Übersicht über Funktionen der ANSI-Standardbibliothek

Die Funktionen sind in Headerfiles vereinbart und in der folgenden Zusammenstellung alphabetisch geordnet. Zu jeder Funktion ist die Spezifikation der Typen der Argumente und des Ergebnisses angegeben. Vergleiche dazu auch die Manualseiten man(3).

assert.h Diagnoseinformationen

void **assert**(int expr); /* Diagnoseausgaben, wenn expr wahr */

ctype.h Behandlung von Zeichen (Makros)

Die Makros liefern den Wert TRUE, wenn c die angegebene Bedingung erfüllt:

int **isalnum**(int c); /* Buchstabe */
int **isalpha**(int c); /* alphanumerisches Zeichen */
int **iscntrl**(int c); /* Steuerzeichen */
int **isdigit**(int c); /* Ziffer 0-9 */
int **isgraph**(int c); /* druckbares Zeichen, 33-126 */
int **islower**(int c); /* Kleinbuchstabe */
int **isprint**(int c); /* druckbares Zeichen */
int **ispunct**(int c); /* Interpunktionszeichen */
int **isspace**(int c); /* Trennzeichen */
int **isupper**(int c); /* Großbuchstabe */
int **isxdigit**(int c); /* Hexadezimalziffer 0-f */

Die folgenden Makros dienen der Zeichenumwandlung:

int **tolower**(int c); /* Groß- in Kleinbuchstaben */
int **toupper**(int c); /* Klein- in Großbuchstaben */

errno.h Definition von Fehlernummern (errno)

locale.h Lokalisierung

char *__setlocale__(int category, const char *locale); /* Setzen der Umgebung für örtliche Besonderheiten z.B. Sprache, Zeit usw. */
struct lconv *__localeconv__(void); /* Bereitstellen der lokalen Information */

setjmp.h Nichtlokale Verzweigungen

int **setjmp**(jmp_buf env); /* Abspeichern Prozeßzustand */
void **longjmp**(jmp_buf env, int val); /* Wiederherstellen Prozeßzustand */

signal.h Signalbehandlung

Definition der Signaltypen (SIG ...)

void (*__signal__(int sig, void (*func)(int)))(int); /* Funktion zur Behandlung eines Signals angeben */
int **raise**(int sig); /* Signal zum laufenden Programm senden */

stdarg.h Funktionen mit variabler Argumentanzahl (Makros)

'type' **va_arg**(va_list ap, 'type'); /* liefert variables Argument */
void **va_end**(va_list ap); /* Beendet Auswertung der Argumente */
void **va_start**(va_list ap, parm'N'); /* Beginnt Auswertung bei Argument 'N' */

stddef.h Definition von Datentypen (size_t)

math.h mathematische Funktionen (Gleitkomma)

double **acos**(double x);	/* Arcuscosinus */
double **asin**(double x);	/* Arcussinus */
double **atan**(double x);	/* Arcustangens */
double **atan2**(double y, double x);	/* Arcustangens von x/y */
double **ceil**(double x);	/* Aufrunden */
double **cos**(double x);	/* Cosinus */
double **cosh**(double x);	/* Cosinus hyperbolicus */
double **exp**(double x);	/* e hoch x */
double **fabs**(double x);	/* Absolutwert, Gleitkomma */
double **floor**(double x);	/* Abrunden */
double **fmod**(double x, double y);	/* Divisionsrest */
double **frexp**(double value, int *exp);	/* Mantisse und Exponent */
double **ldexp**(double x, int exp);	/* x * 2 hoch exp */
double **log**(double x);	/* Natürlicher Logarithmus */
double **log10**(double x);	/* Logarithmus zur Basis 10 */
double **modf**(double value, double *iptr);	/* Gleitkomma-Modulo */
double **pow**(double x, double y);	/* x hoch y */
double **sin**(double x);	/* Sinus */
double **sinh**(double x);	/* Sinus hyperbolicus */
double **sqrt**(double x);	/* Quadratwurzel */
double **tan**(double x);	/* Tangens */
double **tanh**(double x);	/* Tangens hyperbolicus */

