

# Operações em diretórios

## Operações básicas

Cria um novo diretório com as permissões indicadas por mode:

```
#include <sys/stat.h>
#include <sys/types.h>
int mkdir (const char *filename, mode_t mode)
```

Remove o diretório indicado por filename, que deve estar vazio:

```
#include <unistd.h>
int rmdir (const char *filename)

#include <stdio.h>
int remove (const char *filename)
```

Retorna o diretório corrente do processo em buffer (de tamanho máximo size). Caso buffer e size sejam nulos, o buffer será alocado automaticamente:

```
#include <unistd.h>
char * getcwd (char *buffer, size_t size)
```

Ajusta o diretório corrente do processo para filename:

```
#include <unistd.h>
int chdir (const char *filename)
```

Idem, usando o diretório indicado por filedes:

```
#include <unistd.h>
int fchdir (int filedes)
```

## Formato de uma entrada de diretório

Cada entrada de diretório é definida pelo struct dirent, declarado no arquivo dirent.h. Essa estrutura normalmente contém os seguintes campos:

- char d\_name[]: nome da entrada.
- unsigned char d\_namlen: tamanho do nome, sem incluir o “\0” final.
- ino\_t d\_fileno: número do i-node da entrada.
- unsigned char d\_type: tipo do arquivo. As constantes a seguir são definidas para esse valor:
  - DT\_UNKNOWN: desconhecido.
  - DT\_REG: regular file.
  - DT\_DIR: directory.
  - DT\_FIFO: named pipe, ou FIFO.
  - DT\_SOCKET: local-domain socket.
  - DT\_CHR: character device.
  - DT\_BLK: block device.

Os demais atributos da entrada (tamanho, datas, permissões) fazem parte do arquivo em si, e não de sua entrada de diretório.



Algumas constante do campo `d_type` não fazem parte dos padrões Posix e C99. Nesse caso, pode-se usar o C com extensões GNU, ou então usar a função `stat` para obter mais informações sobre o arquivo/diretório.

## Lendo um diretório

Diretórios são abertos através de streams de tipo `DIR*`, definido em `dirent.h`. Os programas não devem alocar variáveis desse tipo, apenas ponteiros para variáveis alocadas pela biblioteca.

Abre o diretório `dirname`, retornando um stream de tipo `DIR*`:

```
#include <sys/types.h>
#include <dirent.h>
DIR * opendir (const char *dirname)
```

Informa o descritor de arquivo associado ao stream `dirstream`:

```
#include <sys/types.h>
#include <dirent.h>
int dirfd (DIR *dirstream)
```

Fecha o stream `dirstream`:

```
#include <sys/types.h>
#include <dirent.h>
int closedir (DIR *dirstream)
```

Lê a próxima entrada do diretório indicado por `dirstream`:

```
#include <unistd.h>
#include <linux/dirent.h>
#include <linux/unistd.h>
struct dirent * readdir (DIR *dirstream)
```

A ordem das entradas não é necessariamente alfabética. Esta função possui uma versão reentrante `readdir_r`, para situações onde várias threads podem acessar `dirstream` de forma concorrente. Para suportar sistemas de arquivos muito grandes, existem as versões `readdir64` e `readdir64_r`.

Reinicia o stream de diretório `dirstream`, fazendo-o apontar novamente para a primeira entrada. Alterações no diretório somente são levadas em conta após esta operação:

```
#include <sys/types.h>
#include <dirent.h>
void rewinddir (DIR *dirstream)
```

Indica qual a posição corrente no stream de diretório `dirstream`:

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

```
off_t telldir (DIR *dirstream)
```

Ajusta a posição corrente do stream de diretório `dirstream` para `pos`:

```
#include <dirent.h>
void seekdir (DIR *dirstream, off_t pos)
```

Esta função permite varrer diretórios de forma bem mais sofisticada, realizando as seguintes operações:

- varre o diretório indicado por `dir`;
- seleciona somente as entradas que atendem um dado critério, definido pela função `selector`;
- ordena as entradas selecionadas, usando a função de comparação `cmp`;
- devolve os resultados em `*namelist`, um vetor de apontadores de estruturas do tipo `dirent`;
- retorna o número de entradas em `*namelist`.

```
#include <dirent.h>
int scandir (const char *dir, struct dirent ***namelist,
             int (*selector) (const struct dirent *),
             int (*cmp) (const void *, const void *))
```

O funcionamento desta função está detalhado no manual da biblioteca Glibc.

Exemplo: listagem simples do diretório corrente:

[listagem.c](#)

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

int main (void)
{
    DIR *dirstream;
    struct dirent *dirent;

    // abre um diretório
    dirstream = opendir (".");
    if ( ! dirstream )
    {
        perror ("Couldn't open the directory");
        exit (1) ;
    }

    // varre as entradas do diretório aberto
    for (;;)
    {
        // pega a próxima entrada
        dirent = readdir (dirstream) ;

        // se for nula, encerra a varredura
        if (! dirent)
            break ;

        // mostra conteúdo da entrada
        printf ("%s\t", dirent->d_name);
    }
}
```

```
switch (dirent->d_type)
{
    case DT_UNKNOWN:
        printf ("(desconhecido)\n") ;
        break ;
    case DT_REG:
        printf (" (arquivo)\n") ;
        break ;
    case DT_DIR:
        printf (" (diretorio)\n") ;
        break ;
    default:
        printf (" (outros)\n") ;
}

// fecha o diretório
(void) closedir (dirstream);

exit (0);
}
```

From:  
<https://wiki.inf.ufpr.br/maziero/> - **Prof. Carlos Maziero**

Permanent link:  
[https://wiki.inf.ufpr.br/maziero/doku.php?id=pua:operacoes\\_em\\_diretorios](https://wiki.inf.ufpr.br/maziero/doku.php?id=pua:operacoes_em_diretorios)

Last update: **2021/01/26 10:37**

