

# Unix System Administration

**Chris Schenk**

Lecture 06 – Thursday Jan 31

CSCI 4113, Spring 2008

# Bash Scripting

- Like nothing you may have done before
  - Even Perl, Python, PHP are more organized
- Bash == Bourne Again SHell
  - Rewritten (but backwards compatible) from Bourne Shell (sh)
- Support for looping, functions, variables, arrays
  - And a host of other strange things
- Used all over your machine
  - Every init.d script is a bash/sh script

# Shell Variables

- Some examples
  - `$PS1` holds your prompt
  - `$PATH` holds your path directories
- We can use variables in scripts
  - And they remain local to scripts
  - Although I get confused with 'exported' variables
- Set a variable in bash:
  - `% myvar="some string of stuff"`

# If Statements

- One of the more cracked out ways I've seen
  - “IF”s are always terminated with “FI”s
- The conditionals are even stranger
  - `[ -f file ] || exit 1`
  - What does it mean?! -- return values!
  - `test` is an alternative
- `if [ "$UID" -ne "0" ]; then echo "You are not root"; fi`
- There are **MANY** types of conditional tests
- They are really picky about spaces and quotes
  - Not like your average language

# Case Statements

- Should be familiar from any C programming

```
- case "$VAR" in
    start)
        <cmd>
        <cmd>
        ;;
    *)
        <cmd>
        ;;
esac
```

- Used in every single `/etc/init.d` script
  - \* is equivalent to 'default'

# Init Script Rules

- There are guidelines for writing init.d scripts
  - Good examples exist in `/etc/init.d` everywhere!
- Start, stop, restart, status, usage (default)
  - These are the basics for startup and shutdown
- Also useful to use the logging library
  - Just another script that has functions
  - `/lib/lsb/init-functions` on Ubuntu
  - Logs things to `syslog` and has nice output on screen

# An IPTables Init Script

- IPTables has no configuration on bootup
  - Why not simply have it load rules from file on boot?
  - Allows us to add configurations in a file instead of with command-line tools
- IPTables does not clear rules with a single command
  - Default policies
  - Extra user-defined chains

# Symbolic (and not) Links

- A smart version of a windows shortcut
  - Can perform actions on links as if it were the file
- Create a hard link:
  - % In <source file> <target link>
  - Looks like a regular file!
- Create a symbolic link:
  - % In -s <source file> <target link>
- Some commands will follow links automatically
  - Some won't, have to specify extra options

# Important Notes on Symlinks

- Work across partitions
  - Have their own INODE that points to another directory entry
  - Hard links directly reference INODEs specific to that partition
- Can point to directories
  - Hard links only link files
- Are less efficient than hard links
  - Extra de-reference to reach the actual file
- Can point to a non-existent file

# SETUID and SETGID

- SETUID – program runs as user who owns file
- SETGID – program runs as group who owns file
  - One or the other (or both) can be set
- Only works on the actual executable running
  - Doesn't work in scripts!
- Real vs Effective user/group
  - Real remains the same
  - Effective is what permissions you have

# Devices

- /dev directory describes all available devices
  - /dev/cdrom, /dev/sda\*
- Different device types
  - block 'b', character stream 'c', pipe 'p'
  - TTYs, sound card, hard drives, cd drives
- Filesystems use block devices
  - /dev/hda, /dev/sda, /dev/scd0
- All devices are exposed as files!
  - Makes interacting with them much easier

# Disk Devices

- Labeling differs for IDE and SCSI devices
- IDE uses `/dev/hd*`
  - Primary IDE channel 1 - `/dev/hda`
  - Primary IDE channel 2 - `/dev/hdc`
    - Many times this is your CDROM
    - `lrwxrwxrwx 1 root root 3 2007-01-21 16:18 /dev/cdrom -> hdc`
- SCSI uses `/dev/sd*`
  - Based on SCSI channel ID
  - SCSI channel 1 - `/dev/sda`
  - SCSI channel 2 - `/dev/sdb`

# Partitions

- A device is divided into physical extents
- Physical extents are raw
  - Filesystems built on top of partitions!
  - Use block devices under /dev
- Numbers are placed after device IDs
  - First partition on IDE device /dev/hda is /dev/hda1
  - Second partition on SCSI device /dev/sdb is /dev/sda2

# Partition Table Editors

- Most installations have automatic partitioners
  - And even if you ask to manually partition, there's a gui involved
- The old-school command line tool is `fdisk`
- usage: `fdisk <device>`
  - `fdisk /dev/sda`
  - 'p' – print partition table
  - 'n' – add new partition
  - 't' – change partition's system ID
  - 'w' – write changes to disk

# Partition Table

- Same sector as the Master Boot Record (MBR)
- 64-byte data structure, each entry 16 bytes long
- Maximum of 4 partitions!
  - Extra partitions handled by the 'extended' type
  - Created as a solution for the 4 partition limit
- 10 fields in each entry in the table
  - Bootable flag
  - starting/ending head/sector/cylinder
  - System ID

# System ID

- Describes the filesystem on each partition
  - independent to the OS, standard format
  - **one** bytes long, 256 total entries available
  - 0x06 (FAT16), 0x07 (NTFS), 0x05 (Extended), 0x82 (linux)
- Read by the OS at boot time
  - need to know which modules to load

# Filesystems

- Partitions must be formatted to a specific filesystem
  - `mkfs` utility
  - different versions, `mkfs.ext3`, `mkfs.reiser`, etc
- Each filesystem requires a module in the kernel
  - `lsmod` (list modules) utility
  - 'used by' number of devices

# Mounts

- Filesystem starts at / (root)
  - not to be confused with root user
  - tree structure starting a /, branching with directories
- A partition may be mounted to any point on the tree
  - partitions always mounted to directories:
    - /dev/sda1 - /
    - /dev/sda2 - /usr
    - /dev/sda5 - /var

# Mounts (cont)

- How do you see your current mounts?
  - `mount` command
  - `df` command (with `-h` parameter)
- `mount` shows options on the filesystem
  - `[no]exec`, `[no]atime`, `defaults`, `nosuid`, `ro/rw`
- Partitions mounted at boot time read from a config file
  - `/etc/fstab`
  - Manually mounted partitions looked up in `fstab`
    - manual override of options

# /etc/fstab

- Six fields for each line
  - device or label, mount point, filesystem type, options, dump enable, fsck priority
- Some devices are marked 'none'
  - filesystems used by the kernel for special tasks
  - process information, device addresses
- CDRoms list 'auto' for filesystem type
  - three types of cdrom filesystems: IS9660, Joilet, HFS
  - Data is the same, but the filesystem metadata is different

# More on mount

- Manual mounting
  - `mount [-o <options>] <device> <mount point>`
- For entries in `/etc/fstab` only the device or mount point is needed
  - `mount /windows`
- Shows more information than simple devices?
  - NFS mounts

# Mount Options

- `defaults` contains a number of options
  - `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, `async`
- `rw/ro` – Mount read/write, readonly
- `[no]exec` – Allow/disallow execution of binaries
- `[no]suid` – Allow/disallow setuid/setgid bits to work
- `[no]user` – Allow/disallow a user to mount the filesystem

# umount

- Used to unmount a filesystem
- Almost identical in usage
  - `umount <device | mount point>`
- Problems may arise
  - `umount: /home: device is busy`
  - no file handles can be open if the device is to be unmounted
  - sitting in a directory on a device means files are open!
    - Your shell has handles open