

Attributs de fichiers

Exercice 1

1) Écrire une fonction qui affiche tous les attributs d'un fichier passé en paramètre.

- type,
- droits,
- nombre de liens,
- propriétaire,
- groupe,
- taille,
- date de dernière modification
- nom du fichier.

Si le fichier est introuvable, la fonction renvoie -1. Nous utiliserons ici l'appel-système `stat`. Le champ `st_mode` de `stat` est fabriqué de la manière suivante :

1	2	3	4	5	6	7	r	w	x	r	w	x	r	w	x
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- Les bits 1 à 4 représentent le type du fichier.
- Le bit 5 représente le `setuid` bit.
- Le bit 6 représente le `setgid` bit.
- Le bit 7 représente le `sticky` bit.
- les bits 8 à 16 représentent les droits. Pour ceux-ci nous avons les constantes symboliques suivantes :

utilisateur	r	S_IRUSR	S_IRWXU
	w	S_IWUSR	
	x	S_IXUSR	
groupe	r	S_IRGRP	S_IRWXG
	w	S_IWGRP	
	x	S_IXGRP	
autres	r	S_IROTH	S_IRWXO
	w	S_IWOTH	
	x	S_IXOTH	

Les droits `rwrx-rx-` correspondent donc à la disjonction `S_IRWXU | S_IRGRP | S_IXGRP | S_IROTH`.

Le type du fichier peut être isolé grâce au masque symbolique `_S_IFMT`.

si (<code>st_mode & _S_IFMT</code>) vaut	Type de fichier	fonction
<code>_S_IFREG</code>	fichier régulier	<code>S_ISREG</code>
<code>_S_IFBLK</code>	fichier mode bloc	<code>S_ISBLK</code>
<code>_S_IFCHR</code>	fichier mode caractère	<code>S_ISCHR</code>
<code>_S_IFDIR</code>	répertoire	<code>S_ISDIR</code>
<code>_S_IFIFO</code>	tube nommé	<code>S_ISFIFO</code>

On utilisera l'appel système `localtime` dont la structure renvoyé est la suivante :

```
struct tm
{
    int      tm_sec;          /* secondes */
    int      tm_min;          /* minutes */
```

```

    int    tm_hour;        /* heures */
    int    tm_mday;        /* jour du mois */
    int    tm_mon;         /* mois */
    int    tm_year;        /* annee */
    int    tm_wday;        /* jour de la semaine */
    int    tm_yday;        /* jour de l'annee */
    int    tm_isdst;
};

```

Enfin les appels système `getpuid` et `getgrgid` permettront d'obtenir le nom de l'utilisateur (resp. du groupe) à partir de l'uid (resp. du gid).

2) Écrire une commande `lsl <rep>` se comportant comme la commande `ls -li`. Cette commande utilisera les appels-système `opendir`, `readdir`, `closedir` définis dans le fichier standard `<dirent.h>`. Elle utilisera la fonction définie dans la question précédente.

Exercice 2 Écrire une commande `ccp` qui permet de copier un fichier. Cette commande pourra avoir 3 options (eventuellement combinables).

ccp -v vérifie si le fichier destination n'existe pas déjà. Si c'est le cas un message d'erreur est renvoyé.

ccp -a Copie en mode append. Le contenu du fichier source est recopié à la fin du fichier destination si celui-ci existe.

ccp -o <off1> <off2> Dans ce cas seuls sont copiés les caractères du fichier source compris entre les positions `off1` et `off2`.