemd(1) for an introduction into the basic concepts

and functionality this tool manages.

systemctl предоставляет возможность включать и выключать сервисы и назначить сервисы в автозагрузку. Например, после установки сервиса апач, чтобы он включился после ребута, нужно написать команду:

systemctl enable apache2

systemctl start apache2

После этого сервис апач будет запущен и будет стартовать при включении. Теперь запускать и останавливать сервис можно командой

service apache2 start

$ man service

NAME

service - run a System V init script

SYNOPSIS

service SCRIPT COMMAND [OPTIONS]

service --status-all

service --help | -h | --version

DESCRIPTION

service runs a System V init script or upstart job in as predictable an environment as

possible, removing most environment variables and with the current working directory set

to /.

The SCRIPT parameter specifies a System V init script, located in /etc/init.d/SCRIPT, or

the name of an upstart job in /etc/init. The existence of an upstart job of the same name

as a script in /etc/init.d will cause the upstart job to take precedence over the init.d

script. The supported values of COMMAND depend on the invoked script. service passes

COMMAND and OPTIONS to the init script unmodified. For upstart jobs, start, stop, status,

and reload are passed through to their upstart equivalents. Restart will call the upstart

'stop' for the job, followed immediately by the 'start', and will exit with the return

code of the start command.

All scripts should support at least the start and stop commands. As a special case, if

COMMAND is --full-restart, the script is run twice, first with the stop command, then with

the start command. This option has no effect on upstart jobs.

service --status-all runs all init scripts, in alphabetical order, with the status com-

mand. The status is [+] for running services, [-] for stopped services and [?] for

services without a 'status' command. This option only calls status for sysvinit jobs;

upstart jobs can be queried in a similar manner with initctl list.

Проще говоря, systemctl нужен для активации сервиса, а service для менеджеринга сервиса. Чтобы посмотреть какие сервисы работают сейчас можно выполнить команду

service --status-all | grep +

Если же хочется увидеть вообще все сервисы, просто убери grep.

Запуск сервиса:

Запустить сервис сейчас, если он не активирован в systemctl, то после ребута он **не** будет запущен.

service <service_name> start

Остановка сервиса:

Остановка процесса демона. Опять же, если он активен в systemctl, то после перезагрузки этот сервис будет автоматически запущен.

service <service_name> stop

Перезапуск сервиса:

Обычно применяют после изменения конфига сервиса, если не поддерживается reload.

service <service_name> restart

Аналогично

service <service_name> stop && service <service_name> start

Перезагрузка конфигурационных файлов без остановки сервиса:

Перечитывание конфигурационных файлов без остановки процесса сервиса. Поддерживается не всегда.

service <service_name> reload

Убрать сервис из systemctl:

systemctl disable <service_name>

Просмотреть статус сервиса:

service <service_name> status

или

systemctl status <service_name>

Просмотр журнала событий:

Если что то не работает и неизвестно почему, то глянуть лог можно командой

journalctl -u <service_name>

Весь лог:

journalctl

Демонами удобно пользоваться для запуска процессов. Как я уже сказал есть сервис апача, который после запуска прочитает конфигурационные файлы находящиеся в "/etc/apache2" (там конфиги самого апача и конфиги виртуальных хостов) и при получении запроса к серверу на порт 80 (или что там будет написано в конфиге хоста) он обработает этот запрос и вернет ответ.

Аналогично работает сервис sshd, который обеспечивает подключение по ssh к серверу. Если же выключить его, то доступа уже не получить.

А теперь ~~создадим ручного демона~~ напишем свой юнит, который будет бекапить нам папку.

Пишем конфиг юнита:

[Unit]

Description=Backups projects

[Service]

RemainAfterExit=true

ExecStop=/usr/local/bin/project_backup

Type=oneshot

[Install]

WantedBy=multi-user.target

[Unit] хранит общие сведения о юните.

* **Description** - Описание.

* **After** - запускать этот юнит после определенных демонов или целей (цель - это набор юнитов). Например, network.target значит, что демон запустится после того как поднимутся сетевые интерфейсы.

[Service] объединяет сведения, необходимые для выполнения юнитом его задач.

* **Type** - тип сервиса. simple - демон запускается и начинает ожидать на консоли и не отфоркивается. forking - демон запускается, потом форкается и завершает родительский процесс. Многие программы именно так и запускаются, чтобы перейти в background режим. Например, nginx запускается по такой схеме. one-shot - используется для запуска скриптов которые запускаются, отрабатывают и завершаются.

* **ExecStop** - указывает скрипт, который должен быть выполнен перед остановкой сервиса.

* **ExecStart** - определяет команду, которая должна быть выполнена сразу после запуска сервиса.

* **RemainAfterExit** - предписывает systemd считать процесс активным после его завершения.

* **Restart** - рестартовать демон, если он завершится/упадет. При always - systemd будет перезапускать демон независимо от того почему он завершился. Можно указать on-failure, тогда будет перезапускаться если демон вышел с ненулевым кодом возврата или был завершен по сигналу (kill DAEMONPID)

[Install] содержит сведения о том, при каких обстоятельствах должен быть запущен сервис.

* **WantedBy** - устанавливает запуск при обычной загрузке компьютера.

Сохраняем юнит с именем "project_backup.service" поместим его в "/etc/systemd/system".

В ExecStop прописан путь до скрипта "/usr/local/bin/project_backup", который должен быть запущен. Его у нас там нет, так что открываем редактор и пишем скрипт

```
#!/bin/bash

current_date="$(date +'%F_%H_%M')"

tar -czf /home/user/.backups/$current_date.tar.gz /home/user/project_dir
```

И сохраняем это в файл /usr/local/bin/project_backup. Даем права на запуск

```
$ chmod +x /usr/local/bin/project_backup
```

Разберемся что в скрипте:

1-ая строка указывает путь к оболочке в которой будет запущен код. Т.е. в данном случае это bash скрипт.

2-ая строка получает текущую дату (date), форматирует в вид "ГОД-МЕСЯЦ-ДЕНЬ_ЧАСЫ_МИНУТЫ" и записывает в переменную.

3-ая строка упаковывает папку с помощью tar.

Теперь осталось создать папку /home/user/.backups, куда будут складываться бекапы и активировать демона:

Релоад конфигурации systemd (нужно при создании/редактировании юнит файлов)

```
# systemctl daemon-reload
```

Активация сервиса

```
# systemctl enable project_backup
```

Запуск

```
# systemctl start project_backup
```

Запуск юнита я делал указывая "project_backup", а не "project_backup**.service**", потому как по умолчанию подрузамевается именно "project_backup.service". Это используется для создание нескольких юнитов одного демона. Например системного ".service" и пользовательского ".{user}", где в место user имя пользователя.

Можно ребутнуться и посмотреть результат. Важно понимать, что ничего не получится если у тебя нечего бекапить. Если вдруг что то пошло не так то смотрим лог:

```
$ journalctl -u project_backup
```