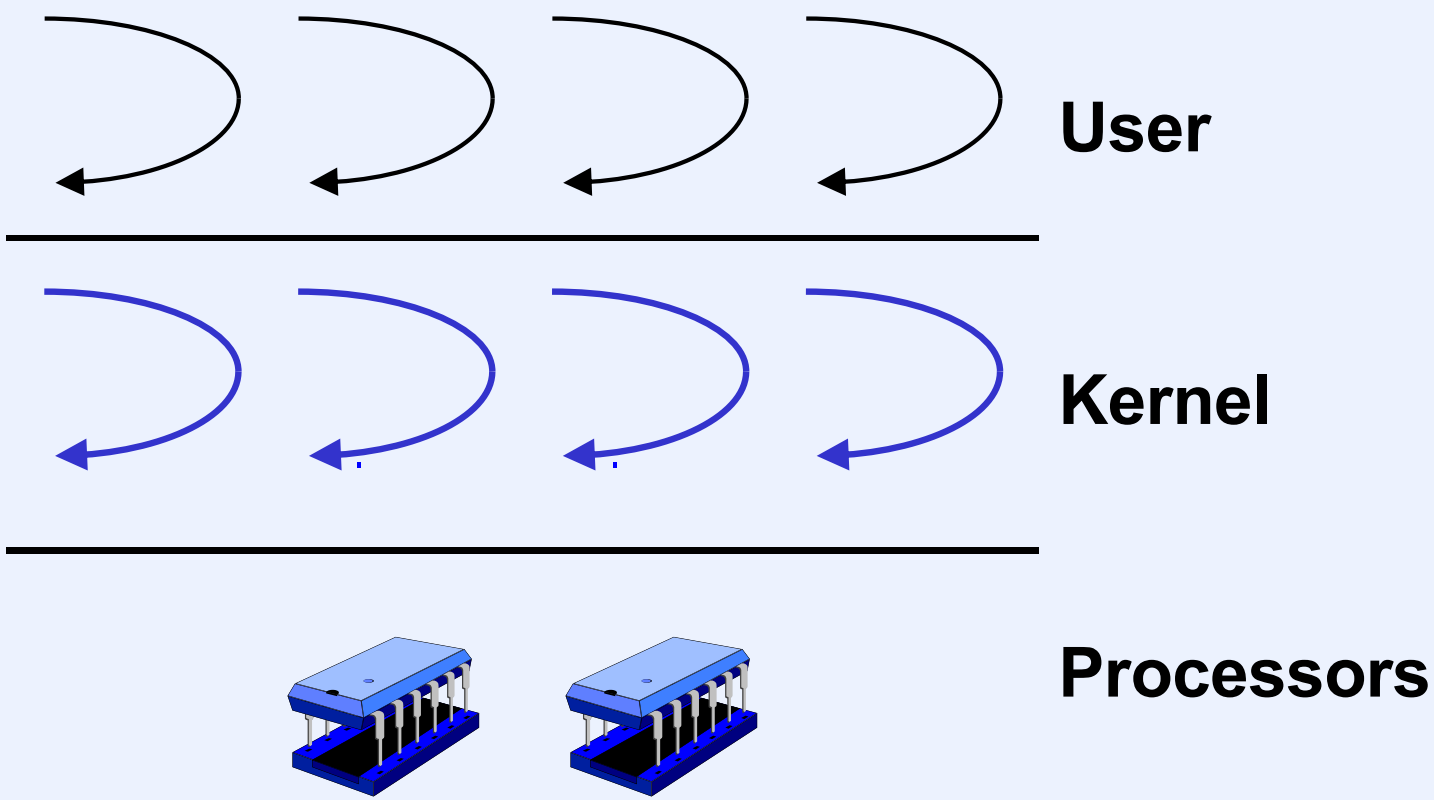
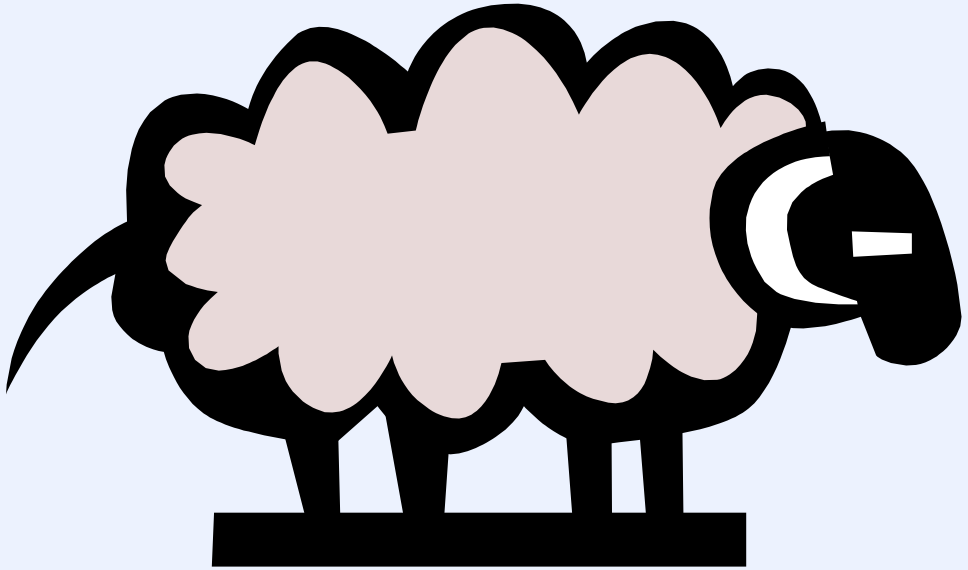
Implementing Threads 3

