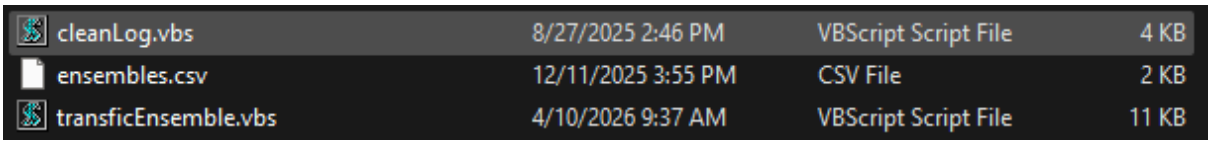
Documentation du Système de Transfert de Fichiers Automatisé

1. Vue d'ensemble de la conception (Architecture)

Ce système est conçu autour d'une architecture modulaire qui sépare la logique d'exécution, la configuration des tâches et la maintenance des journaux (logs). Il s'articule autour de trois composants principaux :

1. **Le fichier de routage (ensembles.csv)** : Un fichier d'index qui associe un identifiant de tâche (ex: "HEBDO-LUNDI-07H00") au fichier CSV contenant les détails de ce transfert spécifique.
2. **Le moteur d'exécution (transfer_ensemble.vbs)** : Le script principal qui lit la configuration, effectue les copies de fichiers, gère le renommage dynamique et le routage spécifique dans des sous-dossiers, tout en journalisant chaque action.
3. **L'outil de maintenance (cleanLog.vbs)** : Un script utilitaire permettant de purger les anciens journaux d'exécution pour éviter que les fichiers logs ne deviennent trop volumineux. (Une contrainte imposée pour ce dernier fut très difficile à mettre en place)



2. Fonctionnement Détaillé des Composants

A. Le Fichier de Configuration (ensembles.csv)

Ce fichier agit comme un annuaire centralisé.

- **Format** : Il s'agit d'un fichier texte avec comme séparateur le point-virgule (;).
- **Structure** : Il contient deux colonnes : ensembleID et FicTransfert.
- **Rôle** : Lorsqu'une tâche planifiée (ex: le planificateur de tâches Windows) lance le script, elle fournit un ensembleID. Le script cherche cet ID ici pour trouver le chemin du fichier CSV contenant la liste réelle des fichiers à transférer (par exemple, la ligne JOURNALIER-10H00;transferts\JOURNALIER-10H00.csv).

  *Il existera donc un fichier CSV par horaire de transfert (si un transfert doit se realiser le dimanche a 2H00 il aura un CSV pour lui tout seul afin de simplifier l'organisation dans un scheduler)

B. Le Script Principal (transfer_ensemble.vbs)

C'est le cœur du système. Il nécessite en argument l'identifiant de l'ensemble à traiter.

Étapes d'exécution :

1. **Initialisation du Log** : Le script crée ou ouvre un fichier log dédié dans le dossier logs\, nommé log_<ensembleId>.txt. Il y inscrit une en-tête d'exécution délimitée par des lignes de 80 signes =.

2. **Recherche de la configuration** : Il lit ensembles.csv pour trouver le fichier de transfert correspondant à l'ID fourni. Si l'ID ou le fichier de transfert est introuvable, une erreur est consignée et le script s'arrête.

3. **Exécution des transferts** : Il lit le fichier de transfert cible ligne par ligne (séparées par ;).
    - **Colonne 1 (Source)** : Le chemin du fichier à copier.
    - **Colonne 2 (Destination)** : Le dossier de destination.
    - **Colonne 3 (Nouveau Nom - Optionnel)** : Si renseigné, le fichier sera renommé à l'arrivée.
    - **Colonne 4 (Traitement Supplémentaire - Optionnel)** : Un mot-clé déclenchant une logique de dossier spécifique.

        *Ces traitements ont été implémentées suite a certaines demandes métier spécifiques

4. **Bilan** : Le script compte les succès et les échecs, puis inscrit un récapitulatif dans le journal.

Fonctionnalités avancées (Renommage Dynamique) : Si un nouveau nom est spécifié (Colonne 3), le script peut remplacer des variables dynamiques à la volée:

- %DATE% : Remplace par la date courte.
- %WEEK% : Remplace par le numéro de la semaine (ex: "S14").
- %YEAR% : Année sur 4 chiffres.
- %MONTH% : Mois sur 2 chiffres.
- %DAY% : Jour sur 2 chiffres.
- %TIME% : Heure au format système.

