```
pthread_t;

pthread_attr_t;

int pthread_equal(pthread_t tid1, pthread_t tid2); Returns: nonzero if equal,
                                                              0 otherwise

pthread_t pthread_self(void);

int pthread_equal(pthread_t tid1 , pthread_t tid2);

int pthread_attr_init(pthread_attr_t *attr);

int pthread_create(pthread_t *restrict tid ,
                   const pthread_attr_t *restrict attr,
                   void * (*start_routine)(void *),
                   void *restrict arg); Returns: 0 if OK,
                                              error number on failure


int pthread_getconcurrency(void);

int pthread_attr_getguardsize(const pthread_attr_t *attr,
                              size_t *restrict guardsize);


int pthread_attr_getstacksize(const pthread_attr_t *restrict attr,
                          size_t *restrict stacksize);

int pthread_attr_getstack(const pthread_attr_t *restrict attr ,
                     void ** restrict stackaddr,
                     size_t stacksize);

int pthread_attr_getdetachstate(const pthread_attr_t *restrit attr,
                                int *detachstate);

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

```
int pthread_attr_setdetachstate(pthread_attr_t *attr,
                                int  detachstate);

int pthread_attr_setstack(const pthread_attr_t *attr,
                    void *stackaddr,
                    size_t stacksize);

int pthread_attr_setstacksize(pthread_attr_t *attr ,
                              size_t *stacksize);

int pthread_setconcurrency(level);

int pthread_attr_setguardsize(pthread_attr_t *attr,size_t guardsize);

int pthread_cancel(pthread_t tid);

void pthread_cleanup_push(void (*rtn)(void *) , void *arg);

void pthread_cleanup_pop(int execute);

int pthread_detach(pthread_t tid);

int pthread_attr_destroy(pthread_attr_t *attr);

void pthread_exit(void *rval_ptr);


fork            pthread_create

exit            pthread_exit

waitpid          pthread_join

atexit          pthread_cancel_push

getpid          pthread_self

abort           pthread_cancel
```

MUTEX:

```
pthread_mutex_t;

pthread_mutexattr_t

pthread_mutexattr_init(pthread_mutexattr_t *attr);

pthread_mutex_init(pthread_mutex_t * restrict mutex,
                   const pthread_mutexattr_t *restrict attr);

pthread_mutexattr_getpshared(pthread_mutexattr_t * restrict attr,
                             int *restrict pshared);

pthread_mutexattr_gettype(const pthread_mutexattr_t * restrict attr,
                          int *restrict type);

int pthread_mutex_lock(pthread_mutex_t *mutex);

int pthread_mutex_trylock(pthread_mutex_t *mutex);




int pthread_mutexattr_settype(pthread_mutexatr_t *attr, int type);

int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr,
                                 int pshared);

int pthread_mutex_destroy(pthread_mutex_t *mutex);
```

```
int pthread_mutexattr_destroy(pthead_mutexattr_t *attr);

int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

CONDITIONAL VARIABLES:

```
pthread_cond_t;

pthread_condattr_t;

int pthread_condattr_init(pthread_condattr_t *attr);

int pthread_cond_init(pthread_cond_t *restrict cond ,
                      pthread_condattr_t *restrict attr);

int pthread_condattr_getpshared(const pthread_condattr_t
                                * restrict attr,
                                int *restrict pshared);

int pthread_cond_wait(pthread_cond_t *restrict cond,
                      pthread_mutex_t *restrict mutex);

int pthread_cond_timedwait(pthread_cond_t *restrict cond,
                           pthread_mutex_t *restrict mutex,
                           const struct timspec *restrict timeout);

struct timespec
{
    time_t tv_sec;  //seconds
    long   tv_nsec; //nanoseconds
}


int pthread_cond_signal(pthread_cond_t *cond);
```

```
int pthread_cond_broadcast(pthread_cond_t *cond);

int pthread_condattr_setpshared(pthread_condattr_t *attr,
                                int pshared);


int pthread_cond_destroy(pthread_cond_t *cond);

int pthread_condattr_destroy(pthread_condattr_t *attr);
```