Advanced Unix System Administration

Lecture 5
February 14, 2007

Steven Luo
<sluo+decal@OCF.Berkeley.EDU>

Shared Libraries

- The dynamic linker
  - Binaries have a “symbol table” containing functions, etc. and their locations
  - Dynamic binaries have tables with blanks – it's the responsibility of the dynamic linker to resolve these
  - Linker loads listed dynamic libraries and tries to resolve the symbols
  - Allows shared code, but incurs a performance penalty on most architectures

Shared Libraries

- Binary compatibility
  - Programs expect the “ABI” offered by a shared library to stay the same (structs, function prototypes, etc.)
  - When this assumption breaks, things go horribly – or worse, subtly – wrong
  - Hence mechanisms for versioning shared libraries and symbols

Startup and Shutdown

- Bootloader
  - Highly architecture-dependent behavior
- Kernel
  - Need to get enough loaded to find root partition, mount it, and launch userspace
  - Traditional fully monolithic kernels load everything here
- init
  - Mounts filesystems, launches daemons, and brings up the system

Startup and Shutdown

- init(8)
  - PID 1, the “ultimate parent”
  - Spawns and respawns various children, according to configuration
  - Two traditional varieties: System V, BSD
- System V init binary
  - Used on Linux, Solaris, most commercial Unix
  - Configured via /etc/inittab
  - Uses “runlevels” to define the stages of boot and what should be running

Startup and Shutdown

- System V style runlevel handling
  - Usually performs actions when changing runlevels based on the contents of /etc/rcN.d, where N is the new runlevel
    - Scripts starting with S are run with argument “start”, scripts starting with K are run with argument “stop”
    - Two-digit number following S or K gives the ordering
  - Runlevel S is notionally invoked at the beginning of startup, 0 and 6 at halt or reboot

Startup and Shutdown

- BSD init binary
  - Used primarily on the BSDs
  - Launches a script /etc/rc when invoked, then spawns and respawns programs based on the contents of /etc/ttys
  - /etc/rc.shutdown is run on shutdown
- BSD style init handling
  - Only used by OpenBSD nowadays
  - /etc/rc and /etc/rc.shutdown do most/all of the work themselves

Startup and Shutdown

- Comparing BSD and SysV init handling
  - BSD-style init handling is simple and straightforward, but difficult to modify automatically
  - SysV init has more flexibility and modularity
  - With appropriate configuration of /etc/inittab or /etc/rc, SysV init binaries can be configured to behave BSD style (i.e. Slackware) and vice versa – or could behave entirely differently from either

Startup and Shutdown

- System shutdown
  - Run shutdown scripts first
  - Kill all processes: send SIGTERM to all processes, wait a few seconds, then send SIGKILL to make sure they're dead
  - Sync/unmount disks, then power down or restart
  - SysV: last two steps actually run from the shutdown scripts, invoked by init
  - BSD: halt(8)/reboot(8) take care of all steps

Startup and Shutdown

- Criticisms of classic init
  - Inefficient – processes not started in parallel, SysV init requires launching lots of shells
  - Manual establishment of order of tasks and daemon load order required
  - Provides no monitoring of services and restarting of those that died
  - Shutdown procedure is an ugly hack

Startup and Shutdown

- “Requires-depends” init handling
  - Used in FreeBSD and NetBSD
  - Standard BSD init binary
  - /etc/rc{,.shutdown} uses a program rcorder(8) to examine “Requires”, “Depends”, “Provides” lines in scripts in /etc/rc.d and provide an order to run them in
  - Allows dynamic ordering of tasks, parallelization (though currently not parallel)

Startup and Shutdown

- Replacing init
  - SMF (Solaris 10+): dependency-based XML config allowing parallel launch of processes and restarting of services; “milestones” separate stages of bootup
  - launchd (OS X 10.4+): dependency-based config allowing parallel launch; also replaces/extends cron and inetd
  - Upstart (Ubuntu 6.10+): event-based structure for controlling processes; also replaces/extends cron and inetd