



WARGAMING.NET

LET'S BATTLE

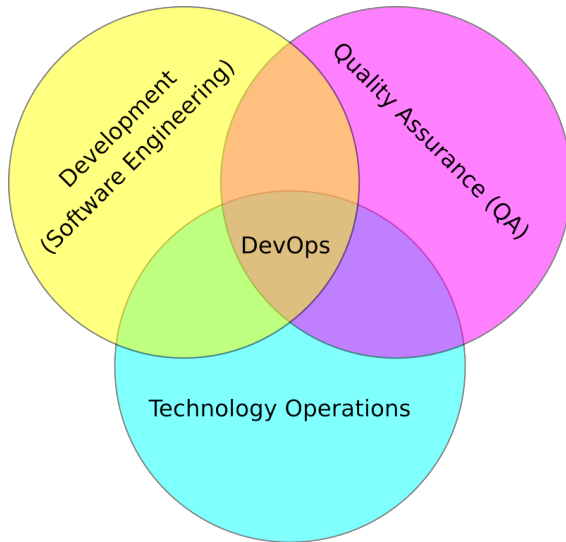
Про Systemd глазами web-разработчика

Стас Рудаков

О чем будем говорить

- ▶ DevOps
- ▶ /sbin/init
- ▶ управление ресурсами
- ▶ journald и бинарные логи
- ▶ сокет активация
- ▶ таймеры как замена cron'a
- ▶ будущее Systemd

DevOps



Загрузка системы

```
NMI watchdog disabled for cpu0: unable to create perf event
:: Loading Initramfs
:: Starting udevd...
done.

INIT: version 2.88 booting

> Arch Linux
> http://www.archlinux.org

-----
:: Starting UDev Daemon [DONE]
:: Triggering UDev uevents [DONE]
:: Waiting for UDev uevents to be processed [DONE]
:: Bringing up loopback interface [DONE]
:: Mounting Root Read-only [DONE]
:: Checking Filesystems [BUSY]
/dev/sda3: clean, 59493/1507328 files, 390592/6018350 block:
/dev/sda1: clean, 265/24096 files, 37560/96356 blocks

:: Mounting Local Filesystems [DONE]
:: Retrying failed UDev events [DONE]
:: Activating Swap [DONE]
:: Configuring System Clock [DONE]
:: Initializing Random Seed [DONE]
:: Removing Leftover Files [DONE]
:: Setting Hostname: arch-virtualbox [DONE]
:: Updating Module Dependencies [DONE]
:: Setting Locale: es_ES.UTF-8 [DONE]
:: Setting Consoles to UTF-8 mode [DONE]
:: Loading Keyboard Map: es [DONE]
INIT: Entering runlevel: 3
:: Starting Syslog-NG [DONE]
:: Starting Network [BUSY]
```

/sbin/init

- ▶ первый процесс в user space
- ▶ ответственен за инициализацию системы
- ▶ реализации: SysVinit, Upstart (Ubuntu), procd (OpenWRT), launchd (Mac OS X)
- ▶ ...ну и Systemd

/sbin/init

```
1 $ pstree -A
2 init+-apache2+-9*[apache2]
3     |           |-apache2---apache2
4     |           '-apache2---17*[{apache2}]
5     |-cron---cron---sh---php
6     |-dbus-daemon
7     |-fail2ban-server---8*[{fail2ban-serve}]
8     |-6*[getty]
9     |-mysqld_safe+-logger
10    |           '-mysqld---20*[{mysqld}]
11    |-postfix-policyd
12    |-rpc.statd
13    |-rsyslogd---3*[{rsyslogd}]
14    |-sshd+-sshd---sshd---bash
15    |           '-sshd---sshd---bash---pstree
16    '-udev
```

Почему Systemd?

