Systems and Network Management

- Manage computers, networks, and people
  - A job that requires you to act as a gopher between programmers and systems, systems and management, and programmers and management
- One of the most essential jobs in any modern organization, but also one of the most thankless
  - System administrator gets blame for everything, from programmer problems to hardware failure
  - Must anticipate the problems before they occur
- Principles to make a system administrator's job more manageable
  - Simplicity – Smallest solution for the problem at hand
  - Clarity – Easily understood
  - Generality – Applicable to many problems; prefer to use open standard protocols rather than proprietary solutions
  - Automation – Eliminate repetitive tasks
  - Communication – Management, customer, programmer and at a level understood by the target audience

History of Unix

- Started in 1969 at AT&T Bell Labs
- Made available to academic users for no cost (and no support) in 1976, and for $21,000 to corporations (with source code)
- Transferred to Unix Support Group and later, to Unix System Laboratories to create the commercial product
- A parallel effort, starting with the source provided by Bell Labs, created BSD Unix under the leadership of Bob Fabry at University of California at Berkeley (Computer Systems Research Group), with support from DARPA
- Notable contributions of Sys V
  - Unix itself, Bourne shell, Documenters' workbench
- Notable contributions of BSD
  - csh, job control, virtual memory, TCP/IP networking
- CSRG created a long-term goal of removing AT&T code from BSD but ran out of funding before the work was completed
- Final collection of AT&T-free code was released as 4.4BSD-lite in 1994
- Both BSD and Sys V flavors of Unix have had an effect on each other, with innovations from one absorbed into another, primarily because of vendors
  - Vendors typically started with one or the other flavor and *enhanced* it
  - In 1988, Sun and AT&T collaborated to jointly develop future versions of Sys V, resulting in Solaris
  - Other companies (IBM, DEC, and HP) formed a consortium to develop OSF/1 which is more important as a standards definition
- Linux appeared in 1991
  - Started as a personal project of Linus Torvalds

  - Started from scratch and does not trace ancestry to either of the two main branches of Unix, though it is more influenced by BSD

- Question: *Is Unix standard?*
  - No, especially if you are a system administrator
  - Systems are usually *enhanced* by vendors *by adding extensions* to the standard
  - None of the existing standards – POSIX, SVID, OSF's Application Environment Specification (AES), XPG4 – talk about system administration
  - Spend time to understand the philosophy rather than specific commands on different versions because philosophy is the same across different versions of Unix

Documentation – RTFM

- Start with the man pages
- Know your man page organization – man page sections
- `catman` to keep formatted copy of man pages (if space available)
- `catman -w` to enable keyword-based search of man pages
- Newer systems and Linux prefer `info` instead of man pages

Tasks of systems administrator – Take care of computers, their users, and management

- Keep systems up and running
  - Especially important when power fails at 2:00am and a project has a deadline for the following morning
- User management – Adding and removing users
  - Assign user's home directory
  - Decide the access for machines
  - Assisting new users to familiarize them with the environment
- Add/remove hardware
  - Make sure that the hardware being ordered works with the existing machines
  - Configure systems to recognize hardware
  - Managing disk space (keeping tabs on people who hog all the disk)
- Backups
  - Almost the most important chore
  - Can be easily automated, depending on the available hardware
  - Develop a policy on backups, as well as to restore files when needed
- Installing and upgrading software
  - Make sure that new software is installed and available to users who need it
  - Local software installed so that OS upgrade preserves it
  - Develop a policy regarding older versions of software (whether they should be removed or supported)
  - Make sure that changes can be easily undone

        * Follow the philosophy of modifying startup files (.login/.cshrc/.profile)

- System monitoring
  - Make sure that network services work correctly
  - Availability of systems resources such as disk space
  - Watch out for users who may consume too much of system time, especially if this is due to an error
  - Monitor log file for early signs of trouble
- Troubleshooting
  - Develop a policy to fix problems (won't work though)
    * A "trouble-ticket" system is the least you can do
    * "Trouble-ticket" helps you track the jobs on hand, whether they are being worked on, and let you show the boss the amount of time you spent on fixing a problem
    * This system leaves a trail that can be followed to fix similar problems in future
    * `bugzilla`
  - Fix network glitches
  - Fix both hardware and software problems
- Documentation
  - Document aspects of system that have been customized
  - Software installed that did not come with OS
  - Maintenance records for both hardware and software
- Security audit
  - Develop a security policy and check that it is not violated
  - Logs and monitoring, again
  - May have to build traps and auditing programs
- Help users
  - Most visible aspect of system administrator's job
  - Could be anything from a simple problem (think secretary) to one coming from serious programmer (my code suddenly stopped working; what did you change; how do I fix it)
- Performance tuning
  - Make the system perform better under the conditions prevalent in the organization
  - Make sure that if a user has large data to be processed, his data is available on the machine locally
- Capacity planning
  - Identify the need for new hardware and peripherals based on the tasks to be performed
  - Do not forget about space and furniture

And do everything above with a smile ...

