Paralelní start systému pomocí systemd

Honza Horák
hhorak@redhat.com

5. 11. 2011

LinuxAlt 2011

Brno

O čem to bude

- Boot - aneb co se děje při startu
- Init proces a jeho historie
- Spouštění služeb na požádání
- Démonizace včera a dnes
- systemd a vše okolo
- Ukázka implementace démona novým způsobem

Co se děje při startu systému

- bootloader z konfiguračního souboru zjistí, kde je uložen kernel
- načtení obrazu kernelu
- spuštění kernelu
- kernel zinicializuje potřebné datové struktury
- spuštění procesu init

Init proces

- manažer služeb a systémový manažer
- první proces spouštěný jádrem s PID 1
- init spouští další procesy, zejména démony
- jeho pád znamená velký problém

Historie implementací init procesu

- první init představen v System III (UNIX od AT&T v roce 1982)
- System V (SysV) ho brzo nahradil a zůstal dlouho jedničkou
- obsahuje skripty v shellu, uložené v adresáři `/etc/init.d/`
- init skripty pracují s démony a reagují na argumenty start, stop, restart, reload, ...

Ukázka velmi jednoduchého init skriptu pro rwhod 1/2

```
. /etc/init.d/functions
RWHOD=/usr/sbin/rwhod
RETVAL=0

start() {
    if [ $UID -ne 0 ] ; then
        #user had insufficient privilege
        exit 4
    fi
    # Check that networking is up.
    if [ ${NETWORKING} = "no" ] ; then
        exit 6
    fi
    echo -n $"Starting rwho services: "
    daemon rwhod
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && touch
/var/lock/subsys/rwhod && touch /var/run/rwhod.pid
    return $RETVAL
}
```

```
stop() {
    if [ $UID -ne 0 ] ; then
        #user had insufficient privilege
        exit 4
    fi
    echo -n $"Stopping rwho services: "
    killproc rwhod
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && rm -f
/var/lock/subsys/rwhod && rm -f
/var/run/rwhod
    return $RETVAL
}

restart() {
    stop
    start
}
```

Ukázka velmi jednoduchého init skriptu pro rwhod 2/2

```
# See how we were called.
case "$1" in
 start)
    start
    ;;
 stop)
    stop
    ;;
 status)
    status $RWHOD
    RETVAL=$?
    ;;
 restart)
    restart
    ;;
 reload)
    RETVAL=3
    ;;
 force-reload)
    restart
    ;;
 condrestart)
     [ -f /var/lock/subsys/rwhod ]
&& restart || :
    ;;
 *)
    echo $"Usage: $0 {start|stop|
status|restart|reload|force-reload|
condrestart}"
    RETVAL=2
    ;;
esac


exit $RETVAL
```

Proč je potřeba nová implementace init systému

- pořadí spouštění služeb je určeno jménem symbolického odkazu - neměnná sekvence
- sekvenční spouštění služeb je příliš pomalé
- zpoždění jedné služby zpozdí celou bootovací sekvenci
- SysV init skripty jsou velmi podobné = duplicita kódů
- interpretace bash skriptu = velký počet procesů

Upstart - nejrozšířenější následník SysV init v současnosti

- Upstart v roce 2006 pomocí Linux Standard Base (LSB) standardizuje, co SysV init implementoval
- LSB init skripty jsou jen mírně upravené SysV init skripty
- možnost používat SysV skripty dovoluje jeho nasazení
- asynchronní spouštění služeb pomocí událostí
- umožňuje částečně paralelní spouštění služeb
- zůstává problém s init skripty
  - velký počet procesů
  - duplicita kódu

Ukázka LSB bloku z init skriptu pro Upstart

```
### BEGIN INIT INFO
# Provides:          scriptname
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start daemon at boot time
# Description:       Enable service provided by
daemon.
### END INIT INFO
```

