The Linux startup process

Jerry Feldman <gaf@hp.com>
The Linux Expertise Center
Hewlett-Packard Company
Document produced via OpenOffice.org

Overview

- The Linux boot process
  - GRUB. This is the default for X86/Linux
  - LILO
  - Other boot loaders
- The Linux Operating modes
  - Single-user mode
  - Multi-user mode.
- Run Levels
  - What are run levels
  - What are the Linux standard run levels
  - How Linux manages run levels

The Linux Boot process

- The PC boot process is a 3-stage boot process that begins with the BIOS executing a short program that is stored in the Master Boot Record (MBR) of the first physical drive. Since this stage 1 boot loader needs to fit in the MBR, it is limited to 512 bytes and is normally written in assembly language. There are a number of boot loaders that can load Linux.
    - GRUB and LILO are the most commonly used ones on X86 hardware.
    - EFI is used on the Intel® Itanium® family.

The GRand Unified Bootloader

- The GRand Unified Bootloader (GRUB) is default boot loader on most distributions today. It has the capability to load a number of different operating systems.
    1. The stage 1 boot resides in the MBR and contains the sector number of the stage 2 boot that is usually located in the /boot/grub directory on Linux.
    2. The stage 2 boot loader presents a boot menu to the user based on /boot/grub/grub.conf or menu.lst. This contains a boot script. It is the stage2 loader actually loads the Linux kernel or other OS.

The Linux LOader

- LILO is the older boot loader for the X86 systems.Like GRUB, it has the capability to load Windows.
  - The /etc/lilo.conf file is the menu that drives LILO.
  - The /sbin/lilo command must be run whenever the lilo.conf file is updated or you change the linux kernel. This will update the information contained in the conf file and the location of the kernel. This is the one feature that effectively obsoleted LILO in favor of GRUB.

Other Boot Loaders

- There are a number of other boot loaders that are employed depending on the hardware platform.
  - EFI – Extensible Firmware Interface is the standard boot loader on the HP Integrity and other Intel® Itanium® platforms. It boots the Linux kernel directly.
  - MILO – This is similar to LILO on Alpha platforms
  - Loadlin is a DOS based loader.
  - PPCBoot and U-Boot are loaders for PowerPC®.
  - And others for SPARC and ARM.

What Happens Next?

- The boot loader loads the Linux kernel
  - As the kernel boots up it scans the hardware, loads the loadable modules it needs, and finally starts process number 1, */sbin/init*.
  - /sbin/init is called the “mother of all processes” in that it is process number 1 in all Unix and Linux systems, and all other non-kernel processes are spawned directly or indirectly by *init*. It is *init* that is responsible for starting the appropriate scripts and terminals.

The Linux Operating Modes

- There are 2 major operating modes in Linux
  - Single User Mode. This is a mode is a mode where only a single command line is active (usually the super user) on the system console.
  - Multi-User mode where multiple users may log in.

Single User Mode

- Single User Mode is used primarily for system maintenance especially if the file systems are in need of repair.
  - This mode is also normally assigned to run level 1.
  - No daemon processes are running.
  - Network is not configured.
  - Only the root file system is mounted by default.
  - No users can log in other than the super user.
  - This mode is used primarily to repair the system.

Multi-user mode

- Multi-user mode is the normal operating mode of a Linux system. When the system boots into multi-user mode the following happens.
  - The file systems are checked using the fsck(8) command. The fsck(8) command causes the file system specific program to be executed.
  - The file systems are mounted.
  - Init(8) parses the /etc/inittab file.
    - The default run level is determined.
    - The command scripts associated with this run level are executed.
  - Service daemons are started.
  - General user logins are permitted.

The Run Levels

- In the early days of Unix, there were only the single-user mode and multi-user mode. *Init(8)* would start a login process on each configured terminal or modem. This was configured by the file, */etc/ttys*.
- AT&T System V replaced the */etc/ttys* file with */etc/inittab*. This introduced run levels that are controlled by the */etc/inittab* file.
- There are run levels for single-user mode, multi-user mode without networking and multi-user mode with networking and of course graphics.

Linux Standard Run Levels

- Today, most Linux distributions use the following standard run levels 0-6.
  0. System halt.
  1. Single-user mode. S and s may also be used here.
  2. Multi-user mode with no remote networking.
  3. Multi-user mode with networking. This is the standard operating mode of a non-GUI based system. All networking daemons are running.
  4. Unused.
  5. Multi-user mode with a GUI. This mode has most of the same daemons as run level 3 plus it is running the X Window System with a display and window manager.
  6. Reboot.

How the run levels work

- */sbin/init* parses the */etc/inittab* file. This file contains
  - An entry for the default run level, which is the run level the system boots up as unless otherwise specified to the boot loader at boot time.
  - Entries to be executed on all or specific run levels. These are of the form: id:runlevels:action:process [arguments]
    - id – generally anything.
    - runlevels are either empty single digit from 0 to 6, or a list of digits.
    - action describes the action to take – described later
    - Process is the program or script to be executed.