string.h Zeichenkettenbehandlung

void *__memchr__(const void *s, int c, size_t n);	/* Zeichen im Speicher suchen */
int **memcmp**(const void *s1, const void *s2, size_t n);	/* Speicherbereiche vergleichen */
void *__memcpy__(void *s1, const void *s2, size_t n);	/* Speicherbereiche kopieren */
void *__memmove__(void *s1, const void *s2, size_t n);	/* Speicherbereiche verlagern */
void *__memset__(void *s, int c, size_t n);	/* Speicherbereiche belegen */
char *__strcat__(char *s1, const char *s2);	/* Verketten von Strings */
char *__strchr__(const char *s, int c);	/* Zeichen suchen im String */
int **strcmp**(const char *s1, const char *s2);	/* Vergleichen zweier Strings */
int **strcoll**(const char *s1, const char *s2);	/* Vergleich nach lokalen Festlegungen */
char *__strcpy__(char *s1, const char *s2);	/* String kopieren */
size_t **strcspn**(const char *s1, const char *s2);	/* Länge eines Teil-Strings */
char *__strerror__(int errnum);	/* String mit Systemfehlermeldung */
size_t **strlen**(const char *s);	/* Länge eines Strings */
char *__strncat__(char *s1, const char *s2, size_t n);	/* n Zeichen anfügen */
int **strncmp**(const char *s1, const char *s2, size_t n);	/* n Zeichen zweier Strings vergleichen */
char *__strncpy__(char *s1, const char *s2, size_t n);	/* n Zeichen kopieren */
char *__strpbrk__(const char *s1, const char *s2);	/* Zeichen im String suchen */
char *__strrchr__(const char *s, int c);	/* letztes Vorkommen des Zeichens */
size_t **strspn**(const char *s1, const char *s2);	/* Länge eines Teil-Strings */
char *__strstr__(const char *s1, const char *s2);	/* String im String suchen */
char *__strtok__(char *s1, const char *s2);	/* Symbole im String suchen */
size_t **strxfrm**(char *s1, const char *s2, size_t n);	/* String-Transformation */

stdio.h Ein-/Ausgabe

void **clearerr**(FILE *stream);	/* Rücksetzen EOF-Bedingung */
int **fclose**(FILE *stream);	/* File schließen */
int **feof**(FILE *stream);	/* Filende prüfen */
int **ferror**(FILE *stream);	/* Fehler bei E/A prüfen */
int **fflush**(FILE *stream);	/* Puffer leeren (ausgeben) */
int **fgetc**(FILE *stream);	/* 1 Zeichen aus File lesen */
int **fgetpos**(FILE *stream, fpos_t *pos);	/* liefert aktuelle Fileposition */
char ***fgets**(char *s, int n, FILE *stream);	/* 1 Zeile aus File lesen */
FILE ***fopen**(const char *filename, const char *mode);	/* File eröffnen */
int **fprintf**(FILE *stream, const char *format, ...);	/* format. Ausgabe in File */
int **fputc**(int c, FILE *stream);	/* 1 Zeichen in File ausgeben */
int **fputs**(const char *s, FILE *stream);	/* 1 Zeichenkette in File ausgeben */
size_t **fread**(void *ptr, size_t size, size_t nmemb, FILE *stream);	/* Zeichen aus File lesen */
FILE ***freopen**(const char *filename, const char *mode, FILE *stream);	/* neues File zuordnen */
int **fscanf**(FILE *stream, const char *format, ...);	/* formatierte Eingabe aus File */
int **fseek**(FILE *stream, long int offset, int whence);	/* verändert aktuelle Fileposition */
int **fsetpos**(FILE *stream, const fpos_t *pos);	/* setzt aktuelle Fileposition */
long int **ftell**(FILE *stream);	/* liefert aktuelle Fileposition */
size_t **fwrite**(const void *ptr, size_t size, size_t nmemb, FILE *stream);	/* Zeichen in File schreiben */
int **getc**(FILE *stream);	/* 1 Zeichen aus File lesen */
int **getchar**(void);	/* 1 Zeichen von stdin lesen */
char ***gets**(char *s);	/* 1 Zeile von stdin lesen */
void **perror**(const char *s);	/* Systemfehlermeldung */
int **printf**(const char *format, ...);	/* formatierte Ausgabe nach stdout */
int **putc**(int c, FILE *stream);	/* 1 Zeichen in File ausgeben */
int **putchar**(int c);	/* 1 Zeichen nach stdout ausgeben */
int **puts**(const char *s);	/* 1 Zeichenkette nach stdout ausgeben */
int **remove**(const char *filename);	/* File löschen */
int **rename**(const char *old, const char *new);	/* File umbenennen */
void **rewind**(FILE *stream);	/* Fileanfang einstellen */
int **scanf**(const char *format);	/* formatierte Eingabe von stdin */
void **setbuf**(FILE *stream, char *buf);	/* Pufferzuordnung */
void **setvbuf**(FILE *stream, char *buf, int mode, size_t size);	/* Pufferzuordnung */
int **sprintf**(char *s, const char *format, ...);	/* formatierte Ausgabe als Zeichenkette */
int **sscanf**(const char *s, const char *format, ...);	/* formatiertes Lesen einer Zeichenkette */
FILE ***tmpfile**(void);	/* erzeugt temporäres File */
char ***tmpnam**(char *s);	/* erzeugt temporären Filenamen */
int **ungetc**(int c, FILE *stream);	/* 1 gelesenes Zeichen zurückstellen */
int **vfprintf**(FILE *stream, const char *format, va_list arg);	/* formatierte Ausgabe mit variablen Arg. */
int **vprintf**(const char *format, va_list arg);	/* formatierte Ausgabe mit var. Arg. */
int **vsprintf**(char *s, const char *format, va_list arg);	/* formatierte Ausgabe mit var. Arg. */

