F4 - Vad förväntas ni kunna

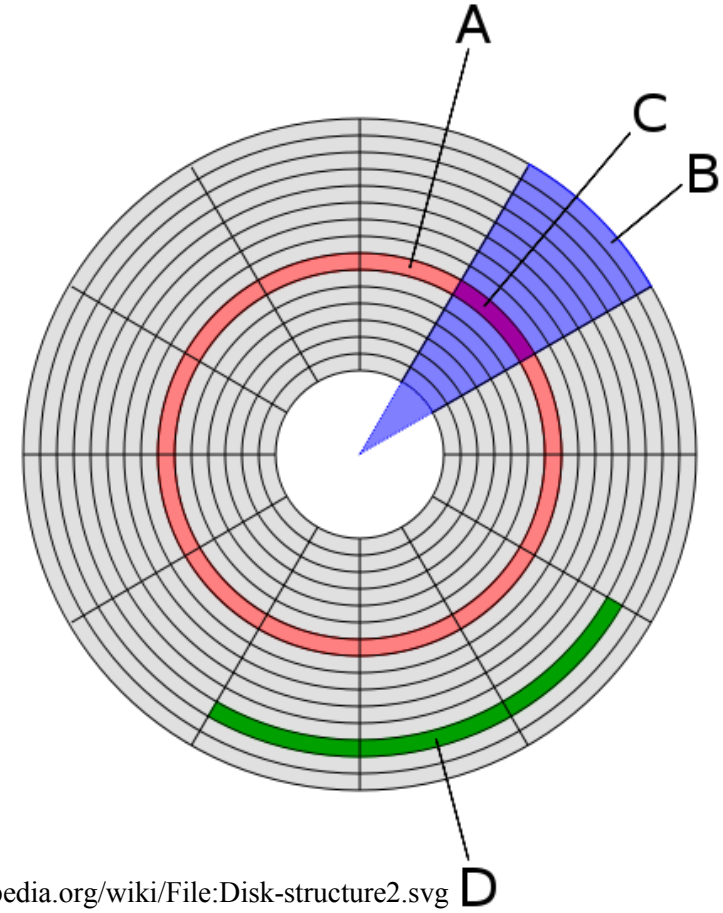
Kunna öppna en fil för läsning och skrivning.
Kunna testa om det gick att öppna den fil man önskat öppna.
Kunna öppna en fil som användaren fått specificera.

Känna till hur man testa om det inlästa tecknet från filen är slut på filen-tecknet.
Känna till hur man kan testa hur många tecken som lästs från en fil.
Kunna läsa ett tecken från en fil.
Kunna läsa en rad från en fil.

Filhantering

Läromål: Hur vet man att man kommit till slutet av en fil?

- I teckentabeller ser ni att tecken representeras av olika tal.
- Ett negativt tal representerar att man kommit till slutet av filen.
- Vilket negativt tal är ej bestämt.
- EOF representerar tecknet slut på filen.
- I C-programmering kan vi jämföra läst tal med EOF.
- Vi kan fortsätta att läsa trots att vi kommit till slutet av filen….
- En sektor = 512 tecken.
- N antal sektorer utgör en kluster
- Data på en hårddisk är vanligtvis adresserad per kluster.
- En fil tar upp utrymme motsvarande minst N antal klustrar.
- Raderade filer innebär bara att ”innehållsförteckningen” har suddats ut.
- Vi kan alltså läsa filer som inte finns listade i ”innehållsförteckningen”… om vi visste vart filen var.



A = Ett spår
B = En geometrisk sektor
C = En sektor (spårsektor)
D = En kluster
Källa (2013-10-24). http://en.wikipedia.org/wiki/File:Disk-structure2.svg

EOF

```
char tecken;
…
if (tecken == EOF) {
```

Läromål: Förstå att man kan testa om man kommit till slutet av en fil.

