OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

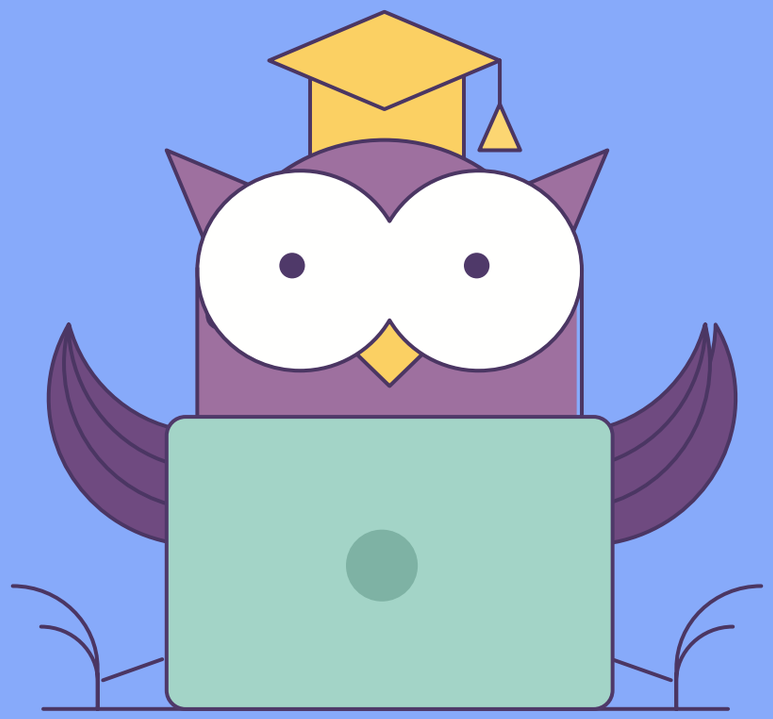
Systemd

Курс «Администратор Linux»

Занятие № 8

Румянцев Александр Леонид Альбрехт

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Цели

- Посмотреть что было до
- Разобраться с systemd
- Написать свой юнит файл

Init

Подсистема инициализации в Unix и ряде Unix-подобных систем, которая запускает все остальные процессы. Обычно это процесс /sbin/init и он имеет PID 1.

Возможно изменить параметром ядра init:

vmlinuz init=/bin/sh

Демон

Обычный процесс, который не дает себя убить

- закрывает открытые по-умолчанию файл-дескрипторы STDIN, STDOUT, STDERR, что бы не получить SIGHUP

- отвязывается от родительского процесса(fork(), setsid()), чтобы родитель не получил SIGCHILD

AT&T vs SysV

AT&T style - один скрипт инициализации, пользовательские программы запускались через rc.local (FreeBSD < 5.0, OpenBSD)

SysV style - все сервисы, появились уровни исполнения

Всё это появилось из-за упоротости консервативности разработчиков init, исповедующих принцип KISS

Классический init и уровни исполнения

/etc/inittab - конфигурация

/sbin/telinit - управление

0 - halt

1 - single user

2 - multiuser w/o NW

3 - full multiuser

5 - X11 GUI

6 - reboot

SysV сервисы

Скрипт, который должен уметь обрабатывать стандартные параметры start
stop
restart

```
#!/bin/sh
case $1 of
start) /usr/bin/mydaemon -p /var/run/mydaemon.pid
 ;;
stop) kill $(</var/run/mydaemon.pid
 ;;
restart) $0 stop; $0 start
 ;;
*) echo "what do you want from me, dude?
esac
```

SysV сервисы

Проблемы:

- линейное исполнение

- отсутствие зависимостей

- невозможность эффективно контролировать процесс (проблема pid-файла и двойного запуска или совсем не запуска, проблема abandoned childs)

- невозможно перезапустить упавший процесс (костыль - monit)

Всё это привело к адовым init-скриптам, в которых демонизация делалась с помощью shell jobs

А как у других?

OS X - launchd

Solaris - SMF

А еще как?

daemontools имени D, J. Bernstein http://cr.yp.to/daemontools.html и его наследник, runit имени G. Pape http://smarden.org/runit/

А как еще было?

Upstart

SystemD

Systemd - это "система инициализации". В кавычках, потому что она вобрала в себя еще огромную кучу функции.

