AGENDA

- W's of File System
- Overall File System Architecture
- File System Features
- Different types of File System
- Different FS Implementation
- Linux Root File System overview
- Building RootFS from scratch

harish.hanumanthappa@aricent.com

Why File system is Required

- Without a file system, information placed in a storage area would be one large body of data
- There is no way to tell where one piece of information stops and the next one begins
- Need to provide mechanism to list, store, retrieve, modify and display objects consistently to all the users of the system
- By separating the data into distant objects and giving it a name, the information contained in the object is easily isolated and identified.
- Taking its name from the way paper-based information systems are named, each group of data is called a "file".

What is File System

- The structure and logic rules used to manage the groups of information and their names is called a "file system".
- FS is used to control how data is stored and retrieved.
- Provides a way to store data about files
  - Filenames
  - Permissions
  - Type of File like data files, special Files like pipe, dev nodes, directories
  - Attributes – Permission
- Operating system needs to understand structure of file system so it can display its contents, open files, and save files to it

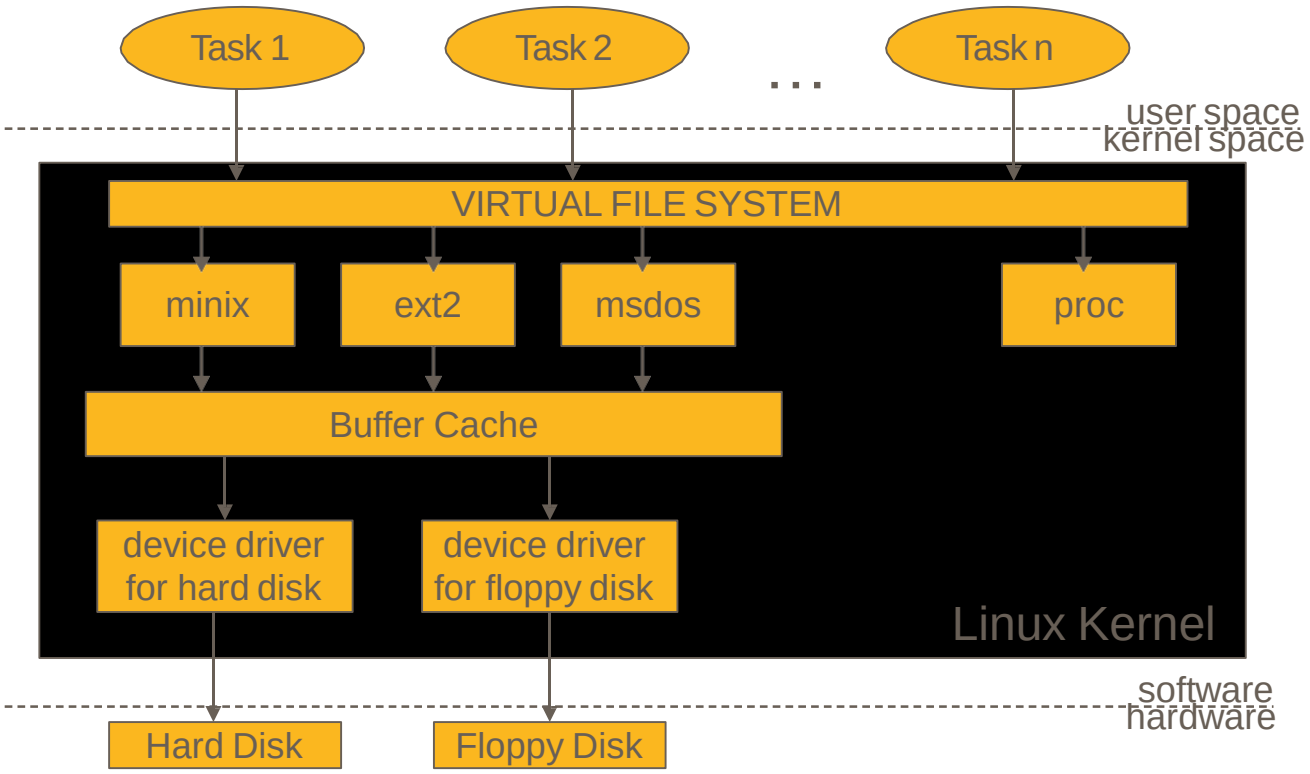
Different Kinds of File Representation

- Files are represented by inodes
- Directories are special files (dentries)
- Devices accessed by I/O on special files
- UNIX filesystems can implement ‘links’

