CE350
Lecture10

SYSTEM ADMINISTRATION

by İlker Korkmaz and Kaya Oğuz

References

- The contents of this lecture are prepared with the help of and based on:
  - the textbook, UNIX Shells by Example (Chapter16)
  - the tutorial, Advanced Bash-Scripting Guide at http://tldp.org/LDP/abs/html/
  - the tutorial, The Linux System Administrator's Guide at http://tldp.org/LDP/sag/html/index.html
    - Linux System Administration, by Tom Adelstein & Bill Lubanovic, an O'Reilly book.
    - Wikipedia, the free encyclopedia

Contents of Lecture10

- Superuser term, setuid programs
- Kernel, init, run level terms
- User administration
- Installing/Removing software
- Backing up system
- Managing disk space
- Managing processes
- System monitoring
- Major services:
  - init, getty, syslog,
  - X Window system
  - networking nfs, samba
  - network login
- cron

Why to cover a lecture on system administration issues?

- You need to understand the basics of system administration at least for the reasons that either you will work as an **admin**istrator of an active multi-user system or you will only manage your system with privileged permissions.

- This lecture is not a whole guide to system administration, rather only an introduction to system administration in respect to the Linux shell.

Superuser

- The terms **superuser** and **root** are often used for the same meanings.
- Regular accounts have access only to the files and processes they own or that give them specific permissions, such as groups and other. Superuser accounts have access to all the files and processes on the system.
- The superuser is omnipotent and has no restriction.
- A superuser's user identification number (uid) is 0 and related shell prompt is a pound sign (#).

Becoming a superuser

- Traditionally, when an administrator wants to run any command requiring root privileges, he/she logs onto an unprivileged account first, and then uses the **su** command to switch to a superuser. A new root-owned shell will be started, and after running privileged commands, the superuser will exit from the root shell and return to the unprivileged user.
- **sudo** may be used in scripts to run a particular command as root (or another user).

System administrator responsibility on scripts

- If you are an **admin**istrator of a system, or you have the superuser rights test your any script carefully before taking it to the system level. An unexpected error could bring a whole system to its knees.

Running scripts as root

- Root's PATH does not include “.” (dot).
  - # myScript
    - # refers to root's prompt sign at command line.
    - The above command fails for a root.
    - Correct usage alternatives are as follows:
      - # ./myScript
      - # /bin/bash myScript

Scripts that run as root (setuid programs)

- Whoever runs the setuid program temporarily becomes the owner of that program and has the same permissions as the owner.
- To identify a setuid program, try **ls -l**
  - The **x** permission will be replaced with an **s** if the program is a setuid program.
- **passwd** is a good example to explain the illusion:
  - When you change your password you temporarily switch to root permissions, but only during the execution of passwd program. By the way, you are able to change the password without going to a system administrator.

System administrator responsibility on setuid scripts

- A shell script may be written as a setuid program and it is a serious security threat if not monitored by the administrator.
- If a shell script is a setuid program, the shell spawned is a root owned shell and commands executed from the script are run as root.
- Bash does not support setuid programs, Korn shell has a privileged mode only if allowed by the OS.

Required terminology

- Understand the following terms:
  - kernel
  - init
  - process
  - PID
  - daemon
  - initialization (init) scripts
- When the system boots the UNIX/Linux kernel is loaded from disk. It is the program that manages the OS from boot-up to shut-down. Firstly, it initializes device drivers, starts the swapper, and mounts the root filesystem (/). Then, it creates the first process init.

Run level

- A run level, also called system state, determines which set of processes are available on the system.
- Usually there are 8 run levels:
  - 0-6 and s (or S)
- Those run levels are categorized in three main states:
  - halted
    - run level 0
    - UNIX is not running in this state
  - single-user
    - run level is either 1 or S
    - only root is logged on
  - multiuser
    - run levels 2-5
    - multiuser mode allows users to log into the system
- The command “who -r” shows the run level of current init process.
- The command “init 0” tries to move the system into run level 0, which is a halted state, and so that command tries to shut the system down.