- ▶ Простота использования
- ▶ Плюшки для разработчиков и мейнтейнеров
- ▶ Скорость загрузки
- ▶ Cgroups: изоляция процессов, управление ресурсами
- ▶ Интеграция подсистем

По интерфейсу встречают

- ▶ `# update-rc.d mysql default`
- ▶ `# /etc/init.d/mysql start`
- ▶ `# /etc/init.d/mysql stop`
- ▶ `# /etc/init.d/mysql restart`
- ▶ `# update-rc.d -f mysql remove`

- ▶ `# systemctl enable mysql`
- ▶ `# systemctl start mysql`
- ▶ `# systemctl stop mysql`
- ▶ `# systemctl restart mysql`
- ▶ `# systemctl disable mysql`

init скрипт

```
1 case "${1:-''}" in
2     'start')
3     sanity_checks;
4     # Start daemon
5     log_daemon_msg "Starting MySQL database server" "mysqld"
6     if mysqld_status check_alive nowarn; then
7         log_progress_msg "already running"
8         log_end_msg 0
9     else
10        # Could be removed during boot
11        test -e /var/run/mysqld || install -m 755 -o mysql -g root
12        -d /var/run/mysqld
13
14        # Start MySQL!
15        /usr/bin/mysqld_safe > /dev/null 2>&1 &
```

Listing 1: одна двенадцатая файла /etc/init.d/mysql в Debian 5

unit файл

```
1 [Unit]
2 Description=MySQL database server
3 After=syslog.target
4
5 [Service]
6 User=mysql
7 Group=mysql
8
9 ExecStart=/usr/bin/mysqld --pid-file=/run/mysqld/mysqld.pid
10 ExecStartPost=/usr/bin/mysqld-post
11
12 Restart=always
13 PrivateTmp=true
14
15 [Install]
16 WantedBy=multi-user.target
```

Listing 2: /usr/lib/systemd/system/mysqld.service

SysVinit vs Systemd

- ▶ init-скрипты дают полный контроль над процессом запуска приложения
- ▶ с точки зрения архитектуры ОС это простое и надежное решение
- ▶ шаблоны запуска приложений теряются за десятками строк кода
- ▶ демонизация (скрипты могут запускаться из пользовательских сессий)
- ▶ реализации пишутся мейнтейнерами дистрибутивов, а не авторами приложений
- ▶ дебаг init-скриптов — вмеру сложно

- ▶ Systemd - это еще один слой абстракции
- ▶ и сам Systemd может непредсказуемо сломаться
- ▶ простые шаблоны запуска приложений
- ▶ демонизация в чистом виде становится ненужной
- ▶ реализации unit-файлов могут поддерживаться авторами приложений
- ▶ лучшая изоляция процессов за счет использования CGroups
- ▶ распараллеливание загрузки

Cgroups

```
1 /  
2 /system  
3 /system/slim.service  
4 /system/systemd-journald.service  
5 /system/cronie.service  
6 /system/dbus.service  
7 /system/netctl-auto@.service  
8 /system/netctl-auto@.service/netctl-auto@wlan0.service  
9 /system/getty@.service  
10 /system/polkit.service  
11 /system/rtkit-daemon.service  
12 /system/sshd.service  
13 /system/systemd-logind.service  
14 /system/systemd-udev.service  
15 /user/1000.user/1.session
```

Cgroups: управление ресурсами

```
1 [Unit]
2 Description=Example server
3
4 [Service]
5 ExecStart=/usr/bin/example
6
7 CPUShares=2048
8 MemorySoftLimit=1G
9 #MemoryLimit=1G
10 BlockIOWeight=500
11 BlockIOWriteBandwidth=/dev/sda 5M
12
13 [Install]
14 WantedBy=multi-user.target
```

