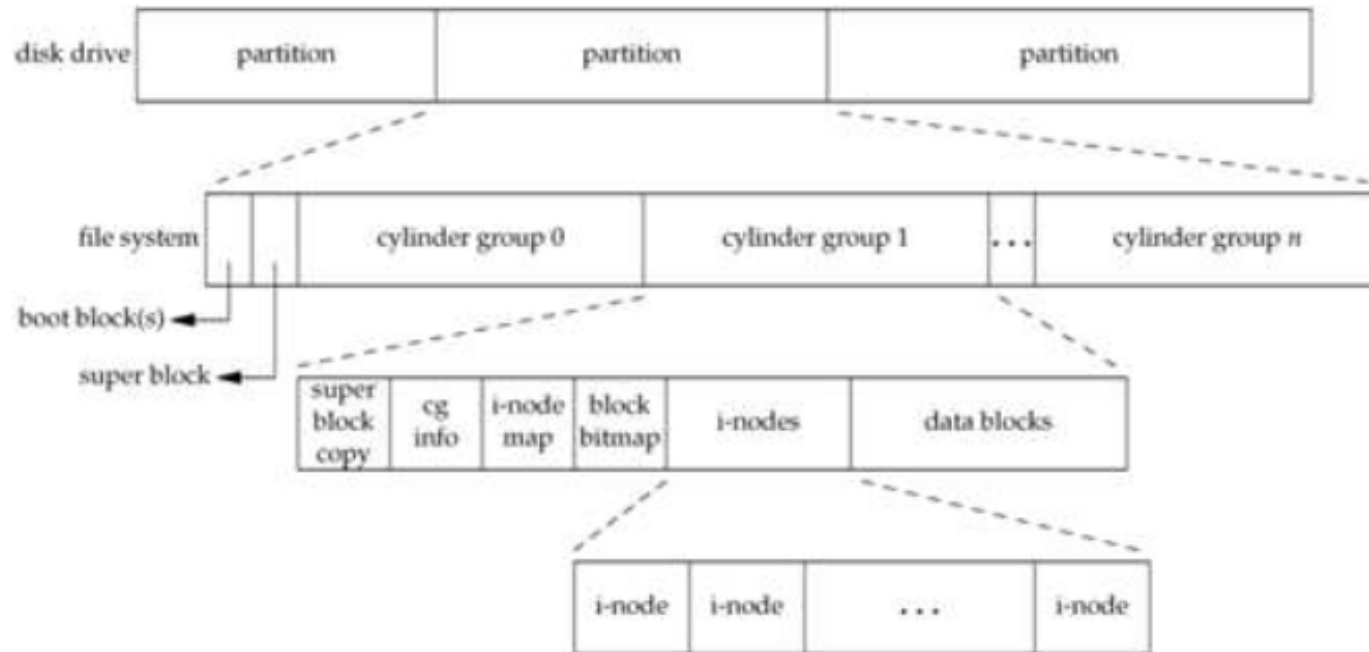


File System

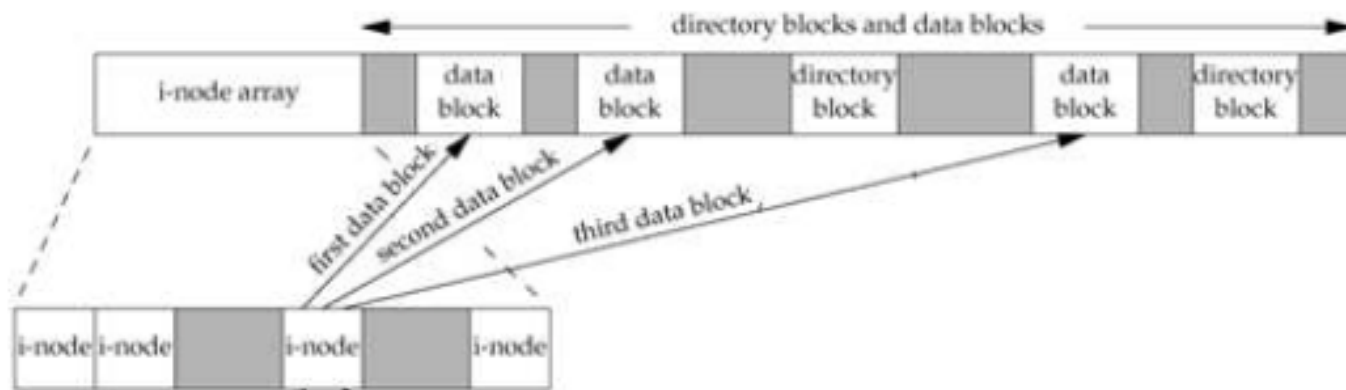
Franco Maria Nardini

File System

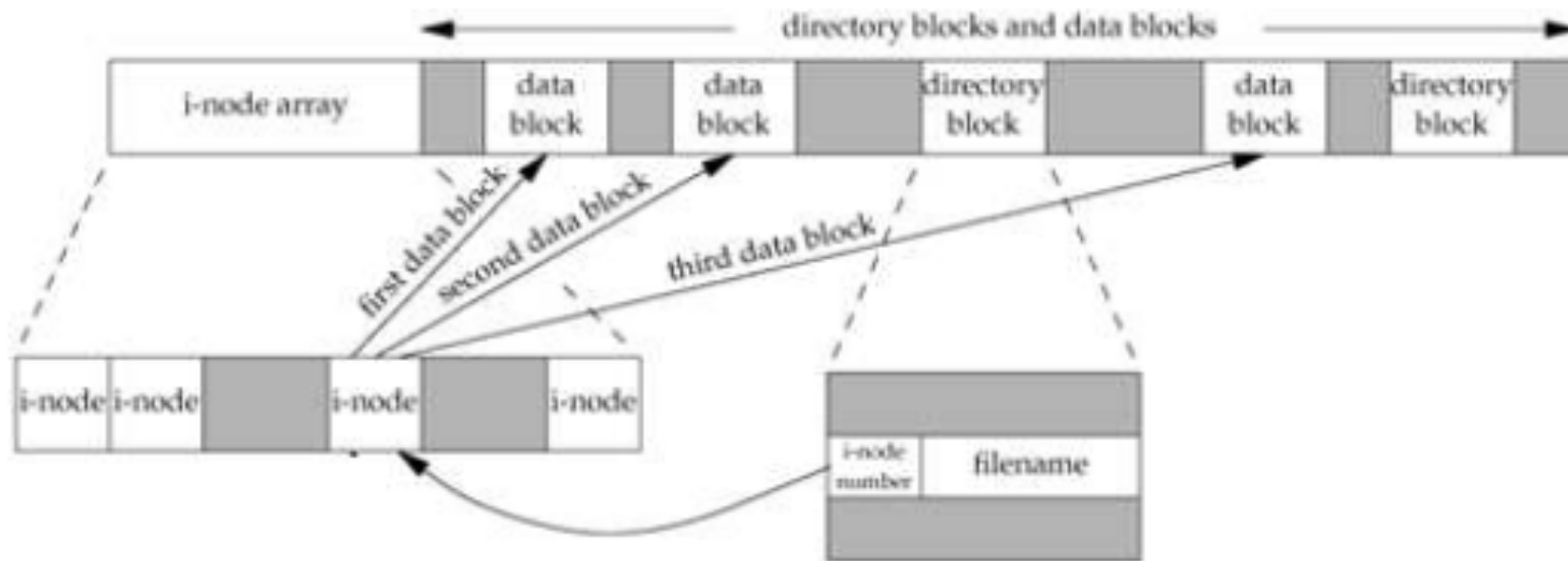


- un disco fisso può essere diviso in *partizioni*
- una *partizione* può contenere un *file system* contenente *cylinder groups*
- ogni *cylinder group* contiene una lista di *i-nodes* (*i-list*), *directory-* e *data blocks*.

