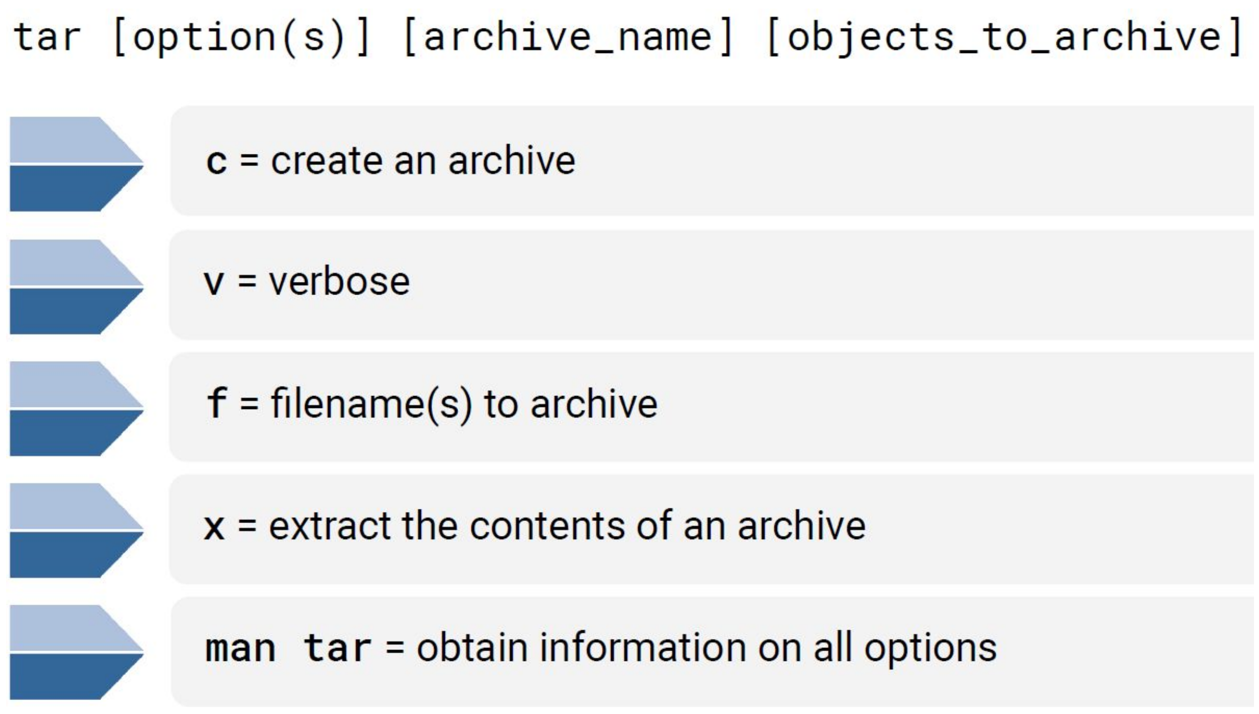
Terminal Commands Summary, Part IV (Lesson 10)

Archiving with Tar

tar (tape archive) - used for basic backup and recovery procedures, as well as archiving documents and distributing software. They are stored in “tarballs,” and can be compressed. Tarball backups are normally completed on one system, compressed and then moved to another system.

Tar Syntax:



```
tar [option(s)] [archive_name] [objects_to_archive]
```

c = create an archive

v = verbose

f = filename(s) to archive

x = extract the contents of an archive

man tar = obtain information on all options

3 Styles of Tar command:

1.) Traditional - Options are a cluster of letters
    a.) tar cvf a.tar /etc
2.) Short - prepends the “-” to the option that is used most in Unix commands
    a.) tar -xvf a.tar or tar -x -v -f a.tar
    b.) Assuming /etc needs to be added (omitted on slides)
3.) Long - spells out the option name
    a.) tar --create --file --verbose a.tar /etc

Listing a tar archive:

- `tar -tvf TarDocs.tar` (verbose)
- `tar -tf TarDocs.tar` (“short”)

Can view directory structure with tree command

Untarring (extracting) examples:

`untar` TarDocs.tar.

