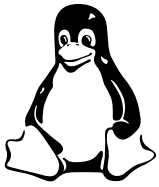

LPIC 102

The boot process, `init` and runlevels

1 Aim

After successfully working through this exercise, You will:

- be familiar with standard runlevels in a Linux system
- be able to rescue a system without the main dynamic link libraries using an alternative `init`
- be able to change through runlevels, and understand their purpose
- be familiar with the main parts of the boot sequence

2 Background

TODO: I wrote this document in Hong Kong, and it needs updating. You may notice discussion of an NFS server; please do something that will work where you are! I also discuss use of `lilo`, whereas GRUB is now used much more widely.

2.1 `init` and runlevels

- Linux has seven modes of operation
- Referred to as *runlevels*
- The Linux Standards Base (`http://refspecs.freestandards.org/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/runlevels.html`) defines the following standard runlevels that all distributions should follow to be compliant:

  0 halt
  1 single user mode
  2 multiuser with no network services exported
  3 normal/full multiuser
  4 reserved for local use, default is normal/full multiuser
  5 multiuser with a display manager or equivalent
  6 reboot

- The `init` process governs runlevels. `init` is *the first program that the kernel runs.*
- To change from one runlevel to another, you can use `telinit` or simply `init`:

```
$ sudo init level
```

For example, if you are currently in runlevel 5, you can change to runlevel 3 by executing the command:

```
$ sudo init 3
```

2.2 Runlevel 1, or Single-user Mode

You can change to runlevel 1 in this way too.

- Mainly used for diagnostic purposes
- Starts only a subset of the possible services, e.g.
  - No networking
  - No mail services
  - No name lookup services
    - Except `/etc/hosts`
  - No file-sharing services etc

You can also select a runevel when the kernel boots. When you see the LILO screen (the one with the red hat logo, before the system boots), press Ctrl-x and then, at the `boot:` prompt:

```
boot: linux 1
```

2.3 An alternative `init`, and when you might need it

Normally the `init` program (`/sbin/init`) is started automatically by the kernel; the location is known by the kernel. But sometimes you need to specify an alternative `init` program, such as when the main dynamic link libraries do not correspond with the executables on the system. In that case, nothing works until they match. A useful tool is a *statically linked* shell, which does not depend on other dynamically loaded libraries. There is one called `sash`, which you will install today. The `sash` package installs `sash` as `/sbin/sash`.

To start the computer using `/sbin/sash` as the `init` process, you can type the following at a LILO prompt:

```
boot: linux 1 init=/sbin/sash
```

and the system will provide you with the limited capabilities of the stand-alone shell, definitely better than no access at all!

3 Procedure

1. Make sure you have sash installed:

   ```
   $ rpm -q sash
   ```

If it tells you it's not installed, then go and install it:

```
$ cd /ftp/rh-7.1-updated/RedHat/RPMS
$ sudo rpm -Uhv sash-3.4-8.i386.rpm
```

2. Verify that you have sash there in the right place:

```
$ ls -l /sbin/sash
-rwxr-xr-x    1 root     root       508268 Aug 14  2001 /sbin/sash
```

(The date and size will be differnent, but similar).

3. Use the `ldd` program to verify that `sash` is not dynamically linked. Then compare the result with running on `bash`:

```
$ ldd /sbin/sash
$ ldd /bin/bash
```

4. Create a boot disk for Red Hat 7.2 like this (assuming that the automounter mounts `ictlab:/var/ftp/pub` on `/ftp`). Start by putting a formatted floppy disk (that has no bad sectors) into the drive (all contents will be replaced), and:

```
$ cd /ftp/redhat-7.2/updates/images/i386
$ dd if=bootnet.img of=/dev/fd0
```

and wait until the data copying and conversion program `dd` has finished creating a boot floppy.

5. boot your system into single user mode, as described in section 2.2 on the preceding page.

6. Change to the `/lib` directory, and examine the important dynamic library loader, `ld-linux.so.2`:

```
# ls -l /lib/ld-linux.so.2
lrwxrwxrwx    1 root     root           11 Dec 15 08:44
/lib/ld-linux.so.2
-> ld-2.2.4.so
```

7. Now you will do something that you will probably never do again: rename `ld-linux.so.2`!

```
# mv ld-linux.so.2 ld-linux.so.2-orig
```

8. Now examine this file:

```
# ls -l ld-linux.so.2-orig
```

Oh dear, it doesn't work! What to do?

4 Getting out of Disaster

Read section 2.3 on page 2, and using the procedure there, recover from this situation, and when finished, make sure that you can have the normal `ls` command work.

Hints:

1. Type `help` when you have sash running
2. type `-ls` within `sash`

5 Now again!

1. rename the file `/lib/libc.so.6`:

   ```
   # mv /lib/libc.so.6 /lib/libc.so.6-orig
   ```

2. This time, boot your computer using the linux installation disk, but this time, at the LILO prompt, enter:

   ```
   boot: linux rescue
   ```

3. Select `NFS`, `ictlab.tyict.vtc.edu.hk` as the NFS server, and the NFS directory: `/var/ftp/pub/redhat-7.2`.
4. Agree to mount your Linux file systems.
5. Make sure that you are working on `your` hard disk, and not `/dev/hdc`, the one that Samson has installed on the internal hard disks!
6. Fix the problem again, and verify that it all works.
7. Show your lecturer.

5.1 More about runlevels

Runlevels give great flexibility to the system administrator. Have a look in the `/etc/rc.d` directory. First, make your terminal window nice and wide, then:

```
$ cd /etc/rc.d
$ ls -lR rc?.d
```

This will list the contents of the dirctories `/etc/rc.d/rc0.d`, `/etc/rc.d/rc1.d`, `/etc/rc.d/rc2.d`, `/etc/rc.d/rc3.d`, `/etc/rc.d/rc4.d`, `/etc/rc.d/rc5.d` and `/etc/rc.d/rc6.d`.

Each directory contains lots of symbolic links to scripts in `/etc/rc.d/init.d`. Some start with `K`*nn*, and others start with `S`*xx*, where *xx* is a two-digit number.

Under the control of `/sbin/init`, these scripts are run with the argument `start` if they start with `S`*xx*, and `stop` if they start with `K`*nn*.

1. Look at the links in `/etc/rc.d/rc0.d`. Do most of them start with `S`*xx* or `K`*xx*?

2. Why do you think this is so?

.....

3. Look at the links in `/etc/rc.d/rc3.d`. Do most of them start with `S`*xx* or `K`*xx*?

.....

4. Why do you think this is so?

.....

5. Try running a few of these scripts by hand:

```
$ sudo /etc/init.d/autofs stop
$ sudo /etc/init.d/autofs start
```

6. What do you think the **chkconfig** program does when you type:

```
$ sudo chkconfig --level 345 autofs off
```

and

```
$ sudo chkconfig --level 345 autofs on
```

Do each of these things, and look for any change in the symbolic links before and after each of the two steps.

.....