POSIX-Threads

- Um die Portierbarkeit von Programmen mit Threads zu ermöglichen, hat die IEEE einen Standard dafür definiert: IEEE-Standard 1003.1c

➢ Thread-Paket **`Pthreads`**:

- wird von den meisten UNIX-Systemen unterstützt.
- definiert über 60 Funktionsaufrufe

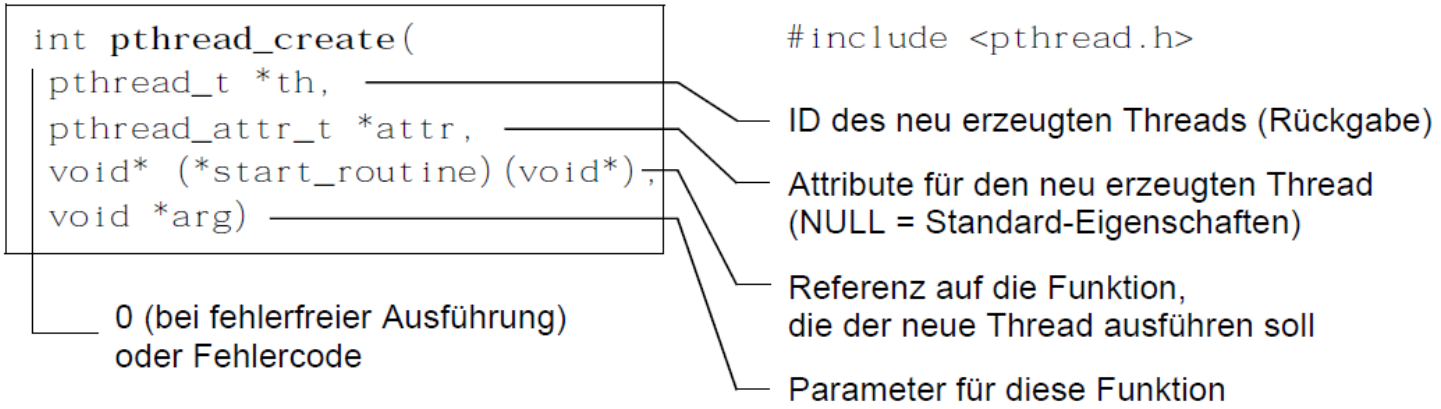
Thread-Aufruf	Beschreibung
pthread_create	Erzeugt einen neuen Thread
pthread_exit	Beendet den aufrufenden Thread
pthread_join	Wartet auf die Beendigung eines bestimmten Threads
pthread_yield	Gibt die CPU frei, damit andere Threads laufen können
pthread_attr_init	Erzeugt und initialisiert eine Attributstruktur
pthread_attr_destroy	Löscht die Attributstruktur eines Threads

Abbildung 2.14: Einige der Pthread-Funktionsaufrufe.

Quelle: [Tanenbaum und Bos, 2016]

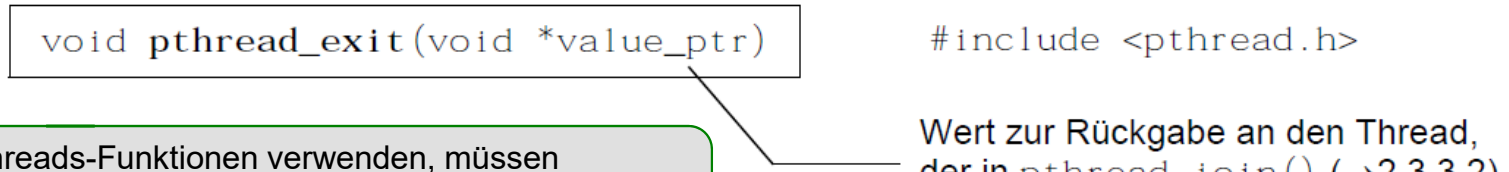
Pthreads erzeugen und beenden

- *Pthread* erzeugen:



  - `pthread_t` ist wie `pid_t` ein systemabhängiger Ganzzahltyp.
  - Die zurückgegebene `ID` eines Threads ist eine eindeutige Nummer.
    - Sie ist zu unterscheiden von der PID des Prozesses, dem der Thread zugeordnet ist.

