CST8177 – Linux II

Services, logging, accounting

Todd Kelley
kelleyt@algonquincollege.com

Topics

- services
- syslog
- logger command line utility
- psacct
- lastcomm
- ac, last, lastlog

chkconfig vs service (review)

The **chkconfig** command allows us to manage the runlevels in which the services are started. It does not do anything to any running process, the daemons that actually provide the services.

To manage the service daemons themselves (as **root**, naturally), we use the **service** command.

```
service daemon command optional-options
```

It's done by running the script in **/etc/rc.d/init.d** with the matching name. For example, say you wanted to stop the **crond** daemon to end the **cron** service. The command is

```
service crond stop
```

What actually runs is this:

```
/etc/rc.d/init.d/crond stop
```

Inside the script

The *command* used in the general **service** command line (previous slide) is passed directly into the script for **crond**. Here's a snippet from inside the crond script ($1 means the first argument):

```
case "$1" in
    start)
        ...
    stop)
        ...
    restart)
        ...
    reload)
        ...
    status)
        ...
    *)
        echo $"Usage: ...
        exit 2
esac
```

System Services: examples

- We've seen several system services by now
- NTP service: keeps clock accurate
  - ntpd : the daemon
  - /etc/ntp.conf : the daemon config file
  - /etc/init.d/ntpd : the SysVinit script
  - ntpd daemon itself IS the client (of an ntp server)
  - ntpd logs to /var/log/messages

System Services: examples

- SSH service: logins with Putty.exe or ssh
  - sshd : the daemon (SSH server)
  - /etc/ssh/sshd_config : the daemon config file
  - /etc/init.d/sshd : the SysVinit script
  - client program: putty.exe, ssh, scp, and more
  - daemon logs its messages to /var/log/secure
  - client manpage: man ssh
  - daemon manpage: man sshd
  - daemon config file manpage: man 5 sshd_config
  - client config file manpage: man 5 ssh_config
- If we stop the sshd daemon, noone can log in over SSH

System Services: daemons

- daemon:
  - a program, launched at boot time usually
    - /etc/init.d/* : the SysVinit scripts for starting, stopping, restarting, getting status of daemons
  - doesn't terminate – it keeps providing the service
  - many daemons, but not all, take requests from a corresponding client program
  - examples:
    - sshd : secure shell daemon, serves the ssh client
    - httpd : web server daemon, serves web browser client
    - crond : cron daemon, no client
    - ntpd: a client that provides a system service and uses an ntp server

System Services: config files

- System Service Daemons have config files
- Normally maintained somewhere in /etc/
- Often end in ".conf"
- Default config file comes with install package
- System administrators customize them
- Config files control various parameters
  - some parameters apply to many services:
    - how should logging be done
    - what port should the service listen on
  - some are specific to an individual service:
    - ntpd: which ntp server should be used
    - sshd: which version(s) of SSH protocol to use

config files (cont'd)

- Often config files have many comments in them
- # is a common comment character
  - usually lines beginning with # are ignored
- Various possible options will be "commented out"
- Those options can be activated by "uncommenting" them

config files (cont'd)

- From /etc/ssh/sshd_config

```
# Logging
# obsoletes QuietMode and FascistLogging
SyslogFacility AUTH
#SyslogFacility AUTHPRIV
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
```

config files (cont'd)

To comment out this line

```
SyslogFacility AUTH
```

it becomes

```
#SyslogFacility AUTH
```

and to uncomment this line

```
#SyslogFacility AUTHPRIV
```

it becomes

```
SyslogFacility AUTHPRIV
```

System Services: clients

- So far in your work with computers you've seen more of the clients than the servers.
- Examples of clients you use all the time
  - ssh, putty.exe
  - Web Browser
  - Explorer (Windows Networking)
  - ftp
  - DHCP (example: network connection at Algonquin)
- Each of these clients requires a server to be running at the "other end"

System Services: logging

- Daemons need a place where they can send their output
- Most daemons use the rsyslog (syslog for short) logging service
- Centralized logs are easier to manage
- rsyslog is itself a daemon
  - daemon : rsyslogd
  - config file : rsyslog.conf
  - client program: logger command line utility, as well as all the other daemons
- daemons send their messages to syslog, and syslog puts those messages in an appropriate place, usually a file under `/var/log/`

Syslog config: /etc/rsyslog.conf

- `/etc/rsyslog.conf` contains a set of rules
- When a log message is sent from a daemon to syslog, the message is marked with a facility and a priority
- The facility is an indictor of what kind of message it is
- The priority is an indicator of how urgent the message is
- `/etc/rsyslog.conf` determines what actions are taken for different priorities of the messages of different facilities

Logging Rules

Each rule takes a single line, with a selector on the left and an action on the right, separated by white space (blanks and tabs).

