05-2 Userspace Initialization – init.d

Initialization

- Kernel Initialization
- Userspace Initialization

Userspace Initialization

- At startup
  - Kernel initializes
  - Mounts a root file system
  - Executes set of initialization routines
- We'll start with a minimal filesystem and build on it

Root File System: Top-Level Directories

```
bone$ sudo
bone$ tree
/
|-- bin
|-- dev
|-- etc
|-- home
|-- lib
|-- sbin
|-- usr
|-- var
`-- tmp
```

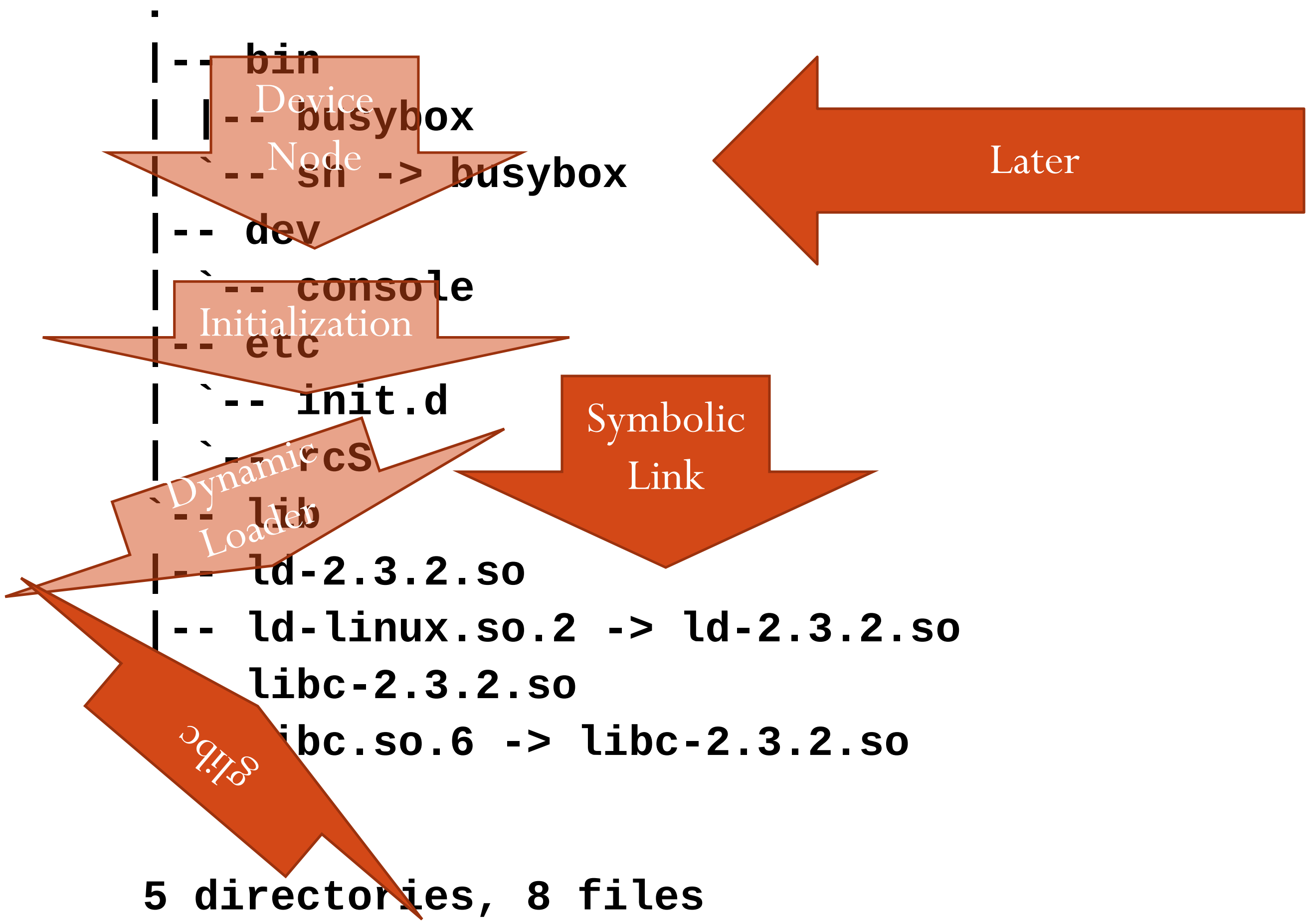
Directory	Contents
bin	Binary executables, usable by all users on the system
dev	Device nodes (see “Device Driver Basics”)
etc	Local system configuration files
home	User account files
lib	System libraries, such as the standard C library and many others
sbin	Binary executables usually reserved for superuser accounts on the system
usr	A secondary file system hierarchy for application programs, usually read-only
var	Contains variable files, such as system logs and temporary configuration files
tmp	Temporary files

