

CST8177 – Linux II

System Administration

Todd Kelley

kelleyt@algonquincollege.com

Today's Topics

- ▶ system administration
- ▶ user and group management

System Administration

- ▶ <http://www.gnu.org/fun/jokes/know.your.sysadmin.html>
- ▶ The system administrator role in a nutshell is to keep the system healthy and the users as productive as possible
- ▶ OK, what's a system? Examples:
 - multi-user Linux machine like our CLS, 245 users
 - multiple Linux workstations (lab in T127)
 - individual Linux workstations (primary user is a sysadmin too, they come to you for help)
 - Web Servers, Mail Servers, Document Servers...
- ▶ OK, what does it mean for a system to be healthy?

Healthy Multi-User System

- ▶ an account has been created for everyone who should have one (the users)
- ▶ every user is authorized to read, write, and execute exactly what they should be able to
 - not more
 - not less
- ▶ every user can access the resources they need
 - disk space
 - software applications/libraries
 - processes, memory, CPU time
 - resource hogs don't affect the work of other users

Healthy Multi-User System (cont'd)

- ▶ Accessible to its users
 - accessible remotely if applicable (ssh)
 - good uptime with reasonable maintenance windows
- ▶ Secure from attack
 - inaccessible to unauthorized users (external attack)
 - no unauthorized or stolen access to user accounts
 - resistant to internal attacks
 - users cannot elevate their privileges
 - users don't bring system down without trying
 - prevent cross-user attacks
 - ensure users cannot interfere with each other's
 - confidentiality of files
 - integrity of files
 - availability of files

Regular Maintenance

- ▶ backups
- ▶ security patches (software updates, below)
- ▶ monitor and manage disk space
 - find and educate and control "space hogs"
 - add new disk space
 - replace failed disk space
- ▶ software installation
- ▶ software updates
- ▶ system upgrades (preferably not often)
- ▶ monitor the system logs for issues

Red Hat Enterprise Linux

- ▶ Red Hat Enterprise Linux is the industry standard for production Linux servers
- ▶ Red Hat and Oracle are the two big companies offering Linux IT services
 - Red Hat uses RHEL (Red Hat Enterprise Linux)
 - Oracle uses RHEL (with Red Hat Trademarks removed)
 - CentOS is RHEL (with Red Hat trademarks removed)
- ▶ RHEL Versions have a lifespan on the order of 10 years
 - 4.x, 5.x, 6.x, (7.x is coming)
- ▶ Each version of RHEL contains a set of packages, each of a particular version

Software Update

- ▶ After a server is installed with a RHEL version, configured, tested, and put into production, you want to keep the individual packages up to date
- ▶ For our purposes, a Software Update is a new version of an individual package
 - new version of openssh, httpd, etc
 - it is important to install these
- ▶ Red Hat backports security fixes to packages
 - This means Red Hat provides software updates to put new security fixes in the old versions of packages in the old versions of RHEL

System Upgrade

- ▶ For our purposes, a System Upgrade is a move from a major RHEL version to a later one
- ▶ Later RHEL versions can be quite different from each other
 - different packages (sysVinit versus upstartd versus systemd)
 - different major revisions of packages (apache 1.x versus apache 2.x; etc)
- ▶ A System Upgrade typically involves a new installation, configuration, testing, deployment, switch over from old system to new system

System Upgrades

- ▶ The switch-over to a new version of a busy server system is a high-stress event that takes much preparation
- ▶ That's why the 10-year EOL cycle of RHEL is important
- ▶ You do not want to do major upgrades to a new system often
- ▶ You DO want to keep all of the individual packages in your current version up-to-date with security fixes.

CentOS VMs

- ▶ minimal install
 - when setting up a server, it's a "best practice" to start with the minimum and add only what you need
 - in our case, we will learn how to grow a system
 - add software
 - add disks
 - don't do any playing with or customization of or installations to your CentOS until you're told
- ▶ Feel free to play with Linux, Linux GUIs, etc
 - do it in a clone of your CentOS VM
 - do it in a VM you create for playing (what, you haven't already done this?)