One-Level Model

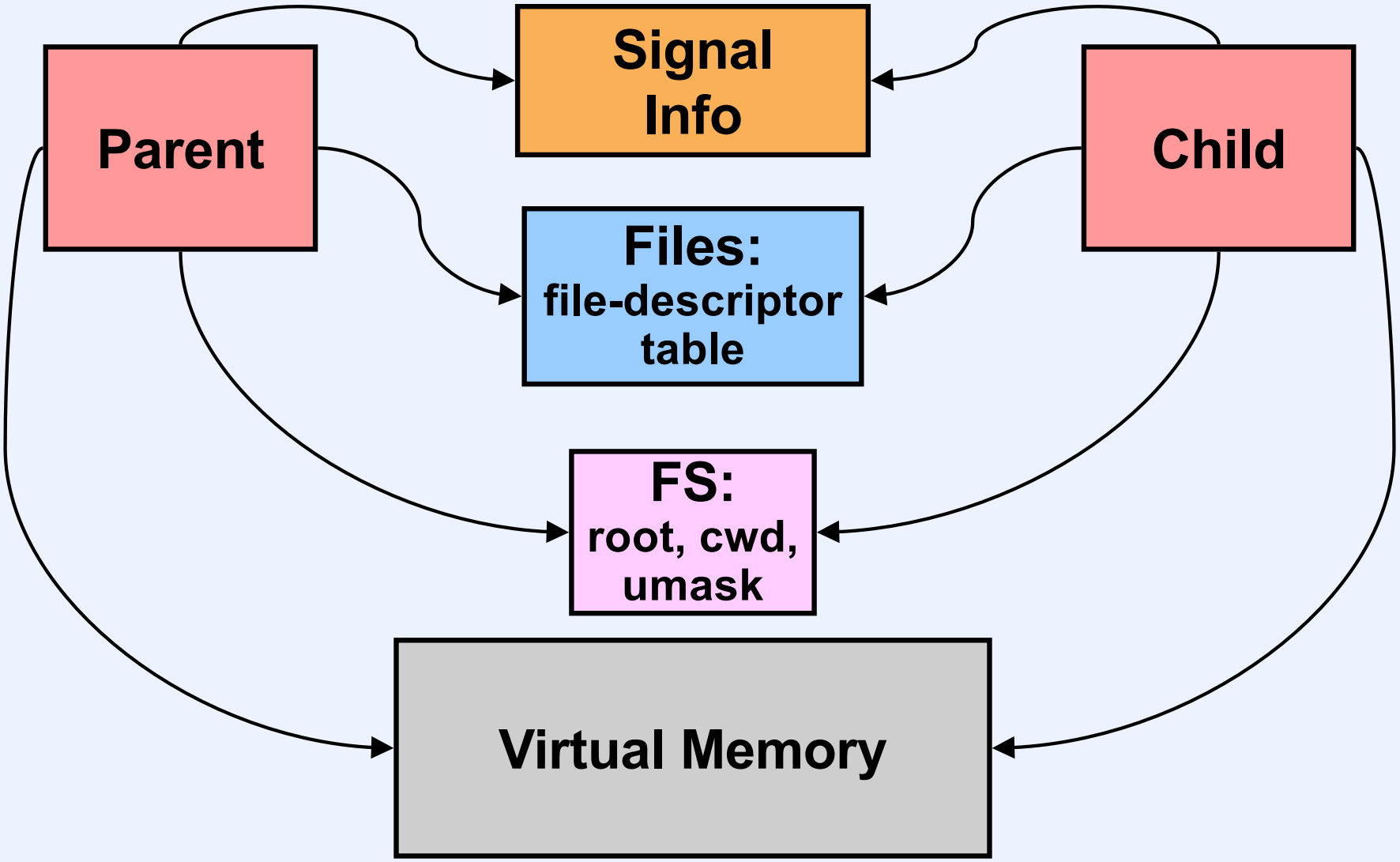


Variable-Weight Processes

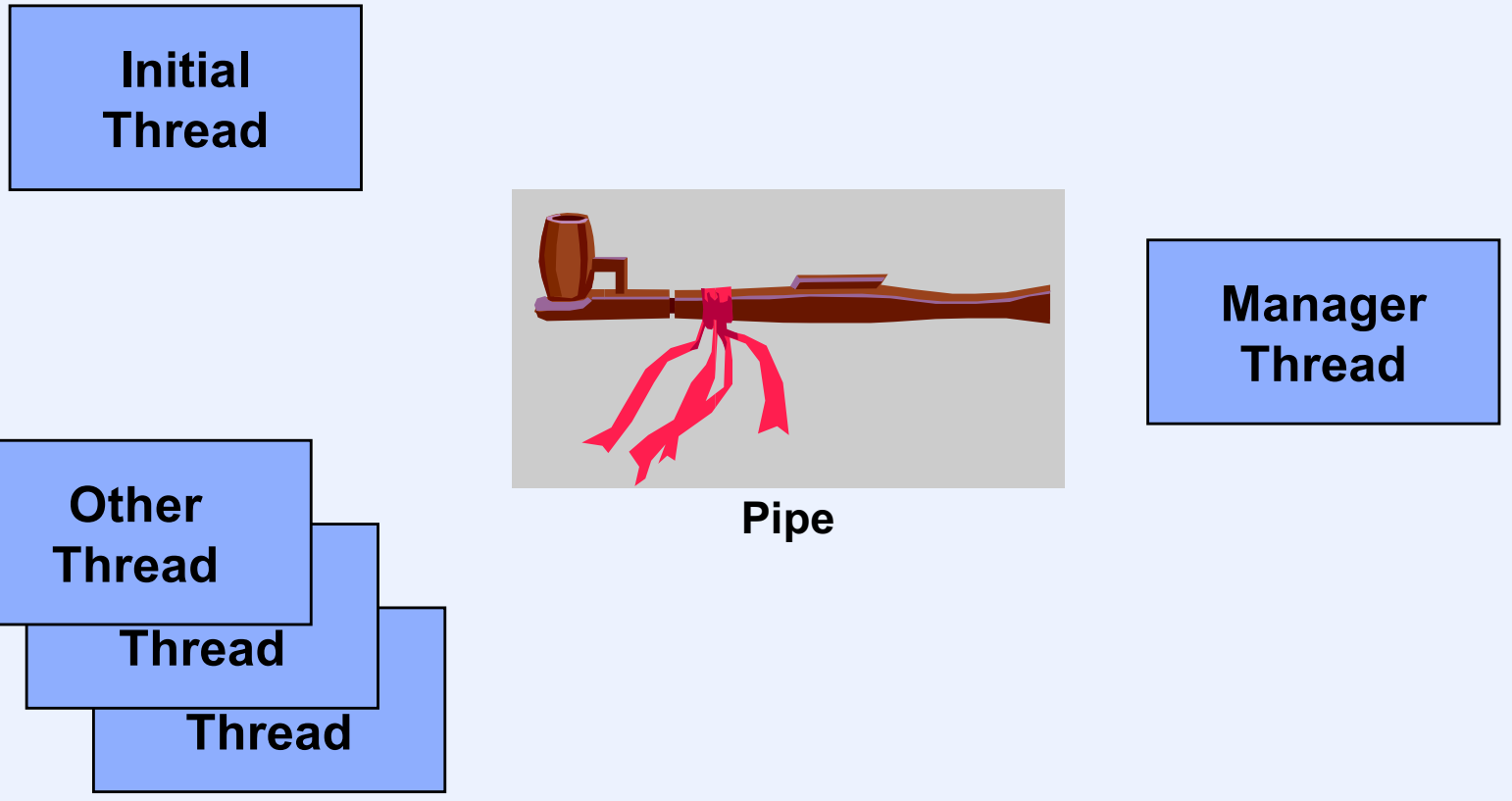
- Variant of one-level model
- Portions of parent process selectively *copied* into or *shared* with child process
- Children created using *clone* system call



