Linux for Systems Administrators

Expertise

- Linux user since 1998
- B.S. 2002
- Linux/Unix admin since 2004
- Unix Administrator @ UTK -- 2004
- M.S. 2007
- System Administrator @ ISU -- 2009
- RHCSA & RHCE -- 2013

Goals

- Linux Administration basics
- Command Line Toolset
- Remote Access
- Firewall Administration
- RHEL 6 vs RHEL 7
- Kickstart
- Ansible

Linux Administration Basics

- Why use Linux?
- Linux Distributions
- Skills required
- Skills acquired
- Installation and Configuring
- General Administration Issues

Why use Linux?

- Troubleshoot other computers
- Server stability and security
- Open environment:
- Home servers : media centers, routers, etc...
- Understand how computers function, not just how to accomplish certain tasks!

Distributions

- Red Hat / Fedora
  - In use at ISU
- Debian
  - and all the variants
    - Ubuntu
    - Mint
    - Raspbian (Linux microdevices)
- Arch / Gentoo
  - No handholding, bare bones

Skills Required

- Little-to-no knowledge required
- Installations:
  - Graphical, point-and-click applications.
  - typically take 10 - 30 minutes to complete on slow hardware
- Maintenance:
  - typical OS and software updates via package managers (apt, yum, yast, etc…)

Skills Acquired with use

- Deeper knowledge of:
  - Layer 2 and Layer 3 Networks (hardware and address encapsulation)
  - programming / scripting
  - computer hardware and OS abstraction layers
  - Virtual Machines
  - Systems administration
  - How computers (in general) function

Installing and Configuring

- Best to use a VM hypervisor until comfortable
  - VirtualBox: free, easy to use
- Download installation and/or live images
- Create VM
- Attach peripherals

General Administration Issues

- Authentication
- Authorization
  - Files and Permissions
  - ACLs and Extended ACLs
  - Access to certain directories
- Firewall
  - “Help, I can’t do *x* from *y*!!!!”
  - Is this by design, or is it a problem?
- Security Enhanced Linux contexts

Command Line Toolset

- Why?
- Common Commands
- Useful Programming Languages
- General rules of thumb

Why?

PROS

- Quicker
- More Utilities
- Better understanding of tools
- Command Line operations are standardized.
- Piping:
  - Output from one command becomes input for the next.
- System Automation

CONS

- Requires
  - more domain knowledge
  - more forethought about certain commands
- GUI is often easier to understand
- Can be much easier to cause unintended consequences

Common Commands

- File and Directories
- Remote Access
- Archival
- File Manipulation
- Other

File and Directory

- **cp**: copy
- **mv**: move or rename
- **rm**: delete
- **rmdir**: delete empty directories
- **mkdir**: create new directory
- **chown**: change ownership
- **chgrp**: change group ownership

Remote Access

- “Secure” commands: All run off the same engine
  - **ssh**: Secure Shell
  - **scp**: Secure Copy
  - **sftp**: Secure FTP
- **xfreerdp**: Remote Desktop application

Remote Access

- **DO NOT USE:**
  - telnet
  - rsh
  - rlogin
  - ftp
- These commands have no encryption!

Archival

- Multi-file archival
  - tar
  - zip
- Compression
  - zip
  - gzip
  - bzip2
  - xz

File Manipulation

- cat
- wc
- grep
- head / tail
- cut / join / sort / uniq
- tr

Other Useful commands

- rsync / wget / curl
- more / less
- diff / patch
- locate / slocate
- nano / vi / vim / emacs
- w / who / top / htop
- sudo

Other useful commands

- IP Tools and Utilities:
  - tcpdump
  - ss: (socket statistics, formerly netstat)
  - ip n: (ip neighbors, formerly arp)
  - ip a: (ip address, formerly ifconfig)
- crontab
- sar
- getfacl / setfacl

File Redirection - Output

- *command > outFile*
  - Run ***command***
  - Write the results into the file named ***outFile***
- *command >> outFile*
  - Run ***command***
  - Append the results into the file named ***outFile***

File Redirection - Input

- *command < inFile*
  - Run ***command***
  - Use the *inFile* file as input for ***command***
- *command << delimiter*
  - Read input until finding a line containing *delimiter*.

    All lines up until *delimiter* are fed into standard input of *command*

File Redirection - Pipes

- *command1* | *command2*
  - Run ***command1***
  - Use the output of ***command1*** as input for ***command2***
  - No limit to amount of chained commands

Redirection Examples (possibly contrived)

```
$> cat file1 > file2

$> mail bbritt@iastate.edu < to_do

$> ps -eaf | grep -v root | awk '{print $2, $6}' | egrep "(pts|tty)" | cut -d ' ' -f 1 |
xargs kill -9
```

Useful Programming Languages

In order from most to least useful:

- Bash
- Sed
- Awk
- Perl
- Python
- C[++]

Bash Scripting

- More memory intensive
- Slower
- More powerful:
  - programs are roughly equivalent to functions
- Can chain outputs

Example: sorting an array

C

```
#include <stdio.h>

void main()
{
   int i, j, a, n, *number;

   for (i = 0; i < n; ++i) scanf("%d", &number[i]);
   for (i = 0; i < n; ++i) {
      for (j = i + 1; j < n; ++j) {
         if (number[i] > number[j]) {
            a =  number[i];
            number[i] = number[j];
            number[j] = a;
         }
      }
   }
   for (i = 0; i < n; ++i) printf("%d\n", number[i]);
}
```

