Objects

```
typedef struct _nandfs_object {
        uint32 id;
        uint32 version;
        uint32 page;            /* Logical page */

        nandfs_type type;

        uint16 hashname;
        uint32 parent_id;

        /* This object is a dir or a file */
        union {
                struct _nandfs_object *files;   /* Dir => objects */
                struct {
                        /* File => chunks */
                        uint32 allocsize;
                        uint32 *pages;
                        uint32 *versions;
                } chunks;
        } childs;

        /* This object is a dir/file/symlink/hardlink */
        nandfs_object_header *header;

        /* This object is a dir/file/hardlink/symlink */
        /* It's a simple linked list, as O(n) will not be a problem here */
        struct _nandfs_object *hardlinks;

        /* The object is a file */
        uint32 max_version;

        uint16 flags;           /* Space for flags */

        /* Linked Lists */
        struct _nandfs_object *bro_prev, *bro_next;     /* brothers */
        struct _nandfs_object *hash_prev, *hash_next;   /* Linked List (hashtab
} nandfs_object;
```

What is an object ?

Everything except data chunks is an object. Every object get :

- An unique Id
- A version number
- A Logical page
- A type
- Some flags

Common

They are associated to their metadata (obj->header, see ObjectHeader)

These objects are stored in the nandfs->objects hashtable.

object->hash_next/hash_prev

for the hashtable linked list

object->bro_next/bro_prev

for the objects who share the same parent.

obj->hardlinks

All hardlinks are linked to equivalent object/hardlinks. The first element in the list is the original object.

obj->hashname

a hash for a faster lookup

obj->parent_id

to find the parent

Directory

Use object->childs.files to keep a linked list of the childs.

File

Use object->childs.chunks to store chunk's pages and versions.

FIFO/Socket/Device (linux only)

Like files, but without data chunks