systemd – ściągawka

Zimowisko TLUG 2014

Tomasz Torcz
<tomek@pipebreaker.pl>

Rozpiska

dlaczego

po co

co z tego dla admina

dobre praktyki

Know your tools

nie śpię, bo restartuję twittera

Why would I care?

połowa linuksianego świata: Fedora, Arch, Mageia

druga połowa: Debian 8.0 “Jessie”

trzecia połowa: RHEL7 → CentOS, Scientific etc.

czwarta połowa: Ubuntu!

no cake: ChromeOS, Android/Linux
aczkolwiek: SailfishOS (Jolla), GENIVI, LSB5.0

Pisownia

systemd

~~SystemD~~ ~~systemD~~ ~~SyStEmD~~

Co to robi?

zarządza systemem i usługami

dzisiaj skupimy się na usługach

Usługi

program lub zespół programów

działający cały czas lub od czasu do czasu

opis/definicja usługi to *unit*

uptimed.service

```
[Unit]
Description=Uptime record tracking daemon
Documentation=man:uptimed(8) man:uprecords(1)

[Service]
ExecStartPre=/usr/sbin/uptimed -b
ExecStart=/usr/sbin/uptimed -f

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

Usługi

program lub zespół programów

działający cały czas lub od czasu do czasu

opis/definicja usługi to *unit*

unity zebrane w katalogach:
/usr – systemowe; /etc – administratora;
/run – konfiguracja nietrwała (*runtime*)

Katalogi

/usr/lib/systemd/system – systemowe nie ruszać
(/lib/systemd/system – sprawdzić, czy nie Debian)

/etc/systemd/system – konfiguracja właściwa
systemctl enable, systemctl set-property

/run – generatory, najwyższy priorytet

systemctl start exim4

Czy już działa? Nie, więc:

czy jest **/run/**systemd/system/exim4.service ? nie

czy jest **/etc/**systemd/system/exim4.service ? nie

czy jest **/usr/lib/**systemd/system/exim4.service ?

a może jest **/etc/init.d/**exim4

Skąd ty?

 # systemctl status polipo-purge.service

polipo-purge.service - flush polipo disk cache

Loaded: loaded (**/etc/systemd/system/polipo-purge.service; static**)

[...]

 # systemctl status hp-health.service

hp-health.service - LSB: hp System Health Monitor and Command line Utility Package.

Loaded: loaded (**/etc/rc.d/init.d/hp-health**)

[...]

Unity

1) usługi zdefiniowane są w unitach

2) najważniejsza linijka to ExecStart= wskazująca binarkę

3) kolejność szukania: /run → /etc → /usr (→ SysV)

Cykl życia i śledzenie wykonania

start → gotowość → działanie → zakończenie

start: lockfile nie potrzebny, czyste środowisko

gotowość: można uruchamiać zależne

działanie: procesy potomne w ramach usługi

6811 ? S 0:00 pickup -l -t unix -u

systemctl status **6811**

postfix.service - Postfix Mail Transport Agent
  Loaded: loaded (/usr/lib/systemd/system/postfix.service; enabled)
  Active: active (running) since Thu 2014-02-06 16:39:22 CET; 46s ago
 Process: 6735 ExecStart=/usr/sbin/postfix start (code=exited, status=0/SUCCESS)
 Process: 6732 ExecStartPre=/usr/libexec/postfix/chroot-update (code=exited, status=0/SUCCESS)
 Process: 6727 ExecStartPre=/usr/libexec/postfix/aliasesdb (code=exited, status=0/SUCCESS)
Main PID: 6810 (master)
  CGroup: /system.slice/postfix.service
          ├─6810 /usr/libexec/postfix/master -w
          ├─**6811 pickup -l -t unix -u**
          └─6812 qmgr -l -t unix -u

Feb 06 16:39:22 mother.pipebreaker.pl postfix/master[6810]: daemon started -- version 2.10.2, configuration /etc/postfix
Feb 06 16:39:22 mother.pipebreaker.pl systemd[1]: Started Postfix Mail Transport Agent.

Cykl życia i śledzenie wykonania

start → gotowość → działanie → zakończenie

start: lockfile nie potrzebny, czyste środowisko

gotowość: można uruchamiać zależne

działanie: procesy potomne w ramach usługi

zakończenie: OK czy crash? restartować?

SuccessExitStatus= Restart= OnFailure=

Typy

określanie gotowości

start przed timeout'em

spełnienie zależności

określanie zakończenia

pomyślne wyjście?

restart?

6 różnych, 3 podstawowe:

simple **forking** **oneshot**

Type=simple

najprostszy

brak sygnalizacji gotowości

domyślny, trywialna demonizacja

```
[Service]
ExecStart=/root/bin/looper.sh
```

Type=forking

dla tradycyjnych demonów

gotowość: fork() + exit()

zalecany PIDFile=

Type=oneshot

dla skryptów

gotowość: po zakończeniu pracy

przydatny RemainAfterExit=

Jak rozpoznać typ?

Jak rozpoznać typ?

 # /usr/sbin/daemon

Copyright 2014 Foo Bar Baz Corp.

Serving Requests...

→ Type=simple

 # /usr/sbin/otherdaemon

#

→ Type=forking

Pozostałe typy

dbus – dla serwisów D-Bus/KDBus

gotowość: zarejestrowanie nazwy

idle – jak simple

uruchomienie następuje „po zbootowaniu”

notify – demon komunikuje się z systemd

wymaga minimalnego patcha

Na złość mamie

simple zamiast forking?

zaraz po starcie usługa zostanie ubita

forking zamiast simple?

po czasie TimeoutStartSec= usługa ubita

simple zamiast oneshot?

usługa przejdzie w stan failed

Typy

domyślny **simple**

pozwalają na określenie gotowości

błędne podanie kończy się płaczem

Po swojemu

Po swojemu

zmiany wprowadzamy w katalogu /etc/...

1) skopiować unit z /usr/... i edytować

2) .include i zmiany tego, co nas interesuje

3) *.d (drop-in dirs)

Dropins

zachowana hierarchia run/etc/usr

katalog <unit.type>.d/, pliki *.conf

pamiętać o [Sekcja]

satellite.target

% cat /etc/systemd/system/**satellite.target.d**/mount.conf

```
[Unit]
RequiresMountsFor=/var/satellite
```

transmission-daemon.service.d/

% systemctl status transmission-daemon.service

transmission-daemon.service - Transmission BitTorrent Daemon
  Loaded: loaded (**/usr/lib/**systemd/system/transmission-daemon.service; enabled)
  Drop-In: **/etc/**systemd/system/transmission-daemon.service.d
          └─**10-user-zdzichu.conf, 20-io-sched-idle.conf, 30-exec-noauth.conf, 40-restart.conf**
   Active: active (running) since Sat 2014-01-18 12:22:28 CET; 4 days ago

transmission-daemon.service.d/

10-user-zdzichu.conf:

```
[Service]
User=zdzichu
```

20-io-sched-idle.conf:

```
[Service]
IOSchedulingClass=idle
```

30-exec-noauth.conf:

```
[Service]
ExecStart=
ExecStart=/usr/bin/transmission-daemon -f --log-error -T
```

40-restart.conf:

```
[Service]
Restart=always
```

systemd-delta

przegląd zmian

[OVERRIDDEN] /etc/systemd/system/auth@.service → /usr/lib/systemd/system/auth@.service
--- /usr/lib/systemd/system/auth@.service 2013-08-05 11:05:34.000000000 +0200
+++ /etc/systemd/system/auth@.service 2013-01-27 17:28:54.973741536 +0100

@@ -4,5 +4,5 @@

[Service]
User=ident
-ExecStart=/usr/sbin/in.authd -t60 --xerror --os -E
+ExecStart=/usr/sbin/in.authd -t60 --xerror --os
StandardInput=socket

Wartości domyślne + zewn. konfig

/etc/default/* i /etc/sysconfig/*

```
EnvironmentFile=/etc/sysconfig/ladvd
ExecStart=/usr/sbin/ladvd $OPTIONS
```

różnice między distro

poszukanie konfiguracji przez admina

Wartości domyślne + zewn. konfig

1) przenieść konfigurację do unita

2) zminimalizować zmienne środowiskowe

3) reguły nadpisywania, *system-delta*

Wartości domyślne + zewn. konfig

. /etc/sysconfig/nfs

/usr/sbin/rpc.mountd

~~${MOUNTD_PORT:+-p $MOUNTD_PORT}~~

[Service]

Environment=MOUNTD_PORT=9001

EnvironmentFile=-/etc/sysconfig/nfs

ExecStart=/usr/sbin/rpc.mountd -p $MOUNTD_PORT

Po swojemu

Konfiguracja w /etc/

Kopiowanie, .include, **dropiny**

systemd-delta *systemctl cat (v209)*

Unity nie gryzą

dzielić na mniejsze

- ładowanie modułów? /etc/modules-load.d/
- tworzenie katalogów, plików? tmpfiles.d
- różne czynności? ExecStartPre= lub inne unity
- wspólne restarty? PartOf=

Condition*= Twoimi przyjaciółmi

sshd@.service

```
[Unit]
Description=OpenSSH per-connection server daemon
Wants=sshd-keygen.service
After=auditd.service sshd-keygen.service
```

 # cat sshd-keygen.service

```
[Unit]
Description=OpenSSH Server Key Generation
ConditionPathExists=|!/etc/ssh/ssh_host_rsa_key
ConditionPathExists=|!/etc/ssh/ssh_host_dsa_key
ConditionPathExists=|!/etc/ssh/ssh_host_ecdsa_key
```

abrt-vmcore.service

abrt-vmcore.service - Harvest vmcores for ABRT

Loaded: loaded (/usr/lib/systemd/system/abrt-vmcore.service; enabled)

Active: inactive (dead)

start **condition failed** at Thu 2014-02-06 16:42:11 CET; 5s ago

ConditionDirectoryNotEmpty=/var/crash was not met

Feb 06 16:42:11 mother.pipebreaker.pl systemd[1]: Started Harvest vmcores for ABRT.

Unity nie gryzą

dzielić na mniejsze

- ładowanie modułów? /etc/modules-load.d/
- tworzenie katalogów, plików? tmpfiles.d
- różne czynności? ExecStartPre= lub inne unity
- wspólne restarty? PartOf=

Condition*= Twoimi przyjaciółmi

Type=notify

pełna informacja

rozszerzony status

- sd_notify("STATUS=Robię dwójkę")
- systemd-notify --status="Przetwarzam plik 42 z 69."

rozszerzony status

httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
  Active: active (running) since czw 2014-02-13 11:10:46 CET; 4 days ago
  Process: 27130 ExecReload=/usr/sbin/httpd $OPTIONS -k graceful (code=exited, status=0/SUCCESS)
Status: "Total requests: 3432; Current requests/sec: 4; Current traffic: 2423 B/sec"
   CGroup: /system.slice/httpd.service
           ├─ 6368 /usr/sbin/httpd -DFOREGROUND
           ├─ 8408 /usr/sbin/httpd -DFOREGROUND

transmission-daemon.service - Transmission BitTorrent Daemon
  Loaded: loaded (/usr/lib/systemd/system/transmission-daemon.service; enabled)
  Drop-In: /etc/systemd/system/transmission-daemon.service.d
           └─10-user-zdzichu.conf, 20-io-sched-idle.conf, 30-exec-noauth.conf, 40-restart.conf
   Active: active (running) since Sat 2014-01-18 12:22:28 CET; 4 weeks 2 days ago
Status: "Uploading 0.00 KBps, Downloading 0.02 KBps."
   CGroup: /system.slice/transmission-daemon.service
           └─9096 /usr/bin/transmission-daemon -f --log-error -T

rozszerzony status - python

```
#!/usr/bin/python
#
# logs temperature from OWFS into postgres database

import glob
import os

import psycopg2
import systemd.daemon
import time

sleep_seconds = 60

systemd.daemon.notify("STATUS=Opening DB connection...")
dbconn = psycopg2.connect("dbname=temperature_log")
dbconn.autocommit=True
cur = dbconn.cursor()
cur.execute("PREPARE put_temperature AS INSERT INTO temperatures (datetime, sensor_id, value) VALUES (NOW(), (SELECT id FROM sensors WHERE SN=$1), $2);")

systemd.daemon.notify("READY=1")
systemd.daemon.notify("STATUS=Entering main loop")
while True:
    for SN in glob.glob("/run/owfs/??.????????????"):
        systemd.daemon.notify("STATUS=Reading sensors...")
        temperature = open("%s/temperature" % SN).readline()
        # we won't be needing full path anymore, trim it
        SN = os.path.basename(SN)

        try:
            cur.execute("EXECUTE put_temperature (%s, %s);", (SN, temperature) )
        except psycopg2.IntegrityError:
            print "New sensor %s! Adding to database, please correct description." % SN
            cur.execute("INSERT INTO sensors (SN) VALUES (%s)", (SN,))
            cur.execute("EXECUTE put_temperature (%s, %s);", (SN, temperature) )

    systemd.daemon.notify("STATUS=Sleeping until %s" % time.ctime(time.time() + sleep_seconds))
    time.sleep(sleep_seconds)

systemd.daemon.notify("STATUS=Cleaning up")
dbconn.close()
systemd.daemon.notify("READY=0")
```

rozszerzony status - python