В основе - cgroups. Кроме прямой задачи - ограничения ресурсов, в контексте systemd cgroups решает задачу принадлежности процесса сервису, так как любой процесс сервиса будет запущен в той-же группе /sys/fs/cgroup/systemd/system.slice

SystemD. Преимущество над SysV

- SystemD запускает процессы параллельно и конкурентно во время загрузки
- Работа процессов отслеживается с помощью cgroups, а не PID
- Улучшенные возможности по управлению процессом загрузки и зависимостями сервисов
- Возможность снимать снепшоты состояний сервисов и восстанавливаться на них
- Мониторинг запущенных процессов. А также возможность перезапускать “упавший” сервис.
- Замена Cron-у
- Журналирование событий. JournalD

SystemD. Загрузка

- Сначала запускается default.target -> graphical.target
- multi-user.target запускает все зависимости нужные для перехода в мульти-юзер режим. Например, запускается сервис отвечающий за авторизацию пользователей
- Так же из зависимостей запускается basic.target и sysinit.target
- sysinit.target - один из самых важных уровней. Он отвечает за монтирование ФС и Swap, например.
- В итоге, это просто запуск зависимых друг от друга сервисов (в контексте systemd)

systemd-analyze time

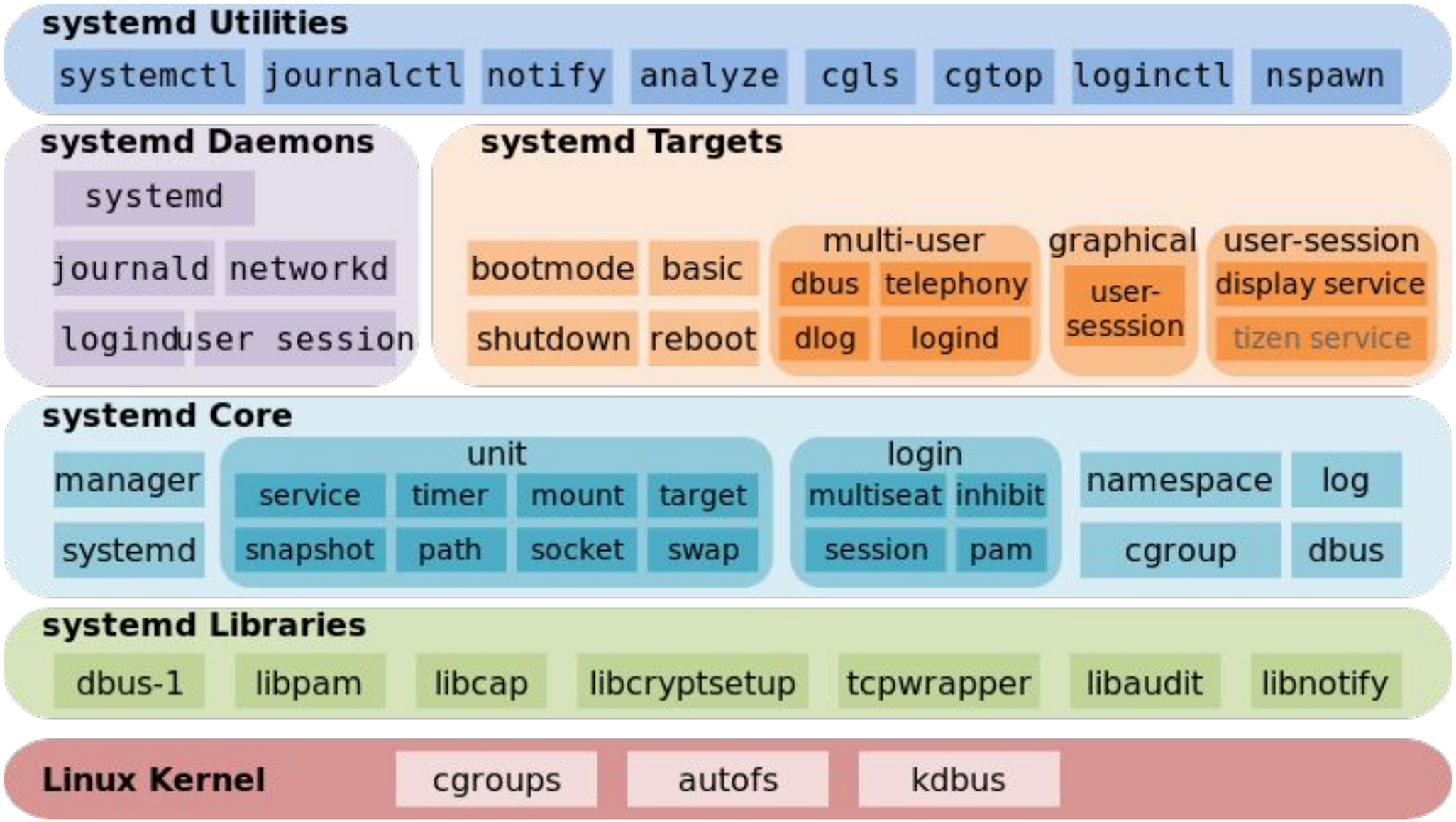
systemd-analyze blame

Systemd conf

При запуске операционной системы помимо выполнения задач инициализации systemd так же забирает параметры из файла /etc/systemd/system.conf

man systemd-system.conf

SystemD


systemd Utilities
systemctl
journalctl
notify
analyze
cgls
cgtop
loginctl
nspawn
systemd Daemons
systemd
journald
networkd
logind
user session
systemd Targets
bootmode
basic
shutdown
reboot
multi-user
dbus
telephony
dlog
logind
graphical
user-sesssion
user-session
display service
tizen service
systemd Core
manager
systemd
unit
service
timer
mount
target
snapshot
path
socket
swap
login
multiseat
inhibit
session
pam
namespace
log
cgroup
dbus
systemd Libraries
dbus-1
libpam
libcap
libcryptsetup
tcpwrapper
libaudit
libnotify
Linux Kernel
cgroups
autofs
kdbus

SystemD. Компоненты

journald - системный журнал

logind - демон управления аутентификацией пользователя

