**PurgeTimer V2 – Testing Documentation**

**Description:**

Windows DOS batch script which will remove contents of folder every N (e.g 14) days only if the file or sub-folder is older than M (e.g. 3) days: purge\_timer\_v2.bat

The batch script is run in the background using a Virtual Basic script: run\_bg\_purge\_timer.vbs

**Classification?:**

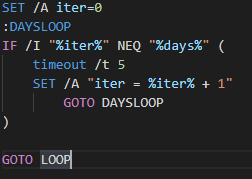
The PurgeTimer script is to be deployed onto a clinical server that holds patient data (specifically DICOM files). The script itself is not performing any calculation nor is making changes to the patient data. It is meant to purge files that are older than a specified data and have been further processed backed up already.

However, the script has potential to impact patient care if it fails. There may be potential loss of patient data that hasn’t been backed up/further processed as needed if the script functions contrary to what is expected.

Based on its function **the script does not qualify as Health software or Medical device software**, but due to the clinical risks it poses it must still undergo some level of testing before being deployed onto the live server.

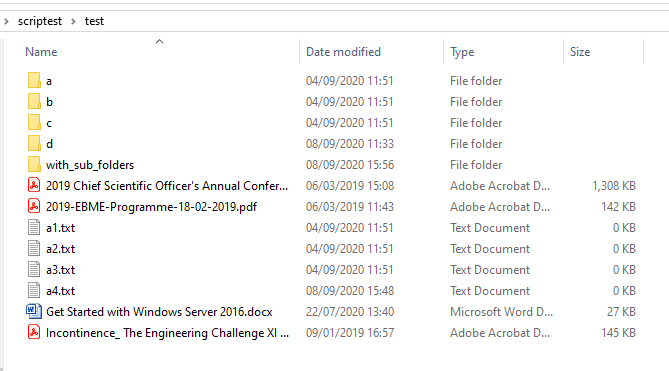
**The script was tested for functionality using accelerated time:**

(Timeout set to 5 sec instead of 86400 sec (1 day) for accelerated time)



**Test set**

* Test set consisted of a mix of files, folders etc. with varying last modified dates



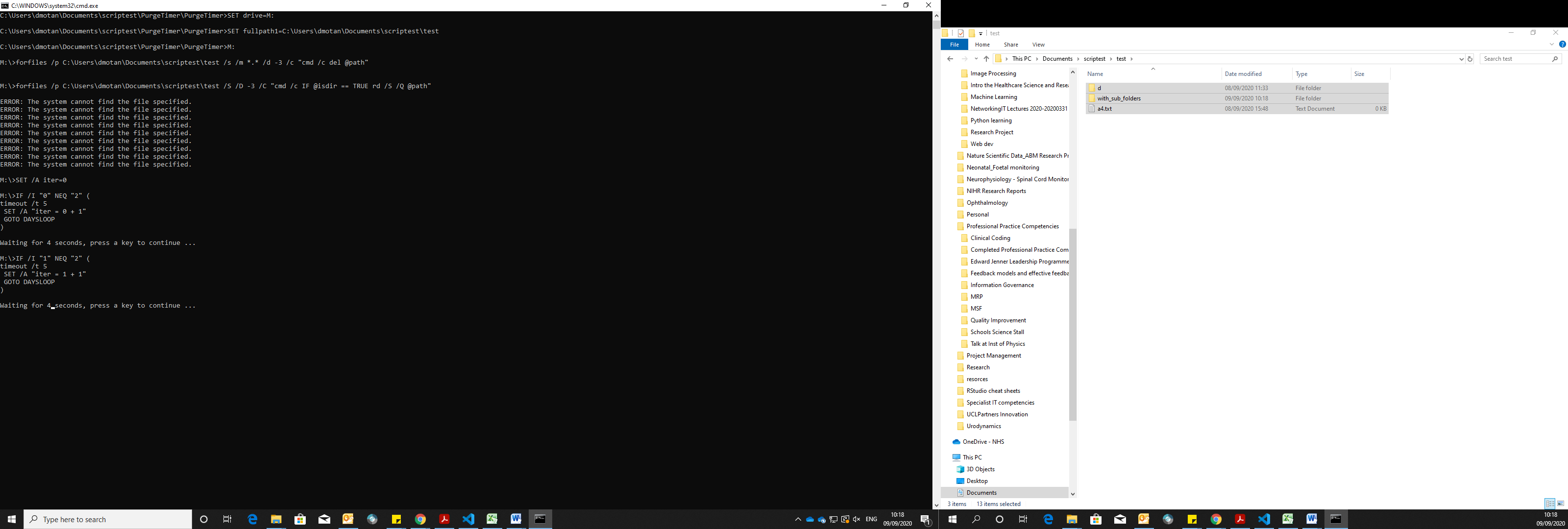
Initial tests using the following parameters:

