Standardfunktioner i C

Include-filer i standarden: (enligt C11*)

<assert.h>	<complex.h>	<ctype.h>	<errno.h>
<fenv.h>	<float.h>	<inttypes.h>	<iso646.h>
<limits.h>	<locale.h>	<math.h>	<setjmp.h>
<signal.h>	<stdalign.h>	<stdarg.h>	<stdatomic.h>
<stdbool.h>	<stddef.h>	<stdint.h>	<stdio.h>
<stdlib.h>	<string.h>	<tgmath.h>	<threads.h>
<time.h>	<uchar.h>	<wchar.h>	<wctype.h>

<. . .> även i C99
<. . .> bara i C11
<. . .> frivilliga

* Standarden C11 antogs 2011, sista utkast kan laddas ner gratis från
http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1539.pdf

<assert.h>

Debug-hjälpmedel

Ex.

```
#define NDEBUG
#include <assert.h>

. . .

assert(expression);
```

Assertion failed: *expression*, function *abc*, file *xyz*, line *nnn*.

<ctype.h>

Funktioner för att testa och ändra enskilda tecken.

```
int isalpha(int c);      // bokstav?
int isdigit(int c);      // siffra?
int isxdigit(int c);     // hexadecimal siffra ?
int isalnum(int c);      // bokstav eller siffra?
int isprint(int c);      // skrivbart tecken?
int isspace(int c);      // "vitt" tecken?
int isupper(int c);      // stor bokstav?
int islower(int c);      // liten bokstav?
int iscntrl(int c);      // styrtecken?
int isgraph(int c);      // skrivbart? (ej ' ')
int isprint(int c);      // skrivbart? (även ' ')

int tolower(int c);      // stor bokstav -> liten
int toupper(int c);      // liten bokstav -> stor
```

<float.h>

Makron som anger gränser för floating point-typer.

Ex.

FLT_DECIMAL_DIG, DBL_DECIMAL_DIG,

FLT_MIN_10_EXP, DBL_MIN_10_EXP, FLT_MAX_10_EXP, DBL_MAX_10_EXP

<limits.h>

Makron som anger gränser för heltalstyper.

Ex.

SHRT_MIN, SHRT_MAX, INT_MIN, UINT_MAX, LONG_MIN, LONG_MAX,

SCHAR_MIN, SCHAR_MAX, UCHAR_MAX, CHAR_MIN, CHAR_MAX

<iso646.h>

Makron för alternativ syntax:

and	&&
and_eq	&=
bitand	&
bitor	|
compl	~
not	!
not_eq	!=
or	||
or_eq	|=
xor	^
xor_eq	^=

<math.h>

Parametrar och resultat av typen `double`:

```
acos  asin  atan  cos   sin   tan   cosh  sinh   round
tanh  exp   log   log10 sqrt  ceil  floor fabs   trunc

pow(x,y)  fmod(x,y)  fmax(x,y)  fmin(x,y)
atan2(x,y)  // arctan y/x
```

Parametrar och resultat av typen `float`:

```
acosf    asinf  atanf  etc.
```

Parametrar och resultat av typen `long double`:

```
acosl    asinl  atanl  etc.
```

Funktioner som ger heltalsresultat:

```
lround  lroundf  lroundl     // ger long int
llround  llroundf  llroundl  // ger long long int
```

<stdarg.h>

Macron för funktioner med variabelt antal parametrar

```
void va_start(va_list ap, parmN);    // parmN är sista namngivna param

type va_arg(va_list ap, type);       // ger nästa parameter, typ: type

void va_end(va_list ap);             // avslutar
```

```
#include <stdarg.h>
#define MAXARGS 31
void f1(int n_ptrs, ...) {
  va_list ap;
  char *array[MAXARGS];
  if (n_ptrs > MAXARGS)
    n_ptrs = MAXARGS;
  va_start(ap, n_ptrs);
  while (ptr_no < n_ptrs)
    array[ptr_no++] = va_arg(ap, char *);
  va_end(ap);
  f2(n_ptrs, array);
}
```

<stdbool.h>

Macron:

`bool` → **`_Bool`**

`true` → `1`

`false` → `0`

`__bool_true_false_are_defined` → `1`

<stddef.h>

Typer:

`size_t`

`wchar_t`

`ptrdiff_t`

`max_align_t`

Macron:

