Les fichiers

Qu'est-ce qu'un fichier et comment est-il utilisé pour stocker des données en programmation ? Cet article explore les types de fichiers, leur structure, leur usage en programmation et les principes de manipulation des fichiers dans divers langages.

6TQ 5TTR 4TTR niveau

Un **fichier** est une unité de stockage utilisée pour enregistrer des informations de manière durable sur un support de mémoire (comme un disque dur, une clé USB ou un SSD). Contrairement aux données manipulées en mémoire vive (RAM), qui sont volatiles et disparaissent à la fermeture du programme ou de l'ordinateur, les fichiers permettent de sauvegarder les données pour un usage futur.

Définition d'un fichier

Un fichier est une séquence organisée de données stockée sur un support non volatil (disque dur, SSD, etc.). Il est identifié par un **nom** et une **extension** (comme `données.txt`, `image.jpg`, ou `base.csv`), qui indiquent son contenu ou son rôle.

Un fichier est constitué de :

1. **Données** : Le contenu réel du fichier, organisé en caractères, bits ou blocs binaires.
2. **Métadonnées** : Des informations sur le fichier (taille, date de modification, permissions) gérées par le système de fichiers de l'ordinateur.

Types de fichiers

Fichiers texte

Les fichiers texte stockent des données sous forme de caractères lisibles par les humains. Ils sont utilisés pour des formats simples comme `.txt`, `.csv` ou `.json`.

- **Exemple de contenu (fichier `.txt`)** :

```
Alice,25
Bob,30
Charlie,28
```

- **Cas d'utilisation** :

- Sauvegarde de configurations
- Stockage de données structurées dans des formats comme CSV ou JSON
- Scripts ou journaux

Fichiers binaires

Les fichiers binaires contiennent des données encodées en format binaire. Ils sont destinés à être interprétés par des programmes spécifiques. Exemples : fichiers d'image (`.jpg`), d'exécutable (`.exe`), de vidéo (`.mp4`).

- **Exemple de contenu binaire (non lisible directement)** :

  ```
  ÿØÿà JFIF ....Exif
  ```

- **Cas d'utilisation** :
  - Stockage de médias (images, audio, vidéo)
  - Fichiers d'applications (logiciels, bases de données binaires)
  - Sauvegarde de données complexes (par exemple, modèles d'apprentissage machine)

Comment les fichiers sont utilisés pour stocker des données en programmation

Les fichiers sont une méthode essentielle pour sauvegarder des informations générées ou utilisées par des programmes. Voici les étapes typiques pour utiliser un fichier en programmation.

Étape 1 : Création ou ouverture d'un fichier

Pour interagir avec un fichier, un programme doit d'abord **ouvrir** le fichier, soit pour le lire, soit pour y écrire.

- **Modes d'ouverture communs** :

Mode	Description
`r`	Lecture seule. Le fichier doit exister.
`w`	Écriture seule. Crée un fichier vide ou écrase l'existant.
`a`	Ajout à la fin du fichier (sans écraser).
`r+`	Lecture et écriture combinées.

Étape 2 : Lecture ou écriture des données

Les données peuvent être **lues** depuis un fichier pour être utilisées dans un programme ou **écrites** pour y être sauvegardées.

- Exemple de lecture d'un fichier texte ligne par ligne :

```python
with open("données.txt", "r") as fichier:
    for ligne in fichier:
        print(ligne.strip())  # Supprime les espaces et retours à la li
```

- Exemple d'écriture dans un fichier texte :

```python
with open("données.txt", "w") as fichier:
    fichier.write("Alice,25\n")
    fichier.write("Bob,30\n")
```

Étape 3 : Fermeture du fichier

Une fois les opérations terminées, le fichier doit être **fermé** pour libérer les ressources système. Dans de nombreux langages (comme Python), cela est automatisé grâce à des structures comme `with`.

Cas d'utilisation des fichiers en programmation

Stockage persistant des données

Lorsqu'un programme génère des données (comme des résultats de calcul, des journaux ou des scores de jeu), ces données doivent être sauvegardées pour éviter de les perdre après la fin du programme.

```python
# Sauvegarder une liste d'utilisateurs dans un fichier
utilisateurs = ["Alice,25", "Bob,30", "Charlie,28"]
with open("utilisateurs.txt", "w") as fichier:
    for utilisateur in utilisateurs:
        fichier.write(utilisateur + "\n")
```

Lecture de données pour configurer un programme

Un programme peut lire un fichier pour obtenir des paramètres ou des données d'entrée.

```python
# Charger les utilisateurs depuis un fichier
with open("utilisateurs.txt", "r") as fichier:
```

```
    for ligne in fichier:
        print("Utilisateur:", ligne.strip())
```

Sauvegarde de données structurées

Pour des structures complexes, les fichiers JSON ou CSV sont très courants.

- **Fichier JSON (JavaScript Object Notation)** :

```
[
    {"nom": "Alice", "age": 25},
    {"nom": "Bob", "age": 30}
]
```

Lecture et écriture en Python :

```
import json

# Écriture
data = [{"nom": "Alice", "age": 25}, {"nom": "Bob", "age": 30}]
with open("data.json", "w") as fichier:
    json.dump(data, fichier)

# Lecture
with open("data.json", "r") as fichier:
    data = json.load(fichier)
    print(data)
```

Gestion de fichiers binaires

Pour manipuler des fichiers binaires, les opérations sont similaires mais nécessitent un mode d'ouverture en binaire (`rb` ou `wb`).

```
# Copier un fichier binaire (exemple : image)
with open("image.jpg", "rb") as fichier_source:
    with open("copie_image.jpg", "wb") as fichier_cible:
        fichier_cible.write(fichier_source.read())
```

Fichiers et limitations

Bien que simples et efficaces, les fichiers présentent certaines limitations pour des applications complexes :

1. **Concurrence** : Plusieurs programmes ne peuvent pas écrire simultanément dans un même fichier sans risque de conflit.
2. **Structure** : Les fichiers nécessitent souvent un formatage manuel pour structurer les données (comme JSON ou CSV).
3. **Recherche inefficace** : Rechercher des données spécifiques dans un fichier volumineux peut être lent, surtout si les données ne sont pas indexées.

Pour surmonter ces limitations, on utilise des bases de données (comme SQLite ou MySQL) qui offrent des fonctionnalités avancées pour stocker, organiser et interroger les données.

Conclusion

Les fichiers sont une méthode fondamentale pour stocker et gérer des données en programmation. Ils offrent une solution simple et rapide pour les besoins de stockage persistant, qu'il s'agisse de données texte ou binaires. Cependant, pour des applications plus complexes ou collaboratives, les bases de données sont souvent préférées en raison de leurs fonctionnalités avancées. Maîtriser les fichiers est néanmoins une compétence essentielle pour tout développeur.