```
import systemd.daemon

systemd.daemon.notify("STATUS=Opening DB connection...")

systemd.daemon.notify("READY=1")
systemd.daemon.notify("STATUS=Entering main loop")

            systemd.daemon.notify("STATUS=Reading sensors...")

      systemd.daemon.notify("STATUS=Sleeping until %s"
          % time.ctime(time.time() + sleep_seconds))

systemd.daemon.notify("STATUS=Cleaning up")

systemd.daemon.notify("READY=0")
```

rozszerzony status - python

% systemctl --user status logtemp

logtemp.service - database logging of temperature from OWFS
   Loaded: loaded (/home/zdzichu/dev/logtemp/logtemp.service; enabled)
   Active: active (running) since Thu 2014-02-06 16:56:23 CET; 1 weeks 3 days ago
 Main PID: 11078 (logtemp.py)
   Status: "Sleeping until Mon Feb 17 13:35:51 2014"
   CGroup: /user.slice/user-1001.slice/user@1001.service/logtemp.service
           └─11078 /usr/bin/python /home/zdzichu/dev/logtemp/logtemp.py

Feb 06 16:56:22 mother.pipebreaker.pl systemd[1251]: Stopping database logging of temperature from OWFS...
Feb 06 16:56:22 mother.pipebreaker.pl systemd[1251]: Starting database logging of temperature from OWFS...
Feb 06 16:56:23 mother.pipebreaker.pl systemd[1251]: Started database logging of temperature from OWFS.

Type=notify

dokładnie określa gotowość usługi

rozszerza informację o statusie

wymaga minimalnego łatanie

systemctl enable debug-shell

debug-shell.service - Early root shell on /dev/tty9 FOR DEBUGGING ONLY
   Loaded: loaded (/usr/lib/systemd/system/debug-shell.service; disabled)
   Active: inactive (dead)
     Docs: man:sushell(8)

Pomocne również (*kernel command line*):

- systemd.confirm_spawn=1
- systemd.unit=
- rescue, emergency

journalctl

logi od samego startu (również *initramfs*)

journalctl /dev/sda

journalctl -u postfix

journalctl --since=yesterday

urlz for lulz

- systemctl help <service>
- **http://www.freedesktop.org/software/systemd/man/**systemd.directives.html
- T-DOSE 2012, Open Source Lennart Poettering, The Systemd Journal
  http://www.youtube.com/watch?v=o1lUeQVYuNs
- CoreOS fleet – distributed init system
  http://www.youtube.com/watch?v=u91DnN-yaJ8
- http://pkgs.fedoraproject.org/cgit/

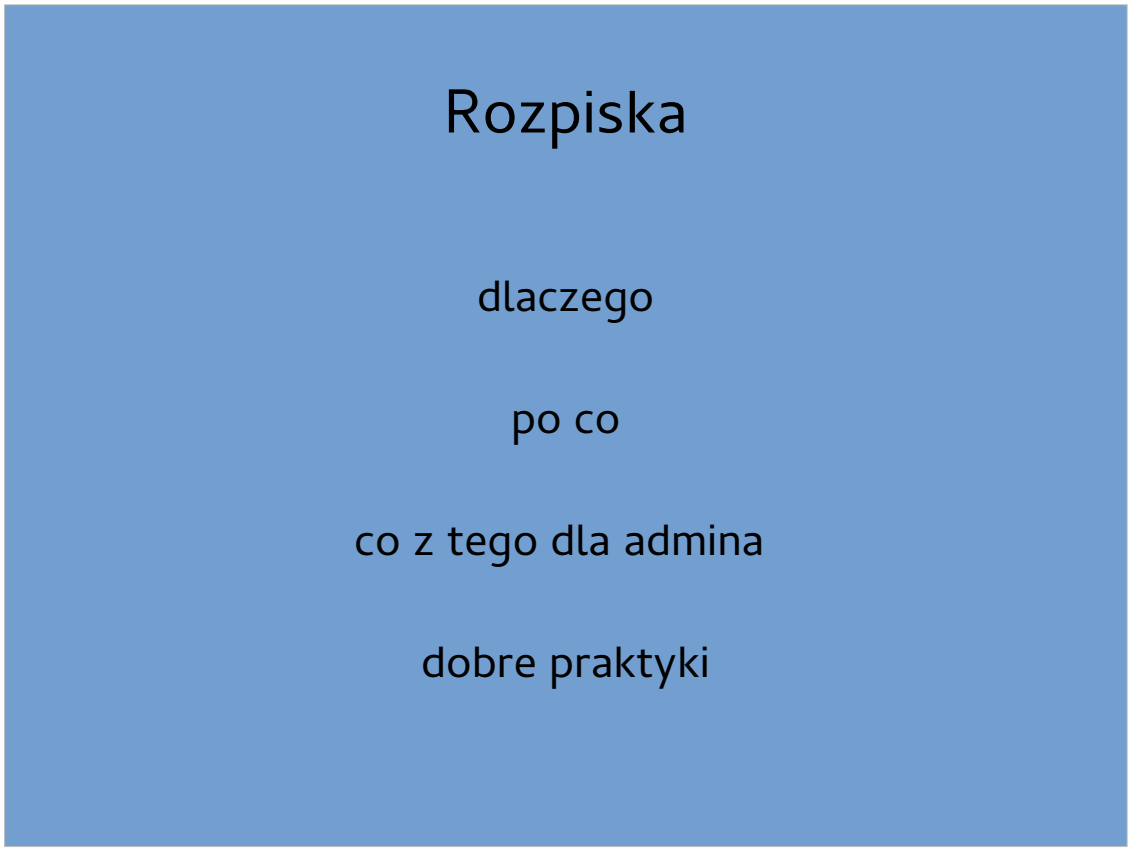
systemd – ściągawka

Dziękuje!

systemd – ściągawka

Zimowisko TLUG 2014

Tomasz Torcz
<tomek@pipebreaker.pl>



Co do prezentacji: materiału jest dość dużo, ale postarałem się zrobić komentarze w slajdach – zachęcam więc do ściągnięcia.
(jeśli to czytasz, to dobrze, że ściągnąłeś :)

Powrót do tematu z zimowiska 2011, dlaczego?

Bo systemd nie przeminął, a wręcz przeciwnie.

Twórca twittera na początku dużo czasu poświęcał na sprawdzanie, czy jego aplikacja się nie wywróciła.
Spał z otwartym laptopem, budząc się co jakiś czas i sprawdzając jej stan.

Gdyby znał swoje narzędzia trochę lepiej, byłby w stanie napisać chociażby *while true; do ./twitter; done*

Prawdziwym wybawieniem był dla niego /etc/inittab

Why would I care?

połowa linuksianego świata: Fedora, Arch, Mageia

druga połowa: Debian 8.0 "Jessie"

trzecia połowa: RHEL7 → CentOS, Scientific etc.

czwarta połowa: Ubuntu!

no cake: ChromeOS, Android/Linux
aczkolwiek: SailfishOS (Jolla), GENIVI, LSB5.0

Debian – decyzja 11 lutego 2014
Ubuntu – 14 lutego 2014

Mimo opanowania 200% linuksowego świata, wciąż nie ma systemd w najpopularniejszej dystrybucji:
As of September 2013, one billion Android devices have been activated

Jednakże są telefony z systemd – system operacyjny Sailfish, telefony Jolla (ciekawostka: sprzedawane od 2013, wersja 1.0 Sailfisha – 21 lutego 2014); również Tizen (Samsung/Intel)

GENIVI to konsorcjum robiące automotive Linux (dla samochodów)

LSB = Linux Standards Base

Pisownia

systemd

~~SystemD~~ ~~systemD~~ ~~SyStEmD~~

Co to robi?

zarządza systemem i usługami

dzisiaj skupimy się na usługach

Dzięki systemd Admin może sobie ułatwić życie na 100 sposobów.
Ale mam tylko jeden wykład, nie 100.

Usługi

program lub zespół programów

działający cały czas lub od czasu do czasu

opis/definicja usługi to *unit*

zespół programów – gdyż usługa może składać się z wielu procesów. W takich przypadkach zazwyczaj wszystkie te procesy potrzebne są do działania. Razem powinny podlegać restartom, stopowi, ograniczaniu ich zasobów.

Typowe demony działają cały czas.
Od czasu do czasu – zadania uruchamianie z **crona**, z **atd**; Np. CUPS – uruchamiany, gdy w obserwowanych katalogu pojawi się coś do wydruku.
Niezależnie od sposobu uruchomienia – tak samo sprawdzamy status, modyfikujemy, zabijamy

opis/definicja *w systemd* to unit

uptimed.service

