Subject: allowing jobs to restart automatically
From: Joseph Picone <joseph.picone@gmail.com>
Date: 6/11/23, 9:01 AM
To: nedc_research <nedc_research@googlegroups.com>
CC: ECE 1111 <temple_engineering_ece1111@googlegroups.com>

(ECE 1111 students won't understand all of this, but it gives you some idea of what we will cover in the course, and highlights the emphasis on problem-solving rather than memorization.)

As I have mentioned before, the Unix "philosophy" is to provide a simple core set of tools that let you build more sophisticated tools by combining these simple tools.

Below is a script that is a good example of this.

When we want to copy a large chunk of data from one machine to another, we use rsync. Rsync is an amazing command that allows you to synchronize or copy files from a src location to a destination location. We use this for backups, and we use this to copy data from one computing network (nedc_009) to another (nedc_000).

Unfortunately, rsync jobs can often take a long time (days) and can crash for a variety of reasons, such as losing a network connection. For example, if you try to rsync data from your laptop to neuronix, you will find out that your wireless network often disconnects in the middle of the night (Comcast loves to run maintenance at 2 AM, which often introduces a network disruption). This will cause a program like rsync to fail.

Rather than lose a night of work, the solution is to create a job that can restart itself when it crashes. But to do this, you need a tool that can figure out where it stopped and can resume from that point. This is the beauty of rsync - you run it repeatedly until it finishes.

You also need an ability to ssh into a machine without typing a password. I have covered this in other emails about ssh keys.

So... putting all these things together, we have a handy little script below that re-launches rsync until the job is complete.

To use this, you have to set up ssh keys, which I have done of course. You also need to know if a Unix command has failed. In shell, we do this with the following line:

```
 RC=$?
```

If a program fails, RC will be set to something other than zero.

The simple script below uses a while loop that runs forever or until RC becomes zero.

If rsync fails, it sleeps for one second, and then starts again.

The cool thing is I can set up a VPN into nedc_009, launch this script via a nohup job (so it runs when I log off), and then log off and exit the VPN. Luckily, even though my VPN connection was terminated, the job keeps running.

You might say why not keep your VPN connection open? The answer is simple - you lose network connectivity with the outside world. You can't send email and can't browse the web.

I don't completely understand why large rsync jobs fail, but they do. Even when you are syncing to local disks on your laptop you will often see them fail. The OS at some point thinks the disk doesn't exist and fails. This has been a chronic problem with operating systems for my 40+ years in the field 🙁

So now you have a very nice way of not losing time because you can keep your rsync jobs restarting automatically until they finish.

-Joe

======

```
nedc_009_[1]: more nedc_rsync.sh
#!/bin/sh
#
# file: nedc_rsynch.sh
#
# this script detects an error with rsync and automatically restarts
# rsync if an error occurs. It is used to keep rsync running over
# long periods of time when you might regularly lose your network conneciton.
#
# Usage: nedc_rsync.sh user@host:/path...
#
# Example:
#  nedc_rsync.sh nedc_tuh_eeg@www.isip.piconepress.com:~/data/tuh_eeg/ .
#

# set up an infinite loop
#
RC=1
while [[ $RC -ne 0 ]]
do
    # display an informational message
    #
    echo "starting rsync..."

    # execute your rsync command
    #
    rsync -auxv svs picone@neuronix.nedcdata.org:/data/isip/data/fccc_dpath/deidentified/v1.0.0/
    RC=$?

    # display an informational message and sleep for a bit
    #
    echo "done with rsync..."
    sleep 1

done

#
# exit gracefully
```