Systemd
и немного про sysv init

Д.В. Иртегов

НГУ

2024

Что такое init в Unix-системах

- Процесс с pid==1
- Создается ядром необычным образом при старте системы
- Обычно, запускает все остальные пользовательские процессы
- По умолчанию, /sbin/init
- Ядру можно указать любой бинарник, например /bin/sh, и оно запустит его в качестве init (полезно если вы даже в single user попасть не можете)

Какие init встречались в Linux

- SystemV-style init (большинство дистрибутивов с 90х)
- BSD-style init (ЕМНИП slackware)
- Upstart (проект Ubuntu, из LTS был только в 12.04)

Что еще было/есть

- Сервисы OS/2 и Windows NT (net start)
- System V R4/Solaris saf (запускается как отдельный сервис из systemv init, но начиная с Solaris 8 запускает большинство системных сервисов)
- Apple MacOS launchd

System V style init (сначала маленький экскурс в историю...)

- Конфигурационный файл /etc/inittab
- Уровень запуска (runlevel)
  - # 0 - halt (Do NOT set initdefault to this)
  - # 1 - Single user mode
  - # 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
  - # 3 - Full multiuser mode
  - # 4 - unused
  - # 5 - X11
  - # 6 - reboot (Do NOT set initdefault to this)

Уровень запуска

- Определяет список процессов (системных сервисов), запущенных на данном уровне
- По умолчанию задается параметром initdefault в /etc/inittab
  - id:5:initdefault
- Систему можно переводить с одного уровня на другой командами
  - init,
  - telinit,
  - shutdown

Откуда берется список процессов?

- В стандартном /etc/inittab есть директивы

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

- Для каждого уровня запуска есть свой каталог /etc/rc[0-6].d
- В этом каталоге лежит набор файлов (скриптов shell)

/etc/rc[0-6].d

```
[root@vzhn boot]# ls /etc/rc1.d
K01dnsmasq          K15httpd        K74ntpd            K89dund
K01setroubleshoot   K20nfs          K74ups             K89hidd
K01smartd           K20vz           K75netfs           K89iscsi
K02avahi-daemon     K24irda         K77vzvpn           K89iscsid
K02avahi-dnsconfd   K25sshd         K80kdump           K89netplugd
K02dhcdbd           K30sendmail     K80vzlicmon        K89pand
K02haldaemon        K35dhcpd        K85mdmonitor       K89rdisc
K02NetworkManager   K35dhcrelay     K85mdmpd           K89vznetcfg
K02oddjobd          K35vncserver    K85messagebus      K90bluetooth
K03yum-updatesd     K35winbind      K85rpcgssd         K90network
K05anacron          K44rawdevices   K85rpcidmapd       K92ip6tables
K05atd              K50netconsole   K86nfslock         K92iptables
K05conman           K50snmpd        K87irqbalance      K95firstboot
K05saslauthd        K50snmptrapd    K87mcstrans        K95kudzu
K05wdaemon          K60crond        K87multipathd      K99microcode_ctl
K10cups             K69rpcsvcgssd   K87portmap         K99readahead_early
K10psacct           K72autofs       K87restorecond     K99readahead_later
K10tcsd             K73ypbind       K88auditd          S06cpuspeed
K10vzlmond          K74acpid        K88pcscd           S26lvm2-monitor
K10xfs              K74lm_sensors   K88syslog          S99single
K15gpm              K74nscd         K88wpa_supplicant
root@vzhn boot]# ls -l /etc/rc1.d/
total 224
lrwxrwxrwx  1 root root 17 Apr  9  2009 K01dnsmasq -> ../init.d/dnsmasq
lrwxrwxrwx  1 root root 24 Nov 30  2008 K01setroubleshoot -> ../init.d/setroubleshoot
lrwxrwxrwx  1 root root 16 Mar 18  2013 K01smartd -> ../init.d/smartd
```

Как это понимать

- S* файлы при входе на уровень запускаются с командой start
- K* файлы запускаются с командой stop
- То и другое – симлинки. в папку /etc/init.d
- Циферки после S. или K определяют порядок запуска
- Скрипты также могут понимать команды restart и status
- Есть утилиты для создания симлинков в правильных местах, но они не стандартные и отличались от дистрибутива к дистрибутиву