File System

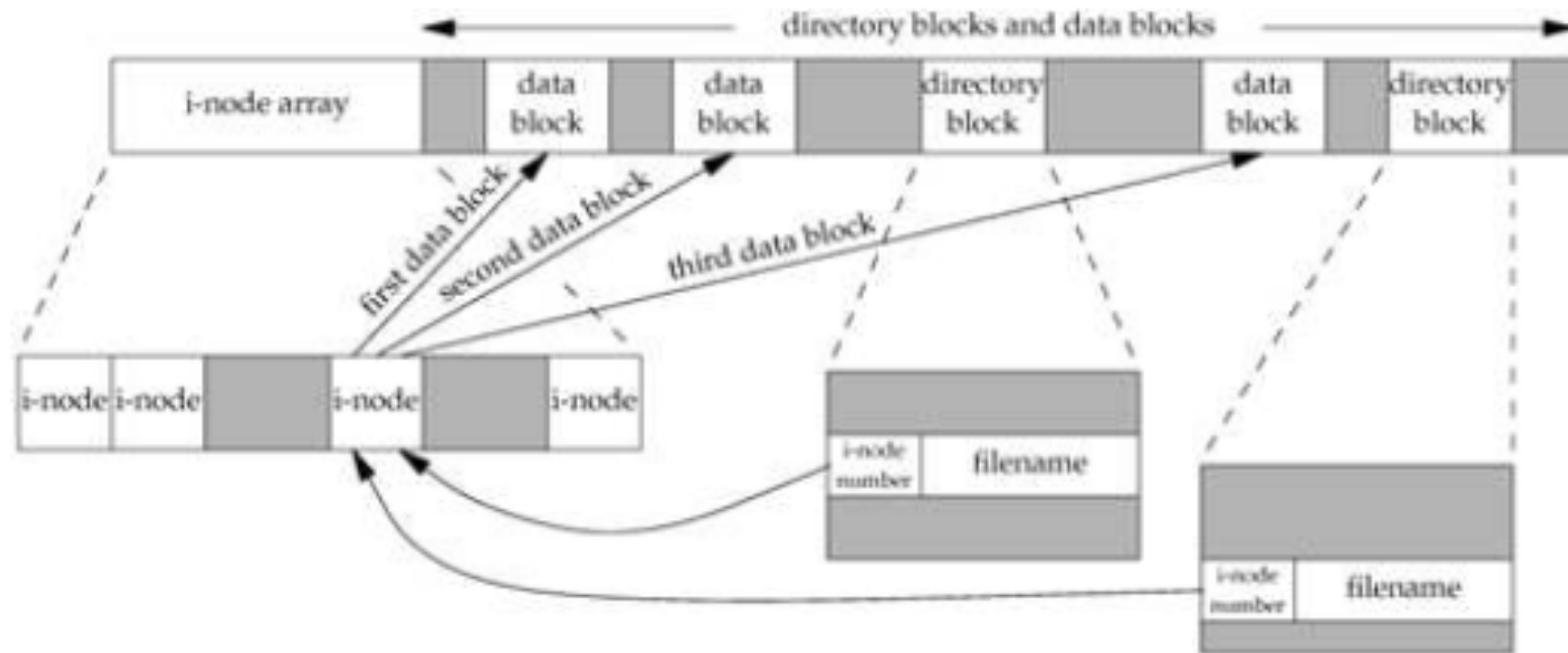


File System



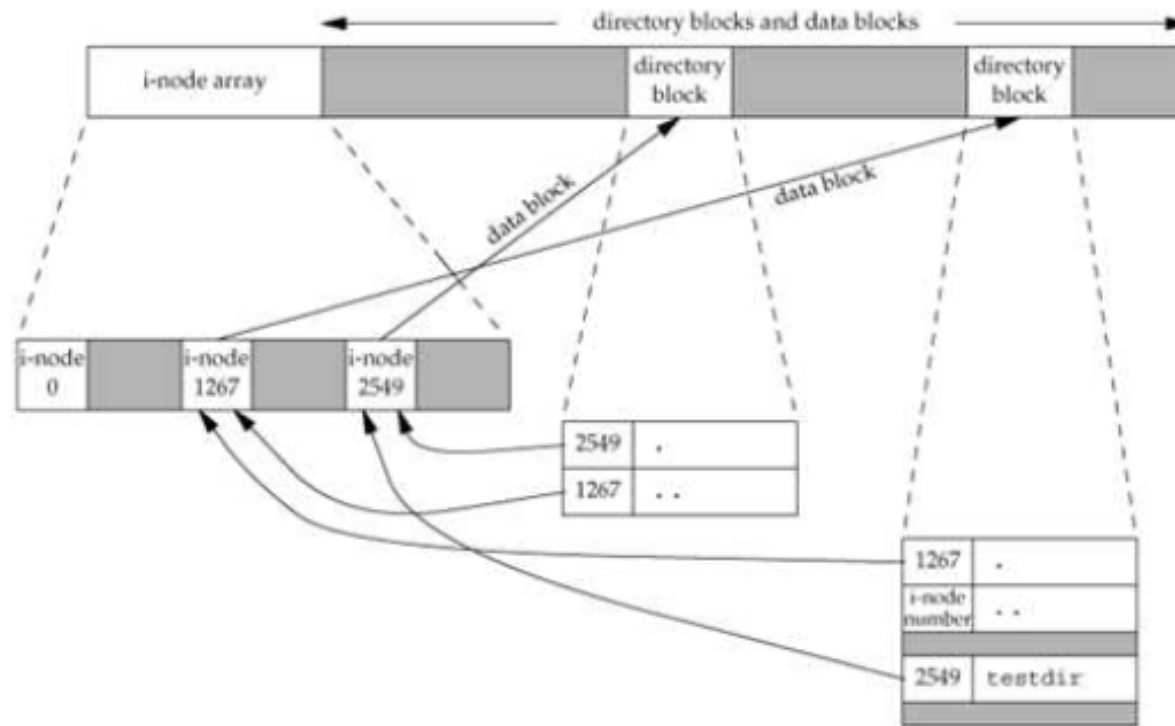
- una directory è un *hard link* che mappa un nome di file ad un *i-node*

File System



- possibili vari mapping allo stesso file!

