I am working with creates 1 to t threads with each thread pointing to the next and the last thread pointing to the first thread. This program allows each thread to sequentially take a turn until all threads have taken n turns. That is when the program ends. The only problem is in the tFunc function, I am busy waiting until it is a specific thread's turn. I want to know how to use semaphores in order to make all the threads go to sleep and waking up a thread only when it is its turn to execute to improve efficiency.

int turn = 1;

int counter = 0;

int t, n;

struct tData {

int me;

int next;

};

void \*tFunc(void \*arg) {

struct tData \*data;

data = (struct tData \*) arg;

for (int i = 0; i < n; i++) {

while (turn != data->me) {

}

counter++;

turn = data->next;

}

}

int main (int argc, char \*argv[]) {

t = atoi(argv[1]);

n = atoi(argv[2]);

struct tData td[t];

pthread\_t threads[t];

int rc;

for (int i = 1; i <= t; i++) {

if (i == t) {

td[i].me = i;

td[i].next = 1;

}

else {

td[i].me = i;

td[i].next = i + 1;

}

rc = pthread\_create(&threads[i], NULL, tFunc, (void \*)&td[i]);

if (rc) {

cout << "Error: Unable to create thread, " << rc << endl;

exit(-1);

}

}

for (int i = 1; i <= t; i++) {

pthread\_join(threads[i], NULL);

}

pthread\_exit(NULL);

}