The init scripts

- **The Linux kernel**
- **The init program**
- **inittab (init's config file)**
- **The init scripts (sysV conventions)**

The init scripts

- **The Linux kernel**
- **The init program**
- **inittab (init's config file)**
- **The init scripts (sysV conventions)**

The Linux kernel

- **During bootup a bunch of stuff happens (self-test, boot loader, etc.) but eventually the kernel gets control.**
- **The kernel does some self-initialization, then gets "user-space" going with the "init" process.**
- **The kernel refers to the first user-specified process as the "init" process, and there is an "init" program. But they don't have to be the same program...**
- **The kernel is hard-coded to look for the init program in a few locations (see next slide)**
- **You can control some kernel parameters from the "kernel command line" which you specify to the boot loader (beyond the scope of this talk). One of those parameters is "which program is going to play the role of the 'init' process"?**
- **Possibilities:**
  - `init=/sbin/init`
  - `init=/bin/sh`
  - `others...`

The Linux kernel

The kernel source is big, divided in several directories:

```
Documentation          kernel
arch                   lib
crypto                 mm
drivers                net
fs                     scripts
include                security
init                   sound
ipc                    usr
```

The Linux kernel

The init directory is concerned with starting off the first userspace program. It has a file called main.c, which contains a function called init which will become the first user process:

And that function ends with:

```
        if (execute_command)
                run_init_process(execute_command);

        run_init_process("/sbin/init");
        run_init_process("/etc/init");
        run_init_process("/bin/init");
        run_init_process("/bin/sh");

        panic("No init found.  Try passing init= option to
kernel.");
```

The init scripts

- **The Linux kernel**
- **The init program**
- **inittab (init's config file)**
- **The init scripts (sysV conventions)**

The init program

- **The init program has a few functions:**
  - **Serves as clonable first user-program instance**
  - **Inherits orphan children and cleans up zombies**
  - **Implements run-levels**

Runlevels

- **What's a run-level?**
  - **computers are complicated systems**
  - **functions can be designed in layers**
  - **higher layers depend on lower layers for services**
  - **run-levels are a way of bringing the system to a certain level of functionality**
  - **Eg:**
    - **runlevel 1 (aka single-user mode, maintenance mode)**
    - **runlevel 2 (Could be multi-tasking, stand-alone system (not networked))**
    - **runlevel 3 (Could be networked, but no gui)**
    - **runlevel 5 (Could be full multi-tasking, gui, fully networked)**

Runlevels in distros

- **Different distros use runlevels for different things**
  - **RedHat does something similar to previous slide**
  - **Debian goes from runlevel 1 (single-user mode) to runlevel 2 (full multi-user, gui, fully networked) (editorial comment: boo, hiss)**
- **The runlevel command tells you what runlevel you are currently in, and the previous one too (or who -r)**
  - **looks at utmp file (/var/run/utmp)**
- **To go to another runlevel**
  - **init 2**
  - **telinit 2**

Runlevels in distros

- Linux distros (and most Unix ones) use runlevels in a hierarchy: higher runlevel numbes mean more functionality
  - building on earlier runlevels
- runlevels can be used to run entirely different services
  - Machine can have different roles in different runlevels
    - backup server -> runlevel 3
    - file server -> runlevel 4
  - Or, in an embedded device, perhaps:
    - normal operating mode -> runlevel 2
    - dormant mode -> runlevel 3
    - upgrade mode -> runlevel 4

Single user mode

- **Booting to single user mode**
- **Add "single" to the kernel command line**
- **(see kernel command line, /proc/cmdline)**
- **try booting with "single" on kernel command line**
- **give "exit" command to continue booting to default runlevel**

different init programs

- **You don't have to run the init program as the init process**
- **You can run /bin/sh**
- **This gets you out of problems with init (like bad inittab)**
- **Can also help when you don't know password(s)**
- **When done, reboot**
- **"exit" from /bin/sh will cause kernel panic! Try it!**

misc advice for init=/bin/sh

- When you use /bin/sh as init process
- init scripts don't run; they don't prep system for you
- You might have to (depending on distro):
  - mount /proc
    - `mount -t proc none /proc`
  - remount / as read-write
    - `cat /etc/mtab  # to show which device is /`
    - `cat /etc/fstab # to show which partitions mount where`
    - `mount -o remount,rw /dev/hda3`
  - possibly other things
- when done:
  - `reboot  # or`
  - `telinit 6`