Cloning



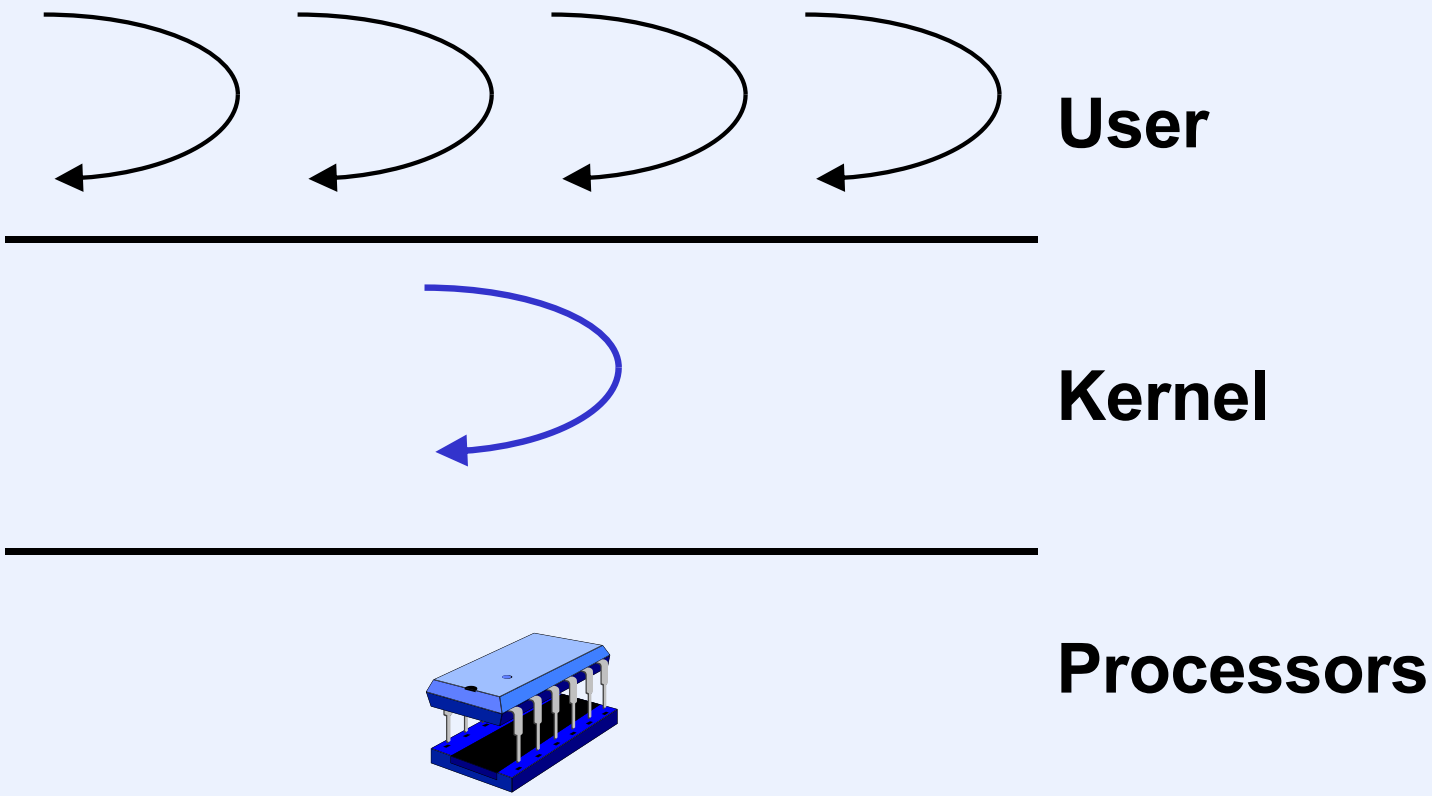
Linux Threads
(pre 2.6)