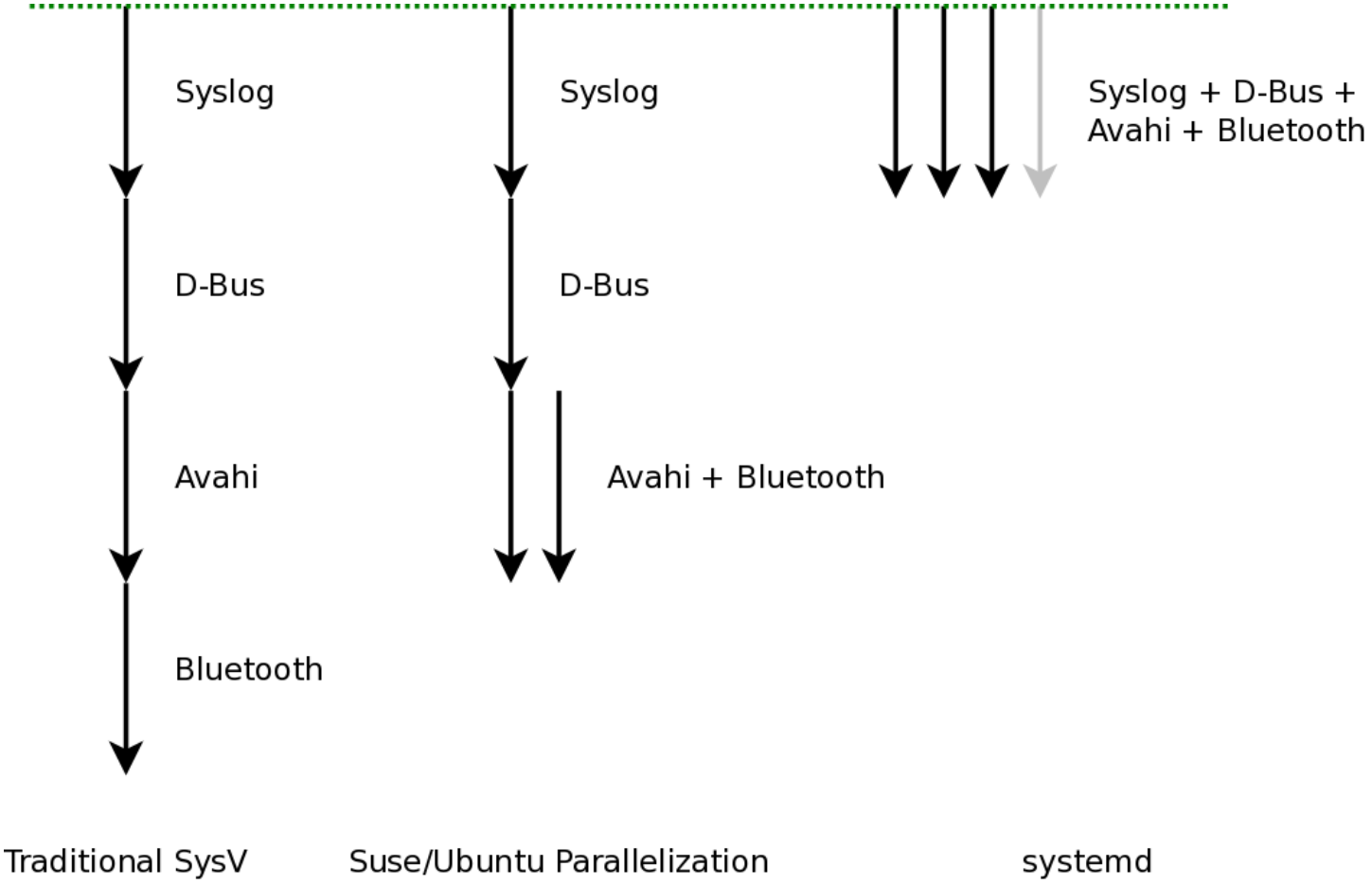
Nový hráč na scéně: systemd

- systemd představeno v roce 2010
- autor Lennart Poettering – PulseAudio a Avahi
- dovoluje používat LSB init skripty, ale pouze jako fallback, doporučují se nativní unit soubory
- maximální paralelizace v bootovací "sekvenci"
- prosazuje spouštění služeb "on demand"

Spouštění služeb na požádání ("on demand")