The init scripts

- **The Linux kernel**
- **The init program**
- **inittab (init's config file)**
- **The init scripts (sysV conventions)**

inittab

- **The init program has a config file called inittab**
- **IT IS NOT A SCRIPT! It's a config file.**
- **It defines what should happen:**
  - **at bootup (system initialization time)**
  - **during regular operations in each runlevel**
  - **at shutdown**
  - **during power-related events (power-low, power-restored)**
  - **some misc stuff**

inittab(5)

- format of inittab(5)
  - `id:runlevels:action:process`
- `id` is a unique key string
- `runlevels` is the set of runlevels in which this line will take effect
- `action` is a keyword which shows what sort of action this is (initdefault, respawn, wait, once, boot, bootwait, off, ondemand, initdefault, sysinit, powerwait, powerfail, powerokwait, powerfailnow, ctlaltdel, kbrequest).
- `process` is a program to start, with any args required (eg, `/etc/rc` or `/etc/getty 9600 tty1`)
- After editing inittab, run `init q` to re-read inittab

inittab(5) example

- **Look at /etc/inittab**
- **(settings/font/huge)**
- **My system is a Debian system (unstable)**
- **Look at:**
  - **initdefault**
  - **sysinit (rcS)**
  - **l0-l6 (wait, rc)**
  - **1-6 (respawn, getty or xdm)**
- **Look at RedHat inittab**

The init scripts

- **The Linux kernel**
- **The init program**
- **inittab (init's config file)**
- **The init scripts (sysV conventions)**

rcS or rc.sysinit

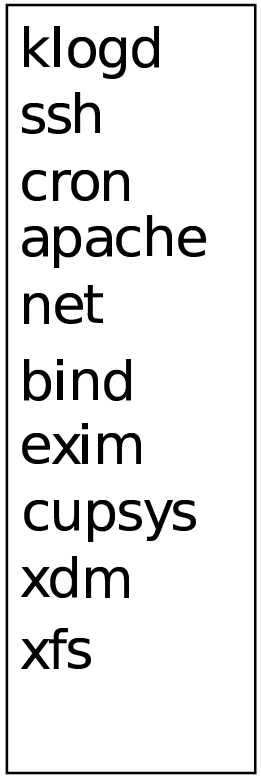
- **/etc/init.d/rcS, /etc/rc.d/rc.sysinit**
- **In RedHat systems, rc.sysinit is a script that does a few things like load modules, mount filesystems, set the time, etc.**
- **In Debian system, rcS script is a loop that calls other scripts (in /etc/init.d/rcS.d) to do the above.**

