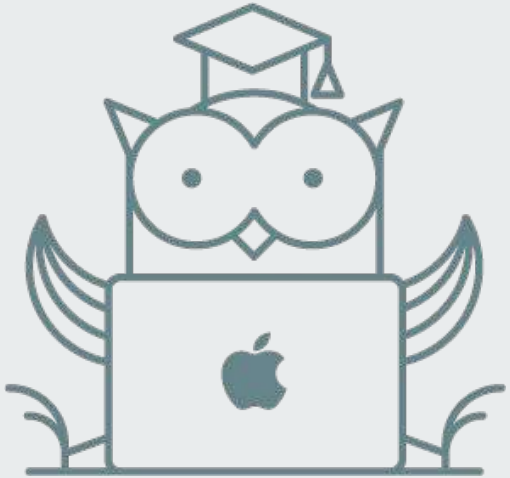
Курс «Администратор Linux»

Запуск системы

Занятие # 5

Дмитрий Молчанов
Григорий Ожегов



Задачи инициализации

Основной задачей системы инициализации является приведение системы в состояние, в котором пользователь может ею пользоваться, а также - управление служебными и фоновыми процессами.

- Запуск ОС
  - Другие: upstart, launchd
  - Sys V Init
  - Systemd

upstart: коротко

Устаревшая система инициализации от компании-производителя Ubuntu Canonical. Изначально разработана для Ubuntu, затем использовалась в том числе и в RHEL дистрибутивах (Fedora, Centos). По структуре сценария очень напоминает systemd: декларативно необходимо определять различные секции, которые отвечают за параметры запуска демона. Upstart также как и systemd следит за состоянием того или иного демона,

upstart: приготовление

```
[ozhegov@p2 init]$ cat /etc/init/nrpe.conf
description           "the Nagios Remote Plugin Executor"
oom -10
start on started network
stop on runlevel [!2345]
respawn
exec /usr/sbin/nrpe -c /etc/nrpe/nrpe.cfg -f

[root@p2 init]# start nrpe
nrpe start/running, process 1111
```

Launchd: коротко

Система инициализации в macOS.

Схожа с SysV Init, точно также существует главный процесс init, который имеет PID 1. Умеет запускать процессы про расписанию, т.е. вшитый cron. В терминологии используются понятия демонов (демоны root’а) и агентов (демоны текущего пользователя).

macOS не базируется на ядре Linux, но в работе системного администратора можно столкнуться с задачами автоматизации своего рабочего ноутбука или с эксплуатацией Mac Mini или Mac Pro (например CI-сервер для тестирования и сборки проектов под macOS)

Launchd: приготовление

В папочку ~/Library/LaunchAgents нужно положить xml-файл:
RunMe.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
     <dict>
          <key>Label</key>
          <string>com.example.app</string>
          <key>Program</key>
          <string>/Users/Me/RunMe.sh</string>
          <key>RunAtLoad</key>
          <true/>
     </dict>
</plist>
```

Также есть директории:

- Демоны /Library/LaunchDaemons
- Системные агенты /System/Library/LaunchAgents
- Системные демоны /System/Library/LaunchDaemons

SysV Init

Наиболее массовая система инициализации ОС. Оперирует runlevel (уровень инициализации/выполнения) и, в соответствии с runlevel, выполняет тот или иной набор скриптов, иными словами runlevel это целевое состояние системы. Основная конфигурация runlevel'ов происходит путем создания в /etc/rcN.d/ (N номер runlevel'а) симлинков вида (S|K)NN_*name* (S - start, K - stop) на исполняемые файлы в /etc/init.d, которые должны обрабатывать параметры start, stop, restart. В момент перехода на следующий уровень выполнения выполняются K-симлинки от предыдущего, а затем S-симлинки нового.

SysV Init: runlevel’ы

0	Halt, Выключение
1	Single User mode, режим обслуживания системы
2	user defined
3	Multiuser Text mode
4	user defined
5	Multiuser Graphical mode
6	Reboot

SysV Init: приготовление

```
[ozhegov@p2 init.d]$ cat /etc/init.d/searchd
#!/bin/sh
#
# sphinx searchd Free open-source SQL full-text search engine
#
# chkconfig:   - 20 80
# description: Starts and stops the sphinx searchd daemon that handles \
#              all search requests.

### BEGIN INIT INFO
# Provides: searchd
# Required-Start: $local_fs $network
# Required-Stop: $local_fs $network
# Should-Start: $remote_fs
# Should-Stop: $remote_fs
# Default-Start:
# Default-Stop: 0 1 2 3 4 5 6
# Short-Description: start and stop sphinx searchd daemon
# Description: Sphinx is a free open-source SQL full-text search engine
### END INIT INFO

# Source function library.
. /etc/rc.d/init.d/functions
```

SysV Init: приготовление (продолжение)

```
exec="/usr/bin/searchd"
prog="searchd"
user="sphinx"
config="/etc/sphinx/sphinx.conf"
lockfile=/var/lock/subsys/searchd

start() {
    [ -x $exec ] || exit 5
    [ -f $config ] || exit 6
    echo -n $"Starting $prog: "
    # if not running, start it up here, usually something like "daemon $exec"
    daemon --user $user $exec --config $config
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    echo -n $"Stopping $prog: "
    # stop it here, often "killproc $prog"
    killproc $prog
    retval=$?
    echo
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}
```

SysV Init: приготовление (продолжение)

```
restart() {
    stop
    start
}

reload() {
    restart
}

force_reload() {
    restart
}

rh_status() {
    # run checks to determine if the service is running or use generic status
    status $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}
```

SysV Init: приготовление (окончание)

