# Unix : concepts et outils avancés
# TD no 5 : système de fichiers, suite


Olivier Aumage
LIP, bureau 302
email : `oaumage@ens-lyon.fr`


25 octobre 2001

## 1 Fichiers longs

- Écrivez un programme `long_fichier` qui créé un fichier (le nom est passé en argument 2 du programme), effectue un `lseek` d'une longueur passée en argument 1 du programme, écrit un caractère dans le fichier, et enfin ferme le fichier (utilisez les appels systèmes d'entrées/sorties).
- A l'aide de la commande `df /var/tmp`, déterminez le nombre de blocs utilisés pour diverses valeurs d'offset de `lseek`. Vous utiliserez pour cela le programme que vous venez d'écrire et vous créerez vos fichiers dans le répertoire `/var/tmp` (pour éviter les perturbations extérieures).
- Après avoir exécuté `long_fichier 10000000 /var/tmp/tst`, déterminez le nombre de blocs utilisés par `tst2` et `tst3`, avec :
  - `tst2` : `cp tst tst2`
  - `tst3` : `cat tst > tst3`

## 2 Gestion de flots

L'objectif de ce second exercice est de recréer une gestion de flot élémentaire *au-dessus* des appels systèmes d'entrées/sorties, à la manière des fonctions `fopen, fwrite,...` Nous ne nous intéresserons qu'à l'écriture, la lecture étant très similaire. Il n'est pas demandé d'ajouter un support pour une éventuelle fonction `my_fseek`.

- Définissez un type `my_FILE` permettant de représenter un flot. Ce type devra contenir un tampon pour l'écriture ainsi que toutes les données supplémentaires que vous jugerez nécessaires.

```
typedef struct s_my_FILE
{
  char *buffer;
  ? ? ?
} my_FILE;
```

- Écrivez les fonctions suivantes :

```
/* my_fopen:
 * Alloue un flot, alloue un tampon, ouvre le fichier.
 * Retourne un pointeur sur le flot alloué, ou NULL en cas d'erreur.
 */
my_FILE *my_fopen(char *path);

```

```
/* my_fclose:
 * Ferme un flot après avoir vidé son tampon et ferme le fichier correspondant.
 * Retourne 0 si OK, ou -1 en cas d'erreur.
 */
int my_fclose(my_FILE* flot);

/* my_fwrite:
 * Écrit 'len' octets dans le tampon de flot. Si celui-ci est plein,
 * le vider dans le fichier et poursuivre le processus.
 * Retourne le nombre d'octets écrits si OK, ou -1 en cas d'erreur.
 */
int my_fwrite(my_FILE* flot, char *ptr, int len);

/* my_fflush:
 * Vide le tampon du flot.
 * Retourne 0 si OK, ou -1 en cas d'erreur.
 */
int my_fflush(my_FILE* flot);
```

– Vérifiez le bon fonctionnement de l'ensemble, notamment dans le cas ou la valeur du paramètre `len` de `my_fwrite` est supérieure à la taille du tampon de votre objet flot.
– Quelles seraient les modifications à apporter à l'ensemble si on souhaitait implémenter une fonction `my_fseek` ? (Il n'est pas demandé de programmer ces modifications).

## 3 Répertoires - Disk Usage

– A l'aide des fonctions `opendir`, `readdir`, `closedir` et des fonctions `stat` et `lstat`, écrivez un progamme `my_du` qui calcule la taille totale d'une arborescence dont la racine est passée en argument (sur le principe de la commande `du`, *Disk Usage*). **Attention :** si un fichier possède plusieurs liens physiques, il ne faudra compter qu'une seule fois la place occupée.