NPTL in Linux 2.6

- Native POSIX-Threads Library
  - full POSIX-threads semantics on improved variable-weight processes
    - threads of a "process" form a *thread group*
      - *getpid()* returns process ID of first thread in group
      - any thread in group can wait for any other to terminate
      - signals to process delivered by kernel to any thread in group

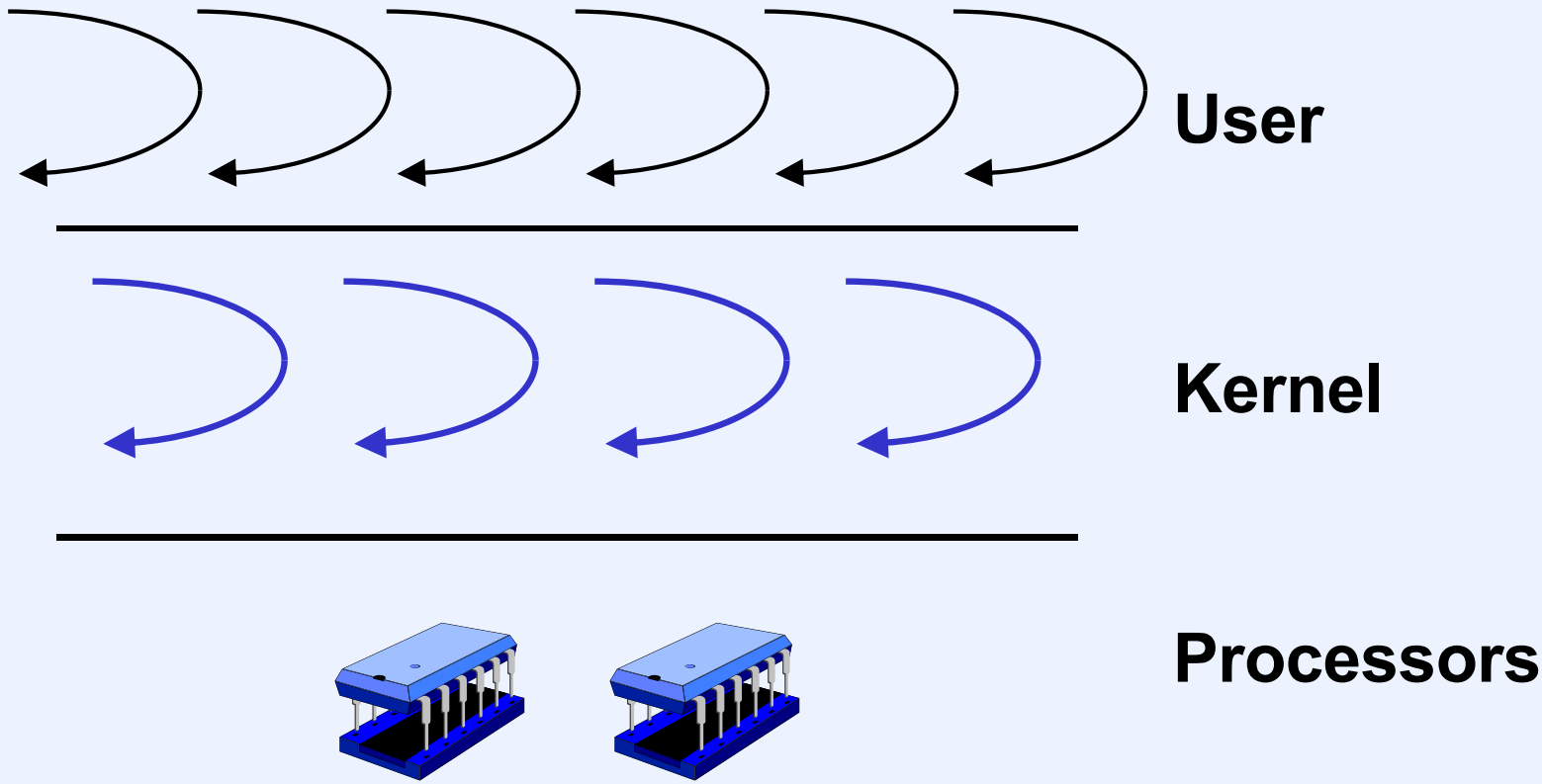
Two-Level Model
One Kernel Thread



Coping ...

```
ssize_t read(int fd, void *buf, size_t count) {
  ssize_t ret;
  while (1) {
    if ((ret = real_read(fd, buf, count)) == -1) {
      if (errno == EWOULDBLOCK) {
        sem_wait(&FileSemaphore[fd]);
        continue;
      }
    }
    break;
  }
  return(ret);
}
```