- Sys admins have to wear many hats
- Must balance the needs and desires of all users and interface well with the management

- People think about a sys admin only when things do not work
- Try to automate as much as possible through scripts

Ethics and responsibility

- Balance authority and availability
- When do you kill processes? Or block users?
  - Can you block users from accessing the system in a corporation?

Booting and Shutting Down

- Must follow a process to bootstrap a Unix box, or to shut it down
- Booting process is hardware dependent; different process for different flavors of Unix

Bootstrapping

- System has to pull itself up by its own bootstraps
- Kernel is loaded into memory and begins to execute (spontaneous processes)
- Perform a number of initialization tasks before making the system available to users
- Problems during bootstrapping
  - Errors in configuration files
  - Missing/unreliable equipment
  - Damaged filesystems
- System is started by boot code stored in ROM
  - Loads and starts kernel
  - Kernel looks at system hardware and then starts `init` (always with PID 1)
  - `init` runs a series of scripts in sequence to check and mount filesystems, and to start system daemons
  - The scripts are referred to as `rc` files
- Automatic and manual booting
  - Automatic boot
    - System comes up on its own
    - Used almost exclusively for most systems
  - Manual boot
    - Systems follows automatic procedure to a point, and then turns control over to an operator, before initialization scripts are run
    - System in single user mode
    - I encounter this when there is a corrupted filesystem
- Steps in the boot process
  - Six phases
    1. Load and initialize kernel

   * Get kernel into memory
   * Typically in the root directory (`/vmunix` in SunOS), or in the directory called `/kernel` (`/kernel/genunix`) in Solaris
   * Kernel tests memory (amount available and other tests)
   * Kernel sets aside some memory for its internal data structures that are static
2. Hardware configuration
   * Locate and initialize each device known to kernel
   * May probe the bus for devices and ask appropriate drivers for information
   * Disable drivers for missing devices
3. Spontaneous processes
   * Created automatically without using `fork(2)`
   * On BSD (SunOS) we get

     ```
     0  swapper
     1  /sbin/init -
     2  pagedaemon
     ```

   * On System V-type machines (Solaris) we get

     ```
     0  sched
     1  /etc/init
     2  pageout
     3  fsflush
     ```

   * On Linux, no process with PID 0
   * Only `init` is a full-fledged user process; others are portions of kernel masquerading as processes for scheduling
   * Kernel's part of bootstrap is over; other processes (`logind` and other daemons) are yet to be started; they are ultimately started by `init`
4. Operator intervention, if needed
   * Need root password to achieve this, and you get a single-user shell
   * Usually, only the root partition is mounted
   * Only the commands available in `/bin`, `/sbin`, and `/etc` are allowed
     · On Solaris, even `/bin` may not be available as it is typically a symbolic link to `/usr/bin`, possibly on a different partition
   * No daemons
   * Filesystem root is mounted as read-only
     · You may have to remount root as read-write to run some commands that need to write on disk (such as `vi`)
   * Use `fsck` command to check and repair filesystems
   * Upon exit from single-user shell, system attempts to continue booting into multiuser mode
5. System startup scripts
   * Typically normal shell scripts
   * Selected and run by `init`
6. Multiuser mode
   * Start `getty` processes to listen in for logins
     · May start other processes such as `dtlogin` and `xdm` to achieve the same
   * On BSD, `init` has only two states – single-user and multi-user
   * On other machines, `init` may have several run levels to determine the resources made available
     · Look at `/etc/init.d` and `/etc/rc?.d` directories on Solaris

Booting PCs

- Difference between PC and proprietary hardware
  - On proprietary hardware, the initial bootstrap code is in firmware
    - Located in a place known to kernel which is a piece of software
    - Has basic information on talking to network and disk-based filesystems
  - On PCs, the same function is achieved by BIOS
    - not as sophisticated as firmware on Unix
    - Several levels of BIOS, for machine, video card, SCSI card, and so on
    - Know about devices on motherboard – IDE controller, keyboard, serial/parallel ports
    - Interactions and conflicts between different BIOSes
    - Confusing if BIOS has to choose the device to boot from
- PC boot process
  - Modern BIOSes allow a user to enter configuration mode during bootstrapping process
  - Still, a fairly rigid process allowing for little leeway to user
    - Boot from floppy. CD-ROM, and disk, in that order
    - Also limited to booting from the first device on the IDE drives
  - Master boot record
    - First 512 bytes on the device
    - Loaded in the first phase of boot process
    - Tells the computer the partition to find the secondary boot program, or boot loader
    - Boot loader responsible to load the kernel
  - LILO –Linux Boot Loader
    - Installed in the MBR of the disk or boot record of the linux root partition
    - Configured and installed with the `lilo` command, using `/etc/lilo.conf` file
    - Must be reconfigured every time you add a new boot partition, and every time you have a new kernel to boot
  - Configuring LILO
    - Basic `lilo.conf` file for Linux kernel

```
boot=/dev/hda        # Put boot loader on MBR
root=/dev/hda1       # Location of root
install=/boo/boot.b
map=/boot/map
delay=20             # 20 tenths of a sec for user interrupt
image=/vmlinuz       # Kernel to boot
    label=linux      # Label for the entry
    read-only
image=/vmlinuz-backup # Backup entry
    label=backup
    read-only
```