File System

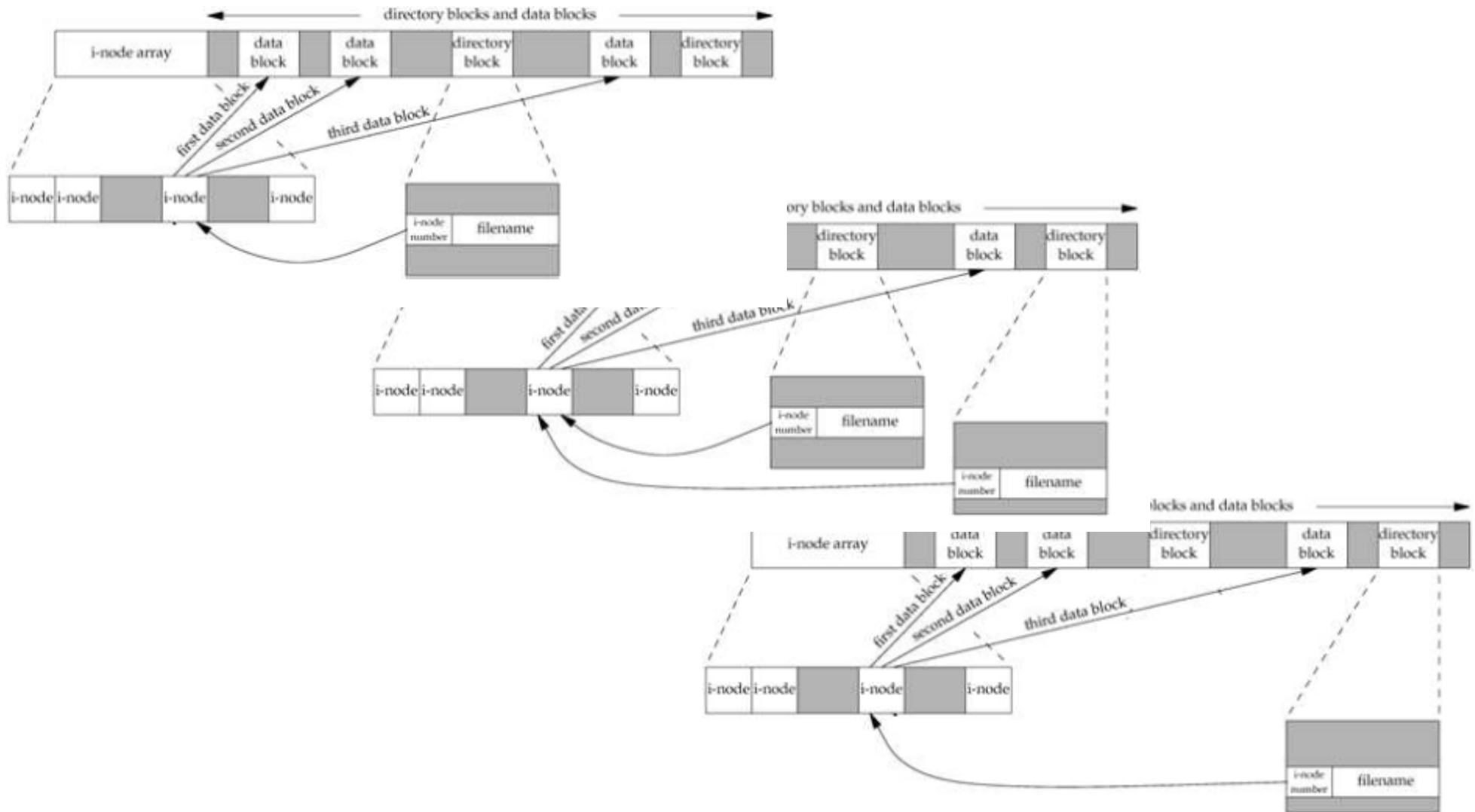


- le directory sono file speciali contenenti *hard link*!
- ogni directory contiene almeno due entry: “.”, “..”
- il link count di ogni directory (`st_nlink`) è sempre almeno 2

