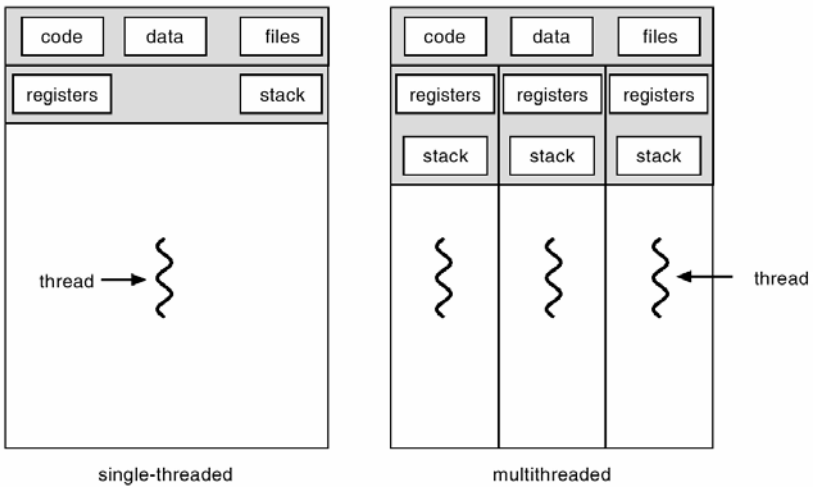
Threads

- Reading: OS Concepts pp.129-131
- A thread is a basic unit of CPU utilization
  - A Heavyweight Process is one with a single thread of control
  - Multi-threaded programs are used to
    - _____
    - _____
    - _____
    - _____

Single vs. Multi threaded

- PCB vs. Thread Control Block



1. Kernel Threads

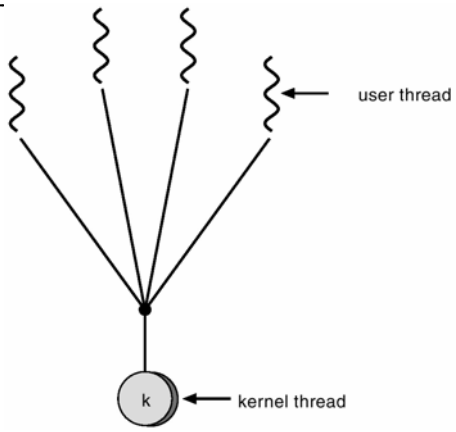
- Scheduled by the operating system
- Hence they are independent of each other
  - _____
  - _____
- _____
  _____
  _____
- Example OSes:
  - _____
  - _____

2. User threads

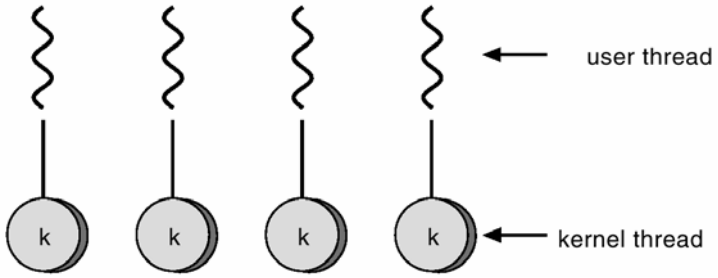
- Supported by a user library and hence scheduling is handled in user mode within the kernel thread.
- Hence they are dependent on each other
  - _____
  - _____
- _____
  _____
- Example:
  - _____
- Which to use?
  - _____
    _____

Many to one threads

- _____
  _____
  _____



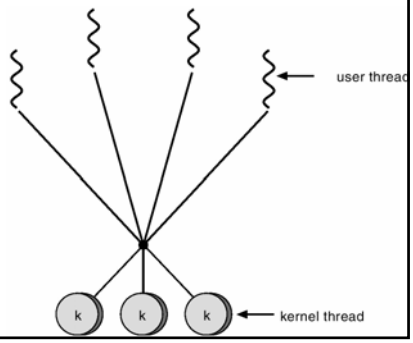
One to one threads



- _____
- Examples: _____

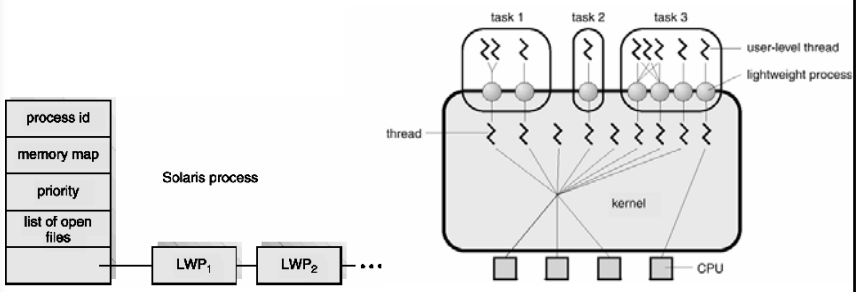
Many to many threads

- _____
  _____
- _____
  _____
- Example _____



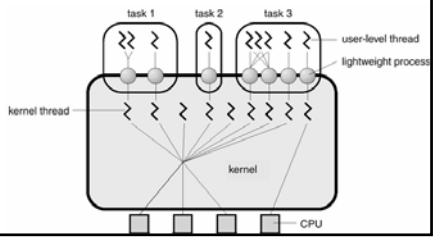
Solaris 2 threads

- Solaris 2 has a many to many threading model
- _____
  _____
- _____
  _____



Solaris 2 threads

- User threads can be
  - _____
  - _____
- Kernel threads
  - _____
- _____
  _____



Threads operations

- fork and exec
  - _____
    _____
  - _____
    _____
- Cancellation of threads can be
  - _____
    _____
  - _____
    _____

Threads operations

- Signal handling. Signal can be handled by
  - _____
  - _____
  - _____
  - _____
    _____
- Most thread libraries also allow threads to maintain data which is specific to the thread.

Pthreads

- Pthreads are a POSIX standard for threads
  - _____
- Some library types and methods.
  - pthread_t is _____
  - pthread_create() _____
  - pthread_join() _____
  - pthread_exit() _____
  - pthread_attr_t _____
  - pthread_attr_init() _____
  - pthread_attr_setscope() _____
  - pthread_attr_setschedparam() _____

Threads in UNIX

Resources/Code/Threads.C

Resources/Output/Threads.SampleOutput

```
#define NUM_THREADS 3
int wait_for_time=2;

void *sleep_test(void *parameters)
{
   int sleep_time = wait_for_time;
   wait_for_time *= 2;
   sleep(sleep_time);
   pthread_exit(0);
}

main(int argc, char *argv[])
{
   pthread_t tid[NUM_THREADS];
   for (int thr_no=0; (thr_no<NUM_THRS); thr_no++)
      pthread_create(&tid[thr_no],NULL,
                                    sleep_test,NULL);
   for (int thr_no=0; (thr_no<NUM_THRS); thr_no++)
      pthread_join(tid[thr_no],NULL);
}
```