* Age set to 1, 3, 10, 100, 500
* Days set to 1, 2, 5 , 10

Initial outcomes:

* Correct folder contents purged only (i.e. folder specified in the script)
* Runs in the background without any problems (no apparent problems with excessive memory usage)
* Functions as needed for specified days and file age
* Handles folders and subfolders as required
* Can be killed as per instructions

**A consistent error that script displays:**

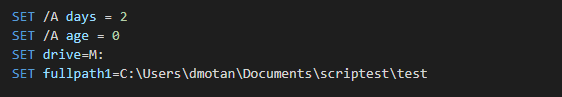


However this does not lead to any functional issues. It is possibly due to the 2 lines of codes running sequentially that respectively check for files meeting the age criteria or files in subdirectories meeting the age criteria.

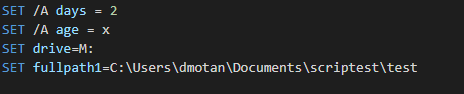
No potential risks envisaged due to this error

**Some exception cases tested:**

* Age set to 0:

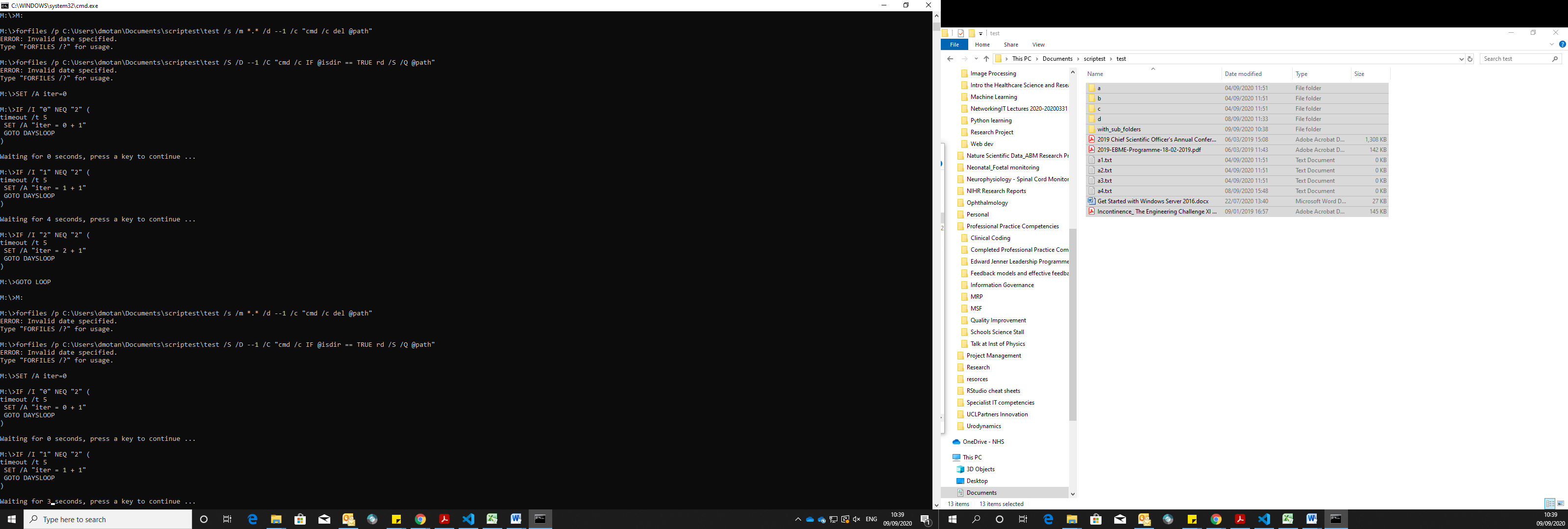
Purges all files as would be expected

* Age set to a non numerical (e.g. z, x, o etc. ):



Purges **ALL** files **(potential risk here)**

* Age set to negative number (eg. -1)