```
[Unit]
Description=Uptime record tracking daemon
Documentation=man:uptimed(8) man:uprecords(1)

[Service]
ExecStartPre=/usr/sbin/uptimed -b
ExecStart=/usr/sbin/uptimed -f

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

To jest przykładowy unit.

Pierwsza sekcja zawiera sprawy opisowe. Może również zawierać odniesienia do innych unitów (definicję zależności)

Druga - to „usługa właściwa”

Trzecia zawiera tylko informację, w jakich warunkach ma być uruchomiony. multi-user.target to taki typowy pełen start systemu.

Usługi

program lub zespół programów

działający cały czas lub od czasu do czasu

opis/definicja usługi to *unit*

unity zebrane w katalogach:
/usr – systemowe; /etc – administratora;
/run – konfiguracja nietrwała (*runtime*)

zespół programów – gdyż usługa może składać się z wielu procesów. W takich przypadkach zazwyczaj wszystkie te procesy potrzebne są do działania. Razem powinny podlegać restartom, stopowi, ograniczaniu ich zasobów.
Typowe demony działają cały czas.
Od czasu do czasu – zadania uruchamianie z crona, z atd; Np. CUPS – uruchamiany, gdy w obserwowanych katalogu pojawi się coś do wydruku.

opis/definicja *w systemd* to unit

Katalogi – ważna sprawa, więc od niej zaczniemy; to daje punkt zaczepienia – gdzie szukać

Katalogi

/usr/lib/systemd/system – systemowe nie ruszać
(/lib/systemd/system – sprawdzić, czy nie Debian)

/etc/systemd/system – konfiguracja właściwa
systemctl enable, systemctl set-property

/run – generatory, najwyższy priorytet

do /usr trafiają rzeczy przygotowane przez dystrybucję – instalowane w pakietach
dlaczego systemd/system (masło/maślane)?

bo to są usługi **systemowe**; użytkownicy mogą mieć swoje
systemctl –user szuka podobnie, ale s/system/user/ i dodatkowo w ~/.config/systemd/user

/run – rzeczy tworzone w trakcie działania systemu, znikają przy reboocie

systemctl start exim4

Czy już działa? Nie, więc:

czy jest **/run/**systemd/system/exim4.service ? nie

czy jest **/etc/**systemd/system/exim4.service ? nie

czy jest **/usr/lib/**systemd/system/exim4.service ?

a może jest **/etc/init.d/**exim4

To jest przykład wyszukiwania unitów

Jak zadziała tyłowe polecenie, które startuje usługę exim4?

Sprawdzone wszystkie ścieżki i wciąż nie ma?

SYSV compatibility!

Mask – te czerwone mogą być linkiem do /dev/null
- zabranianie startu (zwłaszcza na żądanie) – mocniejsza wersja disable
- nie można też wystartować ręcznie

Skąd ty?

 # systemctl status polipo-purge.service
polipo-purge.service - flush polipo disk cache
   Loaded: loaded (**/etc/systemd/system/polipo-purge.service; static**)
[...]

 # systemctl status hp-health.service
hp-health.service - LSB: hp System Health Monitor and Command line Utility Package.
   Loaded: loaded (**/etc/rc.d/init.d/hp-health**)
[...]

Wow, tak dużo katalogu, tyle możliwości. Jak się połapać, skąd był wzięty unit?

Odp: systemctl status

Unity

1) usługi zdefiniowane są w unitach

2) najważniejsza linijka to ExecStart= wskazująca binarkę

3) kolejność szukania: /run → /etc → /usr (→ SysV)

fallback do sysv umożliwia powolną migrację: można utworzyć unit tak samo nazwany jak skrypt SysV i go przetestować

A po co? Musi być definicja ExecStart=, gdyż systemd śledzi cykl życia

Cykl życia i śledzenie wykonania

start → gotowość → działanie → zakończenie

start: lockfile nie potrzebny, czyste środowisko

gotowość: można uruchamiać zależne

działanie: procesy potomne w ramach usługi

Cykl, więc wyobraźcie sobie strzałkę z powrotem na początek

start: nie da się dwa razy tej samej usługi uruchomić, więc nie trzeba się obawiać kilku *systemctl start*;
środowisko minimalne, więc nie ma ryzyka, że jakieś zmienne są dziedziczione z rootshell;
brak ryzyka: w testach po ssh działa, po boocie już nie (bo inne środowisko)

gotowość: usługa zainicjowała się i obsługuje zapytania

działanie: ograniczenia zasobów per usługa
tutaj QUIZ z rozpoznawaniem

6811 ? S 0:00 pickup -l -t unix -u

systemctl status **6811**

postfix.service - Postfix Mail Transport Agent
  Loaded: loaded (/usr/lib/systemd/system/postfix.service; enabled)
  Active: active (running) since Thu 2014-02-06 16:39:22 CET; 46s ago
 Process: 6735 ExecStart=/usr/sbin/postfix start (code=exited, status=0/SUCCESS)
 Process: 6732 ExecStartPre=/usr/libexec/postfix/chroot-update (code=exited, status=0/SUCCESS)
 Process: 6727 ExecStartPre=/usr/libexec/postfix/aliasesdb (code=exited, status=0/SUCCESS)
Main PID: 6810 (master)
  CGroup: /system.slice/postfix.service
          ├─6810 /usr/libexec/postfix/master -w
          ├─6811 pickup -l -t unix -u
          └─6812 qmgr -l -t unix -u

Feb 06 16:39:22 mother.pipebreaker.pl postfix/master[6810]: daemon started -- version 2.10.2, configuration /etc/postfix
Feb 06 16:39:22 mother.pipebreaker.pl systemd[1]: Started Postfix Mail Transport Agent.