The run level actions

- Some of the common actions are:
  - respawn. The process will be restarted whenever it terminates.
  - wait start. the process when the specified runlevel. is entered and wait for its termination.
  - bootwait. The process will be executed during system boot, init waits for its termination. The runlevels field is ignored.
  - initdefault. specifies the runlevel which should be entered after system boot

A generic inittab file

```
# The default runlevel is defined here
id:5:initdefault:
# First script to be executed, if not booting in emergency (-b) mode
si::bootwait:/etc/init.d/boot

# /etc/init.d/rc takes care of runlevel handling
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# what to do when CTRL-ALT-DEL is pressed
ca::ctrlaltdel:/sbin/shutdown -r -t 4 now
x:5:respawn:/etc/X11/prefdm -nodaemon
```

A generic inittab file continued

- l5:5:wait:/etc/init.d/rc 5

This entry tells init to execute the script /etc./init.d/rc passing 5 in as a parameter only on run level 5, and to wait until that script completes. As you can see in the inittab file, there is one line per run level.

- 1:2345:respawn:/sbin/mingetty --noclear tty1
  2:2345:respawn:/sbin/mingetty tty2

The above entries start a virtual terminal for all multi-user run levels. Tty1 is the system console and is not cleared. Note that when the process ends, it is restarted.

The 3-finger salute

- Common to all Windows and DOS users is the 3-finger salute, ctrl-alt-delete. This is enabled in Linux by the following line: ca::ctrlaltdel:/sbin/shutdown -r -t 4 now
- When the 3-finger salute is performed on the keyboard in non-graphical mode, the shutdown command is executed.
- Some GUIs, such as KDE present a menu to the user and others ignore it altogether.

The run level scripts.

- All the startup scripts are located in the directory /etc/init.d (may be a symbolic link).
- And, there are 7 or more directories in /etc/rc.d that correspond to the run levels and are in the form rc#.d such as rc0.d or rc1.d. SuSE also has an rcS.d directory. These directories contain symbolic links to the scripts in /etc/init.d. These scripts may take the following parameters:
  - start – Start the service or daemon
  - stop – stop the service or daemon
  - restart – restart the service or daemon
  - reload (optional) – reload the service's config.
  - status – display the status of the service or daemon.

Starting the run level scripts

- The run levels are initated as follows:
  1. The /etc/init.d/rc is executed with the desired run level. For example: /etc/init.d/rc 5.
  2. It then cycles through the rcx.d directory to first kill those daemons that need killing at the old run level, x
  3. Then it cycles through the rcn.d directory and starts those daemons that need starting. The run level directories are located in /etc/rc.d.
- It is intelligent enough not to kill a daemon that is already running and shall continue to run in the new run level.

The run level directory links

- The run level directories (rc0.d, rc1.d, ...) contain a series of symlinks that begin with either Knn or Snn (such as K01).
  - The Knn symlinks are symbolic links to scripts in the /etc/init.d directory to kill the appropriate service, for example:

    K18network -> ../network

    This is a symlink in rc.5 to shutdown the network. It is executed after the lower numbered links.
  - The rc script executes the Knn symlinks in alphanumeric order. For example:

    K17syslog -> ../syslog

    The K17syslog script is executed before the K18network script.

The run level directory links continued

- The Snn symlinks are executed in the same manner as the Knn symlinks, but are used to start services, for example:

  S05network -> ../network
  S06syslog -> ../syslog
  In this example, the S05network symlink is executed **before** the the S06syslog symlink.

How to add/remove a service

- Services may be added or removed from a run level in one of several ways:
  - Add or remove the symbolic links from the appropriate run level directories. For example, to add a service, *foo* in run level 3:

```
cd /etc/rc.d/rc3.d #enter directory
ln -s /etc/init.d/foo S25foo #start
ln -s /etc/init.d/foo K56foo #stop
```

  - Use the distribution dependent system management utility:
    - In SUSE based distros use YaST.
    - In Red Hat based distros use Service Manager.

How to set the run levels

- The super user can transition between run levels by typing the command "*init n*" where *n* is the desired run level. This will cause all services unique to the old run level to be terminated and all services unique to the new run level to be started.
- The scripts in /etc/init.d may be executed by hand by using the full path names. For example:

```
# /etc/init.d/ntp start
Starting network time protocol daemon          done
```

- There are also distribution dependent ways to start and stop the scripts.

Starting or stopping a run level service, Red Hat based.

- In Red Hat and Fedora systems the following procedures can be used:
  - The command “service” may be used with the service and its arguments. The following is an example of starting the network time daemon service:

```
# service ntpd start
Starting ntpd:                             [  OK  ]
```

  - The GUI Service Configuration Manager is available to set up the various run level scripts.

Starting or stopping a run level service, Novell/SUSE based

- In Novell/SUSE, there are symlinks to each service prefixed by rc, for example, /usr/sbin/rcntp is symlinked to /etc/init.d/ntp as follows: /usr/sbin/rcntp -> /etc/init.d/ntp.
  - The following is an example of starting the Network Time Daemon:

```
# rcntp start
Starting network time protocol daemon      done
```

  - The SUSE configuration tool, YaST, may also be used either in command line or as a GUI to set up the run level services.