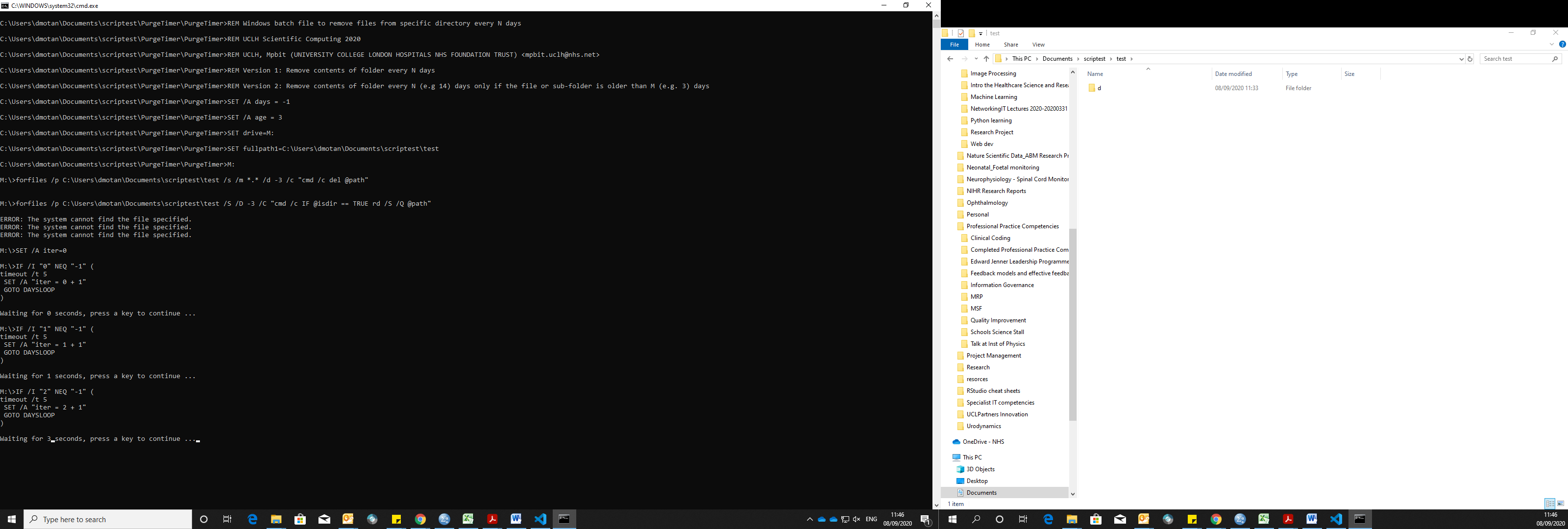
Does not purge any files. Successfully handles by specifying: ‘ERROR: Invalid date specified’

Therefore no potential risk here.

* Days set to 0:

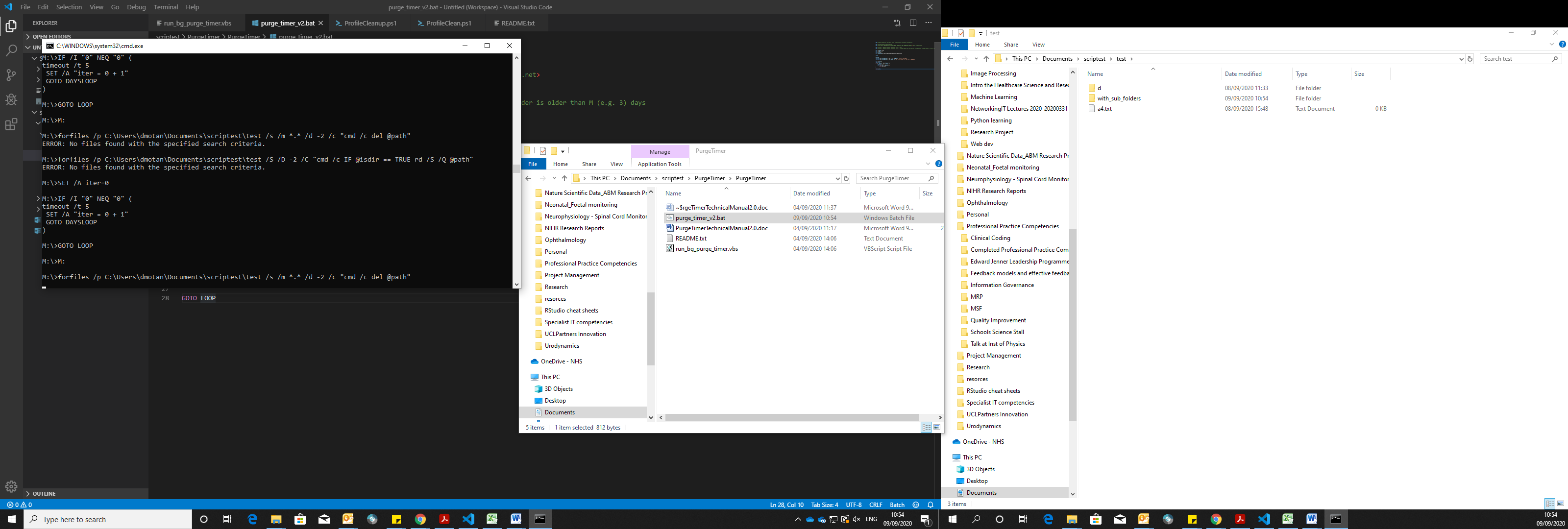
Purge function continuously executes in a loop, but only deletes files that meet the set age criteria

* Days set to -1



Purge function executes once at the beginning, and then not again until script killed and restarted

