Programmierung paralleler Systeme
Threads unter Solaris

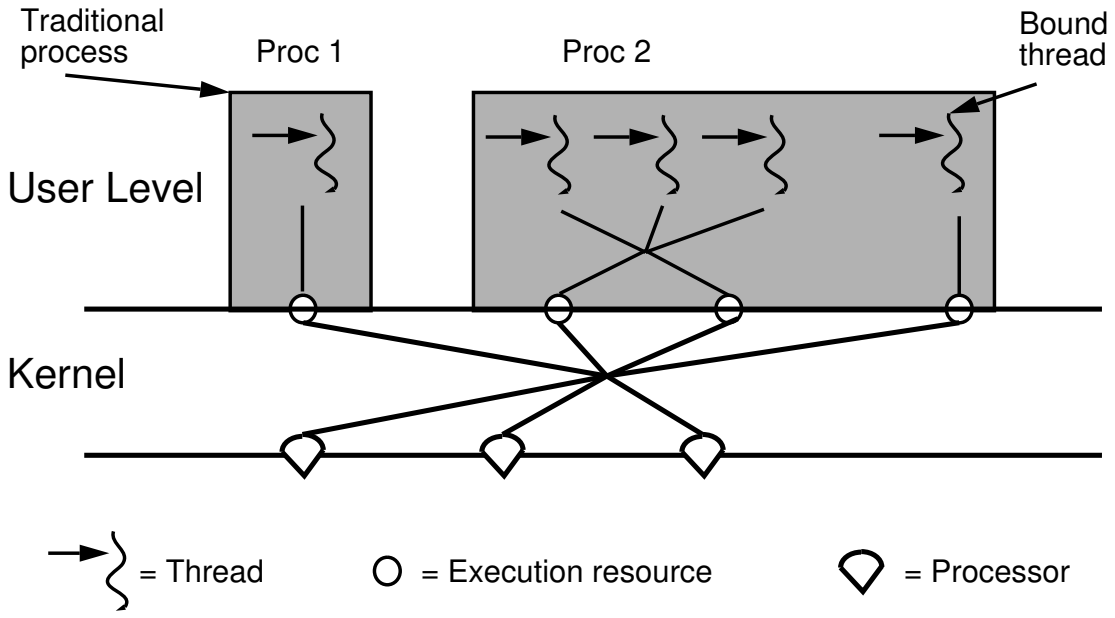
Frank Bellosa

Universität Erlangen-Nürnberg — IMMD IV
bellosa@informatik.uni-erlangen.de

thread.doc: 18.07.95 15:40

Architektur des Thread Interface

❑



thread.doc: 18.07.95 15:40

Thread Zustand

- Kernel Thread (LWP)
  - LWP ID
  - Stack
  - Register Zustand (incl. PC und Stackpointer)
  - Signal Maske
  - User virtual timer alarms
  - User time und System CPU usage
  - Profiling Status
  - Scheduling Classe und Priorität
- User-Level Thread
  - Thread ID
  - Stack
  - Register Zustand (evtl. ohne Floating Point ?)
  - Thread-lokaler Speicher (z.B. errno)
  - Priotität

thread.doc: 18.07.95 15:40

Solaris (faui42)

- Umgebung

```
if ( -d /opt) then
      eval `/usr/openwin/bin/resize`
      set path = ( /opt/SUNWspro/bin /usr/openwin/bin $path)
endif
```

- MANPATH:

```
/opt/SUNWspro/man
```

- Include-Files:

```
#include <thread.h>
#include <synch.h>
```

- Compiler

```
cc -g -mt -o grid grid.c -lthread -lm
```

- Debugger (Compiler-Option -g)

```
debugger
```

- Thread Analyzer (Compiler-Option -Ztha)

```
tha
```

thread.doc: 18.07.95 15:40

Prozeßstart und Terminierung

- FORK

```
thr_create(stack_base, stack_size, start_function, argument, FLAG, thread_id)
```

  - FLAGs:
    - THR_SUSPENDED
    - THR_DETACHED
    - THR_BOUND
    - THR_NEW_LWP
    - THR_DAEMON
  - Beispiel:

```
thr_create(NULL,0, smooth_t, &arg ,THR_SUSPENDED,&thread_id);
```

