Linux For Embedded Systems

For Arabs

Course 102:

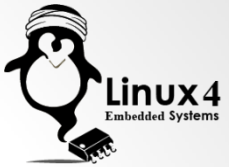
Understanding Linux

Ahmed ElArabawy

Lecture 16:
Process Management (Part 2)

On a UNIX system, everything is a file; if something is not a file, it is a process

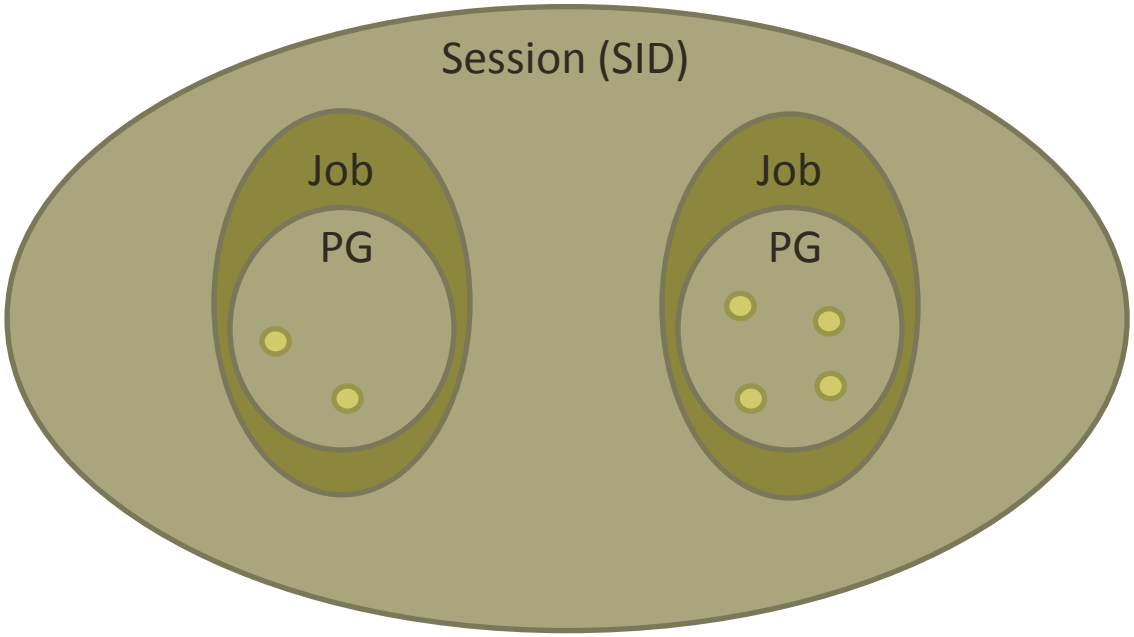
From Process Management (part 1)

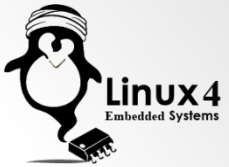


- What is a process?
- Process Owner…..
- Process Tree Hierarchy
- Process IDs (pid, ppid, pgid, sid)
- Types of Processes
  - Interactive Processes
  - Automatic (Batch) Processes
  - Daemon Processes

From Process Management (part 1)

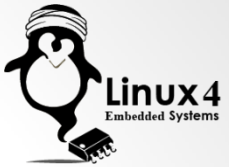
- Interactive Process
  - Started by the user inside a terminal
  - Attached to the terminal (controlling terminal)
  - Can run in the foreground or the background





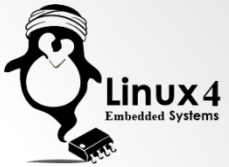
PROCESS TYPES
AUTOMATIC (BATCH) PROCESSES

Process Types
Automatic Process



- Also called a batch process
- This is a process that is not started directly by the user, instead, the user schedule it for a later start
- When started, It is not started inside a terminal, and not attached to a terminal (user does not even needs to be logged in when it starts)
- It is queued in a spooler area, to be executed on a FIFO manner
- It is scheduled in one of the following ways,
  - Scheduled to run at a certain date and time (using the **at** command)
  - Scheduled to run when system load is low (using the **batch** command)
  - Scheduled to run periodically with certain periodicity or interval

Scheduling Automatic Tasks (at Command)



$ at [options] <time> < <script file>

$ at -f <script file> <time>

Schedules jobs described in the file to run at the specified time

- Example:

  $ at 01:35 < job-to-run

  $ at 9am February 2 < job-to-run

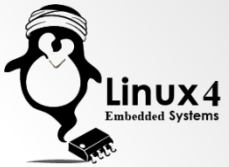
  $ at tuesday +2 hours < job-to-run

  $ at -f job-to-run noon

- Also,

Command	Description
$ at -l ***$ atq***	List scheduled jobs
$ at -r 3 ***$ atrm 3***	Delete scheduled job #3

Run at Low Load times (batch Command)



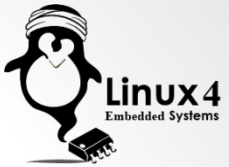
$ batch < <File containing Jobs>

Run the script whenever the system load allows

- Example:

  $ batch < job-to-run

Cron Jobs

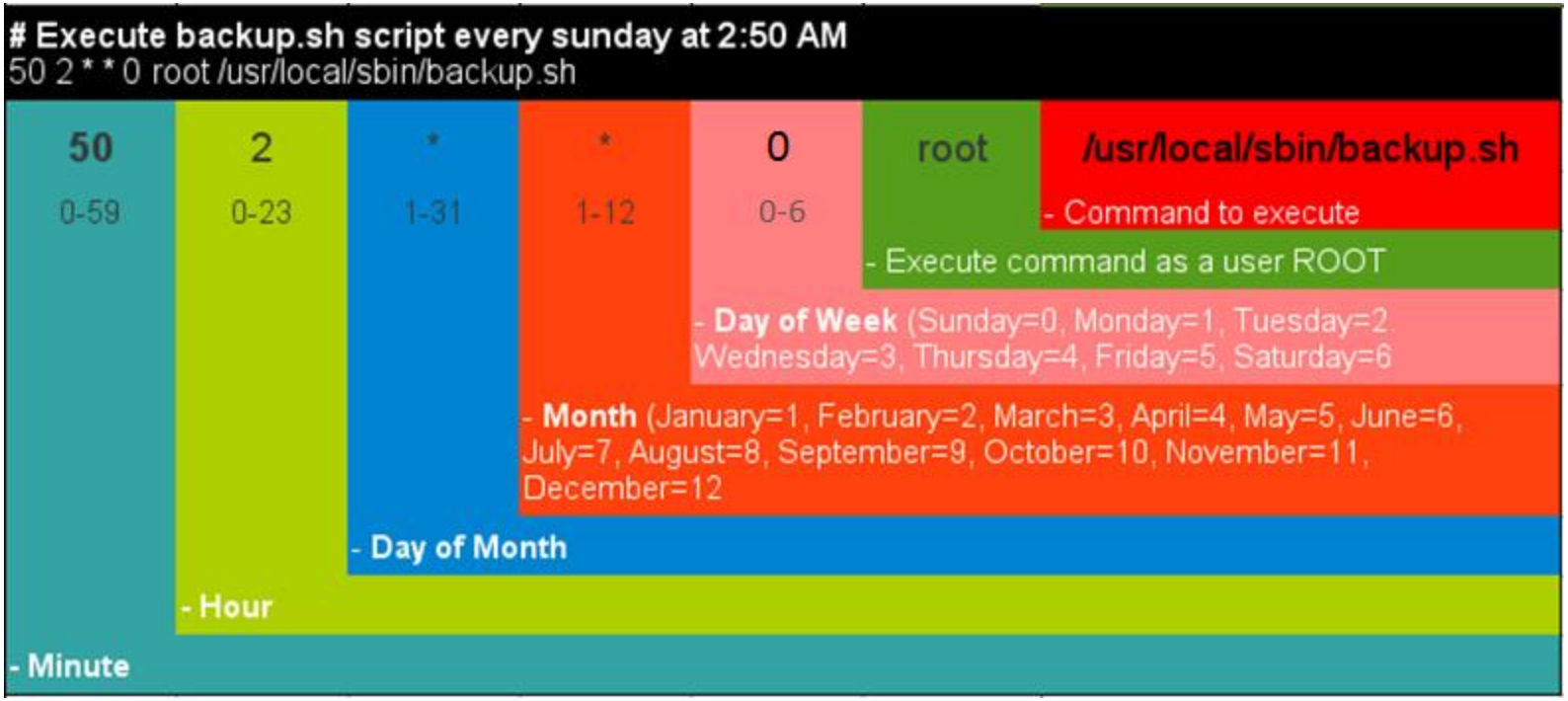


- Cron Jobs are those which are scheduled to run periodically (the word cron comes from Greek word for time)
- Jobs are organized in commands or shell scripts
- They are scheduled by the user to run at,
  - Fixed times
  - Fixed dates
  - Intervals
- Used often to automate repeated tasks (such as maintenance or administration tasks)
- Cron Jobs are organized in a set of configuration files that specify the Job to be run, and the required periodicity
- These are called ***crontab*** files
- Types of ***crontab*** files
  - User ***crontab*** files (per user file)
  - System ***crontab*** files (for root user)
  - Other special files (will be discussed later)

