C program forking Separate Process

```
# include <sys/types.h> # include <stdio.h> #include <unistd.h>
 int main ( )
 {
     pid_t pid;          → datastructure
     pid = fork(); /*fork another process*/

 if (pid<0) { /* error occurred*/
     fprintf(stderr, "fork failed");
     return 1; }
 else if (pid = =0) { /* child process*/
     execlp("/bin/ls", "ls" ,NULL);
  }
 else { /* parent process / parent will wait for the child*/
     wait(NULL);
     printf ("Child Complete");
  return 0;
 }
```

[illegible]

library entirely in user space

~~library~~ Kernel-level library supported by OS

<u>Pthreads</u>

Two general strategies for creating multiple threads.

1. Asynchronous threading
2. Synchronous threading

<u>Thread Program</u>

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

int sum;
void *runner(void *param);
```

```
int main(int argc, char *argv[])
{
        pthread_t tid;
        pthread_attr_t attr;

        if (argc != 2){
          printf(stderr, " usage: a.out <int value> \n");
          return 1;
        if (~~argv~~ atoi (arg[1] <0){
          fprint (stderr, "%d must be ≥0\n", atoi(argv[1]));
          return -1;
        }
  pthread_attr_init(&attr);
  pthread_create(&tid, &attr, runner, argv[1]);

  pthread_join(tid, NULL);
  printf("sum = %d \n", sum);

  void * runner(void * param)
  {
     int i, upper = atoi(param)
        sum=0;
     for(i=1; i<upper; i++)
          sum+=i;
     pthread_exit(0);
  }
```