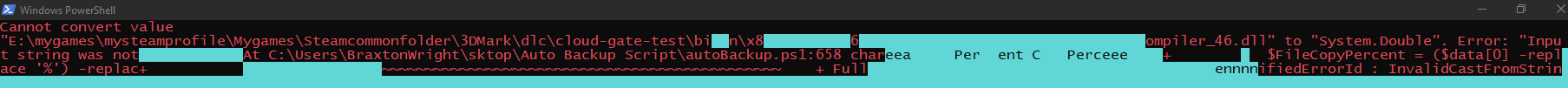
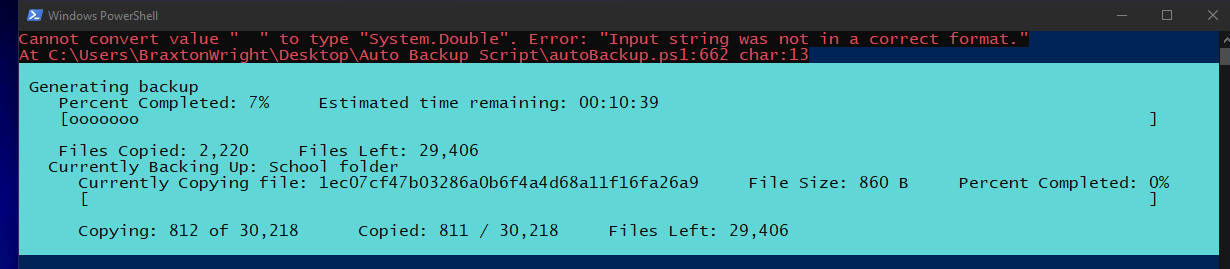
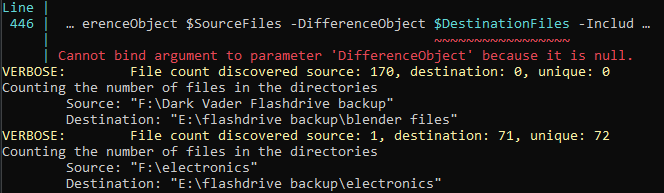
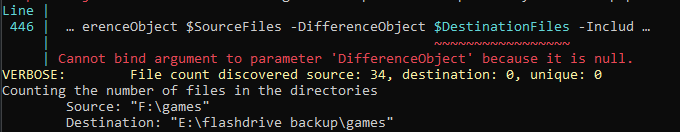
1. See about moving the variables used for the function Assert-Progress into the function Start-Backup so that we will only have one copy of the data that we will need to modify. Then inside the functions Get-RobocopyProgress and Copy-ItemWithProgress we will only have to modify the contents of the variables inside Start-Backup to change its information.
2. Need to see about implementing Start-Threadjob for the counting of the files before the backup progress to speed up the counting of the files. In addition, after you start the jobs, you need to make it so that it automatically starts backing up to make it as fast as possible and the jobs will return the file count after they are done counting, this is because of files like inside unity. One project along contains around 15,000 files. File explore both starts to copy the files while it is counting them and it updates the files remaining portion of the backup.

Also, see about debugging the process and see what is causing it to take so long. This is because when you try to do the job “Unity Projects” it will freeze at the counting of the files and I don’t know at what step it is causing the issue. If it is the counting of the files from the source/destination or comparing the objects.

1. See about finding some logic to better calculate the time remaining for the backup/restore process. It currently doesn’t take into account the size or copy rate of the files, only the number of files copied.
2. See if you can optimize the code to run faster? See if it needs optimized, test by timing how long it takes to copy a set of files using file explorer, a bare bone copy of the robocopy script, and your current iteration of the script.
3. Add a job summary once the backup/restore process has completed. Such as how long it took to complete, how many files were processed, ect. Like how the end of a robocopy, there is that summary table for the job.
4. There is some issue with copying files, it will get a string when it is getting the file size for the function Get-RobocopyProgress. At one time, I got this error in running the script, saying it can’t convert a " " to type "System.Double". However, I can’t seem to replicate this issue I received?
5. Need to see about adding some logic to allow users to continue with the backup/restore if a source directory does not exist. Such as a game that has not been saved and made the required file structure for said file saves. In addition, if a new set of destination paths have been defined but have not been backed up to, then they would not yet exist.
6. There is a bug on the line 446 of the file “AutoBackup.ps1”, I.E. the code “$FileCount = Compare-Object -ReferenceObject $SourceFiles -DifferenceObject $DestinationFiles -IncludeEqual |” because it is saying, “Cannot bind argument to parameter 'DifferenceObject' because it is null.” The cause of this is because; the destination path does not exist.  
   What I am thinking about implementing might fix this issue, instead of using Compare-Object to check to get a list of unique files discovered, we use what we have above inside the above if else statement and while we go through the items, we compare them to the other folder to check to see if they exist. If they do exist, we increment a counter to say there is a matching pair, and if there is not then, we increment a different counter saying it only exists inside the source/destination folder. As shown in the below images:  
   Compare-Object: C:\Users\<user>\Desktop\Auto-Backup-Utility\AutoBackup.ps1:446  
     
   Compare-Object: C:\Users\<user>\Desktop\Auto-Backup-Utility\AutoBackup.ps1:446  
     
   Ect.