Routage intelligent (Traitements supplémentaires) :

La 4ème colonne permet d'activer deux comportements spéciaux qui modifient dynamiquement le dossier de destination :

- **traitsurvop** : Redirige le fichier dans une arborescence basée sur l'année puis la semaine en cours (ex: Destination\2026\S16). Le script crée ces dossiers s'ils n'existent pas.

- **trait_rouge** : Redirige vers un dossier nommé "Suivi PRV" suivi de l'année, puis un sous-dossier de la semaine contenant l'année courte (ex: Destination\Suivi PRV2026\S16(26)). Les dossiers sont également créés à la volée.

  *Bien que ces 2 traitements aient un fonctionnement extremement similaire, les utilisateurs cibles voulaient absolument le format YYYY/SWW(YY)

C. Le Script de Maintenance (**cleanLog.vbs**)

Les logs s'accumulant dans un seul fichier par ID, ce script permet de ne conserver que l'historique récent.

- **Arguments requis** : Il nécessite l'ID du log (logId) qui se construit automatiquement a partir du ensembleID et le nombre de jours à conserver (jours) definit selon le rythme de transfert (jour, semaine, bihebdo, hebdo).
- **Fonctionnement de la purge** :
  1. Il calcule une date limite (cutoffDate) en soustrayant le nombre de jours à la date actuelle.
  2. Il lit le log d'origine et utilise une expression régulière ("EX.CUTION:\s*(\d{2}/\d{2}/\d{4}\s\d{2}:\d{2}:\d{2})") pour détecter la date de chaque bloc d'exécution.(Cette methode n'est pas la plus securisee mais elle etait imposee par le cahier des charges)
  3. Il stocke chaque bloc en mémoire.
  4. Si la date du bloc est supérieure ou égale à la date limite, le bloc est conservé et écrit dans un fichier temporaire (.tmp).
  5. Une fois terminé, il supprime l'ancien fichier de log et renomme le fichier temporaire en tant que log officiel, et confirme que les anciennes exécutions ont été supprimées.
- **Robustesse** : Le script identifie les blocs en repérant les lignes séparatrices composées de 80 caractères = ou plus.

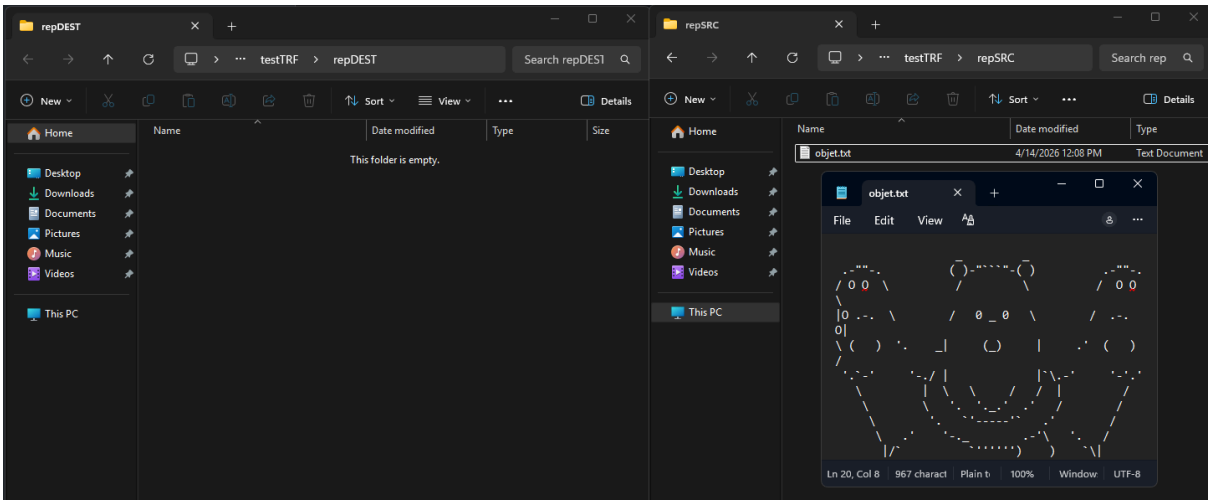
3. Synthèse de l'utilisation

Pour déployer et exploiter ce système :

