ICA07 Gruppe 09
For å definere et virtuel filsystem må vi først lage og få tilgang til en virtuell disk. Dette gjør vi siden filsystemet skal være virtuelt, altså plassert oppå det vanlige filsystemet.

Vi gjør dette ved å definere disken:

```
#ifndef _DISK_H_
#define _DISK_H_

Antall blocks. Halvparten er reservert som data blocks og resten som meta-data.
#define DISK_BLOCKS  16384
Størrelsen på hver block: 8kb.
#define BLOCK_SIZE   8192

Maks filstørrelse er 32MB.

Noen kommandoer for å lage en virtuell disk fil, åpne den, , lukke den, lese og skrive
,Lager en tom virtuell disk:
make_disk(char *name);
Åpner disken:
open_disk(char *name);
Lukker disken som ble tidligere åpnet:
close_disk();

Skriver en block størrelse:
block_write(int block, char *buf);

Leser en block størrelse:
block_read(int block, char *buf);
                        /* read a block of size BLOCK_SIZE from disk   */

#endif
```

For å håndere filsystemet trenger vi tre funksjoner:

Lage filsysteme på den virtuelle disken med disk_namet:

```
int make_fs(char *disk_name);
```

Mounte filsystemet, dvs. Velge filsystemet slik at det kan brukes, 0 på suksess, og -1 kan ikke åpnes eller har et valid filsystem:

```
mount_fs(char *disk_name);
```

Unmounte filsystemet:

```
umount_fs(char *disk_name);
```

Så har vi forskjellige fil system funksjoner, likt til Linux fil system operasjoner. Dette krever at fil systemet tidligere er mounted.

Filen spesifisert av navn åpnes:
fs_open(char *name);
En file descriptor blir brukt for å få tilgang til filen. Bibloteket støtter maks 64 file descriptions som kan være åpne på en gang. Den returnerer -1 ved failure. Når navnet ikke kan bli funnet, eller det allerede er 64 file descriptors aktive er den en failure.

fs_close(int fildes);
Lukker fil descriptoren. Åpnes den ikke eller eksisterer ikke returnerer den -1.

fs_create(char *name);
Lager en ny fil med et navn i root directory på filsystemet. Filen er tom i starten, og max lengde for fil navnet er 32 characters. Det kan være max 128 filer i directoryen. Returnerer -1 ved failure når filnavnet er for langt eller navnet ikke eksisterer, eller det allerede er 128 filer i root directory.

fs_delete(char *name);
Sletter filen med navnet "name" fra root directory, gjør alle data blockene og meta-informasjonen ledig som korresponderte til den filen. Filen som slesses må ikke være åpen, den må ikke ha samme navn som fil descriptoren refererer til. Når navnet ikke eksisterer eller den er åpen returnerer den -1 som failure.

fs_read(int fildes, void *buf, size_t nbyte);
Brukes til å lese filen, leser data fra filen som blir referert via descriptoren inn i bufferen. Bufferen må være stor nok til å holde alle bytsa. Den leser filen og returnerer den, og returnerer -1 ved failure når file descriptoren ikke er gyldig.

fs_write(int fildes, void *buf, size_t nbyte);
Skriver data til filen referert av descriptoren fra bufferen. Filen utvides når enden av filen er nådd så den kan holde flere bytes. Hvis fil descriptoren ikke er valid returnerer den -1.

fs_get_filesize(int fildes);
Returnerer størrelsen via descriptoren.

fs_copy(int fidels, size_t nbyte);
Finner filen via descriptoren, kopierer over bytes til minnet, skriver en ny lik fil med samme bytes ny descriptor.

fs_rename(char*name);
Endrer navnet på filen. Max characters er 32. Det kan ikke være mer enn 32 characters, da returnerer den -1, eller hvis navnet ikke finnes.