rc

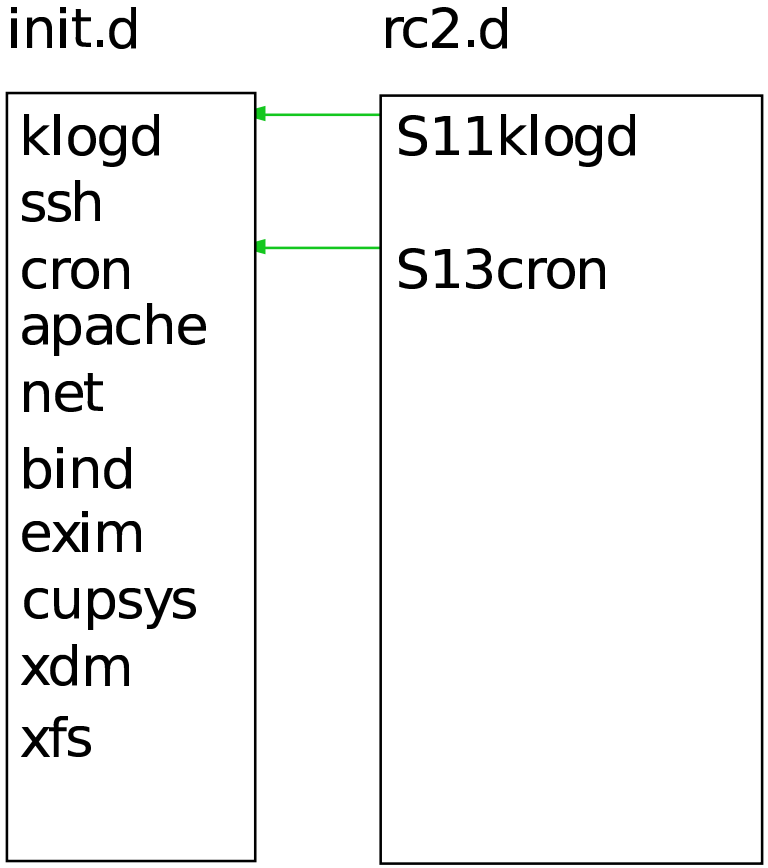
- **In sysV style init scripts, the rc script is used to run a bunch of other scripts (the Start scripts and Kill scripts)**
- **These scripts are in the /etc/init.d directory (sometimes in /etc/rc.d/init.d, depending on distro).**
- **Because you may want to start the same services in various runlevels, there is a directory per runlevel (/etc/rcN.d), and it contains entries that represent the services that could run in that runlevel**
- **Typically these are links to the real scripts in /etc/init.d**
- **Eg, if you have an sshd, there will be a script to start it called /etc/init.d/ssh, and there will be a link from each runlevel directory to the script: /etc/rc2.d/S80sshd, /etc/rc3.d/S80sshd, etc.**