1. **Préparation** : Assurez-vous que vos listes de transfert sont bien dans un dossier (ex: transferts\) et qu'elles sont correctement référencées dans ensembles.csv.
2. **Exécution** : Lancez via invite de commande ou tâche planifiée : cscript transfer_ensemble.vbs <ID_ENSEMBLE> (ex: cscript transfer_ensemble.vbs JOURNALIER-10H00).
3. **Nettoyage** : Planifiez le nettoyage régulier : cscript cleanLog.vbs <ID_ENSEMBLE> <JOURS_A_GARDER> (ex: cscript cleanLog.vbs JOURNALIER-10H00 30).

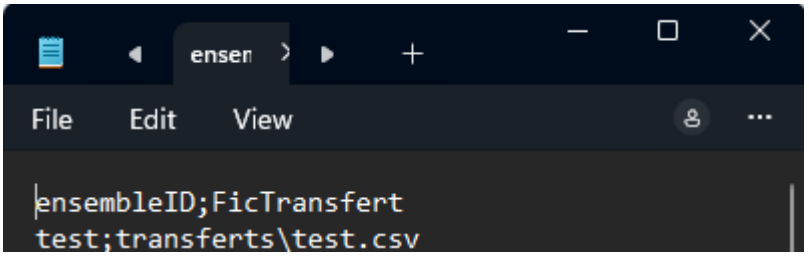
4. Exemple d’utilisation

On va faire une exemple assez simple



L’objectif va etre de copier le fichier objet.txt de repSRC vers repDEST

Pour cela je vais commencer par rajouter une ligne dans ensemble.csv



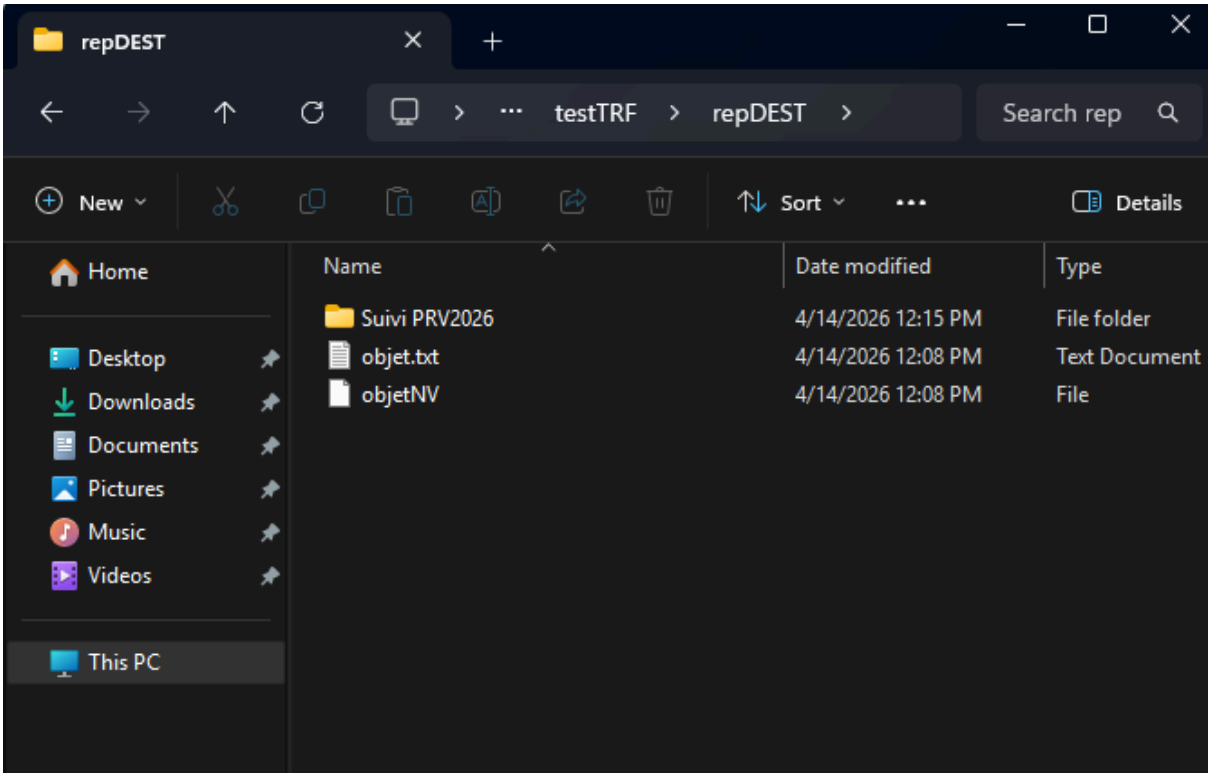
Ensuite je remplis le fichier test.csv avec un panel de fonctionnalités:

```
Source;Destination;Nouveau Nom;Traitement Sup
C:\testTRF\repSRC\objet.txt;C:\testTRF\repDEST;;
C:\testTRF\repSRC\objet.txt;C:\testTRF\repDEST;objetNV;
C:\testTRF\repSRC\objet.txt;C:\testTRF\repDEST;;trait_rouge
```

