Threads

Q. If it takes 23 days for a man to complete a job, how many days will it take 7 men to do the same job??

→ they will complete this in $\lceil 23/7 \rceil$ days because they're working parallely.

This concept is also applicable in computers. Generally parallelization in computers is done to improve performance of application.

Now a days this concept is extensively supported by hardwares like GPU.

A very important construct when doing parallelization is Threads.

Threads are execution entity, very much like process, but these are extensively light weight process

```
#include <stdio.h>

unsigned long addall(){
        int i=0;
        unsigned long sum=0;
        while(i < 1000000){
            sum+=i;
            i++;
        }
        return sum;
}
```

```
int main() {
    unsigned long sum;
    srandom(time(NULL));
    sum = addall();
    printf("%lu\n", sum);
}
```

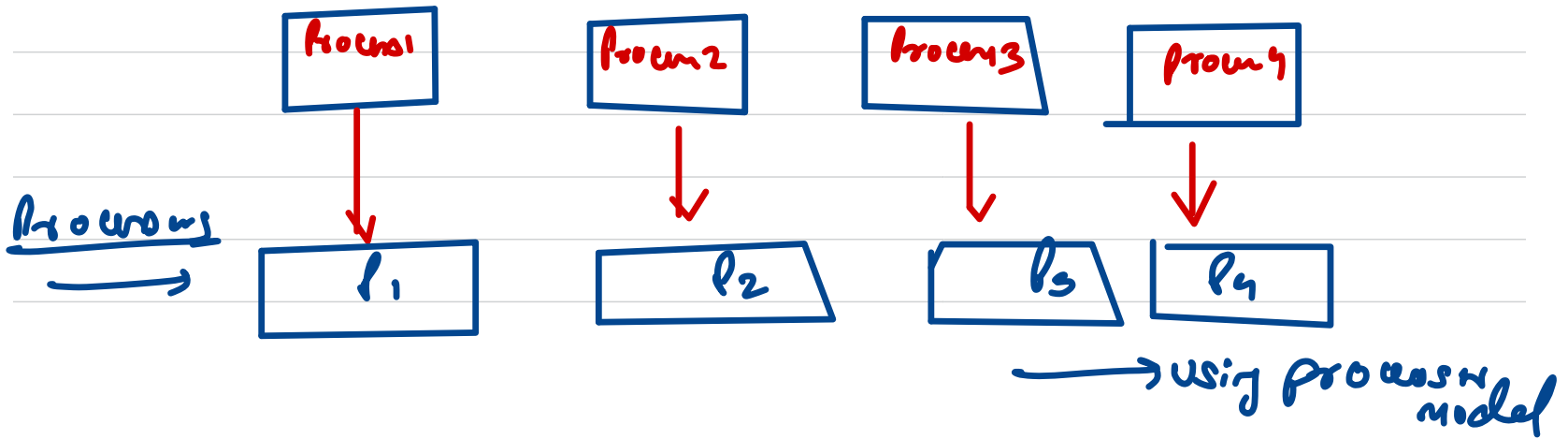
Let's consider what happens if the program is executed on a system with multiple processors.

The program starts execution on one of the processors rest remain idle. This leads to significant time consumption.

Our task was to add first 10^6 no.s.