- Exit

```
thr_exit(return_value)
```

- Join

```
thr_join(thread_id, returned_thread_id, return_value)
```

thread.doc: 18.07.95 15:40

Gegenseitiger Ausschluß I

- Mutex:

```
mutex_init(), mutex_destroy()
mutex_lock(), mutex_trylock()
mutex_unlock()
```

- Semaphoren:

```
sema_init(), sema_destroy()
sema_post()
sema_trywait()
sema_wait()
```

- Reader/Writer-Locks

```
rwlock_init(), rwlock_destroy()
rw_rdlock(), rw_wrlock(), rw_tryrdlock(), rw_trywrlock();
rw_unlock()
```

thread.doc: 18.07.95 15:40

Gegenseitiger Ausschluß II

- Conditionals

```
cond_init(), cond_destroy()
cond_wait(), cond_timedwait()
cond_signal(), cond_broadcast()
```

  - Beispiel

```
mutex_lock(mutex);
while(cond_test() == FALSE) cond_wait(condition,mutex);
....
mutex_unlock(mutes);


mutex_lock(mutex);
......
cond_signal(condition);
mutex_unlock(mutex);
```

thread.doc: 18.07.95 15:40

Yield - Suspend - Resume

- Suspendierung eines anderen Thread

```
thr_suspend(thread_t);
thr_suspend(thr_self);
```

- Resume eines Threads

```
thr_continue(thread_t)
```

- Aufgabe des Prozessors

```
thr_yield()
```

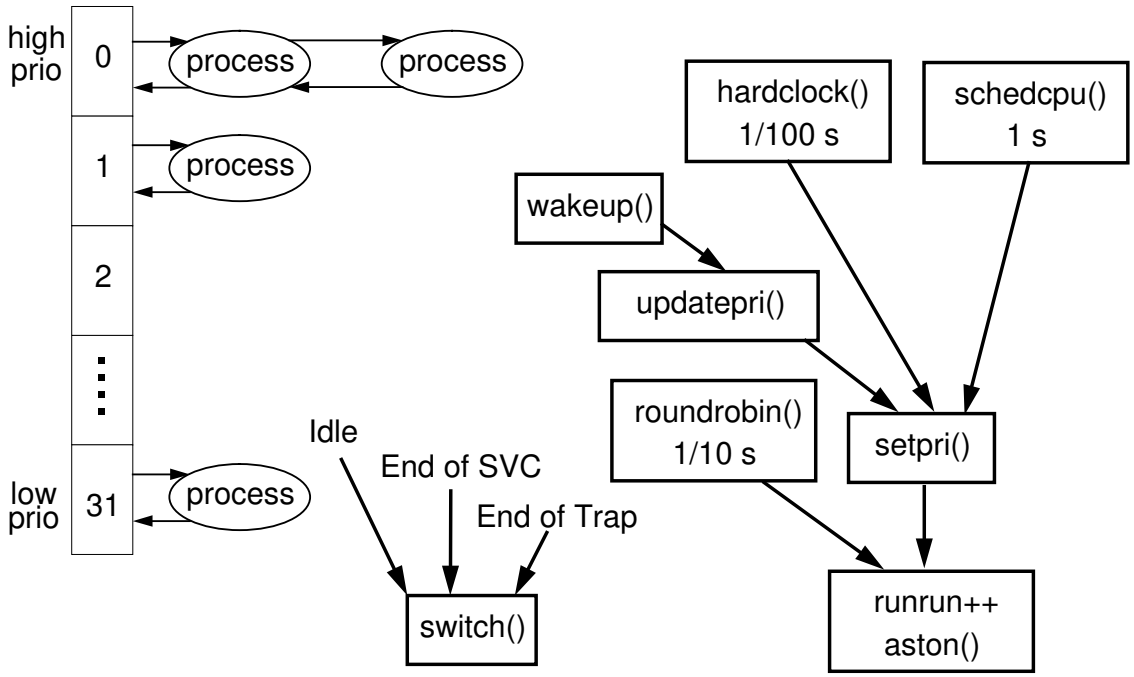
thread.doc: 18.07.95 15:40

Steuerung der Parallelität

- Kontrolle über die Kernel-Threads

```
thr_getconcurrency()
thr_setconcurrency()

.....
```


Aufbau des 4.3 BSD Scheduling