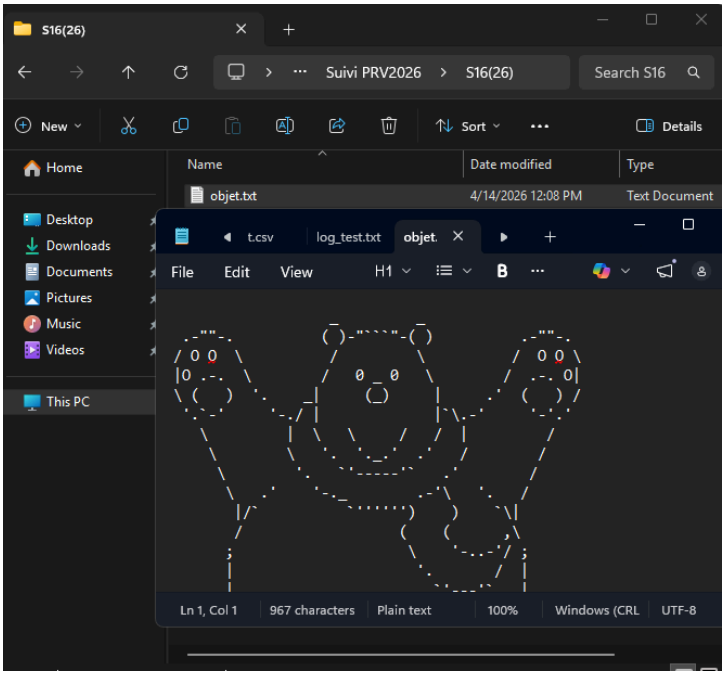
Et une fois enregistre plus qu’a executer, je vais le faire manuellement pour l’exemple, idealement l’agent qui veut rajouter un processus n’as qu’a rajouter une ligne dans le fichier horaire qui lui vonvient ou au pire copier un template de csv, rajouter une ligne dans ensemble.csv et mettre la commande dans le scheduler

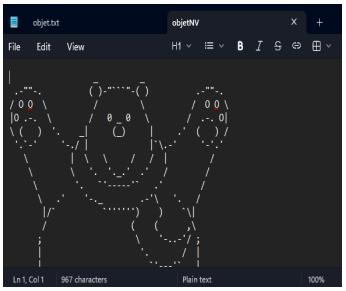
```
PS C:\Users\Utilisateur\Documents\content_portfolio\trf> cscript.exe .\transficEnsemble.vbs test
Microsoft (R) Windows Script Host Version 10.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

Une fois la commande execute voila le repDEST



Et si on ouvre les fichiers on voit bien que la donnée est bonne:





5. Conclusion

Ce projet m’aura permis d'étayer mes compétences en scripting bien que le VBS ne soit pas mon langage de coeur, j’ai su m’en sortir malgré des contraintes qui peuvent me paraître ridicule. Une solution de transfert existait déjà mais se faisait vieillissante, j’ai donc produit une solution robuste et modulable facilement par un utilisateur. Ce script fonctionne avec les partages réseaux et tournes sur un poste robot avec un utilisateur administrateur pour avoir accès aux données restreintes.