Freie Universität Berlin

Die C-Standardbibliothek

Freie Universität Berlin

assert.h – Zusicherungen

- Die Funktion **assert**(3) aus `<assert.h>` implementiert eine *Zusicherung*
- Beispiel: `assert(size > 0);` Ist `size <= 0`, so wird das Programm mit einer Fehlermeldung abgebrochen.
- Zusicherungen können durch Definition von `NDEBUG` abgeschaltet werden.

Freie Universität Berlin

ctype.h – Klassifizierung von Zeichen

- Enthält Funktionen zur Klassifierung von Zeichen und zur Änderung der Groß- und Kleinschreibung
- Klassifizierung: **isalnum**(3), **isalpha**(3), **isblank**(3), **iscntrl**(3), **isdigit**(3), **isgraph**(3), **islower**(3), **isprint**(3), **ispunct**(3), **isspace**(3), **isupper**(3), **isxdigit**(3)
- Umwandlung der Groß- und Kleinschreibung: **tolower**(3), **toupper**(3)

Freie Universität Berlin

errno.h – Errocodes

- Definiert die Variable **errno**(2), die den Fehlercode des letzten Systemaufrufs enthält
- Für jeden Fehlercode gibt es eine Enumerationskonstante, z. B. `ENOENT` »Datei nicht gefunden«, `EPERM` »Zugriff verweigert«.
- mit **strerror**(3) aus `<string.h>` kann man eine Fehlerbeschreibung abrufen
- **perror**(3) aus `<stdio.h>` gibt diese Beschreibung mit wählbarem Präfix aus
- Beispiel:

```
p = malloc(sizeof(*p)); if (p == NULL) perror("malloc");
```

Freie Universität Berlin

limits.h – Typgrenzen

- Definiert maximale und minimale Werte von Typen, generell der Form *xxx*`_MIN` und *xxx*`_MAX`
- Anderer wichtiger Wert: `CHAR_BIT` (Anzahl der Bits in einem Byte)

Freie Universität Berlin

math.h – mathematische Funktionen

- Mathematische Funktionen, zur Verwendung muss mit `-lm` gelinkt werden, z. B. `cc -o prog prog.c -lm`
- Große Bibliothek, Beschreibung aller Funktionen würde den Rahmen sprengen
- Auswahl an Funktionen: **sin**(3), **cos**(3), **tan**(3), **asin**(3), **acos**(3), **atan**(3), **exp**(3), **pow**(3), **log**(3), **sqrt**(3), **ceil**(3), **floor**(3), **round**(3), …

Freie Universität Berlin

setjmp.h – nichtlokale Sprünge

- **setjmp**(3) setzt eine nichtlokale Sprungmarke
- **longjmp**(3) springt zur übergebenen Sprungmarke
- Es darf nur in eine Funktion zurückgesprungen werden, die noch nicht beendet worden ist
- wir werden **setjmp**(3) nicht näher verwenden

Freie Universität Berlin

signal.h – Signalbehandlung

- Signale werden vom Betriebssystem oder anderen Prozessen gesendet, um Prozesse zu steuern
- Beispiel: Tippen Sie Ctrl + C, so wird ein `SIGINT` gesendet
- Prozesse können auf Signale mit Signalhandlern reagieren, die Sie **signal**(3) einrichten können

stdarg.h – Variable Argumentlisten

- Mit `<stdarg.h>` können Funktionen wie **printf**(3) geschrieben werden, die beliebig viele Argumente entgegen nehmen
- Eine Argumentliste hat den Typ `va_list`, wird mit **va_start** geöffnet, mit **va_copy**(3) kopiert und mit **va_end** geschlossen
- Einzelne Argumente können mit **va_arg**(3) aus der Liste gezogen werden – auf richtigen Typ achten!

Freie Universität Berlin

stdbool.h – Boolesche Werte

- enthält `typedef _Bool bool`
- sowie Werte `true` und `false`

Freie Universität Berlin

stddef.h – Basisdefinitionen

- enthält wichtige Basistypen und Makros
- Typen: `ptrdiff_t` (Differenz von Zeiger), `max_align_t` (Typ maximaler Ausrichtung), `size_t` (Größe von Objekten), `wchar_t` (Multibyte-Zeichen)
- Makros: `offsetof(`*`type`*`,` *`member`*`)` (Offset des Feldes *member* in Struktur *type*), `NULL` (Nullzeiger, ist definiert als `((void *)0)`)
- viele Header binden `<stddef.h>` implizit ein

stdint.h – Ganzzahltypen

- Definiert jede Menge Typen fester Länge, siehe Handout vom Tag 2
- Außerdem wichtige Typen: `intptr_t` (Integer, der groß genug ist, um einen Zeiger zu speichern), `uintptr_t` (vozeichenloser Integer, der groß genug ist, um einen Zeiger zu speichern), `intmax_t` (größter Integertyp), `uintmax_t` (größter vorzeichenloser Integertyp)
- Und zugehörige *xxx*`_MIN` und *xxx*`_MAX`-Makros

stdio.h – Ein- und Ausgabe

- Sehr großer Umfang, der im wesentlichen mit `FILE`-Strukturen arbeitet, hier nur wichtige Teile
- Formatierte Ein- und Ausgabe: **printf**(3), **fprintf**(3), **sprintf**(3), **snprintf**(3), **scanf**(3), **fscanf**(3), **sscanf**(3), …
- Zeichenweise Ein- und Ausgabe: **getc**(3), **putc**(3), **getchar**(3), **putchar**(3), **fgetc**(3), **fputc**(3), **ungetc**(3)
- Zeilenweise Ein- und Ausgabe: **fgets**(3), **fputs**(3), **puts**(3), **perror**(3)
- Binäre Ein- und Ausgabe: **fread**(3), **fwrite**(3)
- Dateisteuerung: **fopen**(3), **fclose**(3), **remove**(3), **ftello**(3), **fseeko**(3), **fflush**(3), **setvbuf**(3), **ferror**(3), **feof**(3)

Freie Universität Berlin

stdlib.h – allgemeine Funktionalität

- Wieder sehr großer Umfang, hier nur das Wichtigste
- Zufallszahlen erzeugen: **rand**(3), **srand**(3)
- Speicherallokation: **malloc**(3), **free**(3), **realloc**(3), **calloc**(3)
- Programmabbruch: **abort**(3), **exit**(3)
- Interaktion mit dem Betriebssystem: **getenv**(3), **system**(3)
- Suchen und Sortieren: **bsearch**(3), **qsort**(3)
- Betrag einer Zahl: **abs**(3), **absl**(3), **absll**(3)

Freie Universität Berlin

string.h – Zeichenketten

- Funktionalität für Strings und Puffer
- Kopieren: **memcpy**(3), **memmove**(3), **strcpy**(3), **strncpy**(3), **strcat**(3), **strncat**(3), **memset**(3)
- Strings untersuchen: **strlen**(3), **strcmp**(3), **strncmp**(3), **strchr**(3), **strrchr**(3), **memchr**(3), **strspn**(3), **strcspn**(3), **strpbrk**(3), **strstr**(3)
- Strings zerhacken: **strtok**(3)
- Fehlermeldungen: **strerror**(3)

Freie Universität Berlin

time.h – Datum und Uhrzeit

- Werkzeuge für Datum und Uhrzeit
- aktuelle Zeit bestimmen: **clock**(3), **time**(3)
- Zeit zerlegen und zusammenbauen: **difftime**(3), **mktime**(3), **gmtime**(3), **localtime**(3)
- Zeit formatiert ausgeben: **asctime**(3), **ctime**(3), **strftime**(3)