Overview

- Last Lecture
  - Wireless networking
- This Lecture
  - Scheduled tasks and log management
- Next Lecture
  - DNS
  - Readings:
    - Chapter 6 and 13 in Linux Network Administrator's Guide
    - DNS & BIND (O'Reilly)

Daemon

- A process that runs in the background and is independent of control from all terminals
- Reasons for daemons' independence of terminals
  - Prevent daemons' error message from appearing on a user's terminal
  - Signals generated from terminal keys must not affect any daemons that were started from that terminal earlier
- Typical daemons
  - **crond**, **syslogd**

Daemonization

```
if( (pid = fork()) != 0) exit(0); //parent terminates
setsid(); // become session leader
signal(SIGHUP, SIG_IGN);
if( (pid = fork()) != 0) exit(0); //new child continues
//now it becomes daemon
```

The purpose is to make the process independent of the control from any terminals.

Scheduled tasks

- Automating tasks
  - crond and crontab
  - crond is a very important daemon for automatically executing tasks
  - Tasks can be configured to repeat hourly, daily, weekly,…, or even per minute.
- Possible uses
  - Clean file systems
  - Log rotate
  - Check log files
  - Monitor system status and resources
  - …

syslogd

- How it works?
  - Read the configuration file */etc/syslog.conf*
  - A Unix domain socket is created and bound to the pathname */var/run/log*
  - A UDP socket is created and bound to port 514
  - Runs in an infinite loop that calls *select*, waiting for any one of the above descriptors to be readable, reads the log message, and does what the configuration file says to do with that message.
  - If the daemon receives the SIGHUP signal, it rereads the configuration file

Logging functions

- How start and close logging?
  - Create a Unix domain datagram socket and send out messages to the pathname the daemon has bound, or send them to port 514 by a UDP socket
  - *openlog()* and *closelog()*
- How to send log messages
  - *void syslog(int priority, const char *message, …);*
  - *priority* is a combination of a *level* and a *facility* shown later
  - *message* is like a format string to *printf*, with the addition of a *%m* specification, which is replaced with the error message corresponding to the current value of *errno*.

Example code

- *openlog ("mylibrary", option, priority);*
- *syslog (LOG_INFO, “My library has been invoked”);*
- *closelog ();*

Level

- Level
  - LOG_EMERG (0): system is unusable
  - LOG_ALERT (1): action must be taken immediately
  - LOG_CRIT (2): critical conditions
  - LOG_ERR (3): error conditions
  - LOG_WARNING (4): warning conditions
  - LOG_NOTICE (5): normal but significant conditions
  - LOG_INFO (6): informational
  - LOG_DEBUG (7): debug-level messages, has lowest priority

Facility

- Identify the type of process sending the message
- Facilities
  - LOG_AUTH: security/authorization messages
  - LOG_AUTHPRIV: security/authorization messages (private)
  - LOG_CRON: from *crond*
  - LOG_KERN: kernel messages
  - LOG_MAIL: mail system
  - LOG_USER: random user-level messages (default)
  - LOG_FTP: from *ftpd*
  - LOG_LPR: line printer system
  - LOG_SYSLOG: internal messages from *syslogd*
  - LOG_LOCAL0 – LOG_LOCAL7: local, discretional use by programmers.

klogd

- klogd provides a facility for system admin to check only kernel messages (which can also be checked through syslogd)
- Kernel messages can be read from **/proc/kmsg**
- Use **/proc/sys/kernel/printk** to control the level of log messages.
  - **cat /proc/sys/kernel/printk**

syslog.conf

- Syslogd configuration file
  - /etc/syslog.conf
  - Consists of <facility>.<priority> <target> entries
    - mail.* /var/log/maillog
    - authpriv.* /var/log/secure
    - *.alert root, mal
  - Use "*man 5 syslog.conf*" to find more information about the format of the file

Log processing

- Log scanning and filtering
  - Scanning: use scripts (put as a cron job) to scan key words in log files
  - Filtering: use scripts to remove useless messages from the log files
- Pros and cons of scanning and filtering
  - Scanning: can find useful information, but may have to process a large amount of log files
  - Filtering: can reduce the amount of log files but may miss some useful information.

Log processing (cont.)

- Log rotation
  - Use *logrotate* command
    - logrotate is designed to ease administration of systems that generate large number of log files. It allows automatic rotation, compression, removal, and mailing of log files. Each log file may be handled daily, weekly, monthly, or when it grows too large
  - Configuration file:**/etc/logrotate.conf** (see the manual page for logrotate)
  - Run logrotate as a cron job

Log processing (cont.)

- Store log files in computer archive
  - Legal issues regarding how long log files should be stored.
  - How to process a huge amount of log files efficiently?
- Risks of log management
  - Log files can be changed (MD5?)
  - Log files can be exposed while being transmitted (encryption?)

Cloud services for sharing?

- Privacy issue
- Issue on confidential information
- Who owns the data?
- Enterprise cloud is recommended for sharing confidential documents inside an organization.
- git and svn are good tools for sharing.

Set up cloud services

- It is possible to set up cloud services in Ubuntu using **cloud-init** with a simple recipe.
- Video of using **cloud-init**
  - https://cloud-init.io/

Summary

- Pros and cons of filtering and scanning in log processing
- Log levels for log messages
- Pros and cons of using a cloud service