Two-Level Model:
Multiple Kernel Threads



Quiz

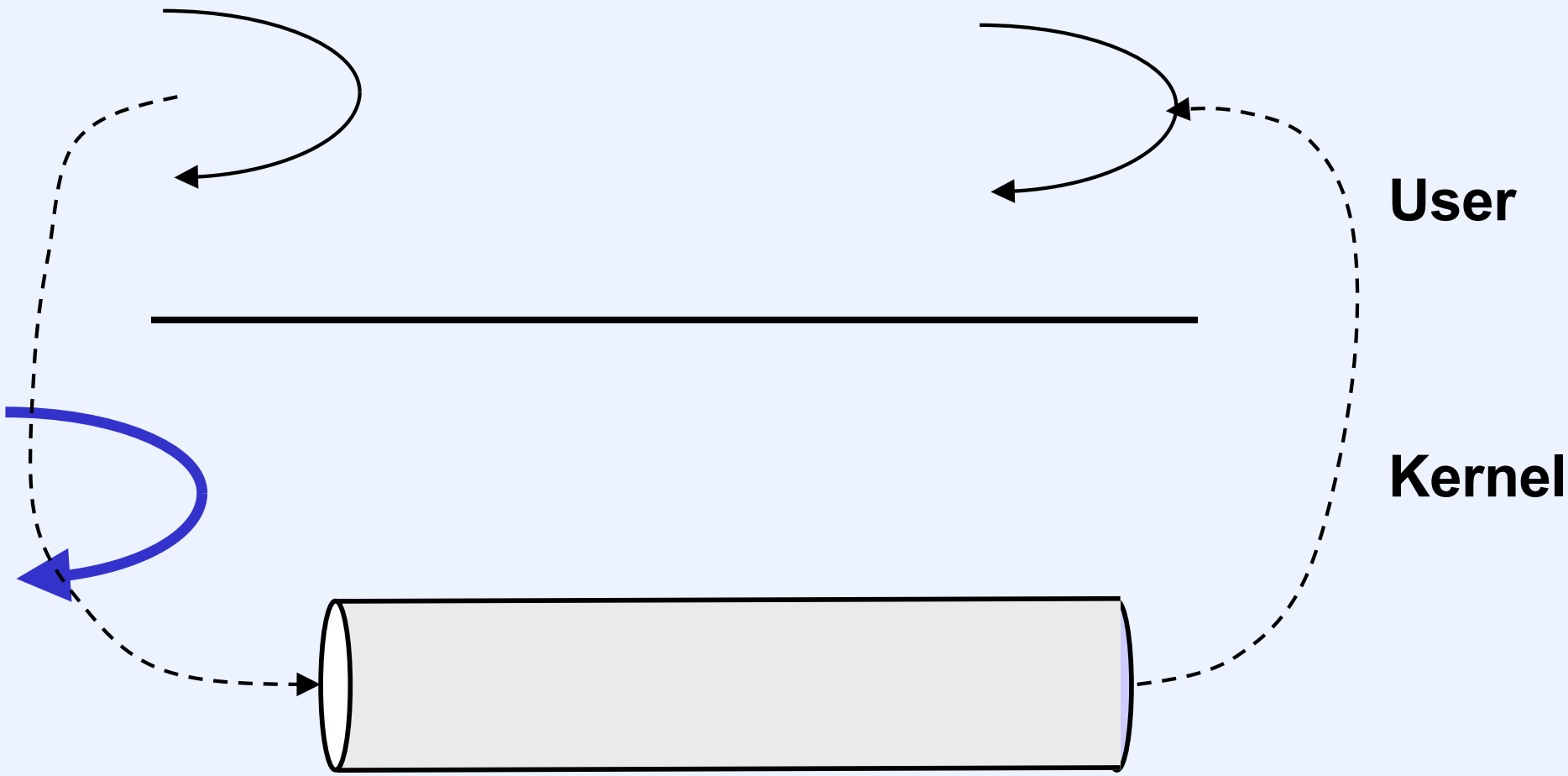
One kernel thread for each user thread is clearly a sufficient number of kernel threads in the two-level model. Is it necessary?

a) there must always be that number of kernel threads for the two-level model to work well.

b) there are situations in which that number is necessary, but they occur rarely.

c) there are no situations in which that number of threads is necessary, as long as there are at least as many kernel threads as processors.