networkd - демон управления сетью

tmpfiles - управление временными файлами

timedated - демон для контроля времени/даты

udevd - девайс менеджер. Поддерживает /dev в актуальном состоянии

libudev - стандартная библиотека для udev, которая позволяет сторонним разработчикам делать запросы к udev

systemd-boot - UEFI менеджер загрузки

SystemD. Юниты

- service - аналог демона или что-либо, что можно запустить;
- device - факт подключения какого-либо устройства (имя юнита генерируется из sysfs-имени устройства);
- target - ничего не описывает, группирует другие юниты;
- mount - точка монтирования файловой системы (имя юнита должно соотвествовать пути до точки монтирования);
- automount - аналог autofs: точки автомонтирования (должен существовать *.mount-юнит с тем же именем);
- timer - аналог cron. Периодический запуск другого юнита (по умолчанию запускаться будет *.service-юнит с тем же именем);
- socket - аналог xinetd. Запуск юнита при подключении к указанному сокету (по умолчанию запускаться будет *.service-юнит с тем же именем);
- path - запуск юнита по событию доступа к какому-либо пути в файловой системе (по умолчанию запускаться будет *.service-юнит с тем же именем);
- slice - группирует другие юниты в дереве cgroups, позволяя иерархично задавать ограничения по используемым ресурсам;

SystemD. Targets

Это аналог уровней исполнения init. На самом деле это такой же юнит файл как и другие, за одним исключением - у него нет ExecStart. Он не запускает сервисы - он их группирует.

```
[root@centos7 system]# ls -l runlevel?.target
runlevel0.target -> poweroff.target
runlevel1.target -> rescue.target
runlevel2.target -> multi-user.target
runlevel3.target -> multi-user.target
runlevel4.target -> multi-user.target
runlevel5.target -> graphical.target
runlevel6.target -> reboot.target
```

SystemD. Targets

Можно просто запустить нужный **target**. Например:

```
systemctl start poweroff.target
```

Но гораздо проще использовать сокращенный аналог, убрав **start** и **target**, т.е.:

```
systemctl poweroff
systemctl halt
systemctl reboot
systemctl rescue
```

SystemD

Основные каталоги:

- /usr/lib/systemd - основной каталог
- /usr/lib/systemd/system - все unit-файлы прилетающие из пакетов
- /etc/systemd - конфигурация
- /etc/systemd/system - оверлей куда пишет администратор, положенные сюда юнит файлы имеют высший приоритет над юнит файлами из /usr/lib/
  - /etc/systemd/system/*.wants - каталоги уровней, в которые кладутся симлинки при systemctl enable
- /etc/sysconfig - каталог, содержащий переменные

Пример unit файла

```
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target
Documentation=man:httpd(8)
Documentation=man:apachectl(8)
[Service]
Type=notify # simple или forking
EnvironmentFile=/etc/sysconfig/httpd
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
ExecStop=/bin/kill -WINCH ${MAINPID}
KillSignal=SIGCONT
PrivateTmp=true
[Install]
WantedBy=multi-user.target
```

systemctl cat sshd.service - мега лайфхак!

SystemD управление

systemctl status somename.service

systemctl start|stop|(try)-restart|reload somename.service

systemctl reload|reload-or-(try)-restart somename.service

systemctl (re)enable (--now)|disable|mask|unmask somename.service

systemctl daemon-reload

SystemD проверка состояние служб

--is-enabled

--is-active

--is-failed

--failed

systemctl list-units (--all) --type=mount

systemctl list-unit-files

systemctl list-dependencies sshd.service

Ulimits

ulimit -d 48000 # Ограничиваем максимальный размер сегмента данных в 48 MB
ulimit -m 48000 # Ограничиваем максимальный размер резидентной части процесса (находящейся в ОЗУ) в 48 MB
ulimit -u 64   # Ограничиваем максимальное число запущенных этим пользователем процессов.
ulimit -n 128   # Ограничиваем максимальное число открытых файлов.
ulimit -f 100000 # Ограничиваем максимальный размер создаваемого файла в 100 MB
ulimit -v 100000 # Ограничиваем максимальный размер используемой виртуальной памяти в 100 MB

Для просмотра текущих настроек: ulimit -a

SystemD Limits

```
[Unit]
Description=Your Service
After=Some.target

[Service]
User=postgres
Group=postgres
LimitNOFILE=50000
LimitFSIZE=infinity
LimitNOPROC=500000
LimitCORE=infinity
LimitRSS=infinity
OOMScoreAdjust=-100
ExecStart=/var/lib/postgresql/11/bin/pg_ctl -D /var/data/base start

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

SystemD Limits

```
Directive          ulimit equivalent     Unit
LimitCPU=          ulimit -t             Seconds
LimitFSIZE=        ulimit -f             Bytes
LimitDATA=         ulimit -d             Bytes
LimitSTACK=        ulimit -s             Bytes
LimitCORE=         ulimit -c             Bytes
LimitRSS=          ulimit -m             Bytes
LimitNOFILE=       ulimit -n             Number of File Descriptors
LimitAS=           ulimit -v             Bytes
LimitNPROC=        ulimit -u             Number of Processes
LimitMEMLOCK=      ulimit -l             Bytes
LimitLOCKS=        ulimit -x             Number of Locks
LimitSIGPENDING=   ulimit -i             Number of Queued Signals
LimitMSGQUEUE=     ulimit -q             Bytes
LimitNICE=         ulimit -e             Nice Level
LimitRTPRIO=       ulimit -r             Realtime Priority
LimitRTTIME=       No equivalent
```

http://man7.org/linux/man-pages/man2/setrlimit.2.html

SystemD шаблонизация

Шаблон в юнит-файле позволяет systemd работать с различными экземплярами юнитов при условии использования единственного юнит-файла.
systemctl start <имя_службы>@<аргумент>.service

- Идентификатор %i позволяет осуществлять передачу аргумента, который будет особым образом отформатирован (из него будут убраны все спецсимволы)
- Идентификатор %I позволяет осуществить передачу аргумента без какой-либо обработки

```
[Unit]
Description=OpenVPN Robust And Highly Flexible Tunneling Application On %I
After=network.target

[Service]
Type=notify
PrivateTmp=true
ExecStart=/usr/sbin/openvpn --cd /etc/openvpn/ --config %i.conf

