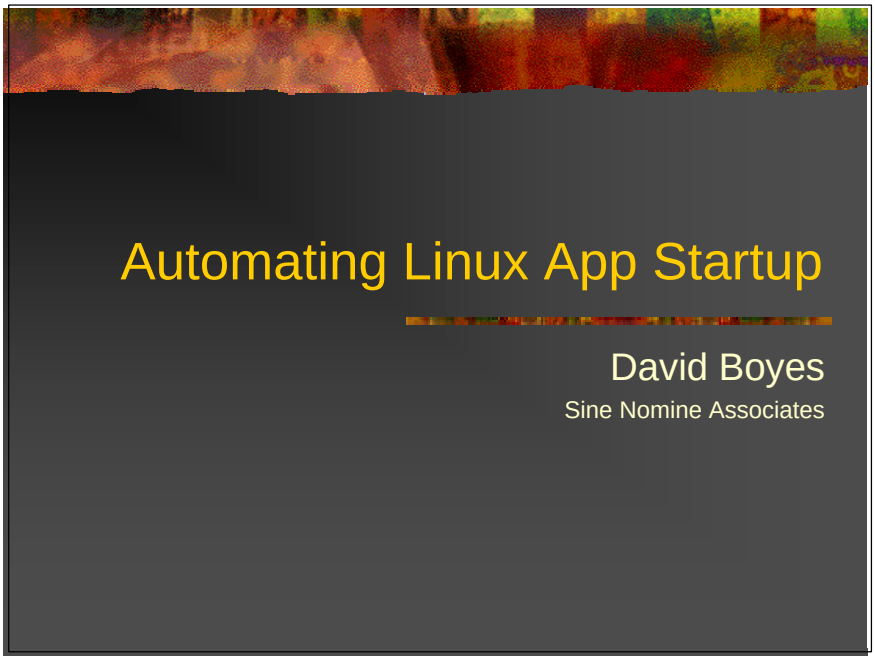

Agenda

- Runlevels, init, and symlinks, oh, my!
- Sequence of events during startup
- A sample application startup script
- Caveats on insserv, yast and friends
- Q&A (if we have time)

Why Do This?

- Like any other production system, applications need to start at boot time without human intervention.
- Presentation spawned by a discussion in the Hillgang (and later the Linux-390) mailing list in early October

Linux (and Unix) Startup

- Bootstrap loader for specific HW (stage1)
  - 3card loader
- Stage 2 loader (grub, zilo, lilo)
- Kernel (vmunix)
- Init
- Runlevel scripts

Runlevels

0 = halt
1/S = single user
2 = single user with network
3 = multiuser with network, no graphics
4 = reserved for future use
5 = multiuser with network, graphical
6 = shutdown/reboot
7-9 = reserved for future use

init

- Always process 1 (careful with job control!)
- Two variations: System V and BSD
  - Linux is System V based
  - Solaris and AIX are SysV
  - FreeBSD can be either (sysvinit package)

BSD init

- /etc/rc.boot
- /etc/rc.local
- /etc/rc.<n>
- Difficult to manage for complex environment

Sys V init

- Mostly compatible at a high level
  - /etc/rc.boot
  - /etc/rc.local
- New:
  - /etc/init.d/
  - /etc/init.d/rc<x>.d/

Sys V init

- Startup scripts go in /etc/init.d
- Symlinks built in /etc/rc<x>.d
  - Snn<scriptname> → /etc/init.d/<scriptname>
  - Knn<scriptname> → /etc/init.d/<scriptname>
  - Both symlinks point to the same physical file
  - Script called with different arguments to determine function

SysV init

- Scripts executed in numerical order on entry and exit from runlevel:
  - Eg, S01xxx before S02xxx
  - Scripts at same number are executed in alphabetic order (S01able before S01baker, etc)
  - Called with ‘start’
  - Kxx scripts called on exit from runlevel
  - Called with ‘stop’ parameter

SysV init

- SuSE and RH use numbering to force prereq management and startup sequencing
  - Prereq/sequence checking based on magic header in file
  - Automated tools will renumber things – caution!
    - Scripts without magic headers are assigned S01
    - THIS MAY MAKE YOUR SYSTEM UNBOOTABLE!

Sample Script Header

```
#! /bin/sh
### BEGIN INIT INFO
# Provides: sshd
# Required-Start: $network
# Required-Stop: $network
# Default-Start: 3 5
# Default-Stop: 0 1 2 6
# Description: Start the sshd daemon
### END INIT INFO

. /etc/rc.status
. /etc/sysconfig/ssh
```

Script Header, Cont.