```
case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?
```

SystemD

В отличие от SysV init systemd оперирует target’ами - так же описывающими состояние системы и unit’ами, которые необходимо запустить для приведения системы в необходимое состояние.

Ключевыми особенностями являются:

- параллельная загрузка
- поддержка зависимостей между unit’ами

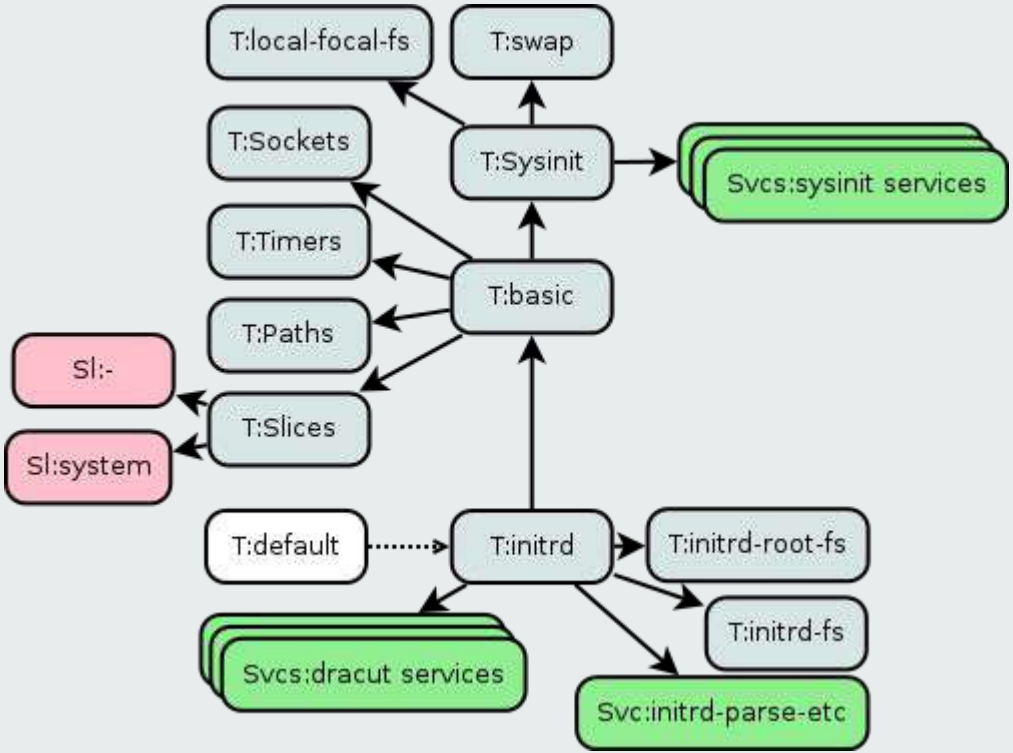
Так же немаловажно:

- отсутствие привязки к скриптам
- отсутствие жестких требований к демонизации процесса

SystemD: компоненты

- systemd
- journald (systemd-journald)
- udevd (systemd-udevd)
- logind (systemd-logind)

SystemD: процесс загрузки



SystemD: Units

- [Unit] (man:systemd.unit)
  - Description
  - Documentation
  - Wants/Requires
  - After/Before
- [Service]
  - Type: simple/oneshot/forking (man:systemd.service)
  - ExecStart(Pre,Post)
  - ExecStop(Post)
  - ExecReload
  - RemainAfterExit
  - PIDFile
- [Install]
  - WantedBy, RequiredBy

SystemD: приготовление oneshoot

```
# cat /etc/systemd/system/test.service
[Unit]
Description="test message sender"
After=sshd1.service
[Service]
ExecStart=/root/test.sh
User=sshd
Type=oneshot
RemainAfterExit=yes
[Install]
WantedBy=multi-user.target
```

SystemD: приготовление forking

```
[root@ozhegov system]# cat /usr/lib/systemd/system/searchd.service
[Unit]
Description=SphinxSearch Search Engine
After=network.target remote-fs.target nss-lookup.target
After=syslog.target

[Service]
Type=forking
User=sphinx
Group=sphinx
# Run ExecStartPre with root-permissions
PermissionsStartOnly=true
ExecStartPre=/bin/mkdir -p /var/run/sphinx
ExecStartPre=/bin/chown sphinx.sphinx /var/run/sphinx
# Run ExecStart with User=sphinx / Group=sphinx
ExecStart=/usr/bin/searchd --config /etc/sphinx/sphinx.conf
KillMode=process
KillSignal=SIGTERM
SendSIGKILL=no
LimitNOFILE=infinity
PIDFile=/var/run/sphinx/searchd.pid

[Install]
WantedBy=multi-user.target
Alias=sphinx.service
Alias=sphinxsearch.service
```

SystemD: приготовление simple

```
[root@ozhegov system]# cat /usr/lib/systemd/system/otus.service
[Unit]
Description=otus gunicorn
After=network.target

[Service]
Type=simple
User=otus
Group=otus
Environment=LANG=en_US.UTF-8
Environment=LC_ALL=en_US.UTF-8
Environment=PYTHONPATH=/opt/otus/src
ExecStart=/opt/otus/env/bin/python /opt/otus/env/bin/gunicorn -c /etc/otus/gunicorn.conf
otus.wsgi:application
ExecReload=/bin/kill -HUP $MAINPID
LimitNOFILE=1048000
LimitNPROC=32768

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

SystemD: Управление

- systemctl {command} {unit}
  - start/stop/reload/restart
  - status
  - enable/disable
  - mask/unmask
  - --type
  - isolate *target*
  - show
  - cat
- journalctl
  - -f - аналог tail -f
  - -u *name* - журналы по выделенному unit'у

Спасибо за внимание

Дмитрий Молчанов
Григорий Ожегов