systembsd: custom dbus daemons emulating systemd behavior on openbsd

ian sutton, 2015

public domain/CC0
who(1)

-senior computer engineering student at syracuse university

-openbsd user for 1 year
  -"plan 9"

-c programmer

-most interested in kernel/low-level drivers development

-aspiring openbsd developer

ian@kremlin.cc
freenode: kremlin-

kremlin.cc
uglyman.kremlin.cc (repos)
bsd.port.mk
ce.gl
-works for spiders, an online platforms development group at syracuse university (3 years)

-started out as a html/css/php/javascript/kitchen sink dev
  -ugh

-now help manage some of the systems hosting our sites
  -many hundreds of wordpress installations

-also doing contract work for a diamond dealer in houston
  -migrating their sales/records system from dbase on dos running on a 486 machine older than i
  -new system uses openbsd, owner very happy with it
history(3)

- starting in 2013 i became increasingly unsatisfied with linux
  - kernel development pace
  - incongruent kern/user space due to -2147483647 distros
    - are you using networkmangler? netct1? wicd? networkd?
  - community
  - license sucks
  - kernel source tree is a gordian knot

- 2013, arch linux replaces system V init with systemd
  - the straw that broke the camel's back
  - "/bin exists in filesystem" (!)
-2014, apply for google summer of code with openbsd, began transitioning to using openbsd
  -2015 gsoc

- found it to be an overall more coherent & contiguous operating system
  - "minimal"

- awesome code quality & documentation

- professional, serious community
  - "what he is trying to do is create a frankenstein." - theo de raadt on my computer

- codebase represents work accomplished by a small number of talented, focused engineers
  rather than by thousands of nebulous individuals

- straightforward & permissive license (11 lines!)
disclaimer

-the opinions in this talk are wholly my own and do not represent the views of my employers, my colleagues, the president, xenu, etc.

-i am an undergraduate college student and shouldn't be taken as an authoritative source

-the systemd project & its developers admittedly catch a lot of undue, inappropriate flak which has created a very angry & political dialogue surrounding systemd, which i am trying to avoid in this talk
systembsd

- my gsoc project last year was to write several dbus daemons emulating the behavior of systemd counterparts
  - from scratch in c

- hostnamed, localed, timedated, and logind

- mentored by ajacoutot@ and landry@, openbsd developers

- project was successful yet ongoing
  - hard to keep up with systemd development pace
  - logind split

- will hopefully end up as a !PORT! installed alongside large DEs like gnome
systembsd: why

- the overwhelmingly common use case for these daemons is for running gnome

- gnome has a history with compatibility/portability
  - gnome 3 requires 3d acceleration
    - but what if I am running it in a VM?
  - old friend nvidia

- gnome 3 now depends on systemd (or at least these 4 daemons)
  - problematic as *bsd will never run systemd

- so we emulate the daemons with code written from scratch as porting a subset of systemd is infeasible

but before we get into that...
was..ist..das
systemd overview
was ist das
systembsd overview
"emulating the daemons' behavior"

- systemd code too co-dependent on other systemd libraries/components
  - can't port/recreate any low-level functionality
    - horrible idea anyway

- we use dbus instead
"emulating the daemons' behavior":

dbus

- dbus is a userspace ipc program
  - general programs can register their service names on the "bus"
  - services with names on the bus can call functions with typed params on each other
    - according to a security policy (polkit)
  - systemd team did a lot of work on it + is tightly integrated into systemd
- dbus is terrible
- dbus is ported to bsd
"emulating the daemons' behavior":

dbus

-we can write programs that register names on the dbus
-those names might happen to match the ones systemd
registers
-APIs, too
what this looks like

```c
node /org/freedesktop/hostname1 {
  interface org.freedesktop.hostname1 {
    methods:
      SetHostname(in s name,
                  in b user_interaction);
      SetStaticHostname(in s name,
                        in b user_interaction);
      SetPrettyHostname(in s name,
                        in b user_interaction);
      SetIconName(in s name,
                  in b user_interaction);
      SetChassis(in s name,
                  in b user_interaction);
    signals:
    properties:
      readonly s Hostname = 'dhcp-192-168-47-11';
      readonly s StaticHostname = 'lennarts-computer';
      readonly s PrettyHostname = 'Lennart\'s Computer';
      readonly s IconName = 'computer-laptop';
      readonly s Chassis = 'laptop';
  }
  interface org.freedesktop.DBus.Properties {
  }
  interface org.freedesktop.DBus.Introspectable {
  }
  interface org.freedesktop.DBus.Peer {
  }
};
```
hostnamed

- controls setting various hostnames
  - regular hostname
  - static hostname
  - "pretty" hostname
  - icon & chassis
  - easy
hostnamed

-hostnames on bsd work differently than linux
- domain

-/etc/hostname.if
- interface-specific name

- notice trend of named daemon doing things outside of its scope
- this is for sysadmins/corporate people
- example: machine type, icon, vm status, etc.
locale

node /org/freedesktop/locale1 {
    interface org.freedesktop.locale1 {
        methods:
            SetLocale(in  as locale,
                in  b user_interaction);
            SetVConsoleKeyboard(in  s keymap,
                in  s keymap_toggle,
                in  b convert,
                in  b user_interaction);
            SetX11Keyboard(in  s layout,
                in  s model,
                in  s variant,
                in  s options,
                in  b convert,
                in  b user_interaction);
        signals:
        properties:
            readonly as Locale = ['LANG=en_US.UTF-8'];
            readonly s VConsoleKeymap = 'de';
            readonly s VConsoleKeymapToggle = "";
            readonly s X11Layout = 'de';
            readonly s X11Model = "";
            readonly s X11Variant = "";
            readonly s X11Options = "";
    }
    interface org.freedesktop.DBus.Properties {
    }
    interface org.freedesktop.DBus.Introspectable {
    }
    interface org.freedesktop.DBus.Peer {
    }
}
localed