$10^6/4 \rightarrow 25 \times 10^4$

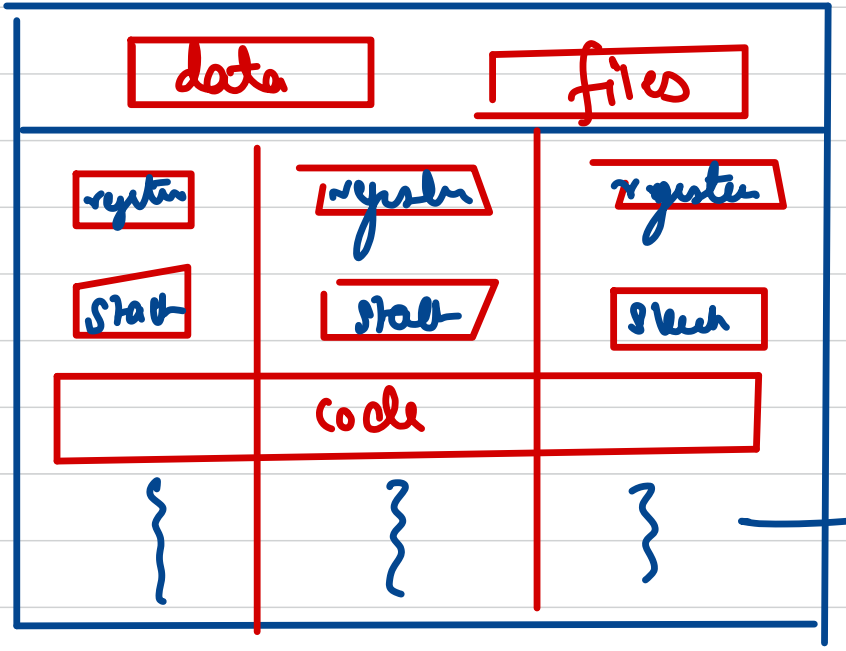
↳ Create 4 processes, each loop does 1/4th of my work.



Thread Mode

Create 1 process with 4 threads, each loop does 1/4th of the work. It is equivalent to having 4 seperate contents running on different processors.

Each thread has its own stack i.e own execution content.



- Separate stream of execution within a single process.
- Threads are not isolated from each other.

Each thread contains stack, registers etc.

Why threads ??

1. lightweight.
2. Efficient context Switch
3. allows comm. between entities.

→ A thread has no data segment & heap, but process has.

→ A process has atleast 1 thread, but a thread cannot live on it's own.

→ If process die, all threads die. But if a thread dies, it's stack is Reclaimed - as each thread has it's own stack.

How to make threads ??

pthread library

#create a thread

TID (thread identifier)

specifies properties for threads

```
int pthread_create (pthread_t * thread,
                    const pthread_attr *attr,
                    void * (* start_routine)(void *),
                    void * args);
```

pointer to a func which starts execution in a diff. thread.

args to f".

Destroying a thread.

```
void pthread_exit (void *retval);
```

Join → wait for a specific thread to complete.

```
int pthread_join (pthread_t thread, void **retval);
```

thread: TID of the thread to wait for

retval: exit status of thread

Task: Use this library to implement the sum fn using n threads.

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

unsigned long sum[4];

void *thread_fn(void *args) {
    long id = (long) args;
    int start = id * 25 * 10^4;
    int i=0;
    while(i < 250000) {
        sum[id] += (i + start);
        i++;
    }
    return NULL;
}
```

