NOM
stat, fstat, lstat - Obtenir le statut d'un fichier (file status).

SYNOPSIS
```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int stat(const char *file_name, struct stat *buf);
int fstat(int filedes, struct stat *buf);
int lstat(const char *file_name, struct stat *buf);
```

DESCRIPTION
Ces fonctions renvoient des informations à propos du fichier indiqué. Vous n'avez besoin d'aucun droit d'accès au fichier pour obtenir les informations, mais vous devez avoir le droit de parcours de tous les répertoires mentionnés dans le chemin menant au fichier.

stat récupère le statut du fichier pointé par file_name et remplit le buffer buf.

lstat est identique à stat, sauf que dans le cas d'un lien symbolique, il donne l'état du lien lui-même plutôt que celui du fichier visé.

fstat est identique à stat, sauf que le fichier ouvert est pointé par le descripteur filedes, obtenu avec open(2).

Les trois fonctions retournent une structure stat contenant les champs suivants :

```
struct stat {

    dev_t     st_dev;     /* Périphérique                 */
    ino_t     st_ino;     /* Numéro i-noeud               */
    mode_t    st_mode;    /* Protection                   */
    nlink_t   st_nlink;   /* Nb liens matériels           */
    uid_t     st_uid;     /* UID propriétaire             */
    gid_t     st_gid;     /* GID propriétaire             */
    dev_t     st_rdev;    /* Type périphérique            */
    off_t     st_size;    /* Taille totale en octets      */
    blksize_t st_blksize; /* Taille de bloc pour E/S      */
    blkcnt_t  st_blocks;  /* Nombre de blocs alloués      */
    time_t    st_atime;   /* Heure dernier accès          */
    time_t    st_mtime;   /* Heure dernière modification */
    time_t    st_ctime;   /* Heure dernier changement     */
};
```

Le champ st_size indique la taille du fichier (s'il s'agit d'un fichier régulier ou d'un lien symbolique) en octets. La taille d'un lien symbolique est la longueur de la chaîne représentant le chemin d'accès qu'il vise, sans le caractère NUL final.

La valeur st_blocks donne la taille du fichier en blocs de 512 octets. (Cette valeur peut être plus petite que st_size/512 par exemple si le fichier contient des trous). La valeur st_blksize indique la taille de bloc "préférée" pour les entrées/sorties du système (l'écriture dans un fichier par petits morceaux peut induire de nombreuses étapes lecture-modification-écriture peu efficaces).

Les systèmes de fichiers de Linux n'implémentent pas tous les champs "time". Certains systèmes de fichiers autorise le montage de telle manière que les accès ne modifient pas le champ st_atime (voir l'option 'noatime' de mount(8)).

Le champ st_atime est modifié par les accès au fichier, c'est à dire avec exec(2), mknod(2), pipe(2), utime(2) et read(2) (d'au moins un octet). D'autres routines, comme mmap(2), peuvent ou non mettre à jour ce champ st_atime.

Le champ st_mtime est modifié par des changements sur le fichier lui-même, c'est à dire mknod(2), truncate(2), utime(2) et write(2) (d'au moins un octet). D'autre part le champ st_mtime d'un répertoire est modifié lors de la création ou la suppression de fichiers en son sein. Le champ st_mtime n'est généralement pas mis à jour lors de modification de propriétaire, groupe, mode ou nombre de liens physiques.

Le champ st_ctime est modifié lors d'une écriture, une lecture, ou une modification de données concernant l'i-noeud (propriétaire, groupe, mode, etc...)

Les macros POSIX suivantes sont fournies pour vérifier le type de fichier :

S_ISREG(m) un fichier régulier ?

S_ISDIR(m) un répertoire ?

S_ISCHR(m) un péripherique en mode caractère ?

S_ISBLK(m) un périphérique en mode blocs ?

S_ISFIFO(m) une FIFO ?

S_ISLNK(m) est-ce un lien symbolique ? (Pas dans POSIX.1-1996).

S_ISSOCK(m) une socket ? (Pas dans POSIX.1-1996).