- většinu služeb není nutné spouštět hned při startu
- co znamená, že jedna služba vyžaduje druhou?
- například Avahi nebo Bluetooth potřebují D-Bus a D-Bus potřebuje syslog
- v praxi nepotřebuje celou službu, ale pouze komunikační prostředek - socket, D-Bus object path apod.
- systemd vytváří tyto prostředky dopředu a službu spustí, až se tento prostředek použije
- chování podobné démonu xinetd

Porovnání paralelizace init systémů



Zdroj: http://0pointer.de/blog/projects/socket-activation.html

Implementace démonů pro SysV init

- démonizace - vyvázání se od svého rodiče (resp. z terminálu), tzv. Double-fork
  => min. 10 řádků
- inicializace socketu a navázání na port
  => min. 25 řádků
- vytvoření init scriptu (běžně zkopírováním z jiného) => prům. 150 řádků

Problém démonizace - double-fork a duplikace kódu

- musí být provedena složitá procedura, její jednodušší varianta vypůjčená z HTTP démona lighttpd:

```
signal(SIGTTOU, SIG_IGN);
signal(SIGTTIN, SIG_IGN);
signal(SIGTSTP, SIG_IGN);
if (0 != fork()) exit(0);
if (-1 == setsid()) exit(0);
signal(SIGHUP, SIG_IGN);
if (0 != fork()) exit(0);
if (0 != chdir("/")) exit(0);
```

Démonizace novým způsobem

- žádná démonizace
- jeden hlavní proces na popředí, systemd zařídí zbytek
- vhodné pro ladění
- systemd restartuje pokud je potřeba, systemd je dobrá chůva
- programátor se může soustředit na funkci, ne na omáčku okolo

Inicializace socketu starým způsobem

- můžeme nechat čistě na systemd nebo:

```
union {
  struct sockaddr sa;
  struct sockaddr_un un;
} sa;

fd = socket(AF_UNIX,
SOCK_STREAM, 0);
if (fd < 0) {
  fprintf(stderr, "socket():
%m\n");
  exit(1);
}
```

```
memset(&sa, 0, sizeof(sa));
sa.un.sun_family = AF_UNIX;
strncpy(sa.un.sun_path,
"/run/foobar.sk",
sizeof(sa.un.sun_path));

if (bind(fd, &sa.sa, sizeof(sa)) <
0) {
  fprintf(stderr, "bind(): %m\n");
  exit(1);
}

if (listen(fd, SOMAXCONN) < 0) {
  fprintf(stderr, "listen():
%m\n");
  exit(1);
}
```

Inicializace a komunikace pomocí socketu v systemd

- systemd nám vytvoří socket podle detailní specifikace
- pouze v případě neúspěchu ho vytvoříme sami (nutnost pokud implementujeme démona podporujícího i jiné init procesy)
- šikovnou alternativou je přesměrovat stdin/stdout na socket, který nám vytvoří systemd

Unit soubory místo bash skriptů

- systemd podporuje SysV skripty, ale má raději nativní konfigurační soubory pro služby
- služeb je několik druhů:
  - **service**, **target socket**, path, device, mount, automount, swap, timer, snapshot

Praktická ukázka jednoduchého démona

- zadání: chceme vytvořit démona, který na dotaz ve tvaru YYYY-MM-DD vrátí řetězec s předpovědí počasí na daný den
- proces bude spouštěn na pozadí a naslouchat na portu 8289
- pro každé příchozí spojení spustí nový proces
- dotaz a odpověď ve formě řetězců je obecný problém, snadno lze rozšířit na HTTP server, FTP server, apod.

Ukázka jednoduchého unit souboru pro službu

- umístění: /lib/systemd/system/forecastd.service

```
[Unit]
Description=Forecast daemon
After=syslog.target network.target
Requires=forecastd.socket

[Service]
ExecStart=/home/hhorak/Dropbox/forecastd
StandardInput=socket
StandardOutput=socket
StandardError=syslog

[Install]
WantedBy=multi-user.target
Also=forecastd.socket
```

Ukázka jednoduchého unit souboru pro socket

- umístění: /lib/systemd/system/forecastd.socket

```
[Unit]
Description=Forecast Daemon Socket

[Socket]
ListenStream=8289
Accept=yes

[Install]
WantedBy=sockets.target
```

Ukázka efektivní části aplikace

- bez úvodních pozdravů a bez funkce process_request()

```
/* get requests and leave when request = quit */
while (fgets(req, sizeof(req) - 1, stdin) != NULL)
{
  if (strncmp(req, "q", 1) == 0 || strncmp(req, "quit", 4) == 0)
    return;

  process_request(req, resp);

  printf(resp);
  fflush(stdout);
}
```

Jak vyzkoušet uvedený příklad?