Bash

$> sort *filename*

or

```
$> sort << EOF
4
3
5
2
…
EOF
```

General Administration Precepts

- If you break it, learn how to fix it.
  - Then, when you fix it, document how you did it…
  - Log files will tell you exactly how something is broken.
  - If the log file doesn't tell you how something is broken, either
    - Turn up the debugging output, or
    - You aren't reading it correctly.
- Don't login as root unless it is necessary. It's best to not even have a root password.
- If you assume something is setup correctly, it isn't.
- RTFM

Remote Access

- SSH
- Remote Desktop

SSH

- How SSH works:
  - High security for initial communication, key exchange.
  - Negotiation about faster encryption protocol to use ensues.
  - Authentication
  - Access
- Authentication happens over an encrypted channel.

SSH

Usage:

- Standard connection:
  - ssh <username>@<computer>
- Can forward remote ports to local:
  - ssh -L5900:<remote>:5900 …
    - Secure VNC tunnel
- Can forward GUI connection
  - ssh -X [-Y] ...

SSH

Benefits:

- Encryption
- Security
- Port Tunneling
- SSH Keys (authentication without passwords)
    - Note, this does not create a kerberos ticket

Remote Desktop

- xfreerdp
  - provided by freerdp
- Usage:
  - xfreerdp -u <username> -d <domain> <computer>
- Very useful for connecting to Windows machines.

Firewall Administration

- iptables
- firewalld

IPTables

- historical firewall implementation
- Kernel-level firewall
- Concepts:
  - State-based
  - Packet Filtering
    - not application specific
  - IP-Aware

Basic Configuring

- /etc/sysconfig/iptables

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp  --dport 22  -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp  --dport 8080  --src 129.186.0.0/16 -j
ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp  --dport 8080  --src 10.0.0.0/8 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Firewalld

- IPTables with the concept of firewall “zones”
- (arguably) easier to configure than IPTables
- Rules can be assigned programmatically

Firewalld

Typical Usage:

- firewall-cmd <zone> <service> [--permanent]
- firewall-cmd <zone> <source> [--permanent]

Graphical Interfaces are available.

Firewall Best Practices

- Allow only those services through that should be allowed.
- Do not disable firewalls when testing new services; build them to work within constraints.
- When creating access rules, create them with minimal rights

RHEL 7
RHEL 6

- Differences
- Similarities
- Which to use?
- Administration Concepts

Differences

	RHEL 6	RHEL 7
Startup	/etc/init.d (init process)	systemd
Logging	syslogd	journald
Firewall	iptables	firewalld
AD Integration	samba	samba or realmd
Performance	---	tuned
VM Guest Integration	---	open-vm-tools

Systemd

- Suite of system commands
  - console
  - journal (syslog)
  - network
  - login
  - devices
- Handles cgroups
  - Kernel level resource accounting and management

Journald

A service that collects, stores, and indexes log information:

- Kernel logs
- Syslog
- Service output and error messages
- Audit logs (SELinux)

Similarities

- Log file location: **/var/log**
- Configuration file locations: **/etc**
- Remote access: **ssh**

Basically, all standard services not pertaining to system boot.

Which one do I use?

- RHEL 7.x
  - Ansible scripts configured to run with RHEL 7
  - Currently in “Production 1 Phase”
- RHEL 6.x
  - Ansible scripts configure to run with RHEL 6
  - Currently moving to “Production 3 Phase”
- RHEL 5.x
  - Extended Life phase

RHEL Production Phases

	Production 1	Production 2	Production 3	Extended
Self-Help	Yes	Yes	Yes	Yes
RHEL Support	Yes	Yes	Yes	limited
Critical Updates	Yes	Yes	by severity	No
Updated Install	Yes	Yes	No	No
Enhancements	Yes	Time dependent	No	No
Minor Updates	Yes	Yes	No	No

Administration Concepts

- Keep it simple
- Ask for help
- Less is more
- AD joined systems are recommended
- Firewalls are good, they should be used
- “Least Privilege”

Kickstart

- How to kickstart
- Satellite
- Maintaining Kickstart Files

How to Kickstart

- Need:
  - kickstart file
  - Static DNS assignment with MAC Address
  - install media

- Workflow:
  a. boot computer
  b. point it at the kickstart file
  c. wait

Kickstart File

- Easiest to generate through the satellite server.
- Mostly autogenerated for you
- Can customize certain aspects of your install:
  - Root password
  - Installed software

How to kickstart from satellite

1. Boot machine from CD
2. At install screen, hit <TAB>
3. Edit (‘e’ key) the boot line.
4. Append the following to the boot line:

ip=dhcp ks=http://titan.its.iastate.edu/ks/cfg/org/83/label/<ks-label>

Satellite

- Located at https://titan.its.iastate.edu
- Email las-server-tech@iastate.edu for access.

Kickstart Files

- Maintained via the Satellite server

Ansible

- Ansible Repository
- Using Ansible
- Examples

Ansible Repository

- https://git.its.iastate.edu/isuans/ansible.git

Using Ansible

- Documentation is:
  - In the repository, and
  - In the handbook.
- Documentation within the repository is almost always up to date.
- Documentation within the handbook may lag behind a bit.

Examples