Root File System: Top-Level Directories

```
bone$ sudo mkdir /mnt/eMMC
bone$ sudo mount /dev/mmcblk1p1 /mnt/eMMC/
bone$ tree -L 1 /mnt/eMMC
/mnt/eMMC
├── bin/          ├── opt/
├── boot/         ├── proc/
├── dev/          ├── root/
├── etc/          ├── run/
├── home/         ├── sbin/
├── lib/          ├── srv/
├── media/        ├── sys/
├── mnt/          ├── tmp/
                  ├── usr/
                  └── var/
```

```
bone$ tree -L 1 /
/
|-- bin
|-- dev
|-- etc
|-- home
|-- lib
|-- sbin
|-- usr
|-- var
`-- tmp
```

Minimal File System



```
.
|-- bin
|  |-- busybox
|  `-- sh -> busybox
|-- dev
|  `-- console
|-- etc
|  `-- init.d
|  `-- rcS
`-- lib
|-- ld-2.3.2.so
|-- ld-linux.so.2 -> ld-2.3.2.so
|-- libc-2.3.2.so
`-- libc.so.6 -> libc-2.3.2.so


5 directories, 8 files
```

Final Kernel Events

- Final sequence of events for the kernel thread called **`kernel_init`** spawned by the kernel during the final stages of boot
- **`run_init_process()`** function *never returns* if no error conditions
- Memory space in which the calling thread is executing from is overwritten by the called program's memory image
- In effect, the called program directly replaces the calling thread, including inheriting its Process ID (PID)

Kernel’s Last Boot Steps (…/`init/main.c`)

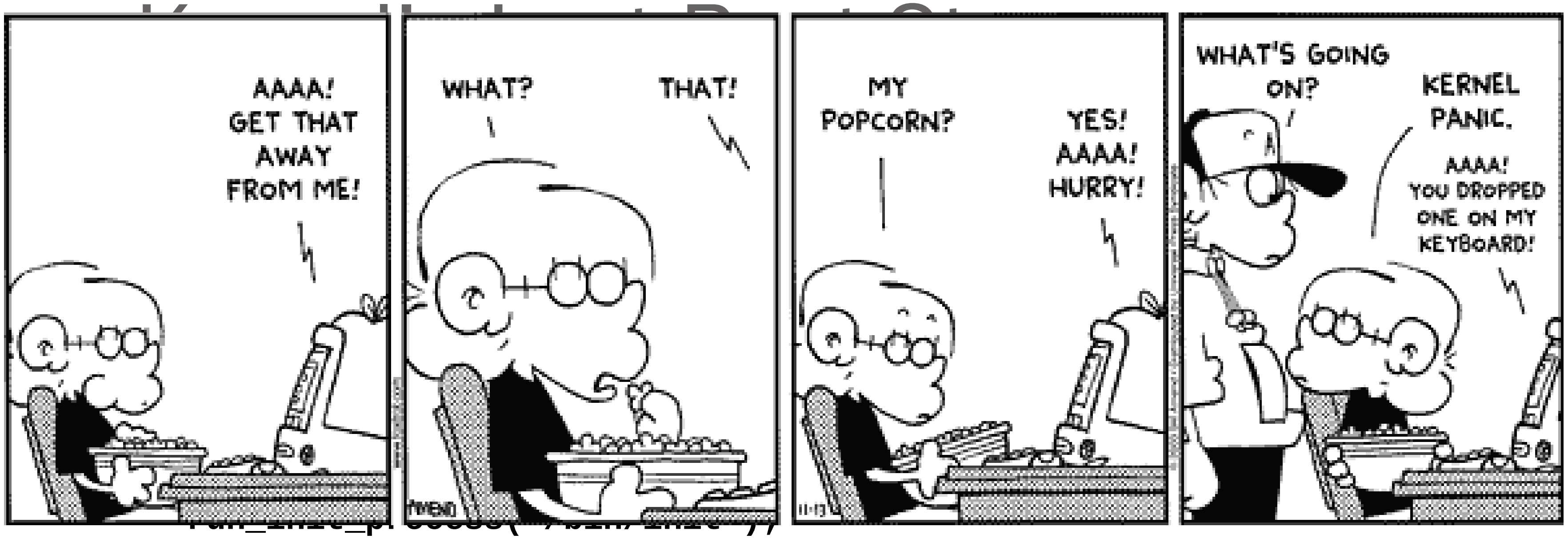
```
   if (execute_command) {
         run_init_process(execute_command);
         printk(KERN_WARNING "Failed to execute %s.  Attempting "
                                 "defaults...\n", execute_command);
   }
   run_init_process("/sbin/init");
   run_init_process("/etc/init");
   run_init_process("/bin/init");
   run_init_process("/bin/sh");

   panic("No init found.  Try passing init= option to kernel. "
              "See Linux Documentation/init.txt for guidance.");
}
```