Different Types of File System

- Disk FS: ext2, ext3, FAT, FAT32 & NTFS
- Network FS: Samba & NFS
- Flash FS: JFFS2
- Special FS: proc FS, sys FS, debugfs, tmpfs

Linux File System Architecture



File system Features

- Hard Links
- Symbolic Links
- Block journaling
- Case sensitive
- File change log
- XIP – Execute In Place
- Filesystem level encryption
- OS support

File system Features

- Limits
  - Maximum Filename Length
  - Allowable characters in directory entries
  - Maximum pathname length
  - Maximum file size

- Metadata
  - File Owner
  - POSIX File permissions
  - Creation Time Stamps
  - Last Access / Read Time stamps
  - Last Meta Data Change Time stamps
  - Mandatory Access Control
  - Discretionary Access Control

Different Implementation of File System

- Each one has different structure and logic, properties of speed, flexibility, security, size, reliability and more.
- Designed specifically to be used for targeted applications.
- There are many different kinds of file systems
  - EXT2
  - YAFFS
  - SQUASHFS
  - INITRAMFS
  - RAMFS
  - EXT4
  - BTRFS
  - SIFFS

Example of SPIFFS for Embedded Market

- Spiffs is a file system intended for SPI NOR flash devices on embedded targets.

- What spiffs does:
  - Specifically designed for low ram usage
  - Uses statically sized ram buffers, independent of number of files
  - Posix-like api: open, close, read, write, seek, stat, etc
  - It can be run on any NOR flash, not only SPI flash - theoretically also on embedded flash of an microprocessor
  - Multiple spiffs configurations can be run on same target - and even on same SPI flash device
  - Implements static wear leveling
  - Built in file system consistency checks

Example of SPIFFS for Embedded Market - Contd.

- What spiffs does not:
  - Presently, spiffs does not support directories. It produces a flat structure. Creating a file with path *tmp/myfile.txt* will create a file called *tmp/myfile.txt* instead of a *myfile.txt* under directory *tmp*.
  - It is not a realtime stack. One write operation might take much longer than another.
  - Poor scalability. Spiffs is intended for small memory devices - the normal sizes for SPI flashes. Going beyond ~128MB is probably a bad idea. This is a side effect of the design goal to use as little ram as possible.
  - Presently, it does not detect or handle bad blocks.
  - One configuration, one binary. There's no generic spiffs binary that handles all types of configurations.

Overview of Filesystem Hierarchy Standard (FHS)

- Defines file system structure like the names, locations, and permissions for many file types and directories.

File System Structure

- The file system structure is the most basic level of organization in an operating system.
- The way an operating system interacts with its users, applications, and security model nearly always depends on how the operating system organizes files on storage devices.
- File systems break files down into two logical categories:
  - Shareable versus un-shareable files
    - *Shareable* files can be accessed locally and by remote hosts; *unshareable* files are only available locally.*Variable*
  - Variable versus static files
    - files, such as log files, can be changed at any time; *static* files, such as binaries, do not change without an action from the system administrator.