    `tar -xvf TarDocs.tar`

Untarring two files from archive:

    `tar -xvf TarDocs.tar -C /tmp --strip-components=2 "TarDocs/Movies/ZOE_0003.mp4"
    "TarDocs/Movies/ZOE_0004.mp4"`

Untarring files of a certain type (i.e., pdf files):

    `tar -xvf TarDocs.tar -C pdfDocuments --wildcards "*.pdf"`

Untarring a “Gibberish” archive

    tar -xvf Gibberish_Folder.tar

Use `--strip-components` to extract the `Movies` directory (--strip-components=1 to strip the first leading component, TarDocs, off of the file name)

    `tar xvf TarDocs.tar --strip-components=1 "TarDocs/Movies"`

Use `--strip-components` to extract the `Movies/ZOE_0004.mp4` file (--strip-components=2 to strip the first two leading components, TarDocs/Movies, off of the file name).

    `tar xvf TarDocs.tar --strip-components=2 "TarDocs/Movies/ZOE_0004.mp4"`

Creating tar examples:

Create a `sample.tar` archive with four files

    `tar cvfW sample.tar file1.txt file2.txt file3.txt file4.txt`

List the contents of the `sample.tar`.

    `tar -tf sample.tar`

Create a `multidir.tar` archive containing the `Pictures` and `Movies` directories.

    `tar cvfW multidir.tar 'TarDocs/Pictures' 'TarDocs/Movies'`

Create a `mp4.tar` archive that contains ONLY the **.mp4** files `Projects/TarDocs` directory.

    `tar cvfW mp4.tar $(find TarDocs -iname "*.mp4")`

Create a `mytest.tar` archive that **excludes** the `Programs` directory in the `Projects/TarDocs` directory.

    `tar cvfW mytest.tar --exclude="TarDocs/Programs" TarDocs/`

Create archive `largefiles.tar` using the `--files-from = FILES` option.

    `find . -size +5000 -print > large-files.txt
    `tar cvfW largefiles.tar --files-from=large-files.txt`

Create a tar archive file tecmint-14-09-12.tar for a directory /home/tecmint in the current working directory

    tar -cvf tecmint-14-09-12.tar /home/tecmint

Create a Tar Archive of IRC_Logs

    tar -cvf IRCLogs.tar IRC_Logs/

Use `tar` to create a bzip2 archive of your `Wallpapers` directory, located in the `Pictures` directory.

    `tar cfvj projects.tar.bz2 ~/Pictures/Wallpapers`

Use `tar` to create a gzip archive of your home directory (`~`), but _exclude_ the `Documents` folder.

```
`tar cfvz home.tar.gz ~ --exclude="~/Documents"`
```

Use tar to unpack the tarball from the previous step into `/tmp`.

```
`tar xvfz home.tar.gz -C /tmp`
```

<u>Updating Tarballs</u>

Add files with `-r`:

```
`tar rvf <existing archive> <new filename>`
tar -rvf update.tar test1.txt test2.txt
```

Update files with `-u`:

```
`tar uvf <existing archive> <updated file>`
tar -uvf update.tar test2.txt
```

Delete files with `-d` or `--delete`:

```
`tar -f <existing archive> --delete <file to delete>`
tar -f update.tar --delete test1.txt
```

<u>Incremental Backups</u>

```
# Creates a level 0 backup of home
  $ tar --create
    --file=/var/backups/home.backup.1.tar
    --listed-incremental=/var/log/home.snar
    /home

  # Add some files to home directory
  $ touch ~/test1.txt ~/test2.txt

  # Copy snapshot file
  $ cp /var/log/home.snar /var/log/home.snar-1

  # Create a level 1 backup
  $ tar --create
    --file=/var/backups/home.backup.2.tar
    --listed-incremental=/var/log/home.snar-1
    /home

  # List contents of backup
  $ tar -tf /var/backups/home.backup.2.tar
    test1.txt
    test2.txt
```