Öppna en fil på hårddisken

Funktionen **fopen** ser ut enligt följande:

```
FILE * fopen(const char *filename, const char *mode)
```

FILE ← **Indikerar att funktionen fopen returnerar en koppling till en FILE på hårddisken**

***** ← **Indikerar att kopplingen till en FILE på hårddisken kommer i form av en adress till en plats i ramminnet där det finns data om vilken fil på hårddisken man ska använda samt hur man kan använda den (mao om man kan läsa eller skriva till den).**

fopen ← **returnerar null om det inte gick att öppna filen. Om det gick att öppna filen returneras en adress (pekare).**

const char *filename ← Innebär att `fopen` behöver veta filnamnet på den fil som ska öppnas i form av antingen `"matlagning.doc"` eller en char-array.

const char *mode ← Innebär att fopen vill veta om du ska läsa eller skriva till filen i form av `"r"` för läsning från filen och `"w"` för skrivning till filen.

stdio.h innehåller funktionen **fopen** som kan användas för att läsa och skriva till filer på en hårddisk.

#include <stdio.h>

För att använda funktionen **fopen** som finns i **stdio.h** skriver vi:

```
FILE *fp = fopen("dagbok.txt", "r");
```

fp blir då en referens till en fil på hårddisken. I detta fall **dagbok.txt**

Läromål: Förstå vad **fopen** returnerar samt förstå vilka två värden **fopen** behöver.

Mer om `const char *mode`

Läge	Beskrivning
"r"	Öppna en fil för läsning. (Filen måste finnas).
"w"	Skapa en ny tom fil som vi kan skriva till. (Finns filen redan så töms den på innehåll!!!)
"a"	Lägg till I slutet på en existerande fil (Filen skapas om den inte finns).
"r+"	Öppna en fil för läsning och skrivning. (Filen måste finnas).
"w+"	Skapa en tom fil för läsning och skrivning.
"a+"	Öppna en fil för läsning och skrivning till slutet av filen.

a, r, w står för engelskans append, read och write.

Läromål: Förstå att man kan öppna en fil på flera olika sätt.

Kodexempel – Öppna en fil

```
#include <stdio.h>
int main(){
// FILE * fopen(const char *filename, const char *mode)
    FILE *fp = fopen("dagbok.txt", "r");

    if (fp == NULL) {
        puts("Filen fanns ej.");
    }
    else {
         puts("Det gick bra att öppna den filen.");
    }

    system("pause");
    return 0;
}
```

Tips!
Provkör denna kod.
Pröva även att byt ut dagbok.txt mot en fil som finns i katalogen där du startar programmet.

Läromål: Kunna öppna en fil. Kunna testa om det gick att öppna filen.

Läsa ett tecken från fil – Ett kodexempel

```
#include <stdio.h>
int main(){
    char tecken;
    // Öppna filen dagbok för läsning (r=read).
    FILE *fp = fopen("dagbok.txt", "r");

    // Läs ett tecken från filen till variabeln tecken
    tecken = fgetc(fp);

    // Skriv ut tecknet i variabeln tecken till skärmen.
    printf("%c", tecken);

    system("pause");
    return 0;
}
```

```
#include <stdio.h>
int main(){
    FILE *fp = fopen("dagbok.txt", "r");
    printf("%c", fgetc(fp));

    system("pause");
    return 0;
}
```

fgetc → file get char
Alltså hämta ett tecken från filen.
En char är alltid ett tecken.
En char-array är alltid 1-n tecken.

Läromål: Kunna använda fgetc till att läsa ett tecken från en fil.

Läsa en rad från fil – Ett kodexempel

```
#include <stdio.h>
#define ARRAY_SIZE 100

int main(){
    char tecken[ARRAY_SIZE];
    int readStatus = 0;
    FILE *fp = fopen("dagbok.txt", "r");

    // Läs en rad från filen till tecken men inte mer än 99 tecken. Varför 99?
    // readStatus får värdet NULL om inget fanns att läsa (slut på filen)
    readStatus = fgets(tecken, ARRAY_SIZE, fp);

    // Vi kollar readStatus
    if (readStatus != NULL) {
        // Skriv ut raden från variabeln tecken till skärmen.
        printf("%s", tecken);
    }

    system("pause");
    return 0;
}
```

fgets → file get string
string → alltså hämta string (flera tecken) från filen.
(Hämtar en rad).

Vad förväntas ni kunna

Kunna öppna en fil för läsning och skrivning.
Kunna testa om det gick att öppna den fil man önskat öppna.
Kunna öppna en fil som användaren fått specificera.

Känna till hur man testa om det inlästa tecknet från filen är slut på filen-tecknet.
Känna till hur man kan testa hur många tecken som lästs från en fil.
Kunna läsa ett tecken från en fil.
Kunna läsa en rad från en fil.