File System

- ogni i-node contiene la gran parte delle informazioni viste nella `struct stat`
- ogni i-node ha un contatore di link (`st_nlink`)
 - rivela quanti oggetti puntano all'i-node
 - reference counting: se il contatore è 0 (e nessun processo ha il file aperto) si può liberare il data block.
 - il riferimento all'i-node in una directory deve puntare all'i-node sullo stesso file system (no hard link tra filesystem)
- per muovere un file, basta muovere la directory entry
 - fatto con creazione di una nuova a destinazione e cancellazione della sorgente

File System



link(2)

```
#include <unistd.h>
```

```
int link(const char *name1, const char *name2);
```

Returns: 0 if OK, -1 on error

- crea un link ad un file esistente (hard link)
- POSIX.1 consente la creazione di hard link tra filesystem diversi.
 - Le principali implementazioni no (SVR4, BSD).
- solo `uid(0)` può creare link a directory (loop!)

unlink(2)

```
#include <unistd.h>

int unlink(const char *path);
```

Returns: 0 if OK, -1 on error

- rimuove una directory entry e decrementa *link count*
- se il file ha *link count* == 0 E nessun processo lo tiene aperto:
 - il file system libera i data blocks associati

Esempio

```
$ wget http://hpc.isti.cnr.it/~nardini/siselab/06/wait-unlink.c
```

```
$ cc -Wall wait-link.c
```

```
$ ./a.out
```

rename (2)

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

```
int rename(const char *from, const char *to);
```

Returns: 0 if OK, -1 on error

- se *oldname* riferisce ad un file
 - se *newname* esiste e NON è una directory, è rimosso e *oldname* diventa *newname*
 - se *newname* esiste e è una directory: errore!
 - permessi w+x sulle directory contenenti *oldname* e *newname*

rename (2)

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

```
int rename(const char *from, const char *to);
```

Returns: 0 if OK, -1 on error

- se *oldname* riferisce ad una directory
 - se *newname* esiste ed è vuota, è rimossa e *oldname* è rinominato in *newname*
 - se *newname* esiste ed è un file: errore!
 - se *oldname* è un prefisso di *newname*: errore!
 - permessi w+x sulle directory contenenti *oldname* e *newname*

Symbolic Links

```
#include <unistd.h>
```

```
int symlink(const char *name1, const char *name2);
```

Returns: 0 if OK, -1 on error

- un file i cui dati sono il path ad un altro file
- tutti possono creare simlinks a directory e files
- alcune funzioni operano sul link, altre sulla destinazione

Symbolic Links

```
#include <unistd.h>
```

```
int readlink(const char *path, char *buf, size_t bufsize);
```

Returns: number of bytes placed into buffer if OK, -1 on error

- Combina `read(2)`, `open(2)` e `close(2)`.
- NOTA: `buf` non termina con NUL

mkdir (2)

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

int mkdir(const char *path, mode_t mode);
```

Returns: 0 if OK, -1 on error

- Crea una directory vuota
 - Permessi di accesso specificati da `mode`
 - `umask (2)` li può cambiare nel processo chiamante

rmdir (2)

```
#include <unistd.h>

int rmdir(const char *path);
```

Returns: 0 if OK, -1 on error

- se *link count* == 0 E nessun processo ha la directory aperta, è rimossa
- deve essere vuota (solo “.” e “..”)

Leggere directory

```
#include <sys/types.h>
```

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

```
DIR *opendir(const char *filename);
```

Returns: pointer if OK, NULL on error

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

Returns: pointer if OK, NULL at end of dir or on error

```
void rewinddir(DIR *dp);
```

```
int closedir(DIR *dp);
```

Returns: 0 if OK, -1 on error

Posizionamento

```
#include <unistd.h>
```

```
char *getcwd(char *buf, size_t size);
```

Returns: *buf* if OK, NULL on error

```
#include <unistd.h>
```

```
int chdir(const char *path);
```

```
int fchdir(int fd);
```

Returns: 0 if OK, -1 on error

Esempio

```
$ wget http://hpc.isti.cnr.it/~nardini/siselab/06/cd.c
```

```
$ cc -Wall cd.c
```

```
$ ./a.out
```

Homework

- Costruire un file system virtuale usando un file: operazioni a blocchi
- Pagine `man` delle funzioni viste.
- Stevens, cap 4, 6.
- Testare hard link, symbolic links: creazione, modifica e rimozione.
- Stessa cosa con le directory.