- controls setting locale
  -$ locale
    LANG=
    LC_COLLATE="C"
    LC_CTYPE="C"
    LC_MONETARY="C"
    LC_NUMERIC="C"
    LC_TIME="C"
    LC_MESSAGES="C"
    LC_ALL=

- these should be utf-8

- locales work differently on openbsd than linux or other bsds
  - /usr/share/locale/
  - posix locales

- also handles setting keymap & "X11 keyboard"
  - keymaps for ttys & X

- also pretty easy
timedate

node /org/freedesktop/timedate1 {
  interface org.freedesktop.timedate1 {
    methods:
      SetTime(in  x usec_utc,
              in  b relative,
              in  b user_interaction);
      SetTimezone(in  s timezone,
                in  b user_interaction);
      SetLocalRTC(in  b local_rtc,
                   in  b fix_system,
                   in  b user_interaction);
      SetNTP(in  b use_ntp,
              in  b user_interaction);
    signals:
    properties:
      readonly s Timezone = 'Europe/Berlin';
      readonly b LocalRTC = false;
      readonly b NTP = true;
  }
  interface org.freedesktop.DBus.Properties {
  }
  interface org.freedesktop.DBus.Introspectable {
  }
  interface org.freedesktop.DBus.Peer {
  }
};
timedated

- handles setting time & date (big surprise)
  - also NTP

- UTC vs. RTC
  - why?
  - unix

- actual systemd linux implementation has a lot more stuff than site lists
  - this was common
  - major development setback
    - have to improvise
logind (the big one)

-api way too huge to list here

-important objects:
  -user (not that kind of user)
  -session
  -seat

-functions:
  -flush devices
  -allocate devices
  -handle creating/destroying users/seats/sessions
  -reboot (!!)
  -shutdown (!!)
  -sleep (!!)
   -acpi hell
logind

-logind is huge and it sucks
- recently split off into its own package
- fstab story
  - Error getting authority: Error initializing authority: Could not connect: No such file or directory (g-io-error-quark, 1)

- `loginctl` to manage via command line
  - almost carbon copy of dbus interface
    - why?

- hard to emulate on openbsd
  - pam vs. bsd_auth
pam

-pluggable authentication modules

-pretty much unstandardized
  -sun 1995
  -linux/freebsd rolls their own
    -api doesn't match up
  -libpam on linux
    -unportable, especially for openbsd (untrusted)

-openbsd's pam port died a while back
  -integrating logind with bsd_auth(3) is hard
    -not sure if possible currently

-end benefit
  -thin clients
systemd // bsd

- systemd major components:
  - lrwxrwxrwx 1 root root 22 Apr 21 21:02 /sbin/init ->
    ../lib/systemd/systemd
      - classic init
      - huge misconceptions

- does pretty much the same job as system V init (unix)
  - inherit zombie pids
  - never ever die, lest panic
systemd // bsd

- systemctl
  - userspace tool for dealing with systemd startup tasks

- similar to /etc/rc.d/foo
  - (as of 5.7) `rcctl` (ajacoutot)

- instead of runlevels, you have service targets (.service files)

- types:
  - simple
  - forking
  - oneshot
  - dbus
  - notify
  - idle (reasons)
systemd // bsd

-one *bsd, we do things the classic way
    -/etc/rc.conf
    -opinion: this is sound methodology

-there are serious problems with systemd startup services
    -ewontfix.com/15
systemd // bsd

- journalctl
  - overview
  - we do NOT WANT THIS
  - binary logs
  - feels more fragmented than syslog

- bsd uses syslog, authlog, /var/log/, etc.

- you're SOL if logs get corrupted
  - "this shouldn't happen" - systemd team

- point of journaling

- fair criticism of systemd
systemd // bsd

(linux)
$ du -csh /var/log/journal
  424M  total

(bsd)
$ sudo du -csh /var
  15.2M  total

-things like this make me want to use bsd over linux
systemd // bsd

- networkd
  - admittedly not that bad
  - naming screwups
    - story
      - eno1, enp0s5, enp0s6

  - uses plaintext config files to define network interfaces
    - very precarious
      - discrepancy between what is defined in configs vs. reality

- on bsd, network interfaces are products of drivers
  - i/f naming schema follow driver naming schema
    - example: run (ralink wifi device driver) puts up interface "run0"
      - makes a lot of sense, to me
systemd // bsd

- cgroups (control groups)
  - kernel-level
  - control resource usage
    - perhaps a logical approach

- compare with bsd login.conf
  - user classes
  - easy to modify, plaintext confs
  - reboot may not be necessary

- used with docker
  - compare with containers, or FreeBSD jails
  - where things are headed, it seems
concluding thoughts

- systemd presents concerns for *bsd
  - easy for developers who only see linux to rely on it
  - we get the short end of the stick
  - cannot port

- the bright side:
  - opportunity for *bsd to outperform on systemd shortcomings
  - stability is key
    - problem as systemd is rapidly being integrated, despite age
  - bugs

- systemd targets audience that would appreciate bsd
  - sysadmins overseeing many machines
  - large scale production servers
concluding thoughts

-i have seen (and am part of) crowd that has switched to bsd because of systemd

-code quality and maintainbility
  -now
  -future

-i am hopeful
questions

- any questions!
thank you