<u>Compression</u>

2 most popular compression standards: gzip and bzip2

Create the `tardocs.tar.gz` archive using the `TarDocs/Programs` directory

```
tar czf tardocs.tar.gz TarDocs/Programs
```

Create the `tardocs.tar.bz2` archive using the `TarDocs/Programs` directory

```
`tar cjf tardocs.tar.bz2 TarDocs/Programs`
```

Compare the size of the `tardocs.tar.gz` and `tardocs.tar.bz2` archives.

```
`wc -c tardocs.tar.bz2`
`wc -c tardocs.tar.gz`
```

Use -cvzf to create a compressed gzip archive file (can also just use .tgz instead of tar.tgz)

```
tar -cvzf tecmint-14-09-12.tar.tgz /home/tecmint
```

Cron

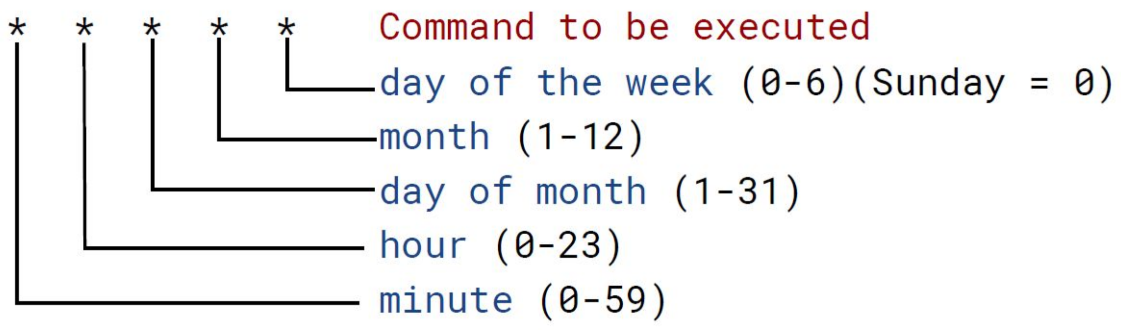
Cron Daemon: standard UNIX program that runs user specific programs at scheduled times

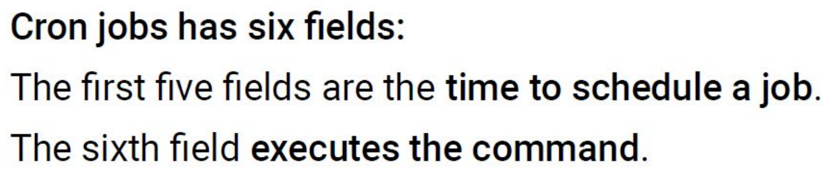
Cron Daemons run Cron Jobs. Cron jobs are lists of tasks located in a cron table.

Check to see if the `cron service` is running.

```
`service cron status`
```

Cron Jobs


* * * * * Command to be executed
day of the week (0-6)(Sunday = 0)
month (1-12)
day of month (1-31)
hour (0-23)
minute (0-59)



Cron jobs has six fields:

The first five fields are the **time to schedule a job**.

The sixth field **executes the command**.

Example 1:

```
0 0 1 */3 * tar -cf quarterly.tar ~/accounts
```

This **cron** job will run on:

```
The first day of every third month and create an archive `quarterly.tar`
file of the `/home/accounts` directory.
```

1. The `minute` field is not assigned a value.
2. The `hour` field is not assigned a value.
3. The `dayofthemonth` field is assigned a 1, to run on the first day.
4. The `month of the year` field is assigned a 3, to run on the third month.
5. The `dayoftheweek` field is not assigned a value.
6. The `command` field contains the tar command.