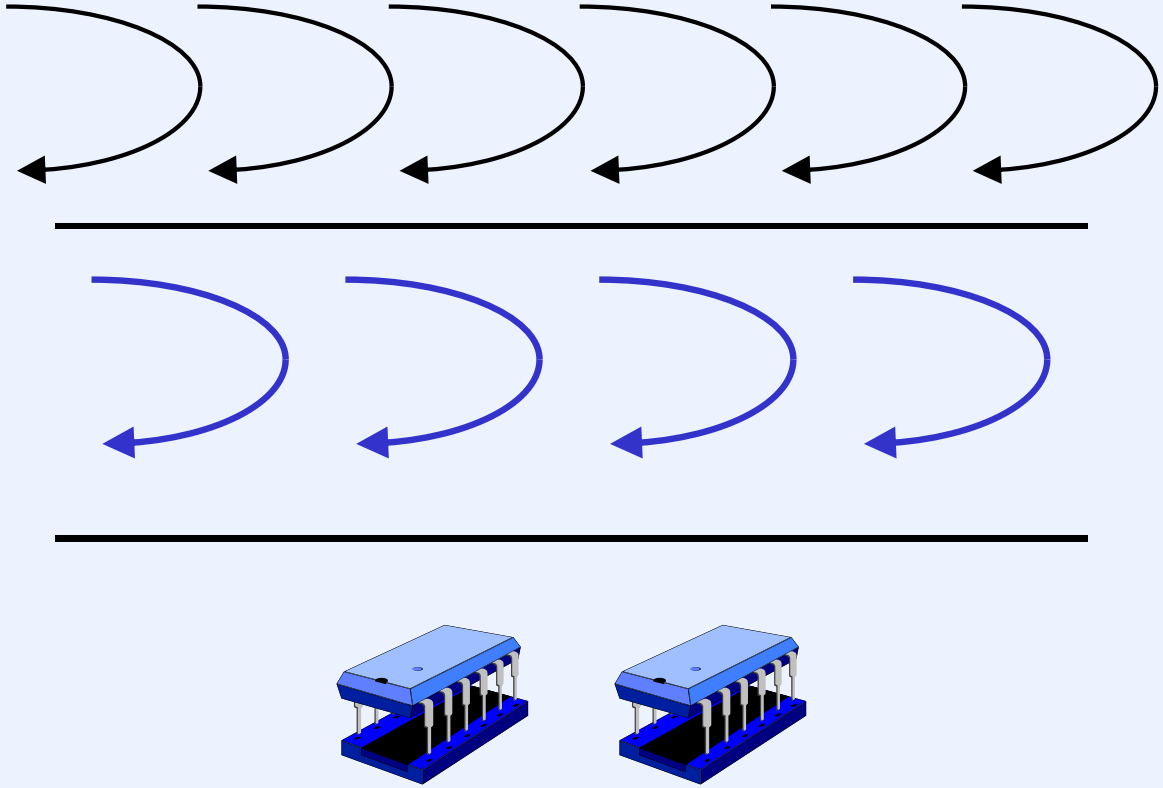
Deadlock



MThreads

- Two-level threads implementation of Uthreads
  - kernel-supported threads are POSIX threads
  - user threads based on your implementation of Uthreads
- Effectively a multiprocessor implementation
  - use POSIX mutexes rather than spin locks
  - use POSIX condition variables rather than the idle loop

Two-Level Model:
MThreads



Thread-Local Storage in Mthreads

- `__thread thread_t *ut_curthr;`
  - **reference to the current uthread**
- `__thread lwp_t *curlwp`
  - **reference to the current LWP (POSIX thread)**

- **Thread-Local Storage accesses are not async-signal safe!**
  - **handler for SIGVTALRM references TLS**
  - **must mask SIGVTALRM when using TLS**

Scheduler Activations