Jak mówiłem przy definicji usługi, na jedną usługę może składać się kilka procesów.
Tutaj są trzy, składają się na postfiksa.

Cykl życia i śledzenie wykonania

start → gotowość → działanie → zakończenie

start: lockfile nie potrzebny, czyste środowisko

gotowość: można uruchamiać zależne

działanie: procesy potomne w ramach usługi

zakończenie: OK czy crash? restartować?
SuccessExitStatus= Restart= OnFailure=

Cykl, więc wyobraźcie sobie strzałkę z powrotem na początek

start: nie da się dwa razy tej samej usługi uruchomić, więc nie trzeba się obawiać kilku *systemctl start*;
środowisko minimalne, więc nie ma ryzyka, że jakieś zmienne są dziedziczione z rootshell;
brak ryzyka: w testach po ssh działa, po boocie już nie (bo inne środowisko)

gotowość: usługa zainicjowała się i obsługuje zapytania

działanie: ograniczenia zasobów per usługa
tutaj QUIZ z rozpoznawaniem

zakończenie: różne reakcje, ubicie wszystkich proc.

Typy

określanie gotowości

start przed timeout'em

spełnienie zależności

określanie zakończenia

pomyślne wyjście?

restart?

6 różnych, 3 podstawowe:

simple **forking** **oneshot**

W nawiązaniu do wcześniej podanego cyklu życia

Gotowość: kiedy usługę można uznać za wystartowaną. Co za tym idzie: uruchomić usługi od niej zależne

Typ podaje się w unicie, definiując usługę.

Type=simple

najprostszy

brak sygnalizacji gotowości

domyślny, trywialna demonizacja

```
[Service]
ExecStart=/root/bin/looper.sh
```

Brak sygnalizacji gotowości

Kompatybilny z niektórymi innymi initami (np. daemontools DJBa)

odpowiednik wpisania do /etc/inittab z opcją *respawn*

zdejmuje z twórców demonów potrzebę forkowania, otwierania logów itp

Type=forking

dla tradycyjnych demonów

gotowość: fork() + exit()

zalecany PIDFile=

fork() tworzy dwa procesy, rodzica i potomny
rodzic kończy pracę – to znak dla systemd, że usługa jest gotowa

jeśli PIDFile= jest podany, to usługa uznawana jest za gotową jeśli w PIDFile zapisany zostanie PID głównego demona.

Type=oneshot

dla skryptów

gotowość: po zakończeniu pracy

przydatny RemainAfterExit=

proces musi się zakończyć

RemainAfterExit=true powoduje, że usługa po zakończeniu jest w stanie „aktywnym”. Czyli nie będzie można jej „uruchomić” drugi raz (np.. przez przypadek), można natomiast zrobić stop/start albo restart.

Przydatne np. do cronjobs

Jak rozpoznać typ?

Pisząc nowy unit, trzeba podać prawidłowy typ.
Jak określić „prawidłowy”?

Najprościej – odpalić z shella

Jak rozpoznać typ?

 # /usr/sbin/daemon
Copyright 2014 Foo Bar Baz Corp.
Serving Requests...

→ Type=simple

 # /usr/sbin/otherdaemon
#

→ Type=forking

Po uruchomieniu program działa „na konsoli”. Można go np.. przerwać przez Control-C
SIMPLE

Drugi program – przechodzi w tło (zapewne forkuje), wraca nam prompt shella.
FORKING

Oczywiście demony mogą mieć przełączniki zmieniające zachowanie:
--foreground
--daemonize
--[no]detach

Czasem
--debug (dodatkowo zwiększa output)

Pozostałe typy

dbus – dla serwisów D-Bus/KDBus
gotowość: zarejestrowanie nazwy

idle – jak simple
uruchomienie następuje „po zbootowaniu”

notify – demon komunikuje się z systemd
wymaga minimalnego patcha

nazwę podaje się w BusName=
podanie BusName= defaultuje typ to dbus
dla startowania on-demand

idle – przez „po zbootowaniu” rozumiany jest moment, kiedy po raz pierwszy od uruchomienia komputera kolejka zadań systemd jest pusta

notify opisany szerzej pod koniec

Na złość mamie

simple zamiast forking?

zaraz po starcie usługa zostanie ubita

forking zamiast simple?

po czasie TimeoutStartSec= usługa ubita

simple zamiast oneshot?

usługa przejdzie w stan failed

A co jeśli podamy zły typ?

simple zamiast forking czyli np., zapomnimy Type= Usługa forkuje, znika wystartowany proces: dla systemd jest to znak, że usługa się zakończyła, ubije pozostałe procesy

forking zamiast simple: czeka na fork i się nie doczeka; domyślny timeout 90 sekund

