IT 341: Introduction to System Administration

Notes for Project #9: Automating the Backup Process

Topics

- **Backup Strategies**
- **Backing Up with the rsync Daemon**
- **The crontab Utility**
- **Format of a crontab File**
- **Creating a Backup Script**

Backup Strategies

- Backups present a classic problem in *rewards* versus *costs*
- Normally, when you perform some action, there is an immediate result
- You do something, and you can see the effects immediately or shortly thereafter
- But this is not true for backups

Backup Strategies

- Backups should be done every day and, if you are lucky, you might *never* need them
- For this reasons backups must be as painless as possible otherwise they will not be done regularly
- The best way to make them painless is to make them automatic

Backing Up with the *rsync* Daemon

- *rsync* comes in two forms
  - A utility
  - A daemon
- The *rsync* daemon is a background process that is always running
- Every time a change is made to the contents of a disk, that change is written to the archive
- This approach has two problems…

Backing Up with the *rsync* Daemon

- **First**, the daemon is constantly running so it continually generating network traffic
- And this traffic is greatest when the system is most heavily used
- In other words, this approach has a high overhead
- For this reason, experienced system administrators don't recommend using the ***rsync*** daemon for backups
- But, there is another problem…

Backing Up with the *rsync* Daemon

- Backing up constantly means you have the latest copies of everything -- ***including mistakes!***
- A much better approach is to make a new backup ***periodically***
- This way, if a mistake is made in the current version of a file you can go back to an *earlier* version

Using *cron*

- *cron* is a daemon that runs specific programs at specific times
- Although *cron* can run any command or program it usually runs scripts
- *cron* is often used for system administration tasks
- The list of programs to be run is contained in a crontab file
- A crontab file is a text file with a list of commands or programs and when they should be run
- The system itself has a crontab file, /etc/crontab

Using *cron*

- Also, every user has his or her own crontab file
- These files are stored in /var/spool/cron/crontabs on Ubuntu systems and in /var/spool/cron on Red Hat installations
- These user crontab files have the same filename as the username
- A crontab file cannot be edited directly
- Instead, you must use the *crontab* utility

Using *cron*

- By default, all users can create a crontab file
- But if a /etc/cron.allow file exists then only users listed there can use *cron*
- If a /etc/cron.allow file does not exist but a /etc/cron.deny does then only users not listed in /etc/cron.deny can use *cron*

The *crontab* Utility

- To create or edit a **crontab** file you must use the *crontab* utility
- *crontab* has two options
  - **-l** List the contents of the crontab file
  - **-e** edit a crontab file

Format of a crontab File

- Each line of a crontab file specifies a command or program and when it is to run
- Each entry in the file must end in a newline character
- Each of these entries is called a cron job
- Each entry has ***6 fields***
  - Minute
  - Hour
  - Day of month
  - Month
  - Day of week
  - Program or command to be executed

Format of a crontab File

- The format for a **crontab** line is

```
* * * * *  command to execute

T T T T T
| | | | |
| | | | |
| | | | └──── day of week (0 - 6) (0 to 6 are
Sunday to Saturday, or use names)
| | | └────────── month (1 - 12)
| | └──────────────── day of month (1 - 31)
| └────────────────────── hour (0 - 23)
└──────────────────────────── minute (0 - 59)
```

Format of a crontab File

- There must be an entry in all fields and each entry is separated from the others by *whitespace*
- If a time field has no number, it must have a *
- If I wanted the script **daily_backup.sh** to run every night at 1:15 AM I would create the following entry

```
15 1 * * *  daily_backup.sh
```

- If I wanted the script **monthly_backup.sh** to run on the first of every month at 2:10 AM on the first of every month I would create the following entry

```
10 2 1 * *  monthly_backup.sh
```

- If I wanted the script **yearly_backup.sh** to run on the last day of the year at 3:20 AM on the last day of the year I would create the following entry

```
20 3 31 12 * yearly_backup.sh
```

Creating a Backup Script

- Before we can use *cron* to run backups, we must create a backup script
- This backup script will run **`rsync`** to create a backup of **/etc** to another machine on the network in a directory whose name is today's date in YYYY-MM-DD format