- # Shell functions sourced from /etc/rc.status:
- # rc_check check and set local and overall rc status
- # rc_status check and set local and overall rc status
- # rc_status -v ditto but be verbose in local rc status
- # rc_status -v -r ditto and clear the local rc status
- # rc_failed set local and overall rc status to failed
- # rc_reset clear local rc status (overall remains)
- # rc_exit exit appropriate to overall rc status

- # First reset status of this service
- rc_reset

Script Header (RH)

```
#!/bin/bash
#
# xinetd       This starts and stops xinetd.
#
# chkconfig: 345 56 50
# description: xinetd is a powerful replacement for
inetd. \
# [… text omitted ..]
# processname: /usr/sbin/xinetd
# config: /etc/sysconfig/network
# config: /etc/xinetd.conf
# pidfile: /var/run/xinetd.pid

PATH=/sbin:/bin:/usr/bin:/usr/sbin

# Source function library.
. /etc/init.d/functions
```

Samples

- Start with the working versions in /etc/init.d.
- ‘man init.d’
- Don’t try to manipulate the links manually

Sample Script

```
case "$1" in
    start)
        if ! test -f /etc/ssh/ssh_host_key ; then
            echo Generating /etc/ssh/ssh_host_key.
            ssh-keygen -t rsa1 -b 1024 -f /etc/ssh/ssh_host_key -N ''
        fi
 […. Code omitted … ]

        echo -n "Starting SSH daemon"
        ## Start daemon with startproc(8). If this fails
        ## the echo return value is set appropriate.

        startproc -f /usr/sbin/sshd $SSHD_OPTS

        # Remember status and be verbose
        rc_status -v
        ;;
```

Sample Script

```
    stop)
        echo -n "Shutting down SSH daemon"
        ## Stop daemon with killproc(8) and if this fails
        ## set echo the echo return value.
        if [ -x /bin/netstat ]; then
          netstat -nlp 2>/dev/null | while read prot a b local remote state pro
g; do
            if [ "${local##*:}" = "22" ] ; then
              if [ -n "$prog" ]; then
                kill -TERM ${prog%%/*}
              fi
            fi
          done
        else
rc_failed 1
        fi # Remember status and be verbose
        rc_status -v
        ;;
```

Sample Script

```
    try-restart)
        ## Stop the service and if this succeeds (i.e. the
        ## service was running before), start it again.
        $0 status >/dev/null &&  $0 restart

        # Remember status and be quiet
        rc_status
        ;;
```

Sample Script

```
restart)
    ## Stop the service and regardless of whether it was
    ## running or not, start it again.
    $0 stop
    $0 start

    # Remember status and be quiet
    rc_status
    ;;
```

Sample Script

```
force-reload|reload)
    ## Signal the daemon to reload its config. Most daemons
    ## do this on signal 1 (SIGHUP).

    echo -n "Reload service sshd"

    [… code omitted .. .]

    rc_status -v

    ;;
```

Sample Script

```
    status)
      echo -n "Checking for service sshd: "
      ## Check status with checkproc(8), if process is running
      ## checkproc will return with exit status 0.

      # Status has a slightly different for the status command:
      # 0 - service running
      # 1 - service dead, but /var/run/  pid  file exists
      # 2 - service dead, but /var/lock/ lock file exists
      # 3 - service not running

      if [ -x /bin/netstat ]; then
        netstat -nlp 2>/dev/null | ( while read prot a b local remote state p
rog; do
[…code omitted … ]

fi
      rc_status -v
      ;;
```

Sample Script

```
    probe)
      ## Optional: Probe for the necessity of a reload,
      ## give out the argument which is required for a reload.

      test /etc/ssh/sshd_config -nt /var/run/sshd.pid && echo reload
      ;;
```

Sample Script

```
    *)
        echo "Usage: $0 {start|stop|status|try-restart|restart|force-reload|relo
ad|probe}"
        exit 1
        ;;
esac
rc_exit
```

Linux Tools for Manipulating init

- insserv
- chkconfig
- /etc/sysconfig
  - Not really a tool, but a place to store config info, Bourne shell syntax

chkconfig

chkconfig -s|--set [name state]
chkconfig -e|--edit [names]
chkconfig -c|--check name [state]
chkconfig -l|--list [--deps] [names]
chkconfig -a|--add [names]
chkconfig -d|--del [names]

Danger, Will Robinson!

- SLES 7 insserv is BROKEN
  - Make sure your scripts have correct headers!
- SLES 8 insserv is OK.
- Works OK in RH
  - Note that comment header is different in RH, but both RH and United Linux headers can be in same file.

Q&A

Contact Info

David Boyes
Sine Nomine Associates
info@sinenomine.net
http://www.sinenomine.net
+1 703 723 6673