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

SystemD типы

- **simple** - процесс описанный в опции ExecStart является основным процессом сервиса. После запуска systemd запускает зависимые юниты.
- **forking** - процесс описанный в опции ExecStart вызовет fork() при запуске. Стандартное поведение UNIX демонов. Здесь рекомендуется использовать PIDFile опцию, чтобы systemd мог правильно идентифицировать основной процесс демона. Systemd запускает зависимые юниты сразу же по выходу родительского процесса
- **oneshot** - похоже на *simple*, но перед стартом зависимых юнитов ожидается, что основной процесс будет завершен.
- **notify** - похоже на *simple*, но запускает зависимые юниты только после того как основной процесс пошлет уведомление через sd_notify или другой системный вызов об успешном старте.

Также у нас имеется "лицензия на убийство":

```
systemctl kill unit_name
```

По умолчанию процесс “убивается” так как описано в ExecStop, но можно послать определенный сигнал с помощью опции -s. Имена сигналов можно получить набрав команду kill -l

```
systemctl kill -s HUP unit_name
```

Хотите перезапустить все юниты и перестроить дерево зависимостей? Не проблема

```
systemctl daemon-reload
```

Не только лишь сервисы:

mount - расширеный вариант опции _netdev в fstab
https://www.freedesktop.org/software/systemd/man/systemd.mount.html

timer - расширенный крон
https://www.freedesktop.org/software/systemd/man/systemd.timer.html

Пример mount сервиса:

/etc/systemd/system/media-nfs.mount:

```
[Unit]
Description=NFS share
[Mount]
What=server.url.example.com:/srv/nfs_share
Where=/media/nfs
Type=nfs4
Options=rw
DirectoryMode=0755
```

/etc/systemd/system/media-nfs.automount:

```
[Unit]
Description=NFS share
Requires=openvpn@vpn.service
Requires=network-online.target
[Automount]
Where=/media/nfs
TimeoutIdleSec=301
[Install]
WantedBy=graphical.target
```

Пример timer сервиса:

/etc/systemd/system/timer.timer

```
[Unit]
Description=Run backup every hour

[Timer]
# Run every one hour or one munite
OnCalendar=*:0/15

Unit=timer.service

[Install]

WantedBy=multi-user.target
```

/etc/systemd/system/timer.service

```
[Unit]

Description=My backup script


[Service]

Type=oneshot

ExecStart=/etc/scripts/backup.sh
```

Systemd-nspawn

Минимальные действия для запуска контейнера:

```
yum -y --nogpg --releasever=7 --installroot=/var/lib/machines/centos7 install \
systemd systemd-networkd passwd yum vim-enhanced bash bash-completion \
less psmisc openssh-server iputils iproute net-tools

systemd-nspawn --machine=centos7
```

Systemd-nspawn

Управление контейнерами:

```
[Unit]
Description=Test Container

[Service]
LimitNOFILE=100000
ExecStart=/usr/bin/systemd-nspawn --machine=centos7
--bind=/usr/share/nginx/html --boot
Restart=always

[Install]
Also=dbus.service
```

Systemd-nspawn

Управление контейнерами:

machinectl list - просмотреть список контейнеров
machinectl status test_container - просмотреть информацию о статусе контейнера
machinectl login test_container - войти в контейнер:
machinectl reboot test_container - перезагрузить контейнер:
machinectl poweroff test_container - остановить контейнер

JournalD

Сервис временного хранения логов.

По-умолчанию STDOUT и STDERR перенаправляется в сокет journald

Просмотр базы логов:
journalctl -u <unit>

journalctl -M <container_name>

Ваши вопросы?

Домашнее задание

1. Написать сервис, который будет раз в 30 секунд мониторить лог на предмет наличия ключевого слова. Файл и слово должны задаваться в /etc/sysconfig

2. Из epel установить spawn-fcgi и переписать init-скрипт на unit-файл. Имя сервиса должно называться также

3. Дополнить юнит-файл apache httpd возможностью запустить несколько инстансов сервера с разными конфигами

4*. Скачать демо-версию Atlassian Jira и переписать основной скрипт запуска на unit-файл

Заполните, пожалуйста, опрос в ЛК о занятии

Спасибо
за внимание!
До встречи в Slack и на вебинаре