Enregistrement des données : sauvegarde dans un fichier (Write)

L'opération de sauvegarde des données est l'opération complémentaire de la lecture. De nouvelles données doivent être enregistrées sur le disque dur en vue d'une consultation future. Le format de sauvegarde peut être de type texte ou de type binaire. Nous présentons ici la sauvegarde des données dans des formats texte.

Comme dans le cas de l'opération de lecture, il faut au préalable définir dans le programme un descripteur de fichier servant de point d'entrée pour les opération d'écriture. On effectue ce qu'on appelle une ouverture en mode "écriture".

Python

```
try :
  f = open('monfichier.dat','w')
except IOError:
  print "Erreur d'ouverture!!"
```

Java

```
try{
  FileWriter d = new FileWriter ('monfichier.dat');
  BufferedWriter f = new BufferedWriter(d);
  }
catch (Exception e){
  System.out.println("Problème d'ouverture!");
  }
```

On notera qu'il existe en python (ainsi qu'en C, C++, ...) plusieurs modes d'ouverture exprimés par le deuxième argument de la fonction open. On retiendra le mode 'w' (création d'un nouveau fichier vide) et le mode 'a' (ajout de nouvelles données à la suite d'un fichier déjà existant).

La sauvegarde dans le fichier s'effectue à l'aide d'un opérateur d'écriture. Dans le cas des chaînes de caractères, l'opérateur d'écriture sauvegarde ces données à la suite des données déjà écrites.

Python

```
f.write("Bonjour!\n")
```

Java

```
f.write("Bonjour!\n");
```

La sauvegarde de données nécessite d'effectuer un choix sur le mode d'encodage, obéissant en général à une norme bien précise (csv, json, xml, etc...). Voir section 1.3.

Une fois les opérations de lecture ou d'écriture terminées, il est nécessaire de fermer le fichier. L'opération de fermeture assure que les données sont effectivement enregistrées sur le disque (et non simplement stockées dans la mémoire tampon - voir section XXX).

Algorithmes de bas niveau (niveau système d'exploitation)

Lors de la création (et mise à jour) d'un fichier, on lui alloue un espace sur le volume. Plusieurs régions peuvent être allouées pour un seul fichier) La taille minimale d'une région d'allocation étant une page, la région allouée à un fichier est composée d'un nombre entier de pages consécutives.

Remarque : le volume possède une table des pages libres (table de bits : chaque bit correspond à une page. Pour allouer, on fait passer le bit de 0 à 1, et on ajoute l'adresse de la page au descripteur de fichier). Lorsque tout ou partie d'un fichier est effacé, certaines pages sont libérées et introduites dans la liste des pages libres.

Stratégies d'allocation

On souhaite :

- minimiser le nombre de régions allouées à un fichier
- minimiser la distance entre deux régions successives

Stratégie par page :

- premier trouvé (la première page libre dans la table)
- meilleur choix (le plus proche de la dernière page allouée)



Stratégie par bloc: on alloue des blocs composés de plusieurs pages (S'il existe des blocs libres consécutifs, ils sont fusionnés en un seul)

- Plus proche choix : la liste des blocs libres est parcourue jusqu'à trouver un bloc de la taille demandée (ou sinon, le premier bloc de taille supérieure, qui est alors découpé en deux blocs).
  - first fit : le premier bloc suffisamment grand pour les besoins
  - best fit : le plus petit bloc qui ait une taille au moins égale à la taille demandée
  - worst fit : le plus grand bloc disponible (qui est donc découpé)

On alloue des blocs de 1,2,4,8,...2K pages. Pour une taille donnée $2i-1 < n < 2i$, on commence par chercher les blocs de taille 2i, puis 2i+1, ... jusqu'à 2K , en divisant ces blocs le cas échéant.

Problème des stratégies par bloc: s'il y a trop de fichiers, on obtient des blocs de taille 1 –> nécessité de réorganiser (défragmenter)

Lecture/Ecriture

Les opérateurs lire_ligne (readline) et écrire_ligne (write) ne travaillent pas directement sur les données du fichier. Les données du fichier sont chargées en mémoire centrale dans une mémoire "tampon". L'objet f servant à décrire le fichier a comme attributs :

- t : table des pages,
- i : numero de la page courante,
- p : tableau d'octets de la page courante (mémoire tampon),

- j : position dans la page courante (tête de lecture).



Lors des opération de lecture, la mémoire tampon est mise à jour au fur et à mesure qu'on avance dans la lecture du fichier par des opérations de lecture sur le disque. En général, plusieurs pages sont chargées en avance. Lors d'une opération d'écriture, la mémoire tampon reçoit les nouvelles données à écrire. Ces données sont effectivement écrites sur le disque lorsque le tampon est suffisamment rempli ou lors de l'opération de fermeture. Au moment de l'écriture effective, le système d'exploitation fait appel à un opérateur d'allocation pour choisir le "meilleur" bloc où stocker les données.

Previous : Consultation des données (Read) Next : Aspect logique

From:
https://wiki.centrale-med.fr/informatique/ - **WiKi informatique**

Permanent link:
https://wiki.centrale-med.fr/informatique/public:std-3:cm1:aspect_physique:2.1.5_fichiers_et_repertoires:enregistrement_des_donnees_write

Last update: **2016/08/31 14:33**