EEE174 –CPE185 INTRODUCTION TO MICROPROCESSORS

LAB 8 – RSYNC

Lab Session: Wednesday 6:30PM - 9:10PM

Section 32385

Lab Instructor: Sean Kennedy

Student Name: Andrew Robertson

TABLE OF CONTENTS

PART 1

OVERVIEW

This is my introduction to RSync, a powerful command line utility for synchronizing data between one location and another. It is able to be used in most linux systems by default, and can be installed on both Windows and Macintosh systems as well. I also learned that Windows 10 has a feature called WSL that can be enabled to run a LINUX subsystem in windows. With this, you can natively run Linux commands in windows through the Powershell.

LAB DISCUSSION

RSync is a very robust utility with many options, flags, and abilities. With this in mind, I wanted this lab to be more of a proof of concept for myself instead of an in-depth dive into the utility. For example, RSync can be used with local or remote directories with the only real difference being the command used to connect to the remote location. Since the core concept remains the same either way, I chose to test the utility locally. Before I get to the steps I took to try RSync, I want to talk about why someone would even bother using this locally instead of simply copying and pasting files. With general file explorers there is no verification of data when copying and pasting from one location to another. This means that if some sort of error occurs and causes data to be very slightly different during a copy, the user won't know it. This is unacceptable for sensitive data and is a reason RSync would still have merit here. RSync uses MD5 hash verification to be sure that the file written is the same as the file from the source.

My first step was to check and see if my distribution had RSync installed at all. Typing "man RSync" in the command line showed me man pages for the utility were available and confirmed it was installed. Next, I created an empty folder on my desktop called "New" that I would use as a destination directory for the sync. Next I attempted to use RSync without any flags:

```
pi@raspberrypi:~/Downloads $ rsync /home/pi/Downloads/ /home/pi/Desktop/New/
skipping directory .
```

From this and a bit of reading I learned that by default, RSync skips directories and subdirectories. The -a flag needs to be used for them to be included:

```
pi@raspberrypi:~ $ rsync -a /home/pi/Downloads /home/pi/Desktop/New
pi@raspberrypi:~ $
```

This time there were no errors so I checked the folder I created on the desktop and found my downloads folder completely intact at the new location – of course keeping the source whole too. After A bit more reading I learned another flag that can be given is one to create a log file for the transaction so naturally I gave it a shot.

```
pi@raspberrypi:~ $ rsync -a --log-file=rsynclog /home/pi/Downloads /home/pi/Desk
top/New
pi@raspberrypi:~ $ cat rsynclog
2018/04/18 16:11:44 [2175] building file list
2018/04/18 16:11:44 [2175] sent 105 bytes  received 22 bytes  total size 0
2018/04/18 16:13:36 [2202] building file list
2018/04/18 16:13:36 [2202] cd+++++++++ Downloads/
2018/04/18 16:13:36 [2202] >f+++++++++ Downloads/testout.txt.txt
2018/04/18 16:13:36 [2202] sent 147 bytes  received 44 bytes  total size 0
```

The flag to do this is a bit long but for good reason. It is formatted as "—log-file=name_of_file_to_use". The way this flag works is intuitive for its purpose. First, if the file name given does not exist, RSync will create it. Second, if the file does exist this utility will append log data instead of overwriting. As can be seen above, I wrote to the same log file twice, once for a situation where the folders were already synced, and a second time after deleting the destination folders contents.

This gave me the general idea for Rsync but again more reading revealed a way to make this utility even more powerful. Pairing RSync with the cron scheduler opens up a wide gamut of automation for backup and sync opportunities. Automated backups using these two can be set to occur when shutting the computer down properly, when plugging in a device, when changes are made to a folder, or even just at a set time of day every day, month, year or any combination thereof. Furthermore, a lot of paid utilities for windows and mac use RSync as a backend so speaking between the two is often still possible.

Work Performed / Solution:

Listing Files(s):

https://linode.com/docs/tools-reference/tools/introduction-to-rsync/

https://medium.com/@sethgoldin/a-gentle-introduction-to-rsync-a-free-powerful-tool-for-media-ingest-86761ca29c34

https://en.wikipedia.org/wiki/Rsync

https://opensource.com/article/17/11/how-use-cron-linux

CONCLUSION

RSync may be a free utility but it is certainly not watered down. With some time and dedication to learn this utility and cron thoroughly I believe anyone can build an automated synchronization tool for their needs that can easily compete with (and often beat) it's paid competition.