Les attributs suivants correspondent au champ st_mode:

S_IFMT	00170000	masque du type de fichier
S_IFSOCK	0140000	socket
S_IFLNK	0120000	lien symbolique
S_IFREG	0100000	fichier régulier
S_IFBLK	0060000	périphérique blocs
S_IFDIR	0040000	répertoire
S_IFCHR	0020000	périphérique caractères
S_IFIFO	0010000	fifo
S_ISUID	0004000	bit Set-UID
S_ISGID	0002000	bit Set-Gid
S_ISVTX	0001000	bit "sticky"
S_IRWXU	00700	lecture/écriture/exécution du propriétaire
S_IRUSR	00400	le propriétaire a le droit de lecture
S_IWUSR	00200	le propriétaire a le droit d'écriture
S_IXUSR	00100	le propriétaire a le droit d'exécution
S_IRWXG	00070	lecture/écriture/exécution du groupe
S_IRGRP	00040	le groupe a le droit de lecture
S_IWGRP	00020	le groupe a le droit d'écriture
S_IXGRP	00010	le groupe a le droit d'exécution
S_IRWXO	00007	lecture/écriture/exécution des autres
S_IROTH	00004	les autres ont le droit de lecture
S_IWOTH	00002	les autres ont le droit d'écriture
S_IXOTH	00001	les autres ont le droit d'exécution

Le bit Set-GID (S_ISGID) a plusieurs utilisations particulières : pour un répertoire, il indique que la sémantique BSD doit être appliquée en son sein, c'est à dire que les fichiers qui y sont créés héritent leur GID du répertoire et non pas du GID effectif du processus créateur, et les sous-répertoire auront automatiquement le bit S_ISGID actif. Pour les fichiers qui n'ont pas d'autorisation d'exécution pour le groupe (S_IXGRP non actif), ce bit indique qu'un verrouillage strict est en vigueur sur ce fichier.

Le bit 'sticky' (S_ISVTX) sur un répertoire indique que les fichiers qui s'y trouvent ne peuvent être renommés ou effacés que par leur propriétaire, par le propriétaire du répertoire ou par root.

VALEUR RENVOYÉE

Ces fonctions retournent zéro si elles réussissent. En cas d'echec -1 est renvoyé, et errno contient le code d'erreur.

ERREURS

EBADF filedes est un mauvais descripteur.

ENOENT Un composant de file_name n'existe pas, ou il s'agit d'une chaine vide.

ENOTDIR Un composant du chemin d'accès n'est pas un répertoire.

ELOOP Trop de liens symboliques rencontrés dans le chemin d'accès.

EFAULT Un pointeur se trouve en dehors de l'espace d'adressage.

EACCES Autorisation refusée.

ENOMEM Pas assez de mémoire pour le noyau.

ENAMETOOLONG
Nom de fichier trop long

EXEMPLE

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>

int main (int nb_args, char * args [])
{
 struct stat                                             sts;

 if (nb_args != 2) {
           fprintf (stderr, "syntaxe : %s <fichier>\n", args [0]);
           exit (1);
 }

 if ( stat (args [1], & sts) != 0) {
           fprintf (stderr, "%s : erreur %X\n", args [0], errno);
           exit (1);
        }