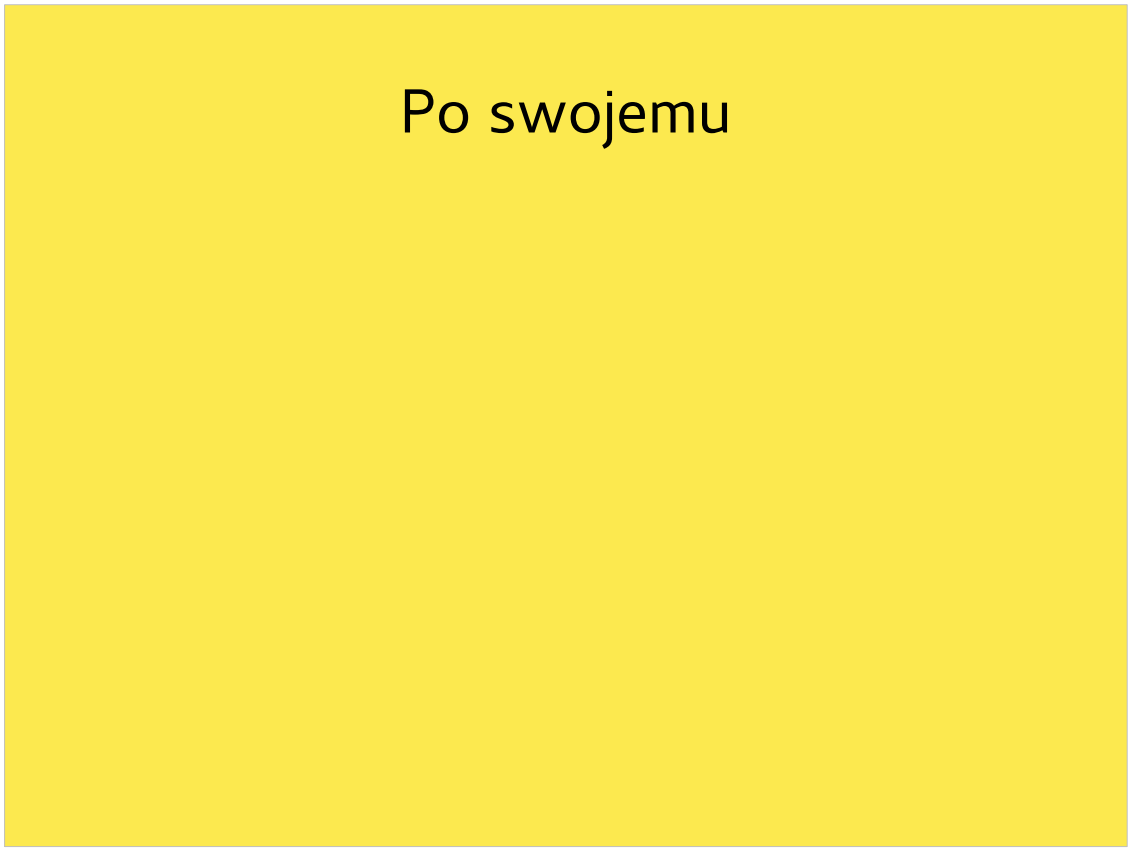
simple (czyli też brak) przy skryptach zamiast oneshot; failed powoduje niespełnione zależności; robi się failed, gdyż skrypt po zrobieniu roboty kończy się

Typy

domyślny **simple**

pozwalają na określenie gotowości

błędne podanie kończy się płaczem



Radziecka maszyna do golenia

Tak samo dystrybucja dostarcza „skryptów startowych”, czyli definicji unitów.

Admin czasem musi je zmodyfikować: dodać zależność, uruchomić powiązanie usługi, zmodyfikować **parametr**

Po swojemu

zmiany wprowadzamy w katalogu /etc/...

1) skopiować unit z /usr/... i edytować

2) .include i zmiany tego, co nas interesuje

3) *.d (drop-in dirs)

pamiętamy o kolejności wyszukiwania, i że adminowe /etc przysłania systemowe /usr
NIE edytujemy w /usr – zostanie nadpisane przy upgradzie pakietu

Trzy metody customizacji, ich wady i zalety:

etc→usr: + widać od razu całość; - przy aktualizacji pakietów może się coś zmienić

.include: + widać zmiany; - czasem trudno nadpisać oryginalne wartości; - metoda uznana za przestarzałą i wycofywana

dropiny: + łatwa automatyzacja; - nieoczywiste nadpisywanie

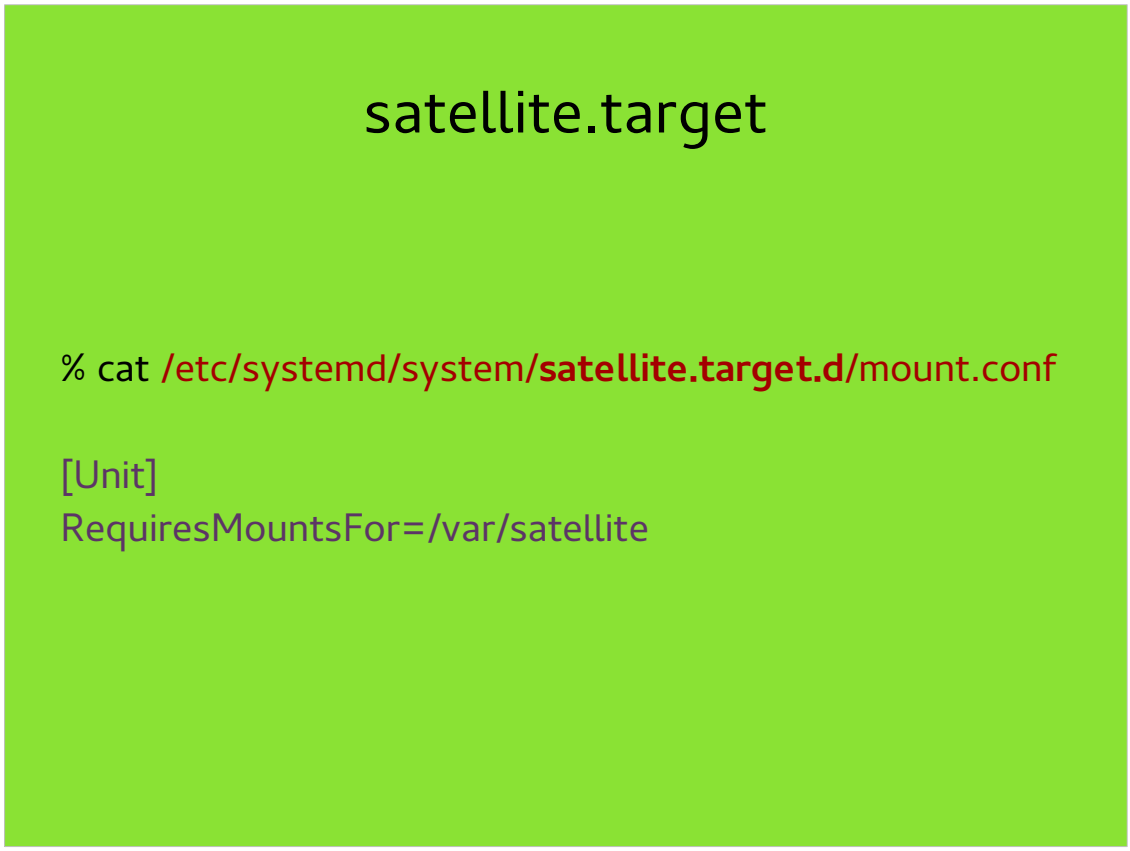
Dropins

zachowana hierarchia run/etc/usr

katalog <unit.type>.d/, pliki *.conf

pamiętać o [Sekcja]

systemctl set-property (ograniczenia via cgroups) – powoduje automatycznie tworzenie dropinów w /etc