Organisering av filer:
Vi kan ha 2 typer clusters:

1. Fil cluster: Som holder innholdet til filen
2. Directory cluster. Som inneholder directory strukturen.

Directory entry:
Directory entries holder filsystemet metadata for alle filene. Inneholder navn, timestamps, starting cluster for filene.

I root har vi første sett av directory entries.

Feks:

File 1	“time stamp”	6 (starting cluster)
Folder 1	“time stamp”	7

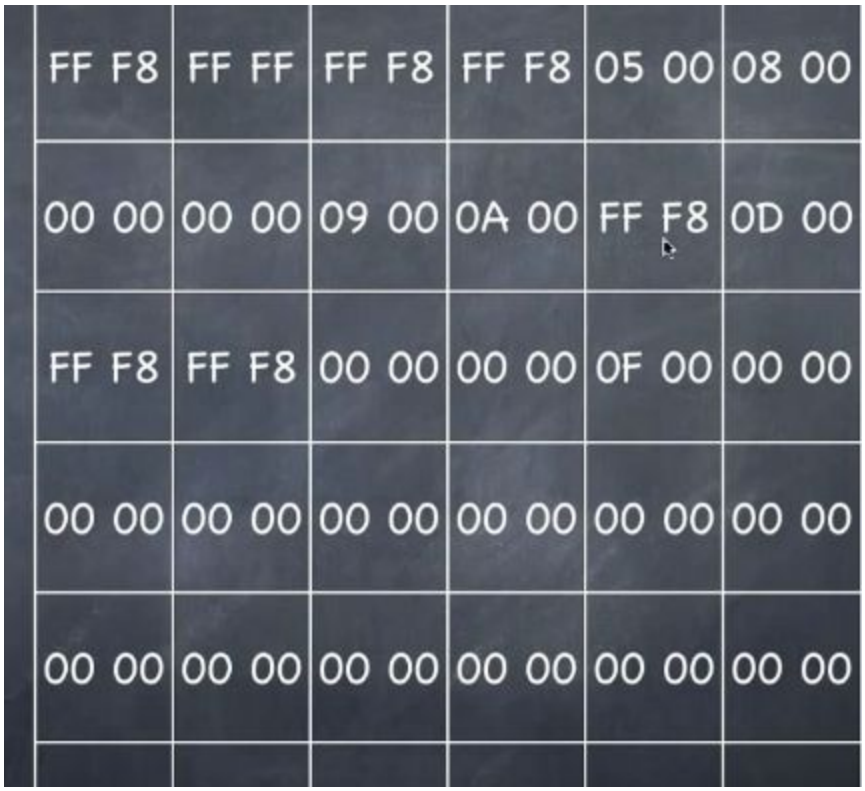
Går vi til cluster 4 finner vi innholdet i filen. Alle filene finnes via root directory. Vi kan også ha en folder, hvor vi finner flere directory entry structures. Starter med (dot). Mens (dotdot).. er parent directory, alle med en cluster tall. Vi kan ha filer og folders i folderen.

Når en folder lages får den en cluster tall som inneholder directories for hva som er i folderen.

Sannsynligheten er at filer trenger flere clusters for å holde den sammen. Det eneste en directory entry inneholder er starting cluster, vi trenger en ny table for å holde filene sammen. Denne tabelen mapper alle clusterene i en table for systemet.

Hver celle representerer en cluster. 0 er unallocated. FFF8, FF8 eller FFF FFF8 for dårlige clusters. FFF9 ,FFF9 eller FFF FFF9 for end of file. Alle andre verdier betyr at de er allocated.

Sigurd.txt … 3 betyr at vi må lete i cluster 3, som er FF F8, starter og slutter i cluster 3.
Sigurd.jpg … 4 den starter i 4, så går den til 5, så til 8, så til 9 så til A(10) der den slutter etter det. Filen er lagret i disse clustene.



Kilder:
https://www.youtube.com/watch?v=HjVktRd35G8
http://www.cs.ucsb.edu/~chris/teaching/cs170/projects/proj5.html