* Days set to 0.5



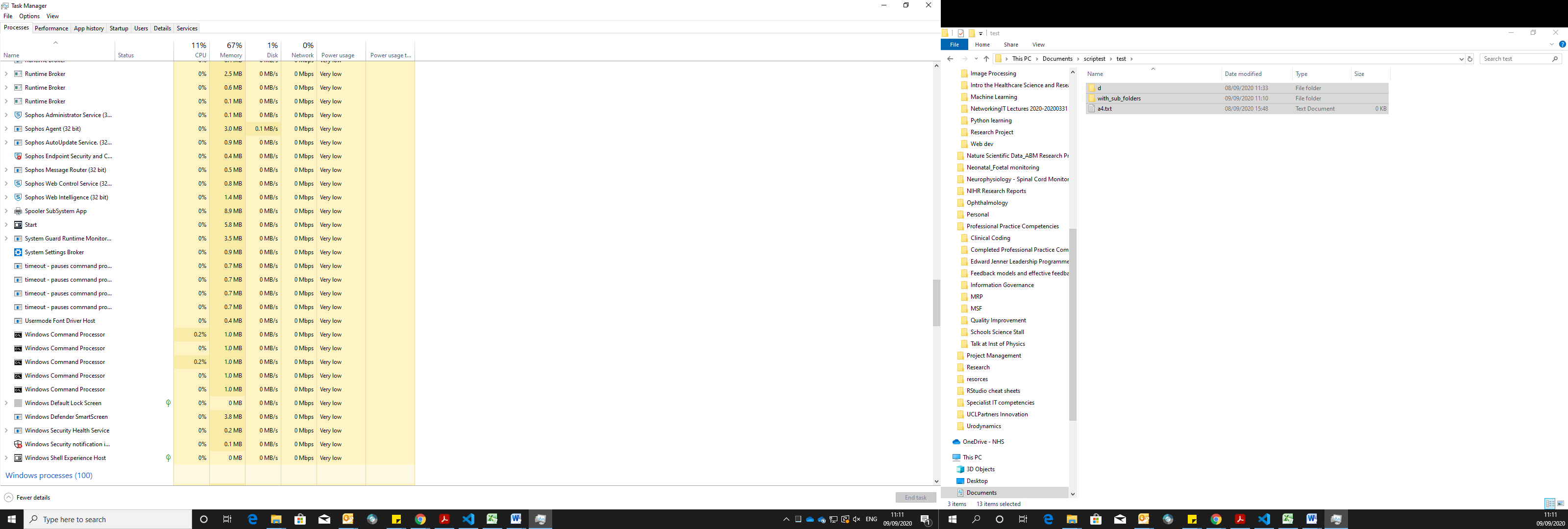
Considers decimal numbers the same as the integer (floored). So days set to 0.5 produces the same result as days set to 0

* Days set to non-numerical (e.g. z, x, o etc. ):

Purge function continuously executes in a loop, but only deletes files that meet the set age criteria

**Further Testing**

* Open multiple instances of the script in the background and see if any effect on function



Each instance of the PurgeTimer script still functions independently without errors.

However, no warning that the script is already running in the background.

**Next possible testing tasks:**

* Add large numbers of files (high memory as would be expected of DICOM files) and see if the script handles it correctly (stress test)
* See what happens when the system/server is restarted while the script is running
* See what happens if system/server is shut/restarted right when the purge function is about to execute

**Overall Possible Hazards:**

* Patient data that hasn’t been backed up/further processed as needed is deleted
* Root files/essential files in the specified directory that may accidentally meet criteria are deleted
* The script running in the background cannot be killed when required

**Overall Possible Controls:**

* Specific file types are purged only (DICOM etc.)
* Script only accepts values for age > for example 3 days
* Script asks to confirm the parameters entered including age and target folder (possibly using a popup) before starting/being deployed in the background
* Script warns if an instance is already running in the background.
* Killing the process running in the background could be made more easier.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hazard Description** | **Potential Clinical Impact** | **Causes** | **Possible Controls** |
| Patient data that hasn’t been backed up/further processed by the ARIA system as needed is deleted | Radiotherapy service delivery delayed.  Patient has to be called in for a re-scan (increased dose exposure) | User enter wrong parameters for Age (e.g. non-numeric value)  A malicious/deliberate attempt at misusing the script by an unauthorized user to purge patient data | Script asks to confirm the parameters entered including age and target folder (possibly using a popup) before starting/being deployed in the background  Script only accepts values for age > for example 3 days  Script only accepts numeric integer values for age and date  Deploying the script in the background requires administrator privileges  Appropriate documentation and training for users authorized to deploy this script |
| Root files/essential files in the specified directory that may accidentally meet criteria are deleted | Radiotherapy service delivery delayed due to failure of target folder. | Essential files are updated while the script is still running in the background. | Script only allows specific file types to be purged (DICOM etc.)  i.e. a more stringent criteria needs to be met before files or folders are purged. |
| The script running in the background cannot be killed when required |  | A user who may not be that familiar with the script finds it difficult to kill it | Killing the process running in the background could be made easier?  Script warns if an instance is already running in the background. |