Final Kernel Events (cont.)

- This is the start of user space processing
- Unless the kernel is successful in executing one of these processes, the kernel will halt, displaying the message passed in the **`panic()`** system call
- If you have been working with embedded systems for any length of time, and especially if you have experience working on root file systems, you are more than familiar with this kernel **`panic()`** and its message!
- If you search on Google for this **`panic()`** error message, you will find page after page of hits for this FAQ

FoxTrot for November 09, 2006



```
    run_init_process("/bin/init");
    run_init_process("/bin/sh");

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

First User Space Program

- Most systems: `/sbin/init` is spawned.

```
|-- bin
|  |-- busybox
|  '-- sh -> busybox
|-- dev
|  '-- console
|-- etc
|  '-- init.d
|  '-- rcS
'-- lib
|-- ld-2.3.2.so
|-- ld-linux.so.2 -> ld-2.3.2.so
|-- libc-2.3.2.so
'-- libc.so.6 -> libc-2.3.2.so
```

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

Busybox is run as the initial process

Resolving Dependencies

- You can't put just any program as **`init`**
- There may be dependencies

```
host$ ldd a.out
linux-gate.so.1 =>  (0x002df000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0x00da8000)
/lib/ld-linux.so.2 (0x00a92000)

bone$ readelf -d a.out | grep NEEDED
 0x00000001 (NEEDED)       Shared library: [libc.so.6]
```

Customized Initial Process

- Look in **`/boot/uEnv.txt`**

```
console=ttyS0,115200 ip=bootp
  root=/dev/nfs init=/sbin/myinit
```

The **`init`** process

- Use standard **`init`**
- Reads **`/etc/inittab`**

```
# /etc/inittab: init(8) configuration.
# $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $

# The default runlevel.
id:5:initdefault:

# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS
```

The `init` process

- `# What to do in single-user mode.`
- `~~:S:wait:/sbin/sulogin`

- `# /etc/init.d executes the S and K scripts upon change`
- `# of runlevel.`
- `#`

- `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`
- `l4:4:wait:/etc/init.d/rc 4`
- `l5:5:wait:/etc/init.d/rc 5`
- `l6:6:wait:/etc/init.d/rc 6`

Runlevels

Runlevel	Purpose
0	System shutdown (halt)
1	Single-user system configuration for maintenance
2	User defined
3	General purpose multiuser configuration
4	User defined
5	Multiuser with graphical user interface on startup
6	System restart (reboot)

- Runlevel scripts are found in **`/etc/rc.d/init.d/`**
- or **`/etc/init.d/`**

Runlevel Directory Structure on 4.4 Beagle