Przykład dropinu

Poza usługami, systemd zarządza też punktami mountowania – głównie w celu umożliwienia wyrażenia zależności od nich

transmission-daemon.service.d/

% systemctl status transmission-daemon.service

transmission-daemon.service - Transmission BitTorrent Daemon
  Loaded: loaded (**/usr/lib/**systemd/system/transmission-daemon.service; enabled)
  Drop-In: **/etc/**systemd/system/transmission-daemon.service.d
          └─**10-user-zdzichu.conf, 20-io-sched-idle.conf, 30-exec-noauth.conf, 40-restart.conf**
   Active: active (running) since Sat 2014-01-18 12:22:28 CET; 4 days ago

Tutaj widzimy: niezmodyfikowana jednostka w /usr

customizacja dokonana dropinami – 4 pliki *conf

Przyjrzyjmy się zmianom...

transmission-daemon.service.d/

10-user-zdzichu.conf:

```
[Service]
User=zdzichu
```

20-io-sched-idle.conf:

```
[Service]
IOSchedulingClass=idle
```

30-exec-noauth.conf:

```
[Service]
ExecStart=
ExecStart=/usr/bin/transmission-daemon -f --log-error -T
```

40-restart.conf:

```
[Service]
Restart=always
```

where the assignment of an empty string removes any previous assignment (ExecStart is defined as a list, so that a "one-shot" service can run a list of processes) and the second assignment gives the desired complete list of command line arguments.

systemd-delta

przegląd zmian

[OVERRIDDEN] /etc/systemd/system/auth@.service → /usr/lib/systemd/system/auth@.service
--- /usr/lib/systemd/system/auth@.service 2013-08-05 11:05:34.000000000 +0200
+++ /etc/systemd/system/auth@.service 2013-01-27 17:28:54.973741536 +0100

@@ -4,5 +4,5 @@

[Service]
User=ident
-ExecStart=/usr/sbin/in.authd -t60 --xerror --os -E
+ExecStart=/usr/sbin/in.authd -t60 --xerror --os
StandardInput=socket

jak się zorientować w zmianach?

systemctl status może nie wystarczać

delta wyświetla modyfikacje, override, masked

Wartości domyślne + zewn. konfig

/etc/default/* i /etc/sysconfig/*

```
EnvironmentFile=/etc/sysconfig/ladvd
ExecStart=/usr/sbin/ladvd $OPTIONS
```

różnice między distro
poszukanie konfiguracji przez admina

Unity to nie skrypty. w skryptach startowych czasem znajdują się funkcje podejmujące decyzję.

Dwa, dystrybucje aby uniknąć modyfikacji samych skryptów startowych, część opcji wynoszą do zewnętrznych plików.

W systemd nie jest to konieczne, gdyż modyfikacje unitów są minimalne, proste i widoczne.

Absolutną herezją są rzeczy typu ENABLED=false w zewnetrznych plikach.
Usługa **niby** włączona, a nie startuje.

Wartości domyślne + zewn. konfig

1) przenieść konfigurację do unita

2) zminimalizować zmienne środowiskowe

3) reguły nadpisywania, *system-delta*

Pozbywanie się zewnętrznych konfigów.

Zamiast podstawiać VAR=32 i potem $VAR, lepiej od razu wpisywać 32 w unicie

Wartości domyślne + zewn. konfig

. /etc/sysconfig/nfs
/usr/sbin/rpc.mountd
~~${MOUNTD_PORT:+-p $MOUNTD_PORT}~~

[Service]
Environment=MOUNTD_PORT=9001
EnvironmentFile=-/etc/sysconfig/nfs
ExecStart=/usr/sbin/rpc.mountd -p $MOUNTD_PORT

Unity to nie skrypty

Ale czasem się nie jednak nie da

jeśli zdefiniowany MOUNTD_PORT, to dodaj „-p $MOUNTD_PORT”

Takiej konstrukcji nie możemy zastosować, bo to shell
.
Over 9000!!!!
1) bezwarunkowo podawany port
2) znak minus przy sysconfig – nie wariuje przy braku

Po swojemu

Konfiguracja w /etc/

Kopiowanie, .include, **dropiny**

systemd-delta *systemctl cat (v209)*

Unity nie gryzą

dzielić na mniejsze

- ładowanie modułów? /etc/modules-load.d/
- tworzenie katalogów, plików? tmpfiles.d
- różne czynności? ExecStartPre= lub inne unity
- wspólne restarty? PartOf=

Condition*= Twoimi przyjaciółmi

Wielkie skrypty SYSV często robią kilka rzeczy naraz.
Do niektórych są lepsze mechanizmy

tmpfiles: 1) tak przydatny, że dawno sportowany do innych dystrybucji i initów; 2) potrzebny przy /run na tmpfs; 3) przesłanianie katalogów podobnie jak przy unitach

ExecStartPre= kilka poleceń wykonanych wg kolejności w pliku; może to być np.. sprawdzenie poprawności konfiga lub coś.
nie powinno to być ładowanie modułów czy mkdir

Jak podzielimy na mniejsze, może nam się zrobić mnóstwo unitów; przyjrzyjmy się warunkom

sshd@.service

```
[Unit]
Description=OpenSSH per-connection server daemon
Wants=sshd-keygen.service
After=auditd.service sshd-keygen.service

# cat sshd-keygen.service
[Unit]
Description=OpenSSH Server Key Generation
ConditionPathExists=|!/etc/ssh/ssh_host_rsa_key
ConditionPathExists=|!/etc/ssh/ssh_host_dsa_key
ConditionPathExists=|!/etc/ssh/ssh_host_ecdsa_key
```

ssh keygen jest wołane za każdym razem, ALE to nie znaczy, że zawsze się wykona, gdyż…

uruchomienie keygen zależy od **braku** plików z kluczami

abrt-vmcore.service

```
abrt-vmcore.service - Harvest vmcores for ABRT
  Loaded: loaded (/usr/lib/systemd/system/abrt-vmcore.service; enabled)
  Active: inactive (dead)
          start condition failed at Thu 2014-02-06 16:42:11 CET; 5s ago
           ConditionDirectoryNotEmpty=/var/crash was not met

Feb 06 16:42:11 mother.pipebreaker.pl systemd[1]: Started
Harvest vmcores for ABRT.
```

Tutaj: uruchomi się, jeśli /var/crash nie jest puste

Unity nie gryzą

dzielić na mniejsze

- ładowanie modułów? /etc/modules-load.d/
- tworzenie katalogów, plików? tmpfiles.d
- różne czynności? ExecStartPre= lub inne unity
- wspólne restarty? PartOf=

Condition*= Twoimi przyjaciółmi

Co do condition, to możemy w ten sposób zastąpić konstrukcję if … then … else z skryptów.
Tworzymy dwa unity, robiące obydwie rzeczy.
Te dwa unity mają sprzeczne Condition.
Usługa faktyczna wymaga obu.
Wykonuje się tylko ten, którego Condition działa.