- *Pthread* beendet:



Programme, die Pthreads-Funktionen verwenden, müssen unter Linux mit der Compiler-Option `-pthread` (oder auch `-lpthread`) übersetzt werden, also z. B. `cc -pthread programmname.c`.

Quelle: [Vogt, 2012]

Auf die Terminierung eines *Pthreads* warten

- Wenn der **`main`**-Thread sein Ende erreicht, beendet sich der gesamte Prozess und mit ihm auch vorzeitig die durch ihn gestarteten *PThreads*.
- Man kann dies durch die Anweisung **`pthread_exit()`** als letzte Anweisung in der **`main`**-Funktion verhindern:
  - In diesem Fall beendet sich lediglich der **`main`**-Thread, nicht aber die durch ihn gestarteten Threads.
  - Dazu müssen jedoch die an die Threads übergebenen Parameter (im folgenden Beispiel **`param1`** und **`param2`**) global deklariert werden, damit sie nach der Beendigung des **`main`**-Threads weiterhin zur Verfügung stehen und dürfen auch während der Ausführung der Threads nicht geändert werden.
- saubere Lösung: das Ende der beiden Threads mit **`pthread_join()`** abwarten:

```
int pthread_join(pthread_t thread, void **value_ptr)
```

0 im fehlerfreien Fall
Fehlercode sonst

```
#include <pthread.h>
```

Rückkehrstatus des Threads
(Rückgabeparameter)

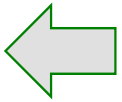
ID des Threads,
auf den gewartet werden soll

Quelle: [Vogt, 2012]

Beispiel für das Erzeugen und Ausführen eines *PThreads*

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#include <sys/types.h>
#include <unistd.h>

/* globale Variable, die für alle Threads zugreifbar ist */
int wertspeicher = 0;

/* Funktion, die von einem Thread ausgeführt werden soll */
void *ausgabe(void *p) {
 printf("Hier ist ein Thread\n");
 printf("Mein Funktionsparameter: %ld\n", *(long *)p);
 printf("Meine PID: %d\n",getpid());
 wertspeicher++;  /* erhöht die gemeinsame globale Variable */
 printf("Wertspeicher: %d\n\n",wertspeicher);
 pthread_exit(NULL);  /* beendet den Thread */
}
```

```
os@os:~/betriebssysteme/Vogt_Beispielprogramme$ gcc prog_2_08.c -oprog_2_08 -pthread
os@os:~/betriebssysteme/Vogt_Beispielprogramme$ ./prog_2_08
PID des Hauptprogramms: 12550
Wertspeicher: 0

Erzeuge den ersten Thread
Thread-ID 140349919319808

Erzeuge den zweiten Thread
Thread-ID 140349910927104

Hier ist ein Thread
Mein Funktionsparameter: 2222
Meine PID: 12550
Wertspeicher: 1

Hier ist ein Thread
Mein Funktionsparameter: 1111
Meine PID: 12550
Wertspeicher: 2

Programm beendet
os@os:~/betriebssysteme/Vogt_Beispielprogramme$
```



```c
/* Hauptprogramm */
int main(int argc, char *argv[]) {

 pthread_t th1, th2;   /* IDs der neu erzeugten Threads  */
 long param1, param2;  /* Parameter zur Übergabe an die Threads */
 int err;              /* Fehlercode von pthread_create() */
 void *status;         /* Rückgabeparameter von pthread_join */

 printf("PID des Hauptprogramms: %d\n",getpid());
 printf("Wertspeicher: %d\n\n",wertspeicher);

