

## List of Functions

### 1. `pthread_create`

- `int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);`
- The `pthread_create()` function starts a new thread in the calling process. The new thread starts execution by invoking `start_routine()`; `arg` is passed as the sole argument of `start_routine()`.

### 2. `pthread_join`

- `int pthread_join(pthread_t thread, void **retval);`
- The `pthread_join()` function waits for the thread specified by `thread` to terminate. If that thread has already terminated, then `pthread_join()` returns immediately.

### 3. `pthread_exit`

- `void pthread_exit(void *retval);`
- The `pthread_exit()` function terminates the calling thread and returns a value via `retval` that (if the thread is joinable) is available to another thread in the same process that calls `pthread_join()`.

### 4. `pthread_self`

- `pthread_t pthread_self(void);`
- The `pthread_self()` function returns the ID of the calling thread. This is the same value that is returned in `*thread` in the `pthread_create(3)` call that created this thread.

### 5. `gettid`

- `pid_t gettid(void);`
- `gettid()` returns the caller's thread ID (TID). In a single-threaded process, the thread ID is equal to the process ID (PID, as returned by `getpid()`). In a multithreaded process, all threads have the same PID, but each one has a unique TID.

### 6. `atoi`

- `int atoi(const char *nptr);`
- The `atoi()` function converts the initial portion of the string pointed to by `nptr` to int.

### 7. `fopen`

- `FILE *fopen(const char *pathname, const char *mode);`
- The `fopen()` function opens the file whose name is the string pointed to by `pathname` and associates a stream with it.

### 8. `fseek`

- `int fseek(FILE *stream, long offset, int whence);`
- The `fseek()` function sets the file position indicator for the stream pointed to by `stream`. The new position, measured in bytes, is obtained by adding offset bytes to the position specified by `whence`. If `whence` is set to `SEEK_SET`, `SEEK_CUR`, or `SEEK_END`, the offset is relative to the start of the file, the current position indicator, or end-of-file, respectively.

## 9. fclose

- `int fclose(FILE *stream);`
- The `fclose()` function flushes the stream pointed to by `stream` (writing any buffered output data using `fflush()`) and closes the underlying file descriptor.

### Problem 0:

Write a C program that creates a thread that prints its POSIX thread ID (returned by `pthread_self()`) and Linux specific thread ID (returned by `gettid()`).

### Solution:

Note: Use the `-pthread` flag during compilation. Ex. `gcc thread1.c -o thread1 -pthread`

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>

void *PrintID(void *arg)
{
    printf("POSIX thread ID is: %ld\n", (long)pthread_self());
    printf("Linux thread ID is: %ld\n", (long)gettid());
    pthread_exit(NULL);
}

int main()
{
    pthread_t thread_id;
    printf("Creating thread... \n");
    int rc = pthread_create(&thread_id, NULL, PrintID, NULL);
    if (rc)
    {
        printf("Error:unable to create thread, %d\n", rc);
        exit(-1);
    }
    pthread_join(thread_id, NULL);
    return 0;
}
```