crontab Files Format

- The ***crontab*** files have the following format
- It contains a line per scheduled job
- Each line has three sections,
  - Time Schedule Section
    - This section describes when this Job is to be executed
  - User Section
    - This section only applicable for some ***crontab*** files
  - Job Description Section
    - Command to execute

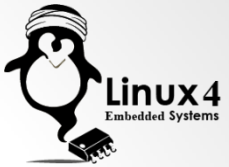
Example:


Execute backup.sh script every sunday at 2:50 AM
50 2 * * 0 root /usr/local/sbin/backup.sh
50
0-59
2
0-23
*
1-31
*
1-12
0
0-6
root
/usr/local/sbin/backup.sh
- Command to execute
- Execute command as a user ROOT
- Day of Week (Sunday=0, Monday=1, Tuesday=2, Wednesday=3, Thursday=4, Friday=5, Saturday=6
- Month (January=1, February=2, March=3, April=4, May=5, June=6, July=7, August=8, September=9, October=10, November=11, December=12
- Day of Month
- Hour
- Minute

More Examples

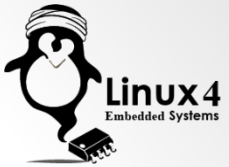
Time schedule					Description
Min	Hr	DoM	Mon	DoW	
30	0	1	1,6,12	*	At 12:30 AM On First of Jan, June, Dec
0	20	*	10	1-5	At 8:00 PM on Every (Mon-Fri) in Oct
0	0	1,10,15	*	*	At Midnight on 1,10,15 on Every Month
5,10	0	10	*	1	At 12:05 and 12:10 AM on Every Monday and 10th of every Month
30	18	*	*	*	At 6:30 PM Every day
*/5	*	*	*	*	Every 5 Minutes
01,31	04,05	1-15	1,6	*	At 4:01, 4:31, 5:01, 5:31 on the first 15 days of Jan and June

Special Strings



String	Description
@reboot	Run once at Startup
@yearly	Run once a year “0 0 1 1 *”
@annually	Same as @yearly
@monthly	Run once a month “0 0 1 * *”
@weekly	Run once a week “0 0 * * 0”
@daily	Run once a day “0 0 * * *”
@midnight	Same as @daily
@hourly	Run once an hour “0 * * * *”

Managing crontab Files (crontab Command)



- You should not edit the ***crontab*** file manually
- Instead use the ***crontab*** command, it performs checking for errors on the file before saving it
- Examples:

  To edit your user specific crontab file

  $ crontab -e

  To display your crontab file

  $ crontab -l

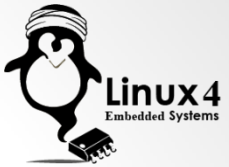
  To remove you crontab file

  $ crontab -r
- Note,
  - Applying the same commands with ***sudo*** performs the same on the system wide ***crontab*** file

Special crontab Files

- So far we talked about
  - Per user crontab file
  - System crontab file (for root user)
- Both of these categories of files are stored at ***/var/spool/cron/crontabs***
- There are other crontab files setup by the System and packages installed on the distribution,
  - The file ***/etc/crontab***
  - crontab files inside ***/etc/cron.d***
- It is not recommended to edit any of those files

/etc/crontab



- This is used by the system
- Not recommended to edit, since system updates will over-write any edits
- It calls scripts inside the directories

  /etc/cron.monthly

  /etc/cron.weekly

  /etc/cron.daily
- Used mainly for system admin tasks

/etc/cron.d

- The ***/etc/cron.d*** directory will contain crontab files installed by the different packages in the system
- Each package will have its own crontab file for periodic tasks required for this applications
- Typical periodic tasks,
  - Checking the web for updates
  - Archiving or Emptying log files
  - Delete temp files
  - Checking the Inbox for new messages

cron Job Restrictions

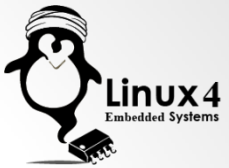
- Sometimes there are restrictions on which users are allowed to run cron jobs
- The restrictions are defined by the files,