`NULL`

`offsetof(`*`type`*`, `*`member-designator`*`)`

<stdio.h>

Typer:

```
FILE
fpos_t
```

Macron, bl.a:

```
EOF
FOPEN_MAX
FILENAME_MAX
SEEK_CUR
SEEK_END
SEEK_SET
```

Fördefinierade strömmar :

```
stderr
stdin
stdout
```

Filer

```
/* Exempel på öppning av filer */
FILE *infil, *utfil, *trans, *mfil;

infil = fopen("kontoplan", "r");     // endast läsning

utfil = fopen("rapport", "w");       // för skrivning, skapa filen
                                     // om den inte finns redan, annars
                                     // töm filen på tidigare innehåll

trans = fopen("transaktion", "a+"); // för tillägg till slutet
                                     // men även för läsning

mfil = fopen("mätdata", "r+b");      // läsning/skrivning av en
                                     // existerande binärdatafil
```

Andra funktioner för hantering av filer;

```
int fclose(FILE *stream);

int fflush(FILE *stream);

FILE *tmpfile(void);

int remove(const char *filname);

int rename(const char *old, const char *new);
```

Enkla funktioner för läsning och skrivning

Motsvarande funktioner finns i <wchar.h> för wide characters, t.ex. putwc, gettwc

```
int fputc(int c, FILE *stream);
int putc(int c, FILE *stream);
int putchar(int c);
int fgetc(FILE *stream);
int getc(FILE *stream);
int getchar(void);
int ungetc(int c, FILE *stream);
int fputs(const char *s, FILE *stream);
int puts(const char *s);
char *fgets(char *s, int n, FILE *stream);
```

```
/* läs fil, kryptera och skriv på två utfiler */
int main() {
   FILE *in, *ut, *logg;
   char rad[500];
   if ((in = fopen("klartext.txt", "r")) == NULL) {
      fputs("Kan ej öppna infilen\n", stderr);
      exit(99);
   }
   if ((ut = fopen("hemlig.txt", "w")) == NULL) {
      fputs("Kan ej öppna utfilen\n", stderr);
      exit(99);
   }
   if ((logg = fopen("loggfil.log", "a")) == NULL) {
      fputs("Kan ej öppna loggfilen\n", stderr);
      exit(99);
   }
   while (fgets(rad, 500, in) != NULL) {
      koda(rad);
      if (fputs(rad, ut) == EOF) {
        fputs("Skrivfel på utfilen\n", stderr);
      }
      if (fputs(rad, logg) == EOF) {
        fputs("Skrivfel på loggfilen\n", stderr);
      }
   }
}
```

Formaterad in- och utmatning

```
int fprintf(FILE * restrict stream, const char * restrict format, ...);
int fscanf (FILE * restrict stream, const char * restrict format, ...);
int printf(const char * restrict format, ...);
int scanf (const char * restrict format, ...);
int sprintf(char * restrict s, const char * restrict format, ...);
int snprintf(char * restrict s, size_t n,
             const char * restrict format, ...);
int sscanf(const char * restrict s, const char * restrict format, ...);
```

Direktaccess

```
long int ftell(FILE *stream);

int fseek(FILE *stream, long int offset, int whence);

void rewind(FILE *stream);

size_t fread(void * restrict ptr, size_t size, size_t nmemb,
             FILE * restrict stream);

size_t fwrite(const void * restrict ptr, size_t size, size_t nmemb,
              FILE * restrict stream);
```

```c
#include <stdio.h>
/* byt plats på två poster i en fil */
struct vpost {
   long varunr;
   int antal;
};

int main()
{
   struct vpost v1,v2;
   FILE *lfil;
   lfil = fopen("lagerfil", "rb+");
   fseek(lfil, 58 * sizeof(struct vpost), SEEK_SET);
   fread(&v1, sizeof(struct vpost), 1, lfil);
   fread(&v2, sizeof(struct vpost), 1, lfil);
   fseek(lfil, -2 * sizeof(struct vpost), SEEK_CUR);
   fwrite(&v2, sizeof(struct vpost), 1, lfil);
   fwrite(&v1, sizeof(struct vpost), 1, lfil);
}
```