```
int main() {

    pthread_t t1, t2, t3, t4;
    pthread_create (&t1, NULL, thread_fn, (void*) 0);
    pthread_create (&t2, NULL, thread_fn, (void*) 2);
    pthread_create (&t3, NULL, thread_fn, (void*) 3);
    pthread_create (&t4, NULL, thread_fn, (void*) 4);
    pthread_join (t1, NULL);
    pthread_join(t2, NULL);
      :
    printf("%lu\n", sum[0]+sum[1]+sum[2]+sum[3]);
    return 0;
}
```

gcc threads.c -lpthread

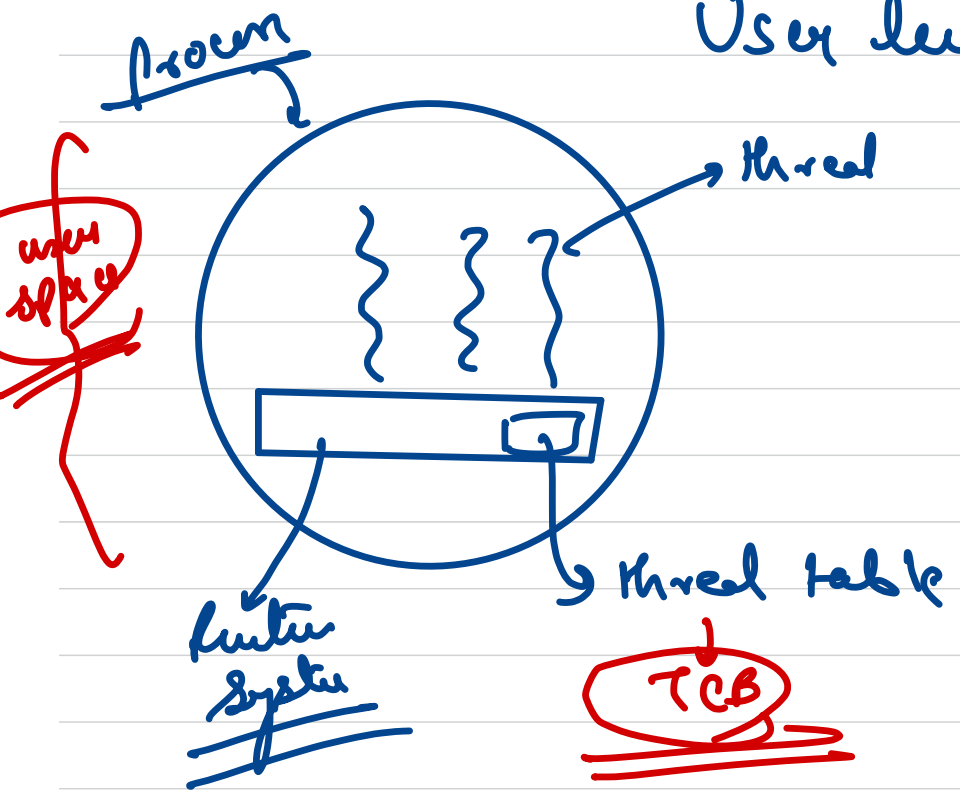
Who manages threads ??

→ Two strategies -

- User thread → thread management is done by a user level thread library. Kernel knows nothing about these threads.

- Kernel thread