The Root Directory

- /bin
- /boot
- /dev
- /etc
- /home
- /initrd
- /lib
- /lost+found
- /media
- /mnt
- /opt
- /proc
- /root
- /sbin
- /usr
- /srv
- /tmp
- /var

1. /bin

- Contains several useful commands that are of use to both the system administrator as well as non-privileged users.
- Usually contains the shells like bash, csh, etc.... and commonly used commands like cp, mv, rm, cat, ls.
- Also contains programs which boot scripts may depend on
- There are no (real) subdirectories in /bin

1. /bin …(cont)

- cat
- chgrp
- chmod
- chown
- cp
- date
- dd
- df
- dmesg
- echo
- false
- hostname
- kill
- ln
- login
- ls
- mkdir
- mknod
- more
- mount
- mv
- ps
- pwd
- rm
- rmdir
- sed
- sh
- stty
- su
- sync
- true
- umount
- uname

1. /bin detail

- cat →Utility to concatenate files to standard output
- chgrp →Utility to change file group ownership
- chmod →Utility to change file access permissions
- chown →Utility to change file owner and group
- cp →Utility to copy files and directories
- date →Utility to print or set the system data and time
- dd →Utility to convert and copy a file
- df →Utility to report filesystem disk space usage
- dmesg →Utility to print or control the kernel message buffer
- echo →Utility to display a line of text
- false →Utility to do nothing, unsuccessfully

1. /bin detail (cont)

- hostname →Utility to show or set the system's host name
- kill →Utility to send signals to processes
- ln →Utility to make links between files
- login →Utility to begin a session on the system
- ls →Utility to list directory contents
- mkdir →Utility to make directories
- mknod →Utility to make block or character special files
- more →Utility to page through text
- mount →Utility to mount a filesystem
- mv →Utility to move/rename files
- ps →Utility to report process status

1. /bin detail (cont)

- pwd →Utility to print name of current working directory
- rm →Utility to remove files or directories
- rmdir →Utility to remove empty directories
- sed →The `sed' stream editor
- sh →The Bourne command shell
- stty →Utility to change and print terminal line settings
- su →Utility to change user ID
- sync →Utility to flush filesystem buffers
- true →Utility to do nothing, successfully
- umount →Utility to unmount file systems
- uname →Utility to print system information

2. /boot

- Contains everything required for the boot process exceptfor configuration files not needed at boot time and the map installer
- Stores data that is used before the kernel begins executing user-mode programs
- May include the system kernel (under symbolically linked)

- /boot/boot.0300 → Backup master boot record.
- /boot/boot.b → The basic boot sector
- /boot/chain.b → Used to boot non-Linux operating systems
- /boot/config-kernel-version → Installed kernel configuration.
- /boot/map → The location of the kernel
- /boot/vmlinuz → Normally the kernel or symbolic link to the kernel
- /boot/vmlinuz-kernel-version

3. /dev

- location of device files

/dev/ttyS0	→Device connected to Com1 (Modem, mouse,..)
/dev/psaux	→PS/2 mouse connection
/dev/lp0	→First parallel port
/dev/dsp	→Sound card (digital signal processor)
/dev/usb	→USB device nodes.
/dev/sda	→First SCSI device
/dev/scd	→First SCSI CD-ROM device
/dev/cdrom	→CD-ROM drive
/dev/fd0	→floppy drive
/dev/had	→The partition on primary hdd

4. /etc

- Contains all system related configuration files
- Local file used to control the operation of a program

- /etc/X11/ : contains all the configuration files for the X Window System
- /etc/X11/XF86Config, /etc/X11/XF86Config-4 : 'X' configuration file
- /etc/ftpchroot : List of ftp users that need to be chrooted
- /etc/ftpaccess : Determines who might get ftp-access to your machine.
- /etc/gateways : Lists gateways for 'routed'
- /etc/group, /etc/passwd. lists the configured user groups and who belongs to them.
- /etc/hostname : Contains the hostname of your machine
- /etc/host.conf : Determines the search order for look-ups

5. /home

- The user home directories
- Accessible only to its owner and the system administrator
- Contains the user's personal configuration files
- Quite large to be used as User's Documents Space

6. /initrd

- Provides the capability to load a RAM disk by the boot loader. This RAM disk can then be mounted as the root file system and programs can be run from it. Afterwards, a new root file system can be mounted from a different device. The previous root (from initrd) is then moved to a directory and can be subsequently unmounted.

- Initrd is mainly designed to allow system startup to occur in two phases, where the kernel comes up with a minimum set of compiled-in drivers, and where additional modules are loaded from initrd.