  /etc/cron.allow

  /etc/cron.deny
- Those files will have list of users that are allowed/denied to/from use of cron jobs
- If both files don't exists, then it is up to the distribution to define the behavior,
  - Some would open the permission for all users
  - Some would limit the permission for the root user

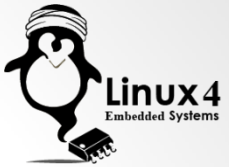
Cron Jobs Output

- Since the Cron Jobs don't run inside a terminal, output does not show on screen
- Progress and Error messages for execution of cron jobs are sent to ***/var/log/syslog***
- By default, the output of the commands go to the user as an email using the command ***email*** (assuming user email has been setup in the system)
- If you don't want to setup the email, you can do the following,
  - Redirect the output to a file to be able to follow progress
  - Redirect the output to **/dev/null** if you want to mute it



PROCESS TYPES
DAEMON PROCESSES

Process Types
Daemon Process

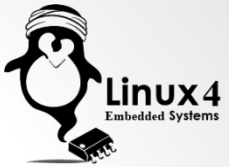


- A Daemon process is a process that runs continuously in the background to perform a task, or waiting for services to be requested from it
- Linux use numerous daemons (normally start them at system startup) to perform things like,
  - Accommodate requests for services from other computers on a network
  - Respond to other programs
  - Respond to hardware activity
- A tradition is to have the daemon name ends with letter '***d***' such as (***syslogd***, ***xinetd***, ***ftpd***)
- Daemons keep listening until they are triggered to do some action, some of the triggers would be,
  - A specific time or date (such as ***atd*** which handles Jobs scheduled by the at command)
  - Passage of a specified time interval (such as ***crond*** which handles cron Jobs)
  - A file landing in a particular directory
  - Receipt of an e-mail or a Web request made through a particular communication means
  - Connection request from a different computer (such as ***ftpd*** which handles FTP requests)

A Daemon Process

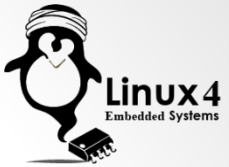
- Since a Daemon process needs to keep running in the background
  - It can not be attached with a terminal (otherwise, it will close with the terminal closure)
  - Accordingly, at its start, it disassociate itself from its contorlling terminal
- A Daemon process **often** needs to have its parent as the ***init*** process (PPID = 1)
  - This is achieved for daemons started at system startup, since they will be launched by ***init***
  - Daemons starting afterwards can have their parent set to the ***init*** process, by launching the daemon process, then killing its parent. This will cause the kernel to re-parent the process to the ***init*** process

Creation of a Daemon



- When Daemons are created, the following happens,
  - The parent of the Daemon is killed (or dies on its own), to make sure the Daemon is re-parented to the ***init*** process
  - The Daemon is detached from his controlling terminal to make sure it remains up even when it closes
  - The Daemon becomes a session leader (its ***SID*** is set to be equal to its ***PID***)
  - The Daemon becomes a process group leader (its ***PGID*** is set to be equal to its ***PID***)
  - Sets its current directory to be the root directory (/) to allow other any file system to unmoun
  - Close any relation that it inherited from its parent process
  - Sets its ***stdin***, ***stdout***, and ***stderr*** to either a logfile, the console, or mute it by using ***/dev/null***

Managing Daemons

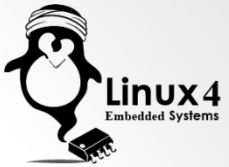


- Since Daemons are detached from their controlling terminal, then we can not manage them from there
- Instead, there are scripts (called ***init scripts***) to
  - Check a daemon status
  - Start a daemon
  - Stop a daemon
  - Restart a daemon
- Scripts are located in /etc/init.d
- For example,

  $ sudo /etc/init.d/bluetooth stop
  $ sudo /etc/init.d/bluetooth start
  $ sudo /etc/init.d/bluetooth restart
  $ sudo /etc/init.d/bluetooth status
- In other systems

  $ sudo restart bluetooth
  $ sudo stop bluetooth
  $ sudo start bluetooth
  $ sudo status bluetooth

Examples of Daemons



Daemon Name	Function
syslogd	It implements the system logging facility
sshd	It services incoming ***SSH*** connections
ftpd	It services incoming ***FTP*** connections (FTP Server)
crond	It executes jobs in ***crontab*** files
atd	It executes jobs scheduled with ***at*** command
Inetd or xinetd	It is responsible for networking
httpd	It is responsible for handling ***HTTP*** requests (web server)

Linux 4 Embedded Systems

http://Linux4EmbeddedSystems.com