Недостатки SystemV-style init

- Для запуска сервиса надо писать относительно сложную программу на shell. »библиотеки» (разделяемые скрипты) для этого появились лишь в 2000е
- **Последовательный запуск** => медленный старт системы
- Очень грубое управление порядком запуска
- Если сервис не стартует, это может привести к неуправляемому каскаду ошибок
- Нет стандартной процедуры кастомизации. Допиливание руками init-скриптов может быть потеряно после установки обновлений

Источники вдохновения (в хронологическом порядке)

- OS/2 LAN Manager/Windows NT services
- System V R4 saf (Service Access Facility)
- Часто также ссылаются на Apple Launchd
- Upstart – первая попытка сделать init-систему на основе событий для Linux
- Главная идея – ввести понятие зависимости между сервисами
- Это позволяет стартовать сервисы параллельно

Основные понятия systemd: unit

- Наиболее важные типы юнитов:
- .service – программа, которую нужно запустить или поддерживать запущенной
- .socket – конечная точка сервиса, обычно обращение к нему приводит к запуску сервиса если он еще не был запущен
- .device – устройство (например, hot-plug), при наличии/появлении которого надо что-то запустить
- .target – промежуточное или целевое состояние системы, например network.target – активируется когда в системе появляется сеть (неважно, каким спссобом)

Пример описания юнита

```
[Unit]
Description=mySQL Server
After=network.target

[Service]
Type=simple
ExecStart=/usr/sbin/mysqld --defaults-file=/etc/mysql/my.cnf --basedir=/usr --
datadir=/var/lib/mysql --pid-file=/var/run/mysqld/mysqld.pid --
socket=/var/run/mysqld/mysqld.sock
ExecStop=/bin/kill -15 $MAINPID
PIDFile=/var/run/mysqld/mysql.pid
Restart=always

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

Формат описания

- Windows.ini-style config
  - Набор параметров ИМЯ=значение, но с секциями
  - В отличие от TOML/YAML секции плоские (только один уровень)
- Наиболее важные секции
  - [unit] – метаданные (описание, зависимости)
  - [service] – собственно описание сервиса: как его запускать, как его останавливать, что делать если он сам помрет
  - В зависимости от типа юнита это может быть [device], [socket] и т.д.
  - [install] – что делать при активации сервиса (systemctl enable)

Где лежат описания юнитов

- /lib/systemd/system – установленные в системе юниты
- /etc/systemd/system – активные (запускаемые при старте)
- В /etc/systemd/system обычно лежат не сами юниты, а симлинки на /lib/systemd/system, причем обычно лежат не в самом system, а в подпапках [target].wants
- /etc/systemd/system/<unit file>.d/override.conf – мержится с файлом юнита, позволяет вносить локальные кастомизации, которые не будут потеряны при установке обновлений

Таржеты

- default.target – симлинк на целевой таржет при загрузке, обычно multi-user или graphical
- poweroff.target – выключение питания
- rescue.target – «однопользовательский режим»
- multi-user.target – типичная конфигурация сервера (без GUI)
- graphical.target
- reboot.target

Как этим всем управлять

- Init/telinit/shutdown – легаси утилиты, используют ранлевелы вместо systemd target
- halt, reboot, poweroff
- service – тоже легаси утилита, но умная: понимает, каким сервисом или инитом управляет (systemV или systemd)
- systemctl – основной инструмент админа

systemctl

- systemctl list-units
- systemctl list-unit-files
- systemctl list-dependencies
- systemctl list-dependencies UNIT_FILE

systemctl (продолжение)

- systemctl status SERVICE
- systemctl show SERVICE
- systemctl start SERVICE
- systemctl stop SERVICE
- systemctl restart SERVICE
- systemctl enable SERVICE – будет стартовать при загрузке
- systemctl disable SERVICE - больше не будет стартовать при загрузке
- systemctl reload-or-restart SERVICE
- systemctl mask SERVICE - /etc/systemd/system/SERVICE.service → /dev/null.
- systemctl unmask SERVICE

Еще полезные команды

- journalctl –u SERVICE
- systemctl --state=failed
- systemctl-analyze

Что еще умеет systemd

- Ресурсные квоты для сервисов (cgroup)
- Пользовательские сервисы
  - /etc/systemd/user (для всех юзеров)
  - $HOME/.config/systemd/user/ (персональные)
- Задания по расписанию (.timer юниты)