

The System Initialization Process: (or *A Daemon is Born*)

By: Micah Altman, (Copyright 1994-8)

Last Revision: June 10, 1998

Overview

This is an overview of the steps the system goes through to initialize network services, starting at the power-up sequence.

The Chain of Events

The Prom Monitor

- Power-up: the system starts up and runs the PROM monitor.
- PROM monitor checks for a keyboard and/or ASCII terminal attached. By default (see the manual page on PROM), if neither is attached the boot-up stops and the system yells:

Cannot connect to keyboard -- check the cable.

- Otherwise, PROM checks the setting of the boot and path variables. And depending on the value of this variable either runs diagnostics and then the kernel, or loads `/unix` immediately (the default), or waits for the user to issue further instructions.

(NOTE: If the volume header is damaged or erased the system may be unable to boot.)

Kernel Startup

- The kernel creates the basic unix processes: `sched`, `init`, `bdflush`, `vhand`.

init Takes Over

- The `init` process checks the `init` state (passed in from PROM), and `/etc/inittab`. It uses the `init` state to determine what lines in `/etc/inittab` to run next.

- Init runs `/etc/bcheckrc`, which checks to see if the root filesystem is dirty (indicating the system was previously not shut down properly). If so, it runs `fsck` to clean up on an `efs` (non-journalled) filesystem, or uses the XFS journal to clean up.
- Init runs `/etc/brc` to clear the mounted file system table, and put an entry for the root filesystem in the mount table.
- If the init state is "2" or "3" (multi-user mode), init runs `/etc/rc2`.

Startup Scripts in the /etc/rc2.d Directory

- `rc2` runs all the scripts in the `/etc/rc2.d` directory by name order. Note that the items in this directory are links to scripts in other directories. These usually include scripts to announce the boot process is starting, mount all local filesystems, start quotas and accounting, autoconfigure the kernel and start all daemons (including the networking daemons). We will look at the autoconfiguration and networking scripts.

Autoconfiguration

- The kernel runs the `autoconfigure` script, which checks `/var/config/autoconfig.options` to determine whether automatic reconfiguration is allowed. If so, (the default), it checks the dates on kernel header files and device driver files in `/var/sysgen`, and looks on the bus for all devices currently installed. If the `/unix` is older than any of the kernel component files, or a device has been added or removed from the bus since the previous kernel build, the kernel will load dynamically loadable device drivers and/or rebuild itself with `lboot`. If a kernel rebuild is required you must reboot to install the new kernel.

The Scripts

- Most services get started by a script in `/etc/init.d`. The network script, `/etc/init.d/network`, is a typical example. This script does the following:

1. It checks if the *verbose* flag has been set with `chkconfig`, to determine whether to echo its actions to the console. Checks whether the network flag has been set with `chkconfig`, in order to determine whether to initialize the network. (Assume it has been set to "on")
2. It then checks the options and configuration files for its services, and configures the network interfaces using `ifconfig`.
3. Finally, it starts the network daemons.

THE SYSTEM IS UP!

Getting More Information from the Startup Process

If a service is failing on startup, or is not being initialized correctly, obtain as much information as possible about the startup process of that service.

- Configure `syslogd.conf` so that all messages from that service are logged.
- Ask the service to report more information. Most daemons support a *trace*, *verbose* or *debugflag* for reporting more information. Check the manual pages.
- Set `chkconfig verbose on` to trace the startup process in more detail.
- If a shell script is run to start the service, use the `-x` flag to trace the execution of that shell script. (For binary files use `par` to trace the program.)
- Check the manual pages, administration documents, and release notes for the existence of additional log files, and the interpretation of messages in those files.