- proces samotný komunikuje pomocí stdin/stdout - možno spustit a ladit bez systemd
- po změně souborů spustíme
  `systemctl daemon-restart`
- socket aktivujeme pomocí
  `systemctl start forecastd.socket`
- vyzkoušet funkci můžeme pomocí
  `telnet localhost 8289`

Jak si hrát se systemd

- defaultní ve Fedoře 15+, volitelný v Ubuntu
- možné stáhnout live image a spustit přes USB disk nebo ve VM
- originální nativní unit soubory uloženy v `/lib/systemd/system`
- soubory v `/etc/systemd/system` mají přednost
- po každé změně unit souboru je potřeba znovu načíst konfiguraci démona:
  `systemctl daemon-reload`

Další "vychytávky" v systemd

- cgroups - organizace procesů do skupin pro lepší izolaci použití zdrojů (CPU, paměť, IO)
- správa runlevelů pomocí symbolických odkazů
- práce s vyrovnávací pamětí - swap
- může nahradit cron, xinetd, automount a další
- systemd dokáže víc než init systém - je to správně?

systemd v praxi (stav okolo Q1 2011)

- nativní soubory neexistují, protože systemd se zatím moc nepoužívá
- systemd nelze používat, protože díky masivní paralelizaci bez nativních souborů nefunguje stoprocentně (některé init skripty stále nejsou LSB kompatibilní)
- problém vejce X slepice

systemd v praxi – pohled uživatele

- příkaz `systemctl` nahrazuje volání `service` a `chkconfig`
- díky fallback módu možno dále používat starší varianty
- snaha nepoužívat pro konfiguraci démonů `/etc/sysconfig` nebo `/etc/default` ale přímo upravené unit soubory – snaha standardizace napříč distribucemi

Konverze init skriptů na nativní systemd unit files

- ...aneb pohled vývojáře ;)
- některé init skripty přidávají vlastní funkcionalitu, která se musí separovat mimo unit soubory
- snaha o:
  - minimalizaci změn chování z pohledu uživatele
  - co nejmenší odlišení od upstreamu
- zatím se nevyužívají veškeré možnosti systemd
- např. mysqld stále používá `mysqld_safe` jako wrapper pro `mysqld` přesto, že může být nahrazen

systemd a současnost

- Fedora 15 - ledy se pohnuly
- nativní unit soubory jsou implementovány
- většina služeb konvertována do Fedory 16
- unit soubory pomalu akceptovány do upstreamu
- porodní bolesti odcházejí, velké časy systemd teprve přijdou

Zdroje & kde hledat další informace

- http://www.gentoo.org/doc/cs/handbook/handbook-x86.xml?part=2&chap=4
- http://en.wikipedia.org/wiki/Init
- http://lateral.netmanagers.com.ar/stories/23.html
- http://en.wikipedia.org/wiki/Unix_System_III
- http://en.wikipedia.org/wiki/UNIX_System_V
- http://blip.tv/linuxconfau/beyond-init-systemd-4715015
- http://www.youtube.com/watch?v=TyMLi8QF6sw
- https://fedoraproject.org/wiki/Systemd
- http://wiki.debian.org/systemd
- http://fedoraproject.org/wiki/How_to_debug_Systemd_problems
- http://en.wikipedia.org/wiki/Upstart
- http://en.wikipedia.org/wiki/Systemd
- http://0pointer.de/blog/projects/systemd-docs.html
- http://0pointer.de/public/systemd-man/daemon.html
- http://0pointer.de/blog/projects/socket-activation.html
- http://0pointer.de/blog/projects/hinter-den-kulissen.html

Děkuji za pozornost