Listing 3: /usr/lib/systemd/system/example.service

Journal

- ▶ journald — сервис, который занимается сбором, сохранением и отдачей логов;
- ▶ логи хранятся в бинарном формате
- ▶ вместе с набором проиндексированных мета-данных (!!!)
- ▶ и контролем целостности
- ▶ (но единственная веская причина создания journal'a — `systemctl status`);
- ▶ работа с логами — через специальную утилиту `journalctl...`
- ▶ ...или Python-модуль `systemd.journal` (запись и чтение);

journalctl

```
1 # journalctl
2 Sep 21 12:15:14 jazzcafe systemd[1]: Startup finished in 2.892s
3 Sep 21 12:15:16 jazzcafe slim[643]: amixer: Unable to find
4 Sep 21 12:15:17 jazzcafe kernel: wlan0: authenticate with
5 Sep 21 12:15:17 jazzcafe kernel: wlan0: send auth to 00:22:b0:
6 Sep 21 12:15:17 jazzcafe kernel: wlan0: authenticated
7 Sep 21 12:15:17 jazzcafe kernel: iwlwifi 0000:03:00.0 wlan0:
8 Sep 21 12:15:17 jazzcafe kernel: iwlwifi 0000:03:00.0 wlan0:
9 Sep 21 12:15:17 jazzcafe kernel: wlan0: associate with 00:22:b0
10 Sep 21 12:15:17 jazzcafe kernel: wlan0: RX AssocResp from
11 Sep 21 12:15:17 jazzcafe wpa_actiond[1864]: Interface 'wlan0'
```

journalctl -o verbose

```
1 Sat 2013-09-21 12:14:26 FET [s=4e6ebc108450474d9fd3d43b96562483
2     _BOOT_ID=0da18fa3d07e43b3af51043d64451f66
3     _MACHINE_ID=b4f307d9cadce8040f7d75b30000147a
4     _HOSTNAME=jazzcafe
5     PRIORITY=5
6     SYSLOG_FACILITY=3
7     _UID=0
8     _GID=0
9     _TRANSPORT=syslog
10    SYSLOG_IDENTIFIER=wpa_actiond
11    SYSLOG_PID=1191
12    MESSAGE=Starting wpa_actiond session for interface
13    _PID=1191
14    _EXE=/usr/bin/wpa_actiond
15    _CMDLINE=wpa_actiond -p /run/wpa_supplicant -i wlan0
16    _SYSTEMD_CGROUP=/system/netctl-auto@.service/netctl-
17    _SYSTEMD_UNIT=netctl-auto@wlan0.service
```


Пишем в журнал

```
1 logging.basicConfig(level=logging.DEBUG)
2
3 # FROM_SOMEBODY will be sent to journal with every record
4 hndlr = systemd.journal.JournalHandler(
5     level=logging.INFO, FROM_SOMEBODY='WITH_LOVE'
6 )
7 logging.root.addHandler(hndlr)
8
9 app_user_id = 100500
10 # oops, APP_USER_ID won't be sent to journal
11 logging.warning('Session started', extra={'APP_USER_ID':
12     app_user_id})
13
14 # APP_USER_ID will be sent to journal
15 systemd.journal.send('Session ended', APP_USER_ID=str(
16     app_user_id))
```

Сокет-активация

- ▶ вместо демона запускается легковесный процесс, слушающий сокет
- ▶ при поступлении данных запускается демон с преинициализированным сокетом
- ▶ изначально механизм был придуман для ускорения инициализации системы
- ▶ сама идея позаимствована из `launchd`

Сокет-активация: простой HTTP-сервер

```
1 class HttpServer(tulip.http.ServerHttpProtocol):
2     @tulip.coroutine
3     def handle_request(self, message, payload):
4         response = tulip.http.Response(self.transport, 200)
5         response.add_header('Transfer-Encoding', 'chunked')
6         response.add_chunking_filter(1025)
7         response.add_header('Content-type', 'text/plain')
8         response.send_headers()
9
10        with open(__file__, 'rb') as fp:
11            chunk = fp.read(8196)
12            while chunk:
13                response.write(chunk)
14                chunk = fp.read(8196)
15
16        response.write_eof()
```

