What is Init?

- Initialization of Userspace
- PID 1: the first process the kernel starts
  - Spawns all other processes
  - Grandparent of all userspace processes
- "Sets up" the system for use
  - Environment, Hostname
  - Filesystem checks, mounts
  - Networking, Services, etc.

BSD Init

- The oldest one, used by traditional UNIX
- Monolithic /etc/rc script
  - rc.local for local changes
- No concept of runlevels
- Later variants support /etc/rc.d
  - packages can install their custom scripts here
  - rcorder used to determine script order
  - dependencies are stored in rc.d scripts
- Arch, Slackware, FreeBSD, etc.

System V Init

- Originated with System V UNIX release
- Configured via /etc/inittab and scripts
- Runlevel-based
- Scripts in /etc/rcN.d, where N is the runlevel
  - Really symlinks to /etc/init.d or similar
  - 'K' for Kill scripts, 'S' for Start scripts
  - Order determined by number after 'K' or 'S'
- Newer variants have dependency tracking
  - "chkconfig" or LSB headers

Upstart

- Originally written for Ubuntu
- Event-based
  - Tasks and Services started/stopped by events
  - Events generated on Task/Service start/stop
  - Respawning supported
- Parallel starting of services
- Backwards compatible with SysVinit
- Native configuration files in /etc/init with deps

systemd

- Brought to you by Lennart Poettering of PulseAudio fame
- Socket- and bus-activated services
  - Automatic parallelization!
- Fine-grained per-process control
  - Environment, rlimits, nice level, etc.
- cgroup-based process tracking
- SysVinit (and LSB) compatible
- Comparable to launchd and Solaris' SMF

The magic of parallelization: Socket and Bus activation

- Remember the inetd superserver?
- systemd creates **all** listening sockets at once
  - Then starts the daemons in parallel
  - Kernel blocks sockets for us
  - Clients (other services) wait until other end becomes available
- Autospawning of services
- No need to manually code dependencies!
- Works with D-Bus too!

Parallelizing Filesystem Setup

- fsck & mounting filesystems takes a long time
- AutoFS provides mount point
- Access to the directory is queued by kernel
- When fsck & mount complete, AutoFS replaced by real mount

Control Groups

- Used instead of process tree (PIDs) to manage services
- Avoids "double-fork" from escaping process control
- Can manage lots of things:
  - rlimits, uids, gids, environment
  - CPU & I/O schedulers, affinity
  - Capabilities (security)
  - Read-only bind mounts for service lockdown

Units

- service: start/stop/restart/reload
  - native & SysV/LSB scripts
- socket: internet & local UNIX sockets
- device: device node in /dev
- mount: mount point
  - /etc/fstab is also supported
- automount: autofs-managed mount

Units (continued)

- target: for grouping
  - kinda like runlevels
- snapshot: used to save/rollback state of all units
    - "emergency shell"
    - suspend/resume

Configuration and Tools

- /etc/systemd/*
- .desktop file (INI) style config
- Symlinks used for target selection
  - /etc/systemd/system/default.target -> /lib/systemd/system/runlevel5.target
- systemctl command
  - /sbin/service and /sbin/chkconfig supported to a certain extent
- systemadm GUI (systemd-gtk)