Operações sobre ficheiros

- Abertura e fecho:

  `fd = open()`: `fd` descriptor do ficheiro aberto.

  `close(fd)`: fecho do ficheiro.

- Leitura e escrita:

  `read(fd, buf, size)`: leitura

  `write(fd, buf, size)`: escrita

- Deslocar a posição corrente do descriptor:

  `lseek(fd, offset, flag)`: desloca `fd` de `offset` bytes relativo a `flag`. Valor de `flag`:

  `SEEK_SET` início do ficheiro (= 0).
  `SEEK_CUR` posição corrente no ficheiro.
  `SEEK_END` fim do ficheiro.

- Truncar ficheiros:

  `ftruncate(fd, tam)`: o tamanho do ficheiro fica em `tam`, perdendo-se toda a informação para lá do novo fim de ficheiro.

Obter informação de um nó-i

O Linux permite obter informação sobre um nó-i usando uma das funções:

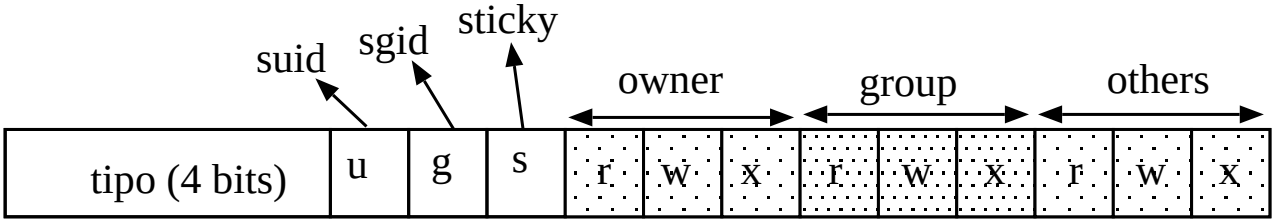
```
int stat(pathname, statbuf)
int lstat(pthname, statbuf)
int fstat(fd, statbuf)
```

`stat()` e `lstat()` procuram com base no caminho do ficheiro. A 1a segue links simbólicos e a 2a. não. `fstat()` usa o descriptor de um ficheiro aberto para retornar informação sobre o seu nó-i. O argumento `statbuf` é uma estrutura como a seguinte:

```
struct stat {
  dev_t  st_dev; // num. periférico onde reside o fich.
  ino_t  st_ino; // núm. nó-i no disco (único no periférico)
  mode_t st_mode;// permissões e tipo de ficheiro
  nlink  st_link;// num. caminhos que referem este nó-i.
  uid_t  st_uid; // user ID criador do fich.
  gid_t  st_gid; // group ID
  dev_t  st_rdev;// para caracterizar fich. como especial
  off_t  st_size;// tamanho do fich. em bytes
  unsigned long st_blksize;// tam. bloco para o SF
  unsigned long st_blocks;// num. blocos atribuídos ao fich
  time_t st_atime;// data/hora último acesso
  time_t st_mtime;// data/hora última modificação no fich.
  time_t st_ctime;// data/hora última mod. dos atributos
}
```

Determinar o tipo de ficheiro

O campo `st_mode` de uma estrutura `struct stat` obtido através da função `stat()`, tem a seguinte estrutura de 16 bits:



Para obter informação sobre o tipo de ficheiro (os 4 1os bits) aplicar a mascara (e.g. `x.st_mode & S_IFMT`) e depois verificar o que se obtém:

`S_IFSOCK`	socket
`S_IFLNK`	link simbólico
`S_IFREG`	ficheiro regular
`S_IFBLK`	block device
`S_IFDIR`	directório
`S_IFCHR`	character device
`S_IFIFO`	fifo

Outra possibilidade é usar uma das macros POSIX:

`S_ISLNK(m)`, `S_ISREG(m)`, `S_ISDIR(m)`, `S_ISCHR(m)`, etc.

De forma análoga se pode verificar as permissões de acesso ao ficheiro. Veja as máscaras através da man-page.

Obter informação de um directório

```
#include <dirent.h>

DIR *opendir(const char *pathname);
int closedir(DIR *dir);
```

`opendir()` retorna um apontador para uma estrutura to tipo DIR (análoga ao tipo FILE) que permite aceder ao conteúdo do directório corrente.

Uma vez o directório aberto, podemos obter as entradas nele contidas usando a função:

```
struct dirent *readdir(DIR *dir);
```

A estrutura `struct dirent` contém vários campos, muitos específicos do sistema, mas inclui dois que são de interesse:

- `d_name` contém o nome do ficheiro correspondente à entrada corrente no directório.

- `d_ino` contém o número do nó-i para esse ficheiro.

Exemplo: listar todos os ficheiros do directório corrente

```
#include <errno.h>
#include <dirent.h>
#include <stdio.h>

int main(void) {
    DIR * dir;
    struct dirent * ent;

    if (!(dir = opendir("."))) { // "." é o dir-corrente
      perror("opendir");
      return 1;
    }

    /* coloca errno=0, para det. se readdir() falha */
    errno = 0;
    while ((ent = readdir(dir))) {
      puts(ent->d_name);
      /* repõe errno, pois o puts() pode modifica-lo */
      errno = 0;
    }
    if (errno) {
      perror("readdir");
      return 1;
    }

    closedir(dir);

    return 0;
}
```

Exemplo: listar os nomes e tipos dos ficheiros de um directório

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

main(int argc,char *argv[]) {
  DIR *dir;
  struct dirent *d;
  struct stat finfo;

  dir=opendir(argv[1]);
  while ((d=readdir(dir))!=NULL) {
    printf("%s: ",d->d_name);
    lstat(d->d_name, &finfo);
    switch (finfo.st_mode & S_IFMT) {
    case S_IFREG:
      printf("regular\n");
      break;
    case S_IFDIR:
      printf("directorio\n");
      break;
    case S_IFLNK:
      printf("link\n");
      break;
    case S_IFBLK:
      printf("block device\n");
      break;
    case S_IFCHR:
      printf("character device\n");
      break;
    default: printf("\n");
    }
  }
}
```