init scripts diagram I

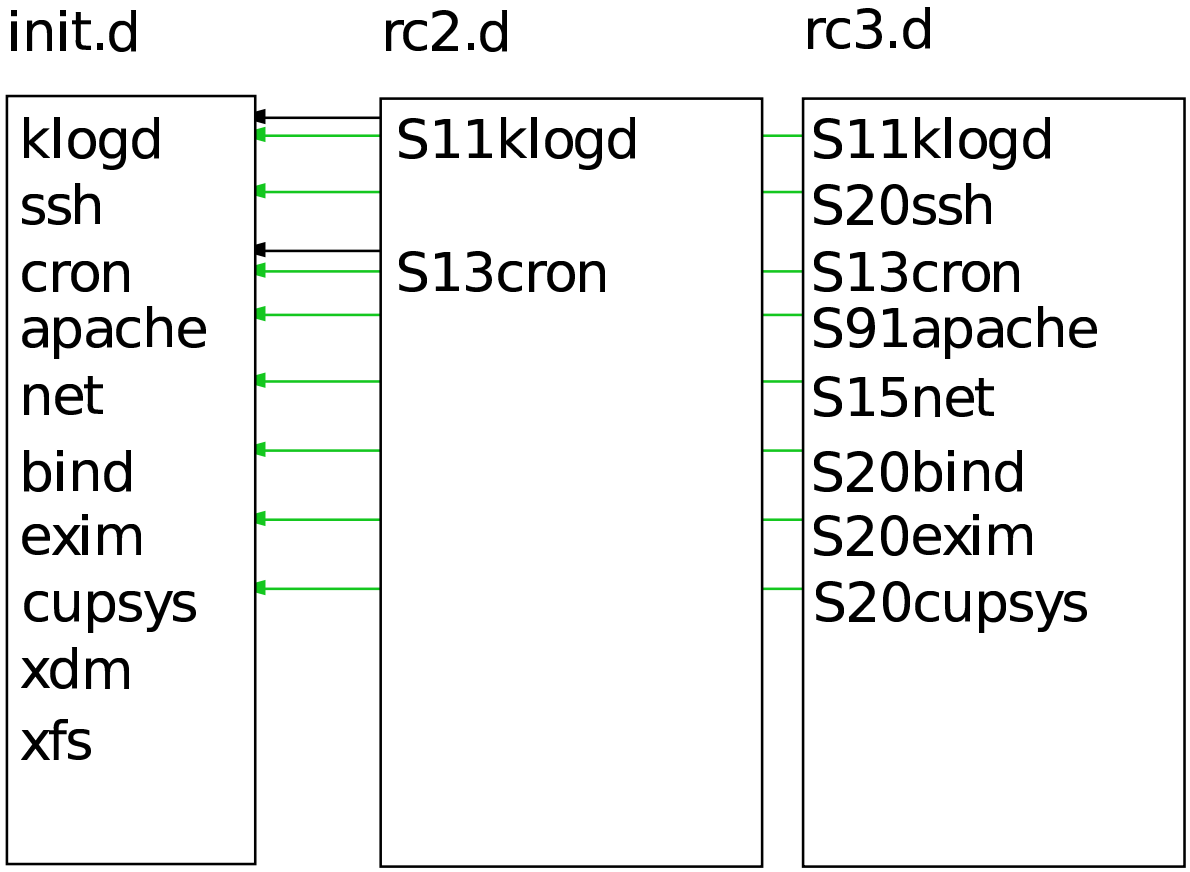
init.d



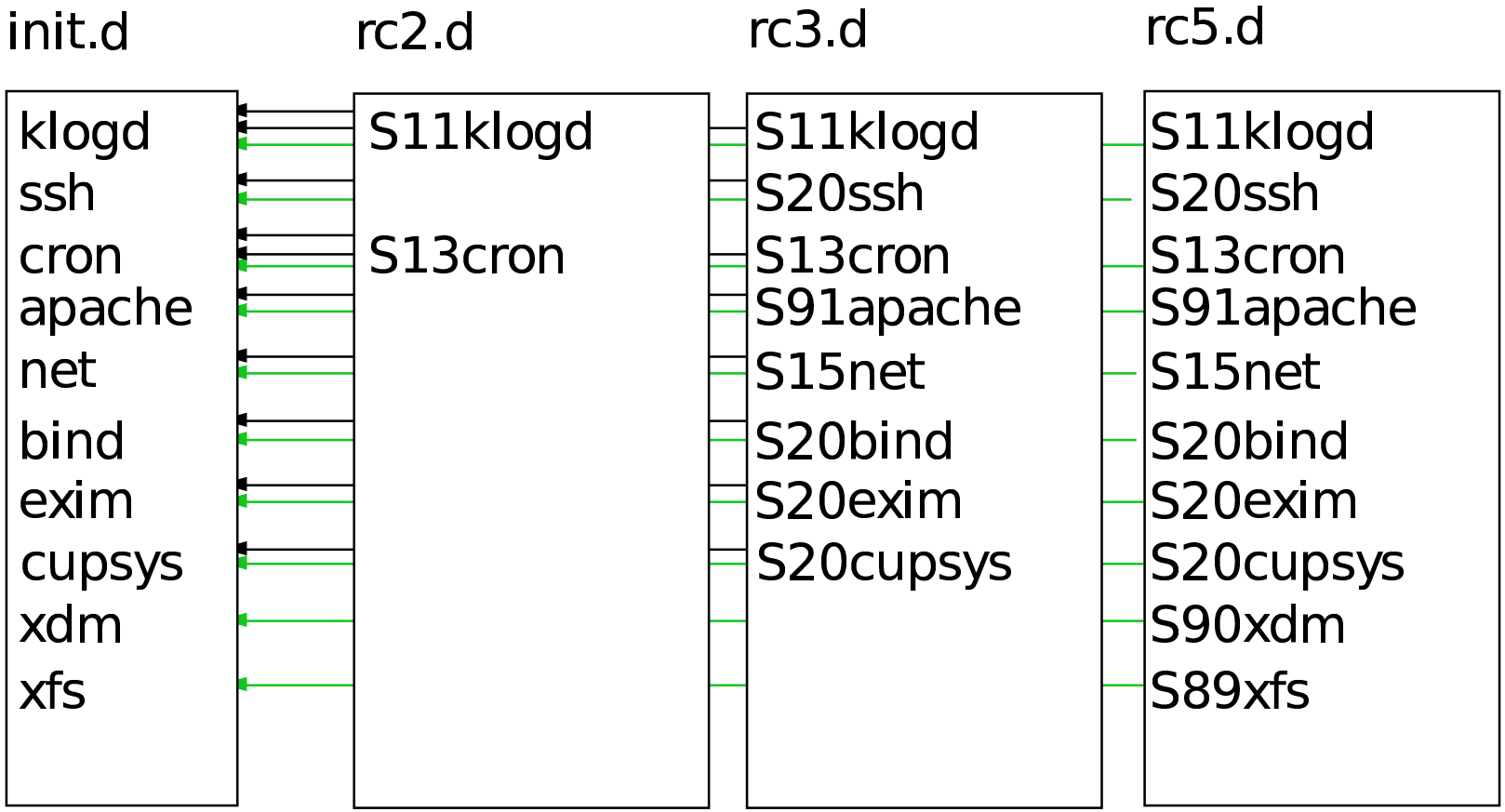
init scripts diagram II



init scripts diagram III



init scripts diagram IV



conventions for init scripts

- **These scripts all follow the same conventions: They can be used to start, stop, restart, or reload a service.**
- **(Look at /etc/init.d/rc)**
- **(Look at a typical init script, eg, klogd or bind)**
- **Many distros have utilities for managing the links:**
  - **Debian: update-rc.d**
  - **RedHat: chkconfig**

managing init scripts

- **To cause a daemon to start on boot, add a link to its init script under rcN.d, where N is the runlevel.**
- **Call the script SNNname, to Start the script "name", or KNNname to Kill it.**
- **The NN is assigned by you to cause the scripts to be run in a certain order.**

RedHat init scripts

- chkconfig lists all available services, and status per runlevel.
  - chkconfig --list
- change which daemons start in which runlevels
  - chkconfig --level N name <on|off|reset>
  - chkconfig --2345 apache on
- creates links in /etc/rc[2345].d called S??apache
  - (and remove links called K??apache from those runlevels)
  - The ?? is obtained from a coment in the init script
    - # chkconfig: 2345 10 90
- start in levels 2,3,4,5; 10 -> S10network; 90 -> K90network

RedHat init scripts

- **chkconfig is good for setting things up for next boot**
- **what about now?**
- **"service" utility**
- **service apache start|stop|whatever_option_initscript_takes**
- **useful for more, see the service program**
  - **which service**
  - **file /sbin/service**
    - **/sbin/service: Bourne shell script text executable**

Debian init scripts

- **update-rc.d**
- **This will help you manage your init script links**
- **does not use comments in init scripts to decide runlevels or ordinal numbers, uses defaults or command line options**
- **to add a link in rcN.d, requires init script to exist in init.d**
- **to remove link from rcN.d, requires ABSENCE of initscript in init.d**
- **to force action in spite of absence or presence of script in init.d, use -f option (force)**

Explore Dynebolic

- **Now that we've seen how the init scripts work, let's explore a little-known distro called Dynebolic (was distributed with the July 2004 Linux Format magazine (LXF55D)).**
- **(cd projects/dyne)**
- **(mount -o loop initrd.img mnt)**
- **...**

URL

Slides will be posted on http://www.linuxbutler.ca

Pop Quiz I

- **What happens if you set the default runlevel to 0 or 6?**

Pop Quiz II

- **If you put init=/bin/sh on the kernel command line to set the init program to /bin/sh, what runlevel does that put you in after you've booted?**