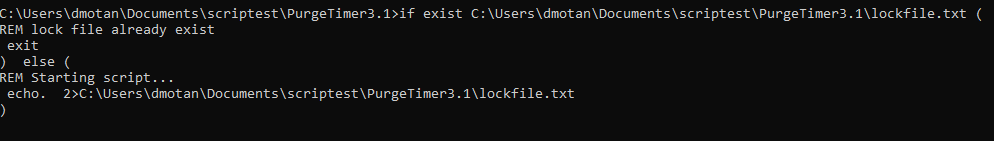
# Controls added in PurgeTimer V3.1:

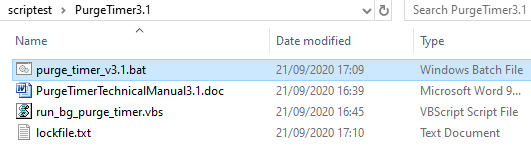
1. A lock file (lockfile.txt) is created when the script is first executed in the directory and this empty file must be manually deleted before running the script again. This guards against multiple instances of the script running in the background. If the file exists then the script execution exits immediately.
2. File Path Confirmation. A variable fullpath2 is set in the body of the script and must match fullpath1 otherwise the script execution exits immediately. This reduces the risk of accidentally deleting the contents of an incorrect folder.

## Testing:

### Lock File

To test if the lock file was functioning to prevent multiple instances of the script running in the background, an attempt was made at running multiple instances

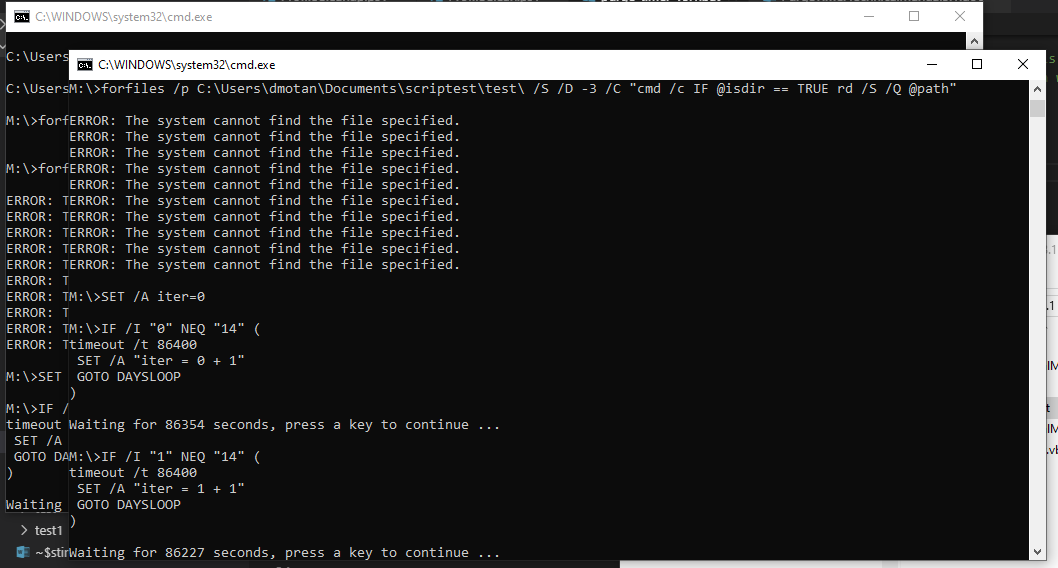




* lockfile.txt created when purge\_timer\_v3.1.bat run for the 1st time (as expected)
* purge\_timer\_v3.1.bat does not execute again when lockfile.txt exists
* even if the running script is killed and starting a new instance is attempted, the script does not execute or delete any files while the lockfile.txt is pre-existent in the directory.

**Deleting lockfile.txt while script is running:**

Manual deletion of the lockfile.txt while 1 instance of the script is running allows running another instance (as expected)



### File path confirmation:

To test if the file path confirmation was functioning as required, firstly the following correct condition was tested