User administration

- Account: http://tldp.org/LDP/sag/html/account.html
- Users and groups: http://tldp.org/LDP/abs/html/system.html

Installing software

- To install a software from scratch:
  - build from source code:
    - First get the source files, then usually:
      - (in the corresponding directory)
      - $ ./configure
      - $ make
      - $ make install
- An example guide to write a script to install programs after a fresh install of Ubuntu Linux: http://ubuntuforums.org/showthread.php?t=290764

Removing software

- Usually, users do not remove a software, upgrade instead.
- If disk space issue becomes problematic, to remove some unused softwares is also considered. Then,
  - usually the related interactive system management package of the system is used, such as **Synaptic Package Manager** in **Ubuntu**.
  - or the related commands are used to uninstall and remove the packages, such as “apt-get remove” command in Ubuntu.

Backing up system

- Types:
  - **full backup**
  - **differential backup**
    - includes changes since last full backup
  - **incremental backup**
    - includes changes since last backup
- The administrator should have a backup strategy
  - Which type on which schedule
  - When to backup
  - What to backup

Main backup tools

- tar
  - to archive files
- rsync
  - to backup to a remote server

Managing disk space

- To save disk space, archiving may be preferred.
- The administrator may need to monitor the disk usages of users
  - **df** shows the free space on a partition
  - **du** shows information about disk space used in a directory

Managing processes

- A process is a program running on the system.
- Process management
  - create/monitor/kill processes

Managing processes: creation

- The kernel identifies the processes with their id numbers, pid.
- A process has a user id, **uid**, and a group id, **gid**. A process has a **ppid** as the parent process id.
- The system starts with an **init** process with the id 1 at boot-up and other processes are descendants of pid 1.

Managing processes: monitor

- **ps** command gives the snapshot of the processes running on a system at that moment.
- **pstree** command also works similar to “ps -f”

Managing processes: kill

- A process can be sent a **signal**.
- **kill** command is used to send a signal to a process. (kill is not just used to terminate a process)
- **KILL** is one of the commonly used signals.

System monitoring

- The administrator needs to monitor the system. However, it is not so easy to clarify the events to be monitored. That's more likely to be handled through the know-how experiences.

- **Daemon-Monitoring Daemon** concept:
  - A **daemon** is mainly a process that runs in the background and provides some particular services.
  - A daemon-monitoring daemon (DMD) is a utility that watches the daemons (services) automatically to restart them when they fail.

Major Services

- Too many services are there...
  - Here in this lecture, we cover the following popular services:
    - init, getty, syslog,
    - X Window system
    - networking nfs, samba
    - network login

Boot scripts

- During a system boot, shell scripts are run to perform tasks such as starting daemons and network services, mounting disks, and so on. They may also be run during system shutdown. Those scripts are known as *boot scripts*, or in some documentations they may be called as startup and shutdown scripts, init scripts, or run control scripts.

init

- The kernel identifies the processes with their id numbers, pid.
- The system starts with an init process with the id 1 at boot-up and other processes are descendants of pid 1.

getty

- init forks a new process and uses an **exec** to run the getty program.
- getty waits for the username and then uses an exec to run the login function to read the password.
- tty refers to teletypewriter.
- http://www.ibiblio.org/pub/Linux/docs/LDP/system-admin-guide/translations/es/html/figuras/logins-via-terminals.png

network login: inetd

- There is a single process inetd that handles all network logins. When it notices an incoming network login (i.e., it notices that it gets a new virtual connection to some other computer), it starts a new process to handle that single login.
- http://tldp.org/LDP/sag/html/login-via-network.html

syslog

- The standard logging solution on UNIX/Linux systems.
- http://en.wikipedia.org/wiki/Syslog

X Window System

- commonly X or X11 refers to X Window System, which provides the framework of a GUI for networked computers.
- http://en.wikipedia.org/wiki/X_Window

NFS, Samba, NIS

- **NFS** is a network file system protocol, which provides the clients to access the files in the network.
- **Samba** is a popular free-software project to implement Microsoft file sharing on non-Microsoft systems. http://samba.org
- **NIS** (Network Information Service) is a service that provides an information database to all machines on the network.

Review on *daemon* term

- daemon word comes from Greek mythology.
- A daemon is a process that runs in the background and performs tasks on a user's behalf.
- A web server, such as **Apache**, is a daemon; it temporarily stays inactive until it is asked for a web page.
- The **cron** boot script (*crond* program) runs the cron daemon.

cron

- cron checks the cron instruction files (*crontab files*) once a minute to see if it is time to run any commands specified there.
- Creating a cron instruction file:
  - The cron daemon reads the crontab files that have been submitted by system users. Each line in the crontab file consists of six fields separated by spaces. The values in the first five fields tell the cron daemon when to run the command that is contained in the sixth field.