  ↳ threads are directly supported by kernel
  → known as light wt. process

User level thread

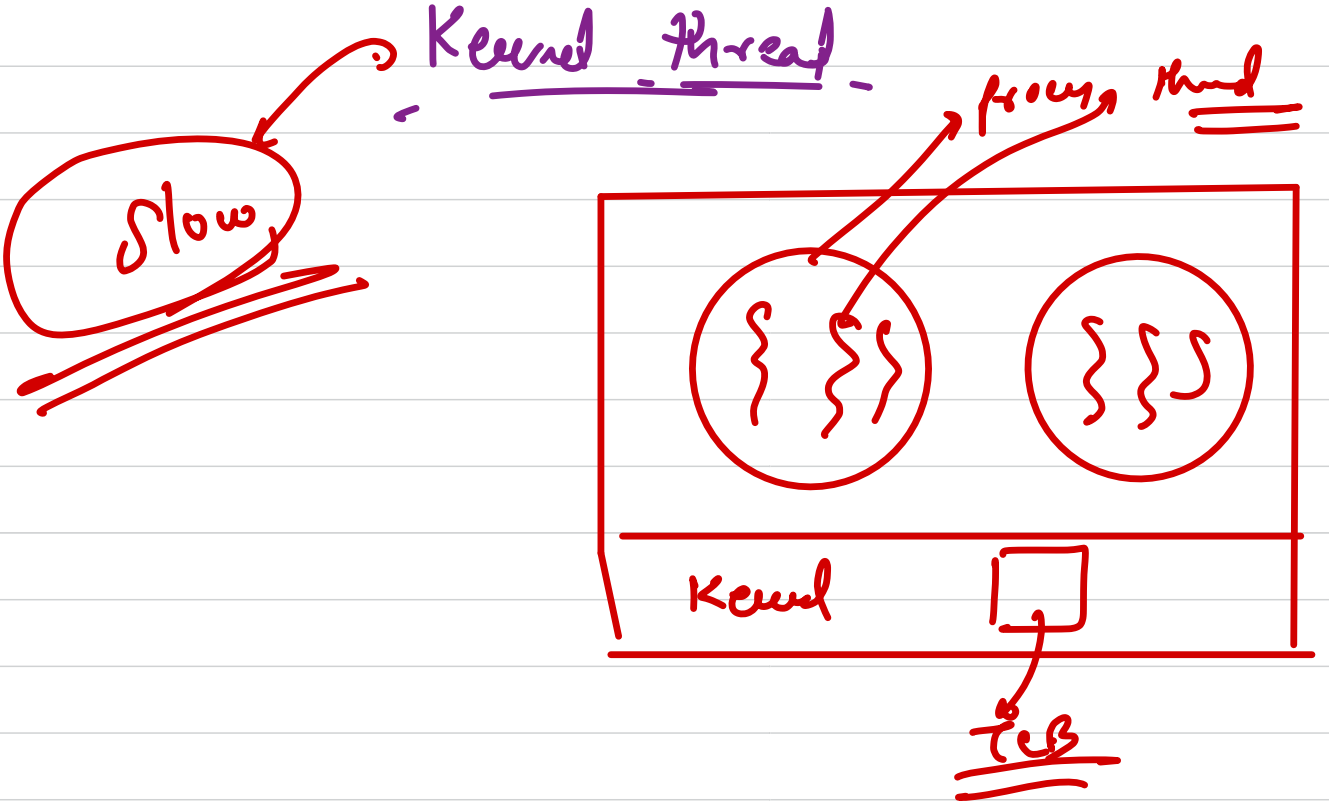


→ advantage -

- fast (no sys call is reqd to manage thread)
- Can be implemented in OS which don't support thread.
- Switching is fast, No Switching from user to protected mode.

→ disadvantage -

- Scheduling can be issue.
- Lack of coordinate in kernel & user thread.


Kernel Thread
Slow
Kernel
TCB

Issues

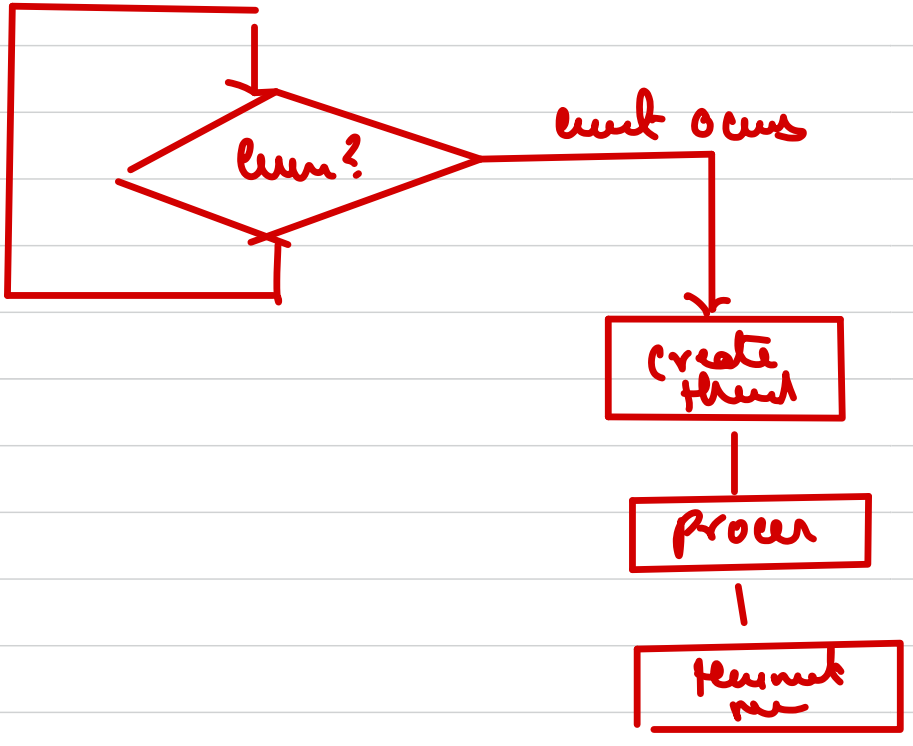
↳ what happens when a thread invokes a fork??

(1) duplicate all thread ×

(2) duplicate only calling thread ?!

↳ what OS should do for seg. fault in thread?!

Thread usage



Thread pools