7. /lib

- Contains kernel modules and those shared library images (the C programming code library) needed to boot the system and run the commands in the root filesystem, ie. by binaries in /bin and /sbin

  - /lib/'machine-architecture' : Contains platform/architecture dependent libraries.
  - /lib/iptables : iptables shared library files.
  - /lib/kbd : Contains various keymaps.
  - /lib/modules/'kernel-version' : The home of all the kernel modules. The organisation of files here is reasonably clear so no requires no elaboration.
  - /lib/oss : All OSS(Open Sound System) files are installed here by default.
  - /lib/security : PAM library files.

12. /proc

- Virtual filesystem, runtime system information (e.g. system memory, devices mounted, hardware configuration, etc).
- The most of them have a file size of 0
- To view, use “cat”. Use “vi” to edit.

13. /root

- The home directory of the System Administrator, 'root'
- Why not in '/home'? Because '/home' is often located on a different partition or even on another system and would thus be inaccessible to 'root' when - for some reason - only '/' is mounted.

14. /sbin

- /sbin should contain only binaries essential for booting, restoring, recovering, and/or repairing the system in addition to the binaries in /bin.
- All executables related to system management, maintenance are located under this folder

15. /usr

- The largest share of data on a system
- the most important directories in the system as it contains all the user binaries, their documentation, libraries, header files, etc.... X and its supporting libraries, and User programs like telnet, ftp, etc.... as well, can be found here.

16. /var

- Contains variable data, files and directories the system must be able to write to during operation, like system logging files, mail and printer spool directories, and transient and temporary files

Building Root File System

General purpose toolbox: busybox

http://www.busybox.net/ from Codepoet Consulting

- Most Unix command line utilities within a single executable! Even includes a web server!
- Sizes less than 1 MB (statically compiled with glibc) less than 500 KB (statically compiled with uClibc)
- Easy to configure which features to include
- The best choice for
  - Initrds with complex scripts
  - Any embedded system!

Busybox commands!

addgroup, adduser, adjtimex, ar, arping, ash, awk, basename, bunzip2, bzcat, cal, cat, chgrp, chmod, chown, chroot, chvt, clear, cmp, cp, cpio, crond, crontab, cut, date, dc, dd, deallocvt, delgroup, deluser, devfsd, df, dirname, dmesg, dos2unix, dpkg, dpkg-deb, du, dumpkmap, dumpleases, echo, egrep, env, expr, false, fbset, fdflush, fdformat, fdisk, fgrep, find, fold, free, freeramdisk, fsck.minix, ftpget, ftpput, getopt, getty, grep, gunzip, gzip, halt, hdparm, head, hexdump, hostid, hostname, httpd, hush, hwclock, id, ifconfig, ifdown, ifup, inetd, init, insmod, install, ip, ipaddr, ipcalc, iplink, iproute, iptunnel, kill, killall, klogd, lash, last, length, linuxrc, ln, loadfont, loadkmap, logger, login, logname, logread, losetup, ls, lsmod, makedevs, md5sum, mesg, mkdir, mkfifo, mkfs.minix, mknod, mkswap, mktemp, modprobe, more, mount, msh, mt, mv, nameif, nc, netstat, nslookup, od, openvt, passwd, patch, pidof, ping, ping6, pipe_progress, pivot_root, poweroff, printf, ps, pwd, rdate, readlink, realpath, reboot, renice, reset, rm, rmdir, rmmod, route, rpm, rpm2cpio, run-parts, rx, sed, seq, setkeycodes, sha1sum, sleep, sort, start-stop-daemon, strings, stty, su, sulogin, swapoff, swapon, sync, sysctl, syslogd, tail, tar, tee, telnet, telnetd, test, tftp, time, top, touch, tr, traceroute, true, tty, udhcpc, udhcpd, umount, uname, uncompress, uniq, unix2dos, unzip, uptime, usleep, uudecode, uuencode, vconfig, vi, vlock, watch, watchdog, wc, wget, which, who, whoami, xargs, yes, zcat

glibc

http://www.gnu.org/software/libc/

- Clibrary from the GNU project
- Designed for performance, standards compliance and portability
- Found on all GNU / Linux host systems
- Quite big for small embedded systems: about ~1.7MB on arm (Familiar Linux iPAQs - libc: 1.2 MB, libm: 500 KB)
- Example “hello world” program size: 12 KB (dynamically linked), 350 KB (statically linked).

