Filhantering

- Repetition:
  - Funktioner för att läsa in data från tangentbordet: **`scanf()`**, **`getchar()`** och **`fgets()`**
  - Funktioner för att skriva ut data till skärmen: **`printf()`** och **`putchar()`**
- Kan också läsa och skriva från filer:
  - Med hjälp av *redirection*: Ett program kan instrueras att läsa från eller skriva till en viss fil genom att ange denna fil *då programmet startas från kommandoprompten/terminalen*
  - Genom att öppna och sedan läsa/skriva till namngivna filer *inne i programmet*

Redirection

- För att styra all utdata från programmet **`a.out`**
  till en fil **`out.txt`**:
  `./a.out > out.txt`
- För att läsa in all indata till programmet **`a.out`**
  från en fil **`in.txt`**:
  `./a.out < in.txt`
- För att läsa från **`in.txt`** och skriva till **`out.txt`**:
  `./a.out < in.txt > out.txt`
- Inte specifikt för c-program, kan styra input/output från vilket program som helst

“End-Of-File”, EOF

- Vid inläsning av data från en fil är det viktigt att veta när slutet på filen påträffas
  - Ofta består programmet av en loop som läser in ett ord eller en rad i taget, tills all information i filen är inläst
- De flesta inputfunktioner returnerar ett speciellt heltalsvärde, **`EOF`**, när slutet på en fil påträffas.
- Följande exempel använder **`getchar()`**, **`putchar()`** och **`EOF`** för att kopiera en fil mha. redirection.

“End-Of-File”, EOF

```
/* Program to copy a file.
Usage: ./a.out < infile > outfile */

#include <stdio.h>

int main(void) {
    int c;

    /* Note that getchar returns an int, not a char.
       Read one character at a time, until we find EOF */
    while ( (c = getchar()) != EOF ) {
        putchar(c);
    }

    return 0;
}
```

Arbeta med namngivna filer

- För att t.ex. läsa eller skriva till flera filer, skapa ett mera användarvänligt gränssnitt och implementera bättre felhantering räcker “redirection”-metoden inte till
- Måste istället används c:s inbyggda funktioner för filhantering
- Allmän princip:
  - Försök öppna filen och kontrollera att operationen lyckades
  - Läs eller skriv till filen
  - Stäng filen
- Filer kan öppnas för läsning, skrivning eller uppdatering

Öppna filer: **`fopen`**

- **`FILE* fopen(const char* path,`**
  `const char* mode)`
- Returvärdet är en speciell “filpekare” som senare kan användas för att läsa till eller skriva från denna fil
- **`path`** är sökvägen till filen
- **`mode`** är en sträng som beskriver hur filen skall öppnas:

  `"r"` => läs
  `"r+"` => läs och skriv
  `"w"` => skapa ny fil och skriv
  `"w+"` => skapa ny fil, läs och skriv
  `"a"` => lägg till data
  `"a+"` => läs och lägg till data

Läs från filer: **getc, fscanf, fgets**

- **`int getc(FILE* stream)`**
  - Läs ett tecken, motsvarar **`getchar()`**
  - Returnerar **`EOF`** när slutet av filen påträffas
- **`int fscanf(FILE* stream,`**
  `const char* formatString, …)`
  - Motsvarar **`scanf()`**
  - Returnerar **`EOF`** när slutet av filen påträffas
- **`char* fgets(char* buffer, int maxsize,`**
  `FILE* inputStream)`
  - Läser in en rad i taget
  - Returnerar **`NULL`** (0) när slutet av filen påträffas

Skriv till filer: **putc, fprintf, fputs**

- **`int putc(int c, FILE* stream)`**
  - Skriv ett tecken, motsvarar **`putchar()`**
  - Returnerar **`EOF`** om fel uppstår
- **`int fprintf(FILE* stream,`**
  `const char* formatString, …)`
  - Motsvarar **`printf()`**
  - Returnerar ett negativt värde om fel uppstår
- **`int fputs(const char* buffer,`**
  `FILE* inputStream)`
  - skriver en rad i taget
  - Returnerar **`EOF`** om fel uppstår

Stänga filer: **`fclose`**

- **`int fclose(FILE* fp)`**
- Returnerar **`EOF`** om fel uppstår
- Viktigt att stänga filer eftersom:
  - Data som skickas till en fil inte alltid omedelbart skrivs till filen. **`fclose`** tömmer ('flushar') utdatabuffern och stänger sedan filen
    - Kan använda **`fflush()`** för att flusha filen utan att stänga den
  - Antalet filer som samtidigt kan vara öppna är begränsat, och varje öppnad fil kräver minne

Arbeta med namngivna filer, exempel

```
/* Program to display a file. */
#include <stdio.h>

int main(void) {
     FILE* infile = fopen("/home/joakim/test.txt", "r");

     if (infile) // Check if file could be opened
     {
          int nextChar;
          while ( ( nextChar = getc( infile ) ) != EOF )
          {
               putchar(nextChar);
               /* Could also have used e.g.
                  putc(nextChar, stdout); */
          }
          fclose(infile);
     }
     return 0;
}
```

Speciella filer: **`stdout, stdin, stderr`**

- Filer eller “strömmar” (*streams*) som automatiskt öppnas när ett program startas, och stängs när programmet avslutas
- Alla tre filer motsvarar normalt terminalfönstret där programmet körs
- **`stderr`** används för felmeddelanden – kan t.ex. styra felmeddelanden till en viss fil
- Funktionen **`freopen()`** kan användas för att ändra på destinationen för dessa filer:

```
FILE* freopen( const char* path,
               const char* mode, FILE* stream )
```

- För att t.ex. skriva felmeddelanden till **/home/joakim/errors.log**:

```
freopen("/home/joakim/errors.log", "a", stderr);
```

stderr, exempel

```
freopen("errors.log", "a", stderr);

char filename[] = "foo.txt";
FILE* f = fopen(filename, "w");
if (f==NULL)
{
     // Output error message to errors.log
     fprintf(stderr,
               "Could not open %s for writing!\n",
               filename);
     exit(0);
}

// Do some writing…

fclose(f);
```