```
bone$ ls -dl /etc/rc*
drwxr-xr-x 2 root root 4096 Aug 28 11:26 /etc/rc0.d
drwxr-xr-x 2 root root 4096 Aug 28 11:26 /etc/rc1.d
drwxr-xr-x 2 root root 4096 Aug 28 11:26 /etc/rc2.d
drwxr-xr-x 2 root root 4096 Aug 28 11:26 /etc/rc3.d
drwxr-xr-x 2 root root 4096 Aug 28 11:26 /etc/rc4.d
drwxr-xr-x 2 root root 4096 Aug 28 11:26 /etc/rc5.d
drwxr-xr-x 2 root root 4096 Aug 28 11:26 /etc/rc6.d
-rwxr-xr-x 1 root root  306 Aug 28 11:18 /etc/rc.local
-rw-r--r-- 1 root root  101 Aug 28 11:21 /etc/rcn-ee.conf
drwxr-xr-x 2 root root 4096 Aug 28 11:27 /etc/rcS.d
```

Example Runlevel Directory on 4.4 Beagle

```
bone$ ls -ls /etc/rc5.d/
total 0
4 -rw-r--r-- 1 root root 677 Apr  6  2015 README
0 lrwxrwxrwx 1 root root  18 Aug 28 11:17 S01bootlogs -> ../init.d/bootlogs
0 lrwxrwxrwx 1 root root  17 Aug 28 11:21 S01hostapd -> ../init.d/hostapd
0 lrwxrwxrwx 1 root root  14 Aug 28 11:17 S01motd -> ../init.d/motd
0 lrwxrwxrwx 1 root root  17 Aug 28 11:21 S01rsyslog -> ../init.d/rsyslog
0 lrwxrwxrwx 1 root root  17 Aug 28 11:21 S02apache2 -> ../init.d/apache2
0 lrwxrwxrwx 1 root root  14 Aug 28 11:21 S03cron -> ../init.d/cron
0 lrwxrwxrwx 1 root root  14 Aug 28 11:21 S03dbus -> ../init.d/dbus
0 lrwxrwxrwx 1 root root  17 Aug 28 11:26 S03haveged -> ../init.d/haveged
0 lrwxrwxrwx 1 root root  21 Aug 28 11:21 S03loadcpufreq -> ../init.d/loadcpufreq
0 lrwxrwxrwx 1 root root  15 Aug 28 11:21 S03rsync -> ../init.d/rsync
0 lrwxrwxrwx 1 root root  13 Aug 28 11:21 S03ssh -> ../init.d/ssh
0 lrwxrwxrwx 1 root root  16 Aug 28 11:21 S03udhcpd -> ../init.d/udhcpd
0 lrwxrwxrwx 1 root root  22 Aug 28 11:21 S04avahi-daemon -> ../init.d/avahi-daemon
0 lrwxrwxrwx 1 root root  19 Aug 28 11:21 S04bluetooth -> ../init.d/bluetooth
0 lrwxrwxrwx 1 root root  17 Aug 28 11:21 S04connman -> ../init.d/connman
0 lrwxrwxrwx 1 root root  22 Aug 28 11:21 S04cpufrequtils -> ../init.d/cpufrequtils
0 lrwxrwxrwx 1 root root  18 Aug 28 11:21 S05rc.local -> ../init.d/rc.local
0 lrwxrwxrwx 1 root root  19 Aug 28 11:21 S05rmnologin -> ../init.d/rmnologin
```

Beagle 3.8

```
beagle$ cat /etc/init.d/README
You are running a systemd-based OS where traditional init scripts have
been replaced by native systemd services files. Service files provide
very similar functionality to init scripts. To make use of service
files simply invoke "systemctl", which will output a list of all
currently running services (and other units). Use "systemctl
list-unit-files" to get a listing of all known unit files, including
stopped, disabled and masked ones. Use "systemctl start
foobar.service" and "systemctl stop foobar.service" to start or stop a
service, respectively. For further details, please refer to
systemctl(1).
```

Beagle 3.8 (cont)

```
beagle$ cat /etc/init.d/README
Note that traditional init scripts continue to function on a systemd
system. An init script /etc/init.d/foobar is implicitly mapped
into a service unit foobar.service during system initialization.

Thank you!

Further reading:
        man:systemctl(1)
        man:systemd(1)
        http://0pointer.de/blog/projects/systemd-for-admins-3.html

http://www.freedesktop.org/wiki/Software/systemd/Incompatibilities
```