Сокет-активация: запускаем HTTP-сервер без сокет-активации

```
1 loop = tulip.get_event_loop()
2 f = loop.start_serving(HttpServer, '127.0.0.1', 8000)
3
4 socks = loop.run_until_complete(f)
5 print('serving on', socks[0].getsockname())
6 try:
7     loop.run_forever()
8 except KeyboardInterrupt:
9     pass
```

Сокет-активация: запускаем HTTP-сервер с сокет-активацией

```
1 loop = tulip.get_event_loop()
2
3 fd = systemd.daemon.listen_fds()[0]
4 if not systemd.daemon.is_socket_inet(fd):
5     raise Exception('Configuration error')
6
7 sckt = socket.fromfd(fd, socket.AF_INET, socket.SOCK_STREAM)
8 sckt.setblocking(0)
9 f = loop.start_serving(lambda: HttpServer(loop=loop), sock=sckt)
10
11 loop.run_until_complete(f)
12 print('serving on', sckt.getsockname())
13 try:
14     loop.run_forever()
15 except KeyboardInterrupt:
16     pass
```

Сокет-активация: останавливаем сервер после 10 секунд неактивности

```
1 class HttpServer(tulip.http.ServerHttpProtocol):
2     def __init__(self, *args, **kwargs):
3         super(HttpServer, self).__init__(*args, **kwargs)
4         self.schedule_shutdown(self._loop)
5
6     @classmethod
7     def schedule_shutdown(cls, loop):
8         future_sysexit = getattr(cls, '_future_sysexit', None)
9         if future_sysexit:
10             future_sysexit.cancel()
11             cls._future_sysexit = loop.call_later(10, sys.exit)
12
13     @tulip.coroutine
14     def handle_request(self, message, payload):
15         # [...]
16         response.write_eof()
17         self.schedule_shutdown(self._loop)
```

Сокет-активация: unit-файлы

```
1 [Socket]
2 ListenStream=127.0.0.1:8000
3 [Install]
4 WantedBy=sockets.target
```

Listing 4: /usr/lib/systemd/system/httpserver.socket

```
1 [Unit]
2 Description=Example HTTP server
3 Requires=httpserver.socket
4 [Service]
5 ExecStart=/usr/bin/httpserver
6
7 [Install]
8 WantedBy=multi-user.target
9 Also=httpserver.socket
```

Listing 5: /usr/lib/systemd/system/httpserver.service

Таймеры - полноценная замена cron'a

```
1 [Unit]
2 Description=Daily Timer
3
4 [Timer]
5 OnBootSec=10min
6 OnUnitActiveSec=1d
7 #OnActiveSec=, OnStartupSec=, OnUnitInactiveSec=, OnCalendar=
8 Unit=logrotate.service
```

Listing 6: /etc/systemd/system/timer-daily.timer

```
1 [Unit]
2 Description=Rotate Logs
3
4 [Service]
5 ExecStart=/usr/bin/logrotate /etc/logrotate.conf
```

Listing 7: /etc/systemd/system/timer-daily.target.wants/logrotate.service

Будущее Systemd

RHEL 7

вторая половина 2013

Настоящее Systemd

- ▶ ArchLinux
- ▶ Fedora
- ▶ Mageia
- ▶ openSUSE
- ▶ Sabayon
- ▶ ну или можно повозиться с Debian, Gentoo и др.

СПАСИБО ЗА ВНИМАНИЕ. ВОПРОСЫ?

Стас Рудаков

<mailto:stas@garage22.net>

https://raw.githubusercontent.com/nott/talks/systemd_webdev.pdf

<http://freedesktop.org/wiki/Software/systemd/>

<http://0pointer.de/blog/projects/the-biggest-myths.html>

<http://0pointer.de/blog/projects>

lists.freedesktop.org/archives/systemd-devel/2013-June/011388.html