- One user thread depends on one kernel thread.
- Example:

Two Level Model

- OS/Thread library supports both one to one and many to many model
- Example:

Threads in Linux - clone()

- In Linux, processes and threads both are referred as "tasks". For each task, a separate "task_struct" is created (instead of PCB or TCB).
- Process can be seen as a new task that doesn't share any section with parent task, but sends SIGCHLD signal to the parent upon its termination.

```
child_task_id = clone(task_fn, stack, SIGCHLD, NULL);
```

- Thread can be seen as a new task that shares parent's virtual address space, open files, fs and signal handler table.

```
child_task_id = clone(task_fn, stack, CLONE_VM|CLONE_FILES|CLONE_FS|CLONE_SIGHAND, NULL);
```

Assignments

1. Create a thread to sort given array of 10 integers using selection sort. Main thread should print the result after sorting is completed.
    - Hint: Pass array to thread function (via arg4 of pthread_create()).

```
void* thread_func(void *param) {
    int *arr = (int*)param;
    // ... code to sort the array
    return NULL;
}
```

2. Create a thread to sort given array of "n" integers using bubble sort. Main thread should print the result after sorting is completed.
    - Hint: `struct array { int *arr; int size; }`
    - Pass struct address to thread function (via arg4 of pthread_create()).
3. One thread prints "SUNBEAM" continuously, and other thread prints "INFOTECH" continhously. Only one should print at a time starting with "SUNBEAM". Hint: using semaphores.
4. Optional - Implement producer consumer across two processes using POSIX semaphores and Mutexes. Hints for communication across the processes.
    - Hint 1: Semaphore and Mutex must be in shared memory.
    - Hint 2: Mutex pshared attribute should be set to PTHREAD_PROCESS_SHARED.
    - Hint 3: Semaphore should be created with non-zero key (arg2 of sem_init()).
    - Hint 4: Use signal handlers to properly cleanup shared memory and synchronization objects.