Creating a root filesystem

- Creating an empty file with a 400K size:
  dd if=/dev/zero of=rootfs.img bs=1k count=400

- Formating this file for the ext2 filesystem:
  mkfs.ext2 -i 1024 -F rootfs.img
  1 inode every 1024 bytes -> 400 files
  instead of 1 inode every 8192 bytes -> only 56 files!

Compiling busybox

- Getting the sources from http://busybox.net
- Configuring BusyBox:
  make xconfig
  Choosing to build a statically, natively compiled executable.
  We used BusyBox 1.10.2 with the following configuration:
  http://free-electrons.com/doc/embedded_lfs/busybox-1.10.2.config
- Compiling busybox:
  make
- Pre-installing busybox (in the _install/ subdirectory):
  make install
- Result: a 500K executable implementing all the commands that we need!

Populating the root filesystem

Logged as root:

- Creating a mount point:
  mkdir /mnt/rootfs
- Mounting the root filesystem image:
  mount -o loop rootfs.img /mnt/rootfs
- Copying the busybox file structure into the mounted image:
  rsync -a busybox/_install/ /mnt/rootfs/
  chown -R root:root /mnt/rootfs/
- Flushing the changes into the mounted filesystem image:
  sync

Creating device files

- Creating device files when programs complain:

```
mkdir /mnt/rootfs/dev
mknod /mnt/rootfs/dev/console c 5 1
mknod /mnt/rootfs/dev/null c 1 3
```

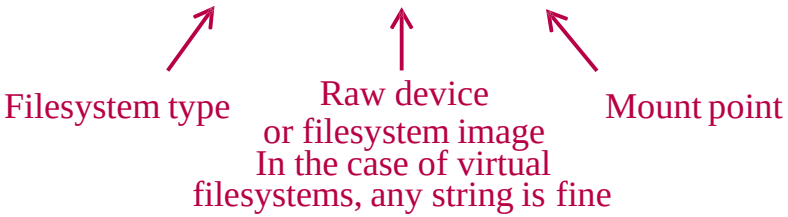
- Taking the GNU/Linux host as an example to find correct major and minor numbers:

```
ls -l /dev/console
ls -l /dev/null
```

Mounting virtual filesystems

Making /proc and /sys available
(required by several command line tools such as ps)

- Mounting /proc:
  mount -t proc none /proc
- Mounting /sys:
  mount -t sysfs none /sys



/etc/inittab file for busybox init

Creating the /etc/inittab file required by busybox init
Getting an example from busybox documentation
(not from the GNU/Linux host... missing features!)

```
# This is run first script
::sysinit:/etc/init.d/rcS
# Start an "askfirst" shell on the console
::askfirst:-/bin/sh
# Stuff to do when restarting the init process
::restart:/sbin/init
# Stuff to do before rebooting
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a -r
```

Setting up networking

- Adding TCP/IP and network card driver to the kernel
- Bringing up the network interface:
  ifconfig eth0 172.20.0.2
- Using the GNU/Linux host as a gateway:
  route add default gw 172.20.0.1
- Testing networking:
  ping -c 3 172.20.0.1
  -c 3: useful when [Ctrl][C] doesn't work (missing tty settings)
- Testing routing:
  ping -c 3 <external address>

