Система инициализации (SysV-init)

Процессы ядра — процессы, которые запускает сама операционная система, процессы, которые запущены в фоне, и процессы, которые запустили мы сами. Например, вот процесс, который запустился, когда мы открыли терминал, и он запустил bash, нашу командную оболочку, при помощи которой мы можем общаться с операционной системой.

Так как мы использовали ключ f, мы видим дерево процессов, то есть можем проследить иерархию: одни процессы порождают другие. Процесс bash порождён процессом терминала. Процесс терминала будет называться родительским, а процесс bash, соответственно, дочерним. Процесс ps будет дочерним для процесса bash.

Процесс с идентификатором 1, **init,** — это процесс инициализации,или первый процесс, который запускается после загрузки операционной системы. Он является родительским для всех остальных процессов (кроме процессов ядра: для них родительским будет процесс с идентификатором 2), то есть запускает все остальные процессы. Он отвечает за то, чтобы правильно загрузились и заработали все необходимые компоненты операционной системы: сеть, разнообразные устройства (например, аудио, Bluetooth), запустился графический интерфейс, различные служебные процессы, например уже знакомый нам планировщик cron.

Кроме того, init предоставляет возможность управлять работающими процессами — в самом простом случае перезапускать их или останавливать.

Веб-сервер же должен работать постоянно, чтобы постоянно отдавать информацию на запросы пользователя. Его, как и, например, cron, нужно автоматически запустить тогда же, когда загружается сервер, и он должен работать, пока сервер не отключат от питанию.

Init — это зонтичный термин. В Linux есть различные имплементации системы инициализации.

Мы познакомимся с тремя:

- SysV-init,
- upstart,
- systemd.

SysV-init

Важнейшим понятием, лежащим в основе **SysV-init**, являются уровни выполнения (runlevels). Они описывают состояние операционной системы, и к ним привязаны наборы сервисов, которые нужно запустить или остановить на соответствующем уровне выполнения.

Вот эти уровни:

0	Halt	Выключение операционной системы
1	Single-user mode	Однопользовательский режим. По смыслу он немного похож на безопасный режим в Windows. В нём запускается очень мало сервисов, работать может только пользователь root. Чаще всего этот режим используется для восстановления после какого-то сбоя
2	Multi-user mode	Работа операционной системы без поддержки работы с сетью
3	Multi-user mode with networking	Нормальная работа операционной системы
4	Не используется	
5	Нормальная работа операционной системы (то же, что и в runlevel 3) + графический интерфейс	
6	Reboot	Перезагрузка операционной системы

Каждому из ранлевелов сопоставлялся набор скриптов, запускающих соответствующий набор сервисов, необходимый для работы операционной системы.

Работа системы инициализации. При запуске процесс init читал файл /etc/inittab, который выглядел приблизительно так.
В нём описывалось, что необходимо сделать на каждом из ранлевелов. Формат записей следующий: идентификатор, затем ранлевел, к которому применяется действие и команда.
Действие wait означает, что команда будет запущена, когда достигнут указанный ранлевел.
В нашем примере выполняется команда rc, и ей в качестве параметра передано число.
Команда rc читает директории /etc/rcN.d/ и выполняет все находящиеся там скрипты.
В директориях находятся скрипты, названия которых начинаются либо с заглавной буквы S, либо с заглавной буквы K, что означает действие — стартовать либо завершить сервисы соответственно.
Например, все файлы в директории /etc/rc0.d будут начинаться на K, так как перед отключением системы мы будем завершать все работающие сервисы.

Файлы в директории /etc/init.d называют **инит-скриптами**. Они описывают запуск, остановку и другие действия с нашим сервисом.

Для того чтобы узнать, на каком уровне выполнения работает операционная система, можно использовать команду *runlevel*.

```
runlevel
```

Чтобы изменить уровень выполнения, используется команда:

```
sudo init N
```

Например, давайте перезагрузим нашу виртуальную машину:

```
sudo init 6
```

Давайте посмотрим, как выглядит инит-скрипт:

```
cat /etc/init.d/nginx
```

Все действия оформлены как функции.

Для управления процессами используется **программа start-stop-daemon.**

Действия, которые мы хотим выполнить, передаются инит-скрипту **параметрами**. Они обрабатываются инит-скриптом в операторе выбора.

Посмотрим на обратную совместимость в действии. Например, если мы попробуем остановить nginx при помощи инит-скрипта, вызвав его с параметром stop:

```
/etc/init.d/nginx stop
```

Операционная система выполнит то, что мы хотим сделать, но подскажет, что на самом деле используется systemctl, который мы использовали раньше.

Поддержка обратной совместимости даёт широкие возможности: например, мы можем писать переносимые скрипты. Они будут работать и на системах с SysV-init, и на системах с systemd.