    - First label is the default
    - Initially, the partition is mounted read-only but remounted by initialization scripts as read-write
    - Running `lilo` without any arguments generates and installs the boot loader
    - Reboot only after a successful run of `lilo`
    - `lilo -t` will test the configuration without really installing it
    - Pressing `<TAB>` key will pause `lilo` to allow you to select an alternate partition
  - FreeBSD boot loader – Reading assignment

  - Multibooting on PCs
    * Only one MBR but possible to have multiple secondary boot loaders – one per partition
    * Decide on which boot loader is the master
      · Typically, Linux is a better choice for LILO
      · Win NT/2000 may need its own record in the MBR
    * Always install a consumer version of Windows before anything else
      · Consumer versions always want to take partition 1 on the first hard disk, and overwrite other boot loaders during installation
    * Same rule applies to Win NT/2000 as well
      · NT/2000 boot loader wants to be installed in MBR
    * Other installation details are left as reading assignment

Booting in single-user mode

- Solaris single-user mode
  - Boot procedure can be interrupted by simultaneously pressing the `STOP` key and the key `a`
  - Type `boot -s` for single-user mode
  - Must enter a full device location to boot from an alternate kernel
    * Locate the device containing kernel (use `df` command, or check in filesystem table)
    * Follow the links to get absolute path name with no links
    * Use the absolute pathname
    * Example: To boot my default kernel
      `boot /devices/pci@1f,0/pci@1,1/ide@3/dad@0,0:a/kernel/genunix`
- Linux single-user mode
  - At the LILO prompt, enter the label of configuration followed by `-s` or `single`
  - You can modify the root device or `init` with boot time options
  - Tries to `fsck` and mount every filesystem before entering single-user mode, with none of the commands being statically linked
    * If shared libraries are not mounted, dynamically linked commands cannot run

Startup scripts

- After exit from single-user mode, system startup scripts are run
- Interpreted by Bourne shell
- Naming scheme
  - BSD systems
    * Reside in `/etc` and have names starting with `rc`
  - SYS V Systems
    * Scripts kept in `/etc/init.d` directory
    * Links in directories `/etc/rc0.d`, `/etc/rc1.d`, ...
    * Cleaner organization
    * Each script can be used to start or stop a service, allowing for orderly shutdown

- Tasks performed in startup scripts
  - Setting name of computer
  - Setting time zone
  - `fsck`
  - Mounting disks
  - Reinitializing `/tmp`
  - Configuring network interfaces
  - Starting up daemons and natwork services
- SYS V style startup scripts
  - Seven run levels to define the services being run by the system
  - Level 0 – Shutdown
  - Level 1 – Single user mode
  - Level 2 through 5 – Multiuser modes
    - Level 2 or level 3 are the most common levels
    - Level 4 and level 5 are rarely used
  - Level 6 – Reboot level
  - Job of each run level defined by the file `/etc/inittab`
  - `init` takes the machine from run level 0 to default run level, traversing levels in sequence; and reverses when shutting down
  - Characters S and K indicate start or kill

Rebooting and shutting down

- What is the big deal about shutting down?
  - Filesystems buffer changes only in memory and is periodically written to disk
  - Disk I/O is fast but filesystem can be left in an undefined state if machine is not properly halted
- Rebooting is not an [attractive] option
- Turning off the power
  - Not a good way to shut down
  - Can potentially lose data and leave system files in inconsistent state
- `shutdown`
  - Safest way to initiate a halt or return to single user mode
  - Sends message to logged in users, warning them of impending halt
  - Disables user logins
- `halt`
  - Simpler way to shut down `shutdown -h`
  - Kills nonessential processes, executes `sync`, waits for filesystem writes to complete, and then halts kernel
- `reboot`
  - Identical to `halt`

  - `shutdown -r`
  - Reboots the machine from scratch
- Sending `init` a `TERM` signal
  - `sync; sync; kill -TERM 1`
- `telinit`
  - Change the run-level of `init`
- Killing `init`