Type=notify

pełna informacja

rozszerzony status

- sd_notify("STATUS=Robię dwójkę")
- systemd-notify --status="Przetwarzam plik 42 z 69."

proste type (simple, forking) nie zawsze wystarczają

Debian – readiness protocol proposal

Pełna informacja: READY=1; nie trzeba zakładać, że po forku usługa już działa

rozszerzony status: proste wywołanie

W shellu też:
np.. długi cronjob (również widziany w systemctl status) może informować o postępie

rozszerzony status

httpd.service – The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
  Active: active (running) since czw 2014-02-13 11:10:46 CET; 4 days ago
  Process: 27130 ExecReload=/usr/sbin/httpd $OPTIONS -k graceful (code=exited, status=0/SUCCESS)
Status: "Total requests: 3432; Current requests/sec: 4; Current traffic: 2423 B/sec"
   CGroup: /system.slice/httpd.service
           ├─ 6368 /usr/sbin/httpd -DFOREGROUND
           ├─ 8408 /usr/sbin/httpd -DFOREGROUND

transmission-daemon.service – Transmission BitTorrent Daemon
  Loaded: loaded (/usr/lib/systemd/system/transmission-daemon.service; enabled)
  Drop-In: /etc/systemd/system/transmission-daemon.service.d
           └─10-user-zdzichu.conf, 20-io-sched-idle.conf, 30-exec-noauth.conf, 40-restart.conf
   Active: active (running) since Sat 2014-01-18 12:22:28 CET; 4 weeks 2 days ago
Status: "Uploading 0.00 KBps, Downloading 0.02 KBps."
   CGroup: /system.slice/transmission-daemon.service
           └─9096 /usr/bin/transmission-daemon -f --log-error -T

rozszerzony status - python

```
#!/usr/bin/python
#
# logs temperature from OWFS into postgres database

import glob
import os

import psycopg2
import systemd.daemon
import time

sleep_seconds = 60

systemd.daemon.notify("STATUS=Opening DB connection...")
dbconn = psycopg2.connect("dbname=temperature_log")
dbconn.autocommit=True
cur = dbconn.cursor()
cur.execute("PREPARE put_temperature AS INSERT INTO temperatures (datetime, sensor_id, value) VALUES (NOW(), (SELECT id FROM sensors WHERE SN=$1), $2);")

systemd.daemon.notify("READY=1")
systemd.daemon.notify("STATUS=Entering main loop")
while True:
    for SN in glob.glob("/run/owfs/??.????????????"):
        systemd.daemon.notify("STATUS=Reading sensors...")
        temperature = open("%s/temperature" % SN).readline()
        # we won't be needing full path anymore, trim it
        SN = os.path.basename(SN)

        try:
            cur.execute("EXECUTE put_temperature (%s, %s);", (SN, temperature) )
        except psycopg2.IntegrityError:
            print "New sensor %s! Adding to database, please correct description." % SN
            cur.execute("INSERT INTO sensors (SN) VALUES (%s)", (SN,))
            cur.execute("EXECUTE put_temperature (%s, %s);", (SN, temperature) )

    systemd.daemon.notify("STATUS=Sleeping until %s" % time.ctime(time.time() + sleep_seconds))
    time.sleep(sleep_seconds)

systemd.daemon.notify("STATUS=Cleaning up")
dbconn.close()
systemd.daemon.notify("READY=0")
```

rozszerzony status - python

```
import systemd.daemon

systemd.daemon.notify("STATUS=Opening DB connection...")

systemd.daemon.notify("READY=1")
systemd.daemon.notify("STATUS=Entering main loop")

            systemd.daemon.notify("STATUS=Reading sensors...")

    systemd.daemon.notify("STATUS=Sleeping until %s"
        % time.ctime(time.time() + sleep_seconds))

systemd.daemon.notify("STATUS=Cleaning up")

systemd.daemon.notify("READY=0")
```

Linijka z sleeping until bardzo pomocna

rozszerzony status - python

% systemctl --user status logtemp

```
logtemp.service - database logging of temperature from OWFS
   Loaded: loaded (/home/zdzichu/dev/logtemp/logtemp.service; enabled)
   Active: active (running) since Thu 2014-02-06 16:56:23 CET; 1 weeks 3 days ago
 Main PID: 11078 (logtemp.py)
   Status: "Sleeping until Mon Feb 17 13:35:51 2014"
   CGroup: /user.slice/user-1001.slice/user@1001.service/logtemp.service
           └─11078 /usr/bin/python /home/zdzichu/dev/logtemp/logtemp.py

Feb 06 16:56:22 mother.pipebreaker.pl systemd[1251]: Stopping database logging of
temperature from OWFS...
Feb 06 16:56:22 mother.pipebreaker.pl systemd[1251]: Starting database logging of
temperature from OWFS...
Feb 06 16:56:23 mother.pipebreaker.pl systemd[1251]: Started database logging of
temperature from OWFS.
```

Jest to wartościowe, gdyż długie zadania z crona

Type=notify

dokładnie określa gotowość usługi

rozszerza informację o statusie

wymaga minimalnego łatanie

Łatki na tyle minimalne, że często upstream przyjmuje.
W przypadku braku systemd funkcje kończa się bez błędu.

systemctl enable debug-shell

debug-shell.service - Early root shell on /dev/tty9 FOR DEBUGGING ONLY
Loaded: loaded (/usr/lib/systemd/system/debug-shell.service; disabled)
Active: inactive (dead)
Docs: man:sushell(8)

Pomocne również (*kernel command line*):

- systemd.confirm_spawn=1
- systemd.unit=
- rescue, emergency

Małe przydatne rzeczy

journalctl

logi od samego startu (również *initramfs*)

journalctl /dev/sda

journalctl -u postfix

journalctl --since=yesterday

O journalu można kolejną prezentację zrobić

zresztą jest, link na końcu

urlz for lulz

- systemctl help <service>
- **http://www.freedesktop.org/software/systemd/man/**systemd.directives.html
- T-DOSE 2012, Open Source Lennart Poettering, The Systemd Journal
  http://www.youtube.com/watch?v=o1lUeQVYuNs
- CoreOS fleet – distributed init system
  http://www.youtube.com/watch?v=u91DnN-yaJ8
- http://pkgs.fedoraproject.org/cgit/

Przed napisanie unita najlepiej sprawdzić, czy już ktoś tego nie zrobił – np.. w Fedorze

Po napisaniu: wysłać do programistów;
man 7 daemon
^^ pełen opis instalacji unitów

systemd – ściągawka

Dziękuje!