Three types of account

- ▶ Root account
 - having a root password is not necessary
 - not having a root password means one less password to manage, one less vulnerability
 - root access is gained by system administrators
- ▶ System Administrator
 - configured in sudoers file
 - gain root privileges with `sudo -s` or `sudo -i`
- ▶ Regular User
 - often named according to a pattern
 - this is the kind of account you have on the CLS

Setting up root

- ▶ common model is to put sysadmins in sudoers file
- ▶ as root, do `visudo`
- ▶ put the following line in
 - `youradminname ALL=(ALL) ALL`
 - `youradminname`: the username you use for admin
 - `ALL`: from any host
 - `(ALL)`: run commands as any user
 - `ALL`: run any command
- ▶ test that you can become root with `sudo -s`
- ▶ put `*` in root password field in `/etc/shadow`

sudo refresher

- ▶ `sudo -s` gives you a shell as root
- ▶ this is not a login shell, it retains your old environment
- ▶ `sudo -i` gives you a root shell and simulates a full login
- ▶ a full login will leave you with root's path
- ▶ root's path will contain `/sbin`, `/usr/sbin`, etc, which are directories not normally needed in a regular user's path

User Management

- ▶ Create, Modify, and Remove User Accounts
- ▶ Create, Populate, Modify, and Remove Groups
- ▶ Password Policy
 - strength of passwords
 - how often passwords must be/can be changed
 - how often passwords can be reused (or based on an old password)
- ▶ Set and Administer File Permissions
- ▶ http://teaching.idallen.com/cst8207/13f/notes/700_users_and_groups.html

passwd command

- ▶ `man passwd`
- ▶ `passwd -l` : lock an account
- ▶ `passwd -u`: unlock an account
- ▶ `passwd -n`: min password lifetime in days
- ▶ `passwd -x`: max password lifetime in days
- ▶ `passwd -w`: number of days warning
- ▶ `passwd -i`: number of days after expiry to disable
- ▶ `passwd -S`: print a summary

creating users

- ▶ by default, useradd creates the new user's home directory
- ▶ the new home directory is populated with the contents of /etc/skel/
- ▶ shadow password suite configuration in /etc/login.defs
- ▶ the defaults for useradd are /etc/default/useradd

creating many new users (cont'd)

- ▶ to create one user:

```
useradd -c "Full Name" user001
```

```
chmod 750 /home/user001
```

```
passwd user001 # and enter passwd by hand
```

creating many new users (cont'd)

- ▶ there are various possible strategies for creating many new user accounts
- ▶ one possibility:
 - use Linux utilities and/or your own script to create a set of commands for each new user (one-off script):

```
useradd -c "User 1" user001          #create the user
usermod -p u75jjvrue5B92 user001  #assign passwd
chmod 750 /home/user001 || exit 1  #homedir perms
```
- ▶ If you were creating 100 users, you'd have 300 commands in your one-off script

creating many users (cont'd)

- ▶ another possibility: the `newusers` command
- ▶ `man newusers`
- ▶ `newusers` takes a file containing info about the accounts you want to create
- ▶ the input file for creating the accounts is in the same format as the `/etc/passwd` file:
uncle:3uncle4:503:503:Uncle Tom:/home/uncle:/bin/bash
aunt:3aunt4:504:504:Aunt Betty:/home/aunt:/bin/bash
- ▶ Manipulating a spreadsheet-based text file into a file for the `newusers` command is a good use of the `sed` editor (we'll need it for Assignment09)

Basic sed

- ▶ we'll use sed to read lines from stdin or a file, and write the modified lines to stdout
- ▶ we'll concentrate on the forms of the substitute (s) command
 - `sed 's/this/that/'`
 - replace first instance of `this` with `that`
 - `sed '/^#/s/this/that/'`
 - notice the regular expression in front of the s
 - on lines that begin with # replace first instance of `this` with `that`
 - `sed 's/this/that/g'`
 - replace all instances (on each line) of `this` with `that`
 - `sed -e 's/this/that/' -e 's/what/who/'`
 - replace first instance (on each line) of `this` with `that` and first instance of `what` with `who`