Example 2:

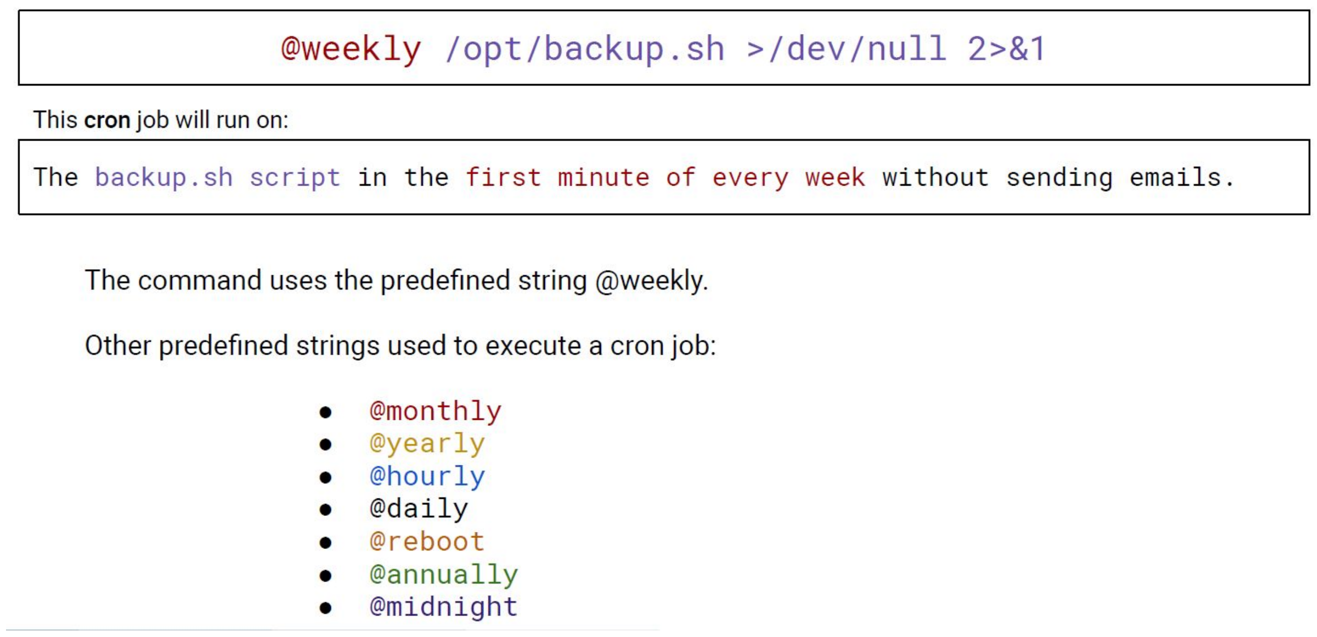
```
0 */4 * * * rsync -avz /home/slynux/data
slyunx@192.168.1.5:/home/backups/data
```

This **cron** job will run on:

```
Every Four hours and create a compressed rsync backup of the
`/home/sylnux/data` directory on the remote machine in the /home/sylnux/data
```

1. The `minute` field is 0.
2. The `hour` field indicates every four hours.
3. The `dayofthemonth` field is not assigned a value.
4. The `month of the year` field is not assigned a value.
5. The `dayoftheweek` field is not assigned a value.
6. The `command` field contains the rsync command.

Example 3:



```
@weekly /opt/backup.sh >/dev/null 2>&1
```

This **cron** job will run on:

```
The backup.sh script in the first minute of every week without sending emails.
```

The command uses the predefined string @weekly.

Other predefined strings used to execute a cron job:

- `@monthly`
- `@yearly`
- `@hourly`
- `@daily`
- `@reboot`
- `@annually`
- `@midnight`

More examples:

Schedule the shell script `/scripts/backup.sh` to run `every 10 hours`.
`0 */10 * * * /scripts/backup.sh`
Create a compressed `pictures.tar.gz` file of the `/home/pictures` directory. The job runs on Monday and Friday at 10 PM.
`0 22 * * mon,fri  tar -czf pictures.tar.gz $HOME/pictures`
Schedule the jobs `/scripts/expenses.sh` and `/scripts/paidexpenses.sh` using one cron entry. The jobs run at 4 am and 5 pm on Tuesday.
`0 4,17 * * tues /scripts/expenses.sh; /scripts/paidexpenses.sh`
Write a cronjob that creates a bzip2 archive of the `/home` directory every Sunday. Save it in `/var/backups`.
`* * * * 7 tar cfvj /var/backups/home.tar.bz2`
Write a cronjob that creates a gzip archive of the `/var/log` directory every day at 5AM. Save it in `/var/backups`.
`* 5 * * * tar cfvz /var/backups/logs.tar.bz2`
Create a `cronjob.tar` tarball that contains only the `text files` in the `~/data/cron/Documents` directory. The job should then `untar` the tarball into a `data/cron/exercises` directory. Job should run Tue, Thu and Sat every 2 minutes

```
2 * * * tue,thu,sat tar -cf $HOME/data/cron/cronjob.tar $(find $HOME/data/cron -name '.txt'); tar -xf $HOME/data/cron/cronjob.tar -C $HOME/data/cron/exercises
```

<u>To add the cron job to the cron table (Edit the crontab file)</u>:
1.)`crontab -e` (select the `nano` editor if needed.)
2.) Scroll down to the bottom of the file and add cron jobs.

3.) * `*/2 * * * * echo "Hello World" >> myhellojob.txt; echo "This is a test" >> myhellojob.txt` (sample: 2 cron jobs on one line).
- Verify job. ls or cat myhellojob.txt in this case
- IMPORTANT: Disable the job after verifying by commenting (prepending `#` to the job)

<u>Displaying and Backing Up a crontab</u>
Display crontab file
	`crontab -l`
Backup crontab file
	`crontab -l > backup-cron-jobs.txt`

<u>Removing and Restoring a crontab (verify with display command above)</u>
Removes your crontab file
	`crontab -i`
Restores your backup
	`crontab backup-cron-jobs.txt`

<u>Logging</u>
Cron jobs create log entries. Logs take up a lot of space and must be backed up. Logrotate and rsyslog can perform automatic rotation, compression, remote backup and removal of files.

Cron jobs are stored in /var/log/syslog.

View messages in `/var/log/syslog` in real time:
	`tail -f /var/log/syslog`
View only `cron` log messages in `/var/log/syslog`.
	`grep -i CRON /var/log/syslog`

<u>Logger examples</u>:
Create a simple message such as "This is a logging message".
	`logger "This is a logging message from <your first name>."`
Confirm that message is displayed in `var/log/syslog`.
	`grep -i <your firstname> /var/log/syslog`
Create a simple message using a `tag`. Use **your name** as the tag and the text `"This is a test using a tag."`
	`logger -t <name> 'This is a test using a tag'`

Display a log message to `standard error` as well as the `/var/log/syslog`.

`logger -s "This message goes to the screen and a log file.`

Create a log message using the `-f` option.
`cat > testfile.txt`
`My name is <your name>`
`I am in class tonight`
`learning about logging`
`logger -f testfile.txt`

Monitoring Log Files
Journalctl and rsyslog administer logs via filtering. Journalctl displays entries managed by the journald daemon.

Journalctl examples:
Obtain the date / time on the system and view the log for `cron` up to that time.
`date`
`journalctl  --until <year-mon-day> <hr:min:sec> -u cron`
View the log for `cron` without `metadata` for the `last 20 minutes`.
`journalctl -o cat --since -20m -u cron`
View the log for `cron` so that the newest entries are displayed first.
`journalctl -r -u cron`
Review and analyze the results from `journalctl` (for the system as a whole) to determine what activities are frequently logged
`journalctl | less`
Export the journal log for `cron` to `exportlog.txt`. This seralizes the data for backups or network transfer.
`journalctl -o export -u cron > exportlog.txt`

Logwatch:
Manually send a digest for **today's cron activity** to `mylogwatch.txt`.
`logwatch --service cron --filename mylogwatch.txt --range today`
View the activity in the `mylogwatch.txt` file. - `nano mylogwatch.txt`