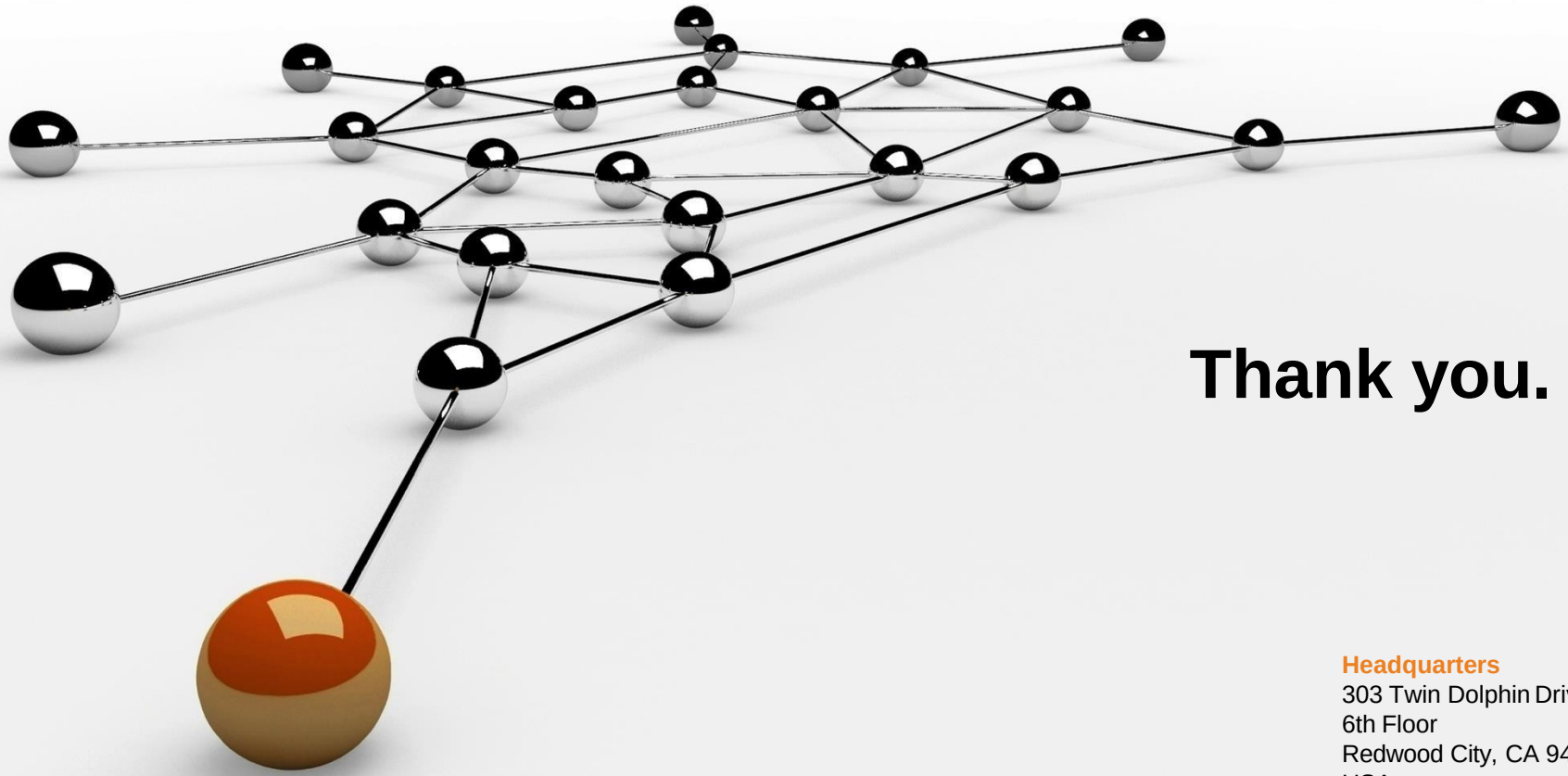
/etc/init.d/rcS startup script

```
#!/bin/sh
 mount -t proc none /proc
 mount -t sysfs none /sys
 ifconfig eth0 172.20.0.2
 route add default gw 172.20.0.1
 /usr/sbin/httpd -h /www/ &
 /bin/sh
```

See how simple this can be!

Aricent®

Engineering excellence. Sourced.



Thank you.

Headquarters
303 Twin Dolphin Drive
6th Floor
Redwood City, CA 94065
USA
Tel: +1 650 632 4310
Fax: +1 650 551 9901

www.aricent.com