Felhantering

```
errno         // sätts automatiskt vid ev. fel
void clearerr(FILE *stream);    // nollställer
int ferror(FILE *stream);       // testar om något fel inträffat
int feof(FILE *stream);         // testar om end of file inträffat
void perror(const char *s);     // skriver ut s + lämplig felutskrift
```

```
<stdlib.h>

double atof(const char *nptr);

int atoi(const char *nptr);

long int atol(const char *nptr);

long long int atoll(const char *nptr);

double strtod(const char * restrict nptr, char ** restrict endptr);

float strtof(const char * restrict nptr, char ** restrict endptr);

long double strtold(const char * restrict nptr,char ** restrict endptr);

long int strtol(const char * restrict nptr,
                char ** restrict endptr, int base);
long long int strtoll(const char * restrict nptr,
                      char ** restrict endptr, int base);
unsigned long int strtoul(const char * restrict nptr,
                          char ** restrict endptr, int base);
unsigned long long int strtoull(const char * restrict nptr,
                                char ** restrict endptr, int base);
```

```
RAND_MAX
int rand(void);
void srand(unsigned int seed);
```

Ex. Implementering

```
static unsigned long int next = 1;

int rand(void) // RAND_MAX assumed to be 32767
{
  next = next * 1103515245 + 12345;
  return (unsigned int)(next/65536) % 32768;
}

void srand(unsigned int seed)
{
  next = seed;
}
```

```
void *malloc(size_t size);

void *calloc(size_t nmemb, size_t size);

void *realloc(void *ptr, size_t size);

void *aligned_alloc(size_t alignment, size_t size);  // ny i C11

void free(void *ptr);

EXIT_FAILURE
EXIT_SUCCESS

void exit(int status);

char *getenv(const char *name);

int system(const char *string);

void *bsearch(const void *key, const void *base,
              size_t nmemb, size_t size,
              int (*compar)(const void *, const void *));

void qsort(void *base, size_t nmemb, size_t size,
           int (*compar)(const void *, const void *));
```

Multibyte (UTF-8)

```
int mblen(const char *s, size_t n);   // n==antal bytes i s

int mbtowc(wchar_t * restrict pwc, const char * restrict s, size_t n);

int wctomb(char *s, wchar_t wc);     // översätter wc till en multibyte


size_t mbstowcs(wchar_t * restrict pwcs,
                const char * restrict s, size_t n);       // *s -> *pwcs

size_t wcstombs(char * restrict s,
                const wchar_t * restrict pwcs, size_t n); // *pwcs -> *s
```

<string.h>

Operationer på char-arrays:

```
void *memcpy(void * restrict s1, const void * restrict s2, size_t n);

void *memmove(void *s1, const void *s2, size_t n);   // ev. Overlap

int memcmp(const void *s1, const void *s2, size_t n);

void *memchr(const void *s, int c, size_t n);  // letar efter c i *s

void *memset(void *s, int c, size_t n);        // lägger in n st c i *s
```

Texthantering:

```
char *strcpy (char *s1, const char *s2);
char *strncpy(char *s1, const char *s2, size_t n)
char *strcat (char *s1, const char *s2);
char *strncat(char *s1, const char *s2, size_t n);

int strcmp (char *s1, const char *s2);
int strncmp(char *s1, const char *s2, size_t n);

size_t strlen(const char *s);

size_t strspn (const char *s, const char *s2);
size_t strcspn(const char *s, const char *s2);


/* räkna antalet hexadecimala siffror i början av str */
antalhexsif = strspn(str, "0123456789ABCDEFabdcef");
```

```
char *strchr (const char *s, int c);
char *strrchr(const char *s, int c);
char *strpbrk(const char *s, const char *s2);
char *strstr (const char *s, const char *s2);
```

```
/* byt alla ? mot ! i strängen str */
char *p = str;
while ((p = strchr(p, '?')) != NULL) {
   *p = '!';
}
```

```
/* byt alla "katt" mot "hund" i strängen str */
char *p = str;
while ((p = strstr(p, "katt")) != NULL) {
   strncpy(p, "hund", 4);
}
```

```
char *strtok(char * restrict s1, const char * restrict s2);
```

Ex.

```
#include <string.h>
static char str[] = "?a???b,,,#c";
char *t;
t = strtok(str, "?"); // t points to the token "a"
t = strtok(NULL, ","); // t points to the token "??b"
t = strtok(NULL, "#,"); // t points to the token "c"
t = strtok(NULL, "?"); // t is a null pointer
```