W różnych interfejsach katalog przedstawiany jest jako rodzaj pojemnika zawierającego pliki i inne podkatalogi - przez analogię do szafy z dokumentami. O ile dokument to plik, wówczas szuflada w której się znajduje to katalog. W zasadzie cała szafa to także katalog, tylko nadrzędny w stosunku do szuflad. Taka reprezentacja jest dość intuicyjna i ułatwia zrozumienie hierarchicznego systemu plików.

Zgodnie z prostą i bardzo konsekwentnie przestrzeganą podstawową zasadą Unixa, że wszystko jest plikiem, także i katalog to formalnie plik. Nieco upraszczając można powiedzieć, że katalog to plik, w którym spisane są wszystkie pliki w nim zawarte poprzez podanie nazwy pliku i odpowiadającego mu numeru inode. Oczywiście podkatalogi i inne pliki specjalne są tak samo traktowane.

Katalog UFS ma specjalną budowę (patrz prezentacja) opisaną w pliku nagłówkowym `ufs_fsdir.h` jako struktura `direct`. Zbudowana jest ona z 4 pól o następującym znaczeniu:

```
uint32_t   d_ino;                  /* inode number of entry */
```

Numer inode pliku. Aby stwierdzić, czy dany plik to plik zwykły, katalog, czy inny plik specjalny należy odczytać ten inode.

```
ushort_t   d_reclen;               /* length of this record */
```

Rozmiar całej struktury `direct` podany w bajtach.

```
ushort_t   d_namlen;               /* length of string in d_name */
```

Faktyczna długość nazwy pliku zapisanej w tablicy `d_name`.

```
char       d_name[MAXNAMLEN + 1];  /* name must be no longer than this */
```

Tablica znakowa zawierająca nazwę pliku. Ponieważ stała `MAXNAMLEN` jest ustawiona na 255, to nazwa plików może składać się z maksymalnie 255 znaków. Dodanie 1 w definicji `d_name` gwarantuje, że zmieści się bajt zerowy kończący napis (null-terminated string). Aby zminimalizować fragmentację, nazwa pliku zaokrąglana jest zawsze do wielokrotności 4 bajtów. To znaczy, że nazwa `etc` zajmować będzie 4 bajty (3 znaki plus bajt zerowy), natomiast nazwa `unix` zajmie 8 bajtów (4 znaki plus bajt zerowy zaokrąglone do wielokrotności liczby 4) .

Warto zwrócić uwagę na wielkość `d_reclen` i na to, co się dzieje, gdy usuwamy plik. Gdyby utworzenie pliku powodowało zawsze dopisanie w katalogu nowej pozycji na końcu, to zauważmy, że w systemie plików, gdzie często ma miejsce tworzenie nowych i usuwanie starych plików, spis plików, czyli katalog rósł by bardzo szybko zajmując coraz więcej miejsca. Byłby to spory problem na przykład w różnych rodzajach cache, gdzie pliki zapisywane są na krótki okres czasu, a potem usuwane. Dlatego ostatnia pozycja w katalogu zawsze zajmuje tyle miejsca ile pozostaje wolnego miejsca w katalogu. W przykładzie z prezentacji wpis pliku `usr` zajmuje 484 bajty, gdzie w sumie efektywnie użyto tylko 12 bajtów. Skąd 484? Rozmiar katalogu jest wielokrotnością 512. Tak więc najmniejszy katalog zajmuje 512 bajtów jak w naszym przykładzie. Razem 3 wpisy wypełniają 16+12+484 = 512 bajtów. Gdy dodamy nowy plik, to od 484 zostanie odjęta odpowiednia wielkość. Gdy usuwamy plik, to zwalniane jest miejsce zajmowane przez jego wpis w katalogu, a poprzedni wpis powiększa się o jego rozmiar. Na przykład, gdybyśmy usunęli plik `etc` to we wpisie pliku `unix` pole `d_reclen` z 16 zostałoby zwiększone z 16 na na 28 = 16+12. Gdy dodawany jest nowy plik, to system wyszukuje wolne miejsce w katalogu, aby utworzyć nowy wpis. Nie musi to być na końcu katalogu, ale w środku, pomiędzy istniejącymi wpisami.

W miejscu gdzie mamy rozmiar pliku dla katalogu `ls` wyświetla ilość miejsca zarezerwowaną na wpisy w tym katalogu:

```
sirius$ mkdir test1
sirius$ ls -ld test1
drwxr-xr-x   2 mariusz  staff        512 lis 29 16:33 test1
```

Nowo utworzony katalog zajmuje 512 bajtów. Większe katalogi jakąś wielokrotność tej liczby:

```
sirius$ ls -ld /etc
drwxr-xr-x  89 root     sys         4608 lis 29 12:01 /etc
```

Ze względu na swoją specyfikę katalog, który formalnie jest plikiem w systemie UFS, będzie traktowany przez programy inaczej niż zwykły plik. Nie obejrzymy go przy pomocy `cat`, program `ls` pokaże listę plików w nim się znajdujących itd.