 /* Erzeugung eines Threads */
 printf("Erzeuge den ersten Thread\n");
 param1 = 1111;
 err = pthread_create(&th1, NULL, ausgabe, &param1);
    /* 1. Parameter: Rückgabewert = ID des neuen Threads
       2. Parameter: Attribute (hier: Standardattribute)
       3. Parameter: Funktion, die der Thread ausführen soll
       4. Parameter: Parameter für die Funktion  */
 if (err!=0) {
  printf("Fehler bei der Erzeugung des ersten Threads\n");
  exit(-1);
 }
 printf("Thread-ID %lu\n\n",th1);

 /* Erzeugung eines zweiten Threads */
 printf("Erzeuge den zweiten Thread\n");
 param2 = 2222;
 err = pthread_create(&th2, NULL, ausgabe, &param2);
 if (err!=0) {
  printf("Fehler bei der Erzeugung des zweiten Threads\n");
  exit(-1);
 }
 printf("Thread-ID %lu\n\n",th2);

 pthread_join(th1,&status);
 pthread_join(th2,&status);

 printf("Programm beendet\n");

}
```

Quelle: [Vogt, 2012]

Beispiel für das Warten auf die Terminierung eines *PThreads*

```
ifadmin@llc-off-site:~/Vogt_Beispielprogramme$ gcc prog_2_09.c -o prog_2_09 -pthread
ifadmin@llc-off-site:~/Vogt_Beispielprogramme$ ./prog_2_09
Main erzeugt Thread
Main wartet auf Thread in pthread_join()
Thread schlaeft 2 Sekunden
Thread ist fertig
Main: Thread beendet mit Status 0
ifadmin@llc-off-site:~/Vogt_Beispielprogramme$
```

```c
#include <pthread.h>
#include <stdio.h>

#include <unistd.h>

/* Funktion, die der Thread ausfuehr
void *schlafe(void *schlafzeit) {
 int sz = *(int*)schlafzeit;
 static int exitcode = 0;
 printf("Thread schlaeft %d Sekunden\n",sz);
 sleep(sz);  /* Thread blockiert sich eine Zeit lang */
 printf("Thread ist fertig\n");
 pthread_exit(&exitcode);  /* Thread beendet sich */
}

/* Hauptprogramm */
int main(int argc, char *argv[]) {
 int schlafzeit = 2;  /* Schlafzeit des Threads */
 pthread_t thread_id; /* ID des Threads */
 void *status;         /* Rueckkehrstatus des Threads */
 printf("Main erzeugt Thread\n");
 pthread_create(&thread_id, NULL, schlafe, &schlafzeit);
 printf("Main wartet auf Thread in pthread_join()\n");
 pthread_join(thread_id, &status);
 printf("Main: Thread beendet mit Status %d\n",*(int*)status);
}
```

Quelle: [Vogt, 2012]

Beispiel für das Beenden eines *PThreads* durch einen anderen Thread

- Terminierung eines Kind-Threads durch die Funktion **`pthread_cancel()`**:

```
 7 #include <pthread.h>
 8 #include <stdio.h>
 9
10 #include <unistd.h>
11
12 /* Funktion, die der Thread aus
13 void *endlosschleife() {
14  while (1)
15   printf(".");
16 }
17 /* Hauptprogramm */
18 int main(int argc, char *argv[]) {
19  pthread_t th;
20  /* Erzeugung eines Threads */
21  pthread_create(&th, NULL, endlosschleife, NULL);
22  /* Beenden des Threads nach einer Sekunde */
23  sleep(1);
24  pthread_cancel(th);
25  printf("Thread ist beendet\n");
26 }
```

```
ifadmin@llc-off-site:~/Vogt_Beispielprogramme$ gcc prog_2_10.c -o prog_2_10 -pthread
ifadmin@llc-off-site:~/Vogt_Beispielprogramme$ ./prog_2_10
.....................................................................................
.....................................................................................
.....................................................................................
                    ...
.....................................................................................
.....................................................................................
.....................Thread ist beendet
ifadmin@llc-off-site:~/Vogt_Beispielprogramme$
```

Quelle: [Vogt, 2012]