

Unix System Administration

Chris Schenk

Lecture 03 – Tuesday Jan 11

CSCI 4113, Spring 2008

A few notes

- The 'root' user
 - An actual user on your machine
- Troubles with ssh'ing to your machine
 - Network is misconfigured, stop by office hours
- Startup scripts in /etc/init.d
 - Chapter 2 of the book talks all about booting
- Under Ubuntu - /etc/inittab replaced with /etc/event.d
- Lecture slides can overlap

A few more shell tricks

- In addition to '!!' and !cmd
- Make an error and can't backspace?
 - First option: set terminal to erase on backspace
 - `% stty erase <hit backspace key>`
 - All successive backspaces will properly erase
 - Second option: erase the input buffer
 - `<ctrl>-u`
 - Even works on mis-typed passwords!
 - Third option: erase a single character
 - `<ctrl>-h`
- There are plenty more, but we'll get them slowly

VIM Quick Notes

- Edit a file
 - `% vim mycooolfile.txt`
- Once in, you are in 'nav' mode or something
- You can enter 'insert' mode with 'i' or insert key
 - Get out of insert mode by hitting escape
- Quit without saving! `:q!`
 - Saves your butt more than once
- Write file `:w`
- Force write of file `:w!`
- Write and quit `:wq`

Shells and Environments

- Look at available shells – `/etc/shells`
- Simply programs executed by the OS
 - Starts with `mingetty`, spawned by `init`
 - Executes `/bin/login`
 - `/bin/login` checks password and executes shell listed in `/etc/passwd`
 - Executed as a **login** shell
 - Privileges changed to your user
- Variables are created
 - `printenv` and `$echo $VARIABLE` to see them

Shells and Environments (cont)

- `$PAGER` is like `$EDITOR`
 - `man` invokes `$PAGER`
 - `update-alternatives` changes both
- Forgot to type `sudo` before your command?
 - `% sudo !!`
- Want to execute the last 'vim' command?
 - `% !vim`
- `Screen` makes your life easier
 - `% screen`
 - `<ctrl>-a, d` and `screen -r`

Filesystems

- Devices
 - all behave like files, permissions like any other file
- Partitions and Types
 - physical separation of hard drive space
 - fdisk - ext2, ext3, vfat, swap, reiserfs, xfs
- Mounts
 - Access to partitions on the system
 - /etc/fstab
- Features
 - File types, modes, ownership

Features

- What most users see
 - I own my files (check with `stat` or `ls -l`)
 - Type: **-** (file), **b** (block dev), **c** (char dev), **d** (dir), **p** (pipe), **l** (symlink), **s** (socket)
 - **Read/Write/eXecute** (rwx), **user**, **group** **other**
 - Size, Timestamps
 - Name!
- What users don't usually see
 - Reference count, INODE number
 - Access and Change time

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

Hard Drive Devices

- Labeling differs for IDE and SCSI devices
- IDE uses /dev/hd*
 - Primary IDE channel 1 - /dev/hda
 - Primary IDE channel 2 - /dev/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!
- 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

fdisk – Partition Table Manipulator

- usage: `fdisk <device>`
 - `fdisk /dev/sda`
 - 'm' – print help
 - '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!

- 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

lsof – List Open Files

- Great utility for debugging many types of problems
- Everything under linux is a file
 - so we can see almost anything with lsof!
- Usage
 - Open files: `lsof <file>`
 - Open ports: `lsof -i tcp:<port>`
 - Processes: `lsof -p <pid>`