Sed examples

▶ `echo this | sed 's/this/that/'`
that

▶ `echo this and this | sed 's/this/that/'`
that and this

▶ `echo this and this | sed 's/this/that/g'`
that and that

▶ `echo this and what | sed -e 's/this/that/' -e 's/what/why/'`
that and why

▶ `echo this and that | sed -e 's/this/that/' -e 's/that/why/g'`
why and why

▶ **Copy file** `orig.txt` **to** `new.txt`, **changing all instances (on each line) of**
this to that

```
sed 's/this/that/g' orig.txt > new.txt
```

▶ **Copy file** `orig.txt` **to** `new.txt`, **changing first instance (on each line) of ::**
to :abc:

```
sed 's/::/:abc:/' orig.txt > new.txt
```

passwd command examples

- ▶ bad idea: set blank password for user
 - `passwd -d username` # shouldn't need to do this
 - sets blank password field in `/etc/shadow`
 - login still prompts for password, so you'd need to jump through hoops to allow for login with blank password
 - `su` will not prompt for passwd

passwd (cont'd)

- ▶ disable passwd authentication for username
 - `passwd -l username #` puts ! in front of passwd
 - `passwd -u username #` undoes the above
- ▶ a ! placed in front of the passwd entry of the shadow file ensures that nothing anybody can type will successfully match this passwd entry
- ▶ * in the passwd entry is similar, and used for accounts for which should never use passwd authentication
- ▶ SSH keys will still work without passwd

passwd example (cont'd)

- ▶ `passwd -n mindays`
- ▶ `passwd -x maxdays`
- ▶ `passwd -w warndays`
- ▶ `passwd -i expireaccountdays`
- ▶ example: allow changing password no more than once per day, force changing every 90 days, warning 10 days in advance of expiry, and if they don't change their password within 2 days after expiry, disable account (not even ssh key login will work):
- ▶ `passwd -n 1 -x 90 -w 10 -i 2 username`

force passwd change on login

- ▶ `chage -d 0 username`
- ▶ thereafter, the first time the user logs in, they will be forced to enter their password
- ▶ all the other aging parameters are unchanged (`maxdays`, `lastday`, `mindays`, etc)

Disable an account

- ▶ `usermod --lock --expiredate 1970-01-01 <username>`

Editing critical files

- ▶ Don't edit files when there's a command that updates the file
 - e.g. "usermod -c 'New User' newuser" instead of changing gecos field in /etc/passwd by hand
- ▶ If you must edit the file, don't edit it directly when there's a command for that purpose (vi will be the default editor):
 - visudo # edit the /etc/sudoers file
 - vipw # edit the /etc/passwd file
 - vigr # edit the /etc/group file
- ▶ normally can specify a different editor in EDITOR or VISUAL environment variables (see man)
- ▶ can set EDITOR or VISUAL in .bashrc, export them!
- ▶ Command line examples (either of these will work):

```
bash$ EDITOR=nano visudo # call visudo with EDITOR=nano
```

or

```
bash$ export EDITOR=nano
```

```
bash$ visudo
```

Group management

- ▶ http://teaching.idallen.com/cst8207/13f/notes/700_users_and_groups.html

group examples

- ▶ **add a group**

```
groupadd mygroup
```

- ▶ **make** `tgk` **and** `idallen` **the members of** `mygroup`

```
gpasswd -M tgk,idallen mygroup
```

- ▶ **add** `ian` **to** `mygroup`

```
gpasswd -a ian mygroup
```

- ▶ **make** `tgk` **the admin of** `mygroup`

```
gpasswd -A tgk mygroup
```

- ▶ **set a password on** `mygroup`

```
gpasswd mygroup
```