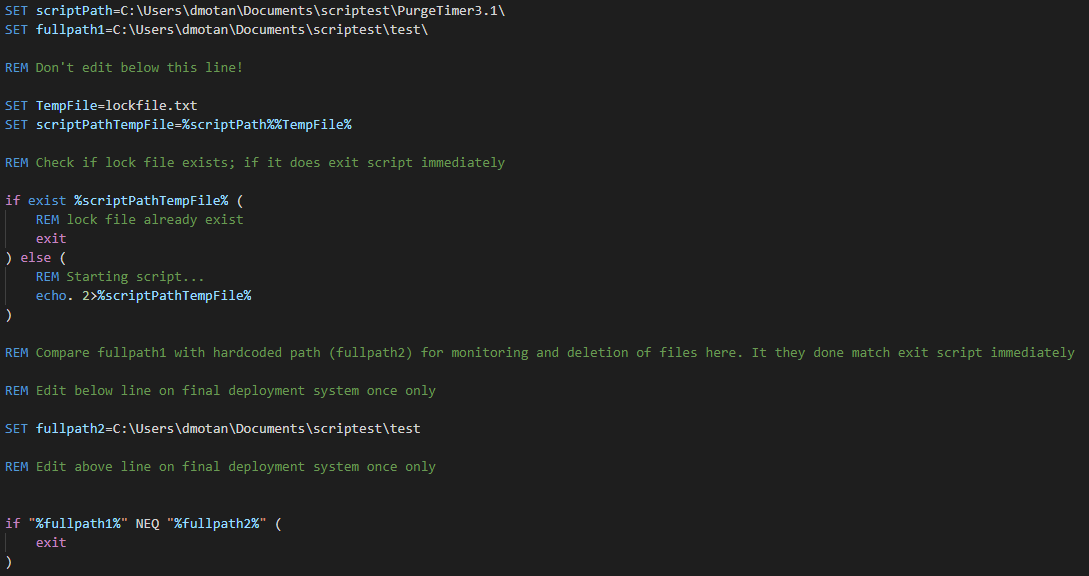
* fullpath1 and fullpath2 match:

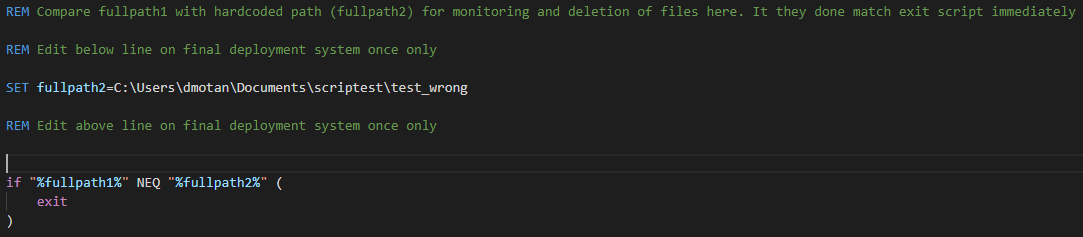
Script executes correctly (as expected)

---------------

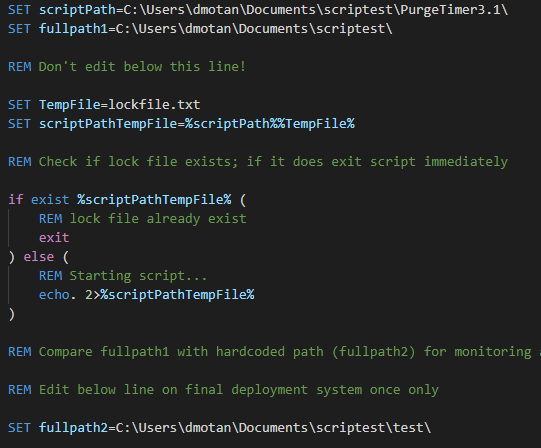
Next the following exception cases were tested to ensure that the file path confirmation is functioning to prevent the script from running when conditions are not met.

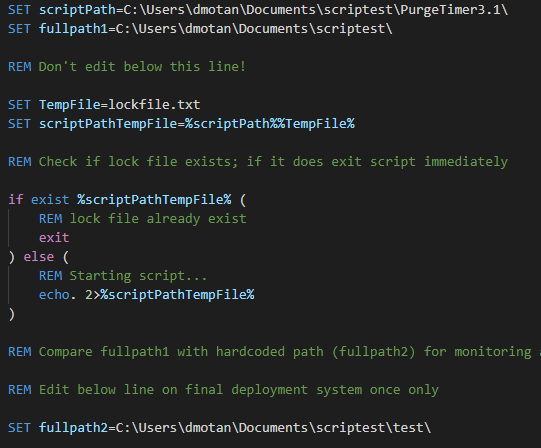
* fullpath2 set incorrectly:



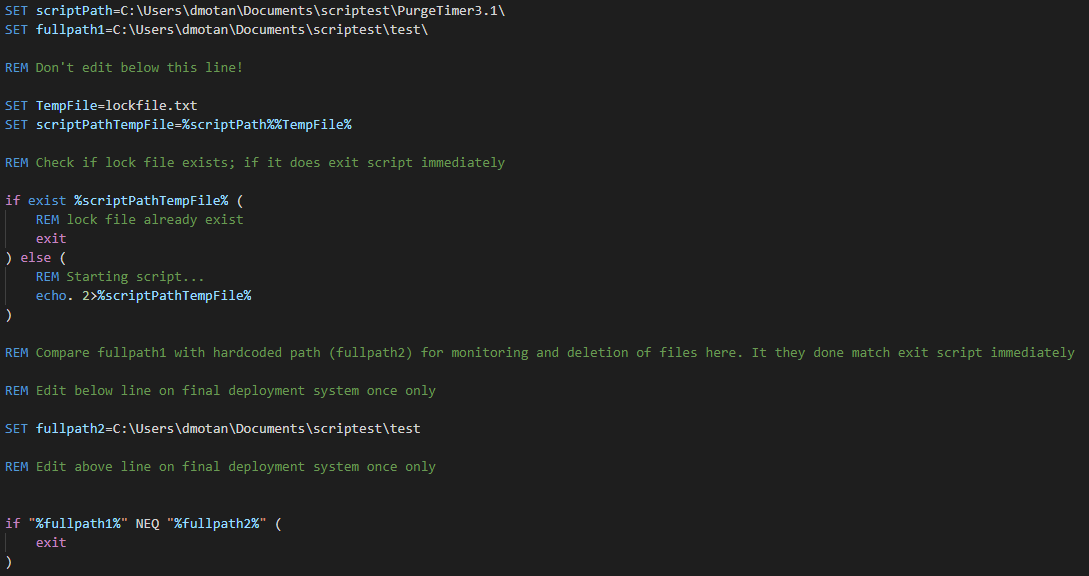


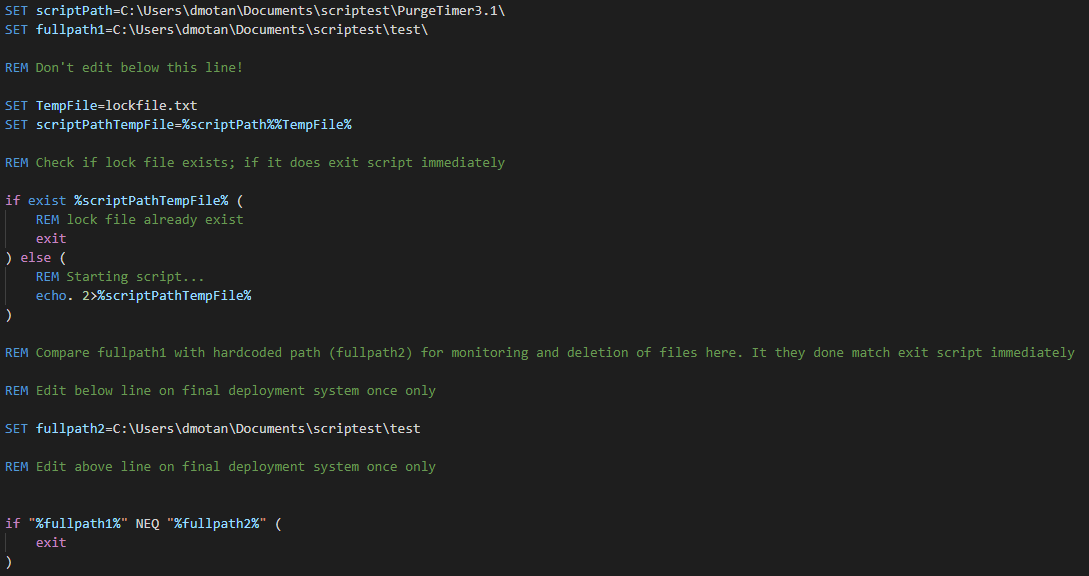
* fullpath1 set to a parent directory:



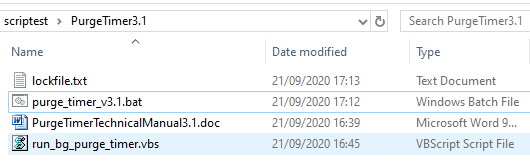


* ‘\’ left out from fullpath2



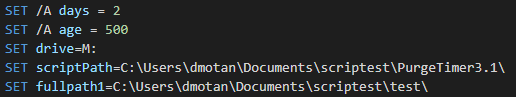


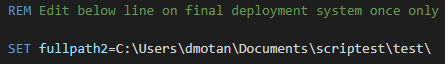
In all 3 cases above the Lockfile.txt is still created (shown below) but script exits without executing Purge (as expected)