 fprintf (stdout, "Périphérique : %d\n", sts . st_dev);
 fprintf (stdout, "Noeud : %ld\n", sts . st_ino);
 fprintf (stdout, "Protection : %o\n", sts . st_mode);
 fprintf (stdout, "nb liens matériels: %d\n", sts . st_nlink);
 fprintf (stdout, "ID propriétaire : %d\n", sts . st_uid);
 fprintf (stdout, "ID groupe: %d\n", sts . st_gid);
 fprintf (stdout, "Taille : %lu octets\n", sts . st_size);
 fprintf (stdout, "Taille de bloc : %lu\n", sts . st_blksize);
 fprintf (stdout, "Nombre de blocs : %lu\n", sts . st_blocks);
}
```

CONFORMITÉ

Les appels stat et fstat sont conformes aux versions SVr4, POSIX, X/OPEN, BSD 4.3. L'appel lstat est conforme aux versions BSD 4.3 et SVr4. SVr4 mentionne des conditions d'erreurs supplémentaires pour fstat EINTR, ENOLINK, et EOVERFLOW. Pour stat, etlstat SVr4 indique les conditions supplémentaires EACCES, EINTR, EMULTIHOP, ENOLINK, et EOVERLOW. L'utilisation des champs st_blocks et st_blksize risque d'être moins portable. Ils ont été introduit dans BSD, et ne sont pas mentionnée dans POSIX. Leur interprétation change suivant les systèmes, voire sur un même système s'il y a des montages NFS.

Posix ne décrit pas les bits S_IFMT, S_IFSOCK, S_IFLNK, S_IFREG, S_IFBLK, S_IFDIR, S_IFCHR, S_IFIFO, S_ISVTX, mais réclame d'utiliser les macros S_ISDIR(), etc. Les macros S_ISLKNK et S_ISSOCK ne se trouvent pas dans POSI.1-1996 mais seront présentes dans le prochain standard POSIX. La première vient de SVID 4v2, la seconde de SUSv2.

Unix V7 (et les systèmes suivants) propose S_IREAD, S_IWRITE, S_IEXEC, là où POSIX préfère leurs synonymes S_IRUSR, S_IWUSR, S_IXUSR.

AUTRES SYSTÈMES

Voici quelques valeurs qui ont été (ou sont) utilisées sur d'autres systèmes

hex	nom	ls	octal	description
f000	S_IFMT		170000	Masque du type de fichier
0000			000000	SCO out-of-service inode, BSD unknown type SVID-v2 and XPG2 have both 0 and 0100000 for ordinary file
1000	S_IFIFO	p|	010000	fifo (tube nommé)
2000	S_IFCHR	c	020000	fichier spécial caractère (V7)
3000	S_IFMPC		030000	fichier spécial caractère multiplexé (V7)
4000	S_IFDIR	d/	040000	répertoire (V7)
5000	S_IFNAM		050000	fichier spécial nommé XENIX avec deux sous-types distingués par st_rdev valant 1 ou 2:
0001	S_INSEM	s	000001	sous-type sémaphore de IFNAM XENIX
0002	S_INSHD	m	000002	sous-type données partagées de IFNAM XENIX
6000	S_IFBLK	b	060000	fichier spécial bloc (V7)
7000	S_IFMPB		070000	fichier spécial bloc multiplexé (V7)
8000	S_IFREG	-	100000	fichier normal (V7)
9000	S_IFCMP		110000	compressé VxFS
9000	S_IFNWK	n	110000	fichier spécial réseau (HP-UX)
a000	S_IFLNK	l@	120000	lien symbolique (BSD)
b000	S_IFSHAD		130000	Fichier shadow Solaris pour l'ACL (invisible)
c000	S_IFSOCK	s=	140000	socket (BSD; aussi "S_IFSOC" sur VxFS)
d000	S_IFDOOR	D>	150000	Solaris door
e000	S_IFWHT	w%	160000	BSD whiteout (non utilisé pour les inoeuds)
0200	S_ISVTX		001000	'sticky bit': garder en mémoire après exécution (V7) réservé (SVID-v2) non-répertoires : ne pas swapper le fichier (SunOS) répertoires : restrictions d'effacement (SVID-v4.2)
0400	S_ISGID		002000	Utiliser l'ID du groupe à l'exécution (V7) répertoires : sémantique BSD propageant le GID
0400	S_ENFMT		002000	Verrouillage strict Système V (partage S_ISGID)
0800	S_ISUID		004000	Utiliser l'ID utilisateur à l'exécution (V7)
0800	S_CD		004000	Fichier répertoire dépendant du contexte (HP-UX)

Une commande sticky est apparue dans la version 32V d'AT&T UNIX.

VOIR AUSSI

chmod(2), chown(2), readlink(2), utime(2)

TRADUCTION

Christophe Blaess, 1997.