```
#include <pthread.h>
pthread_t threadID;
pthread_create (&threadID, NULL, methodName, void *para);
int pthread_join(threadID, void **value_ptr);
void pthread_exit(void *value_ptr);
pthread_t pthread_self(void);
int pthread_equal(thread_t t1, pthread_t t2);

pthread_mutex_t lock; // lock is a mutex variable
pthread_cond_t   a_cond;
pthread_mutex_init( &lock, NULL );
pthread_mutex_lock( &lock ) ;
pthread_mutex_unlock( &lock );
pthread_cond_init (&a_cond, NULL );
pthread_cond_wait(&a_cond, &lock);
pthread_cond_signal(&a_cond);
pthread_cond_broadcast(&a_cond);
-------------------------------------------------------
#include <semaphore.h>
sem_t s; // declares a semaphore variable s
int sem_init(sem_t *s, int pshared, unsigned value);
sem_post(sem_t *s);   // signal,       release
sem_wait(sem_t *s);   // wait,         acquire
sem_trywait(sem_t *s);
-------------------------------------------------------
#include <signal.h>
int sigemptyset(sigset_t *set);
int sigfillset(sigset_t *set);
int sigaddset(sigset_t *set, int signo);
int sigdelset(sigset_t *set, int signo);
int sigismember(sigset_t *set, int signo);
int sigprocmask(int how, sigset_t *set, sigset_t *oset);
     // how: can be SIG_BLOCK, SIG_UNBLOCK, SIG_SETMASK
int sigaction(int signo, struct sigaction *act,
                         struct sigaction *oact);
struct sigaction {
  void (*sa_handler)(int); /* SIG_DFL, SIG_IGN or
                              pointer to function (no return value) */
  sigset_t sa_mask;        /* additional signals to be blocked */
  int sa_flags;            /* special flags and options */
  void (*sa_sigaction) (int, siginfo_t *, void *); /* real-time */
};
int pause(void);
int sigsuspend(const sigset_t *sigmask);
int sigwait(sigset_t *sigmask,   int *signo);
```