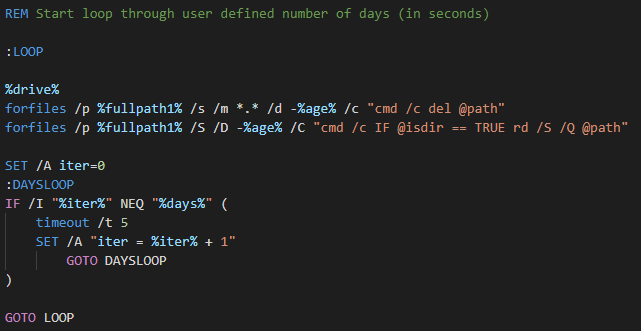


## Final integration testing:

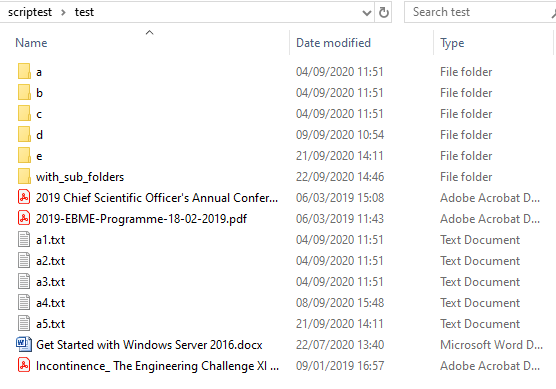
The script was then run using accelerated time and parameters set as shown below:



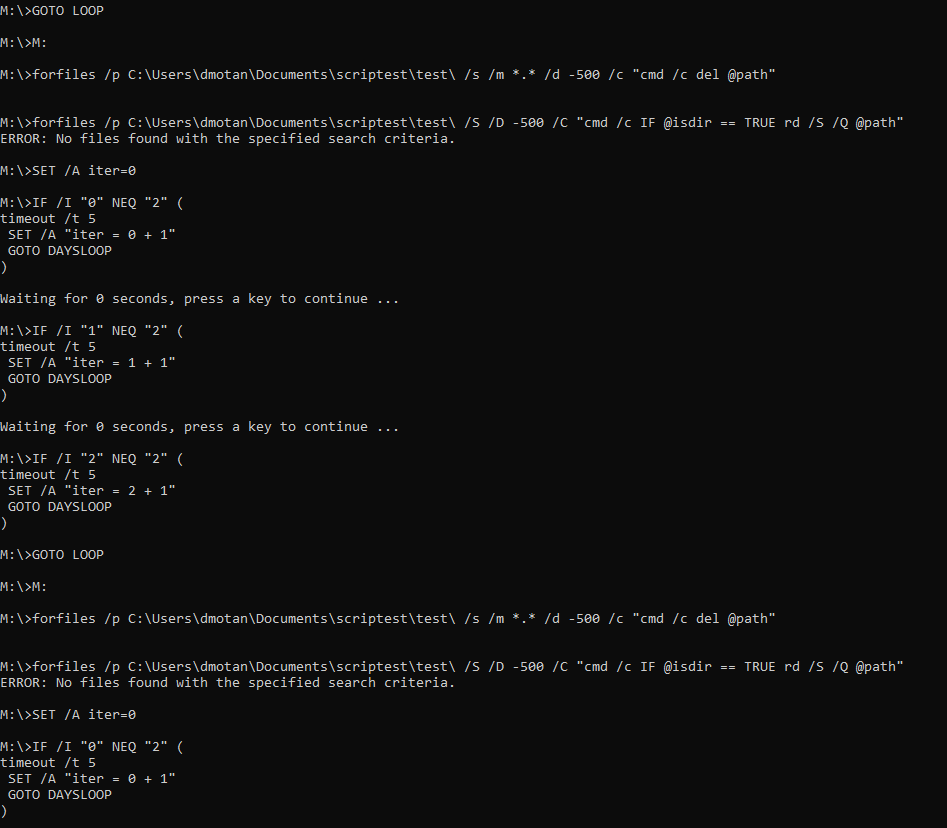




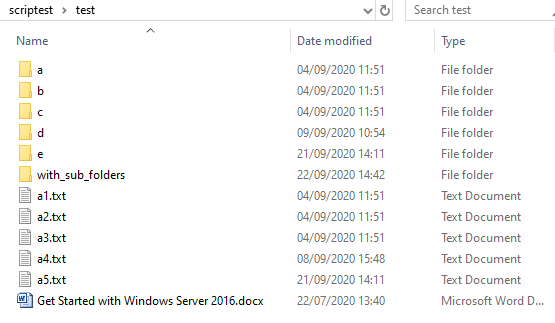
The test set of files is shown below:



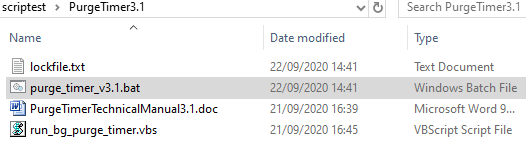
The script loops as expected (shown below)



The correct files are deleted (as shown below)



Finally, the lockfile.txt is also created as expected (shown below)



The same process was repeated by running the batch script in the background using the Virtual Basic script: *run\_bg\_purge\_timer.vbs,* and the same result was obtained.