stdlib.h Allgemeine Funktionen

- Zeichenkettenkonvertierung
  double **atof**(const char *nptr); /* Zeichenkette in float-Zahl */
  int **atoi**(const char *nptr); /* Zeichenkette in int-Zahl */
  long int **atol**(const char *nptr); /* Zeichenkette in long-Zahl */
  double **strtod**(const char *nptr, char **endptr); /* Zeichenkettenteil in double-Zahl */
  long int **strtol**(const char *nptr, char **endptr, int base); /* Zeichenkettenteil in long-Zahl */
  unsigned long int **strtoul**(const char *nptr, /* Zeichenkettenteil in unsigned-long */
  char **endptr, int base);

- Pseudozufallszahlen
  int **rand**(void); /* erzeugt Pseudo-Zufallszahl */
  void **srand**(unsigned int seed); /* neuer Startwert für Berechnung der Zufallszahlen */

- Speicherverwaltung
  void ***calloc**(size_t nmemb, size_t size); /* Hauptspeicher zuweisen */
  void **free**(void *ptr); /* Hauptspeicher freigeben */
  void ***malloc**(size_t size); /* Hauptspeicher zuweisen */
  void ***realloc**(void *ptr, size_t size); /* Hauptspeicherbereich ändern */

- Umgebungsbezugnahmen
  void **abort**(void); /* Prozeß abbrechen */
  int **atexit**(void (*func)(void)); /* Angabe von Funktionen, die bei exit gerufen werden */
  void **exit**(int status); /* Programm beenden */
  char ***getenv**(const char *name); /* Umgebungsvariable lesen */
  int **system**(const char *string); /* Kommandos des umgebenden Systems ausführen */

- Such-und-Sortierfunktionen
  void ***bsearch**(const void *key, const void *base, /* Primäres Suchen */
  size_t nmemb, size_t size, int (*compar)(const void *,
  const void *))
  void **qsort**(void *base, size_t nmemb, size_t size, /* Feldsortierung */
  int (*compar)(const void *, const void *));

- arithmetische-Integerfunktionen
  int **abs**(int j); /* Absolutwert einer int-Zahl */
  div_t **div**(int numer, int denom); /* Division von int-Zahlen */
  long int **labs**(long int j); /* Absolutwert einer long-Zahl */
  ldiv_t **ldiv**(long int numer, long int denom); /* Division von long-Zahlen */

- Multibyte-Zeichen
  int **mblen**(const char *s, size_t n); /* Anzahl Bytes für Multi-char */
  int **mbtowc**(wchar_t *pwc, const char *s, size_t n); /* Code für Multi-char bestimmen */
  int **wctomb**(char *s, wchar_t wchar); /* Multi-char aus Code bilden */
  size_t **mbstowcs**(wchar_t *pwcs, const char *s, /* Multi-char-Folge in Codefolge */
  size_t n);
  size_t **wcstombs**(char *s, const wchar_t *pwcs, /* Codefolge in Multi-char-Folge */
  size_t n);

time.h Datum und Uhrzeit

char ***asctime**(const struct tm *timeptr); /* Umwandlung von Datum und Uhrzeit in Zeichenkette */
clock_t **clock**(void); /* Prozessorzeit des Programms */
char ***ctime**(const time_t *timer); /* Datum und Uhrzeit als Zeichenkette */
double **difftime**(time_t time1, time_t time0); /* Zeitunterschied berechnen */
struct tm ***gmtime**(const time_t *timer); /* Universalzeit im internen Format */
struct tm ***localtime**(const time_t *timer); /* lokale Zeit im internen Format */
time_t **mktime**(struct tm *timeptr); /* Umwandlung der lokalen Zeit in time-Format */
size_t **strftime**(char *s, size_t maxsize, /* formatierte Datum-/Zeit-Ausgabe */
const char *format, const struct tm *timeptr);
time_t **time**(time_t *timer); /* aktuelle Tageszeit intern */