The selector is in two parts, the facility and the priority, joined by a dot.

selector
        action

facility.priority
        action

There are a whole host of possible actions, from the typical one of "put the message into this log file" to "tell everyone logged-in at once!!!" and lots in-between including email.

Syslog Facilities

- auth : authorization
- authpriv : private authorization
- cron : crond
- daemon : other misc daemons
- kern : the kernel
- lpr : printer
- mail : email
- news : usenet news
- syslog : syslog itself
- user : user messages
- uucp : unix to unix copy
- local0 through local7 : local use
- * : all of the above

Syslog Priorities

- The priorities are the level of importance of the message, and are fixed. They are listed here in ascending order, so **debug** has the lowest priority and **emerg** the highest. This is important, since messages are sent to a range of priorities.
- Messages are logged according to the rules for the specified priority and all lower priorities (higher in the list). A **crit** error (for example) is also logged according to any rules for **err**, **warning**, **notice**, **info**, and **debug**.

Syslog Priorities

- In order of increasing priority:
  - `debug` : debug level messages
  - `info` : normal information
  - `notice` : normal but significant, unusual
  - `warning` : not an error, but action should be taken
  - `err` : error condition, non-urgent failure
  - `crit` : critical condition, failure in primary system
  - `alert` : action needed immediately
  - `emerg` : panic, system unusable, notify all
  - `*` : all of the above
  - `none`: none of the above for the given facility

Actions

- The action part of a logging rule specifies what to do when rsyslog receives a message of that priority for that facility
- A common action is to put the message in a file, and this action would be specified as an absolute pathname to a file

Syslog actions

- Absolute pathname of a file
  - put the message (log entry) in that file
  - dash in front means omit syncing on every entry
- Terminal or Console : ex `/dev/console`
  - write the message on that screen
- `@hostname` : remote machine
  - send the message to syslog on the remote machine
- username
  - write the message on that user's terminal
- `*` : everyone logged in
  - write the message on everyone's screen
- named pipe (fifo) : useful for debugging

More detail on Selectors

A selector is a facility, a dot, and a priority level.

A simple selector might be **mail.info** to log **mail** issues at **info** and **debug** priorities to whatever action is defined.

Selectors can be combined in certain ways. For example, **news,mail.info** logs both **news** and **mail** issues. ***.emerg** will handle all facilities **emerg** priority messages the same way. And **news.warning;mail.err** will use the same action for each as if it were a separate selector.

,	combine facilities
***.**	all facilities
.*	all priorities
;	combine selectors
.none	no priority
=priority	only this priority
!priority	not this priority, only higher

rsyslog.conf examples

- Here is the rule that says for all facilities except mail, authpriv, and cron, log messages of priority info and higher to /var/log/messages

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none       /var/log/messages
```

- Here is the rule that says for all facilities, if the priority is emerg, write the message on all screens of logged-in users (the * action) (the above rule will also apply)

```
# Everybody gets emergency messages
*.emerg                                        *
```

- authpriv messages of all priorities go into /var/log/secure

```
# The authpriv file has restricted access.
authpriv.*                                     /var/log/secure
```

logger command

- logger: the command line utility for putting an entry in the logs
- can be used to log the activity of scripts

```
logger [-is] [-f file] [-p pri] [-t tag] [message ...]
```

Examples:

```
logger -p user.info -t logger "this is a test"


logger -p authpriv.info -t kelleyt "another test"
```

logwatch

- With all this logging information being recorded, a sysadmin should be monitoring it
- You would think someone would have written a script to "grep" through logs, or summarize them
- Yes, they have!
- yum install logwatch
- logwatch is run daily:
  - /etc/cron.daily/0logwatch

Daily sysadmin tasks (cron revisited)

- /etc/crontab is the main system crontab
- On CentOS 6.5, we can see it's configured to run hourly, daily, weekly, and monthly jobs
- To do a system admin task daily, for example, put a script that does the job into the `/etc/cron.daily` directory

- Similarly for hourly, weekly, monthly jobs
- `/etc/crontab` can be considered "root's" crontab file, but notice the "userid" field – the job will run as "userid"

logwatch

- On our CentOS 6.5 minimal systems, logwatch is not installed by default
- After it's installed there is a link to a perl script in `/etc/cron.daily`
- By default, it emails a summary of the logs to root, with "low" detail
- We put custom configuration in

`/etc/logwatch/conf/logwatch.conf`

logwatch.conf

- Examples:

```
MailTo = root          # email sent to root
Detail = Low            # low detail in the summary
```

- Detail can be specified as a number 0 to 10
- Another example

```
MailTo = tgk00001    # email my sysadmin user
Detail = High               # lots of detail
```

logrotate

- log files grow under normal use of the system
- eventually they would fill the disks
- the logrotate facility manages the log files
- it will save a log file as a "backlog" file, and start a new empty version of that log file
- old backlogs can be deleted, or emailed
- logrotate is another process run daily through a shell script in `/etc/cron.daily`
- the logrotate process is configured by

`/etc/logrotate.conf`

logrotate.conf

- how often should log files be rotated
- how big should log files get before they're rotated
- how many old backlog files should be kept
- what permissions should new empty log files get, and who should own those files

logrotate.conf

- daily, weekly, or monthly rotations
- `rotate 5` : keep 5 backlog files
- specific log files can have specific config
- `/etc/logrotate.conf` is configured to include individual package configurations from the directory `/etc/logrotate.d/`
- Example:`/etc/logrotate.d/yum`

```
/var/log/yum.log {
    yearly    # this applies to yum only
}                 # over-riding the setting in /etc/logrotate.conf
```

Services/Logging overview

- Services are provided by programs running in the background, with no terminal
- No terminal means these programs are
  - not designed to be controlled by a keyboard
  - not designed to display output to a screen
- They are started/stopped/restarted by their init script in /etc/init.d/
  - at bootup or change-of-runlevel automatically, or
  - manually with service command
- They send their "output" messages to the syslog service

Services/Logging overview

- Their behavior is controlled by a config file, where the admin specifies settings
- They often provide the "server side" for a client:
  - ftp client program running on a local computer connects to an ftp server program running on a remote computer
  - ssh client program running on a local computer connects to an ssh server program running on a remote computer
  - a web browser running on a local computer connects to a web server running on a remote computer

Examples

- SSH
  - configure port
  - configure logging
- HTTP
  - install it
  - chkconfig it
  - start it
  - firewalling
  - play with default document
  - configuration file to change its behaviors

Process Accounting

- Process accounting is concerned with keeping track of commands executed and resources used
- Linux kernel is capable of keeping process accounting records for the commands being run, the user who executed the command, the CPU time, and more.
- The process accounting service `psacct` is used to turn accounting on and off
- The `psacct` rpm includes accounting commands `sa, ac, lastcomm`

psacct

- psacct is the process accounting service
- not installed by default on CentOS 6.5
- When the psacct service is started, the kernel begins keeping track of resource usage
- various commands in the package make use of that resource usage info: `ac, sc, lastcomm`
- install and enable it with
  - `yum install psacct`
  - `chkconfig psacct on  # default runlevels 2,3,4,5`
  - `service psacct start`

ac, last, lastlog, faillog

- ac: print statistics about users' connect time

man ac

```
$ ac -p -d
```

`p`: individual totals

`d`: daily totals

last

- last: listing of last logged in users

```
last -t  YYYYMMDDHHMMSS
```

who was logged in at that time

lastlog

- lastlog : reports the most recent login of all users or of a given user

```
lastlog -u LOGIN
```

for username LOGIN

```
lastlog -b DAYS
```

before DAYS ago

```
lastlog -t DAYS
```

since DAYS ago

lastcomm

- with psacct enabled, we can view info on previously executed commands

```
lastcomm
lastcomm --user USERNAME
lastcomm --command COMMAND
```

sa

- `sa` : summarize accounting information
- `man sa` for more detail
- Fields:

cpu	sum of system and user time in cpu seconds
re	"real time" in cpu seconds
k	cpu-time averaged core usage, in 1k units
k*sec	cpu storage integral (kilo-core seconds)
u	user cpu time in cpu seconds
s	system time in cpu seconds