Baboon Project Report.

Bradly Ross, Ziad Arafat, Luis Mendoza, Angel Camacho

Abstract:

Withing this project we will be taking on problem one. Where we will be conduncting an incremental aproch to solve the baboon problem such that FIFO order is preserved. This will be solved utilizing C Pthreads library.

Introduction:

Withng this project we will be working on baboon problem where baboons can only cross a canyon by swinging hand-over-hand on the rope. It is important to note the fact that in this problem "baboons" should not enter from oposite directions and meet in the middle other wise they will fall to their death, not to mention no more then three baboons may be on the rope other wise the rope rips and they fall to their death.

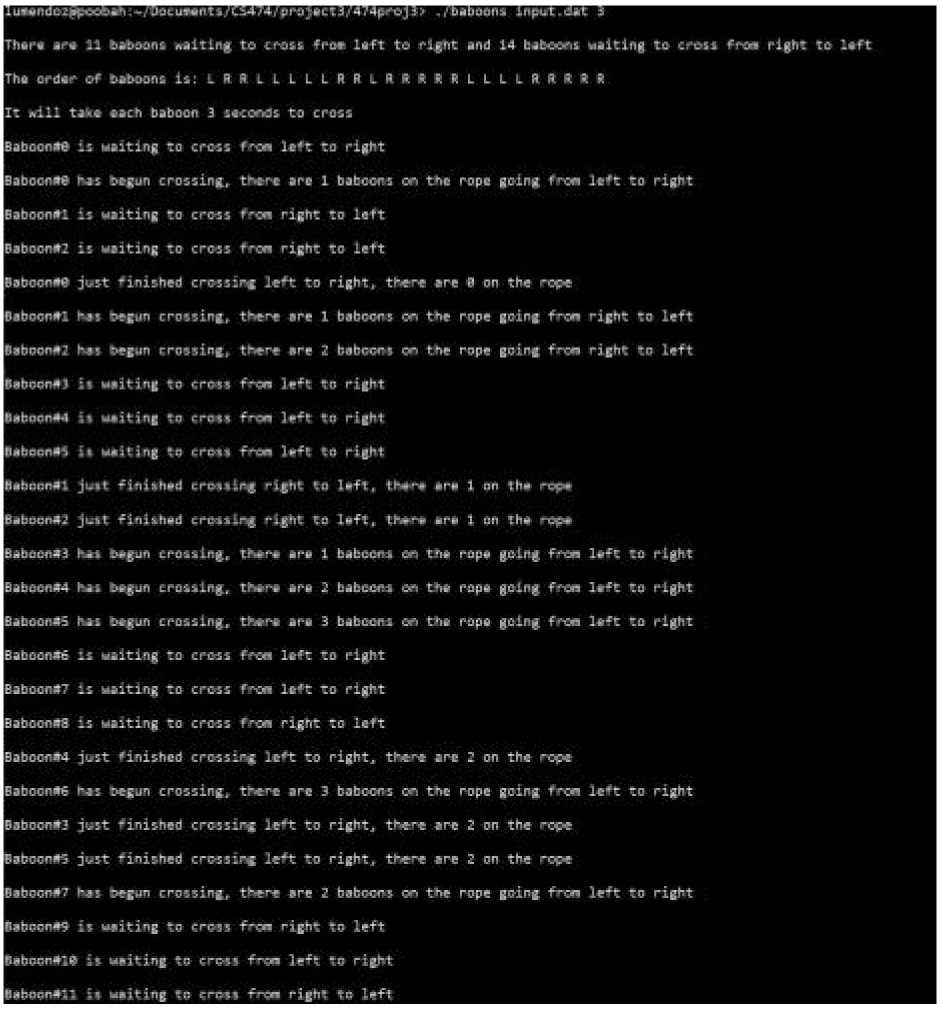
Methodology:

First we read the file wich includes the instructions that dictate weather the baboos will enter from the left or from the right. To represent each baboon we will be essentially utilizing a thread, the rope will be an array of one hundred indexes which queue the baboons directions. Further more we will also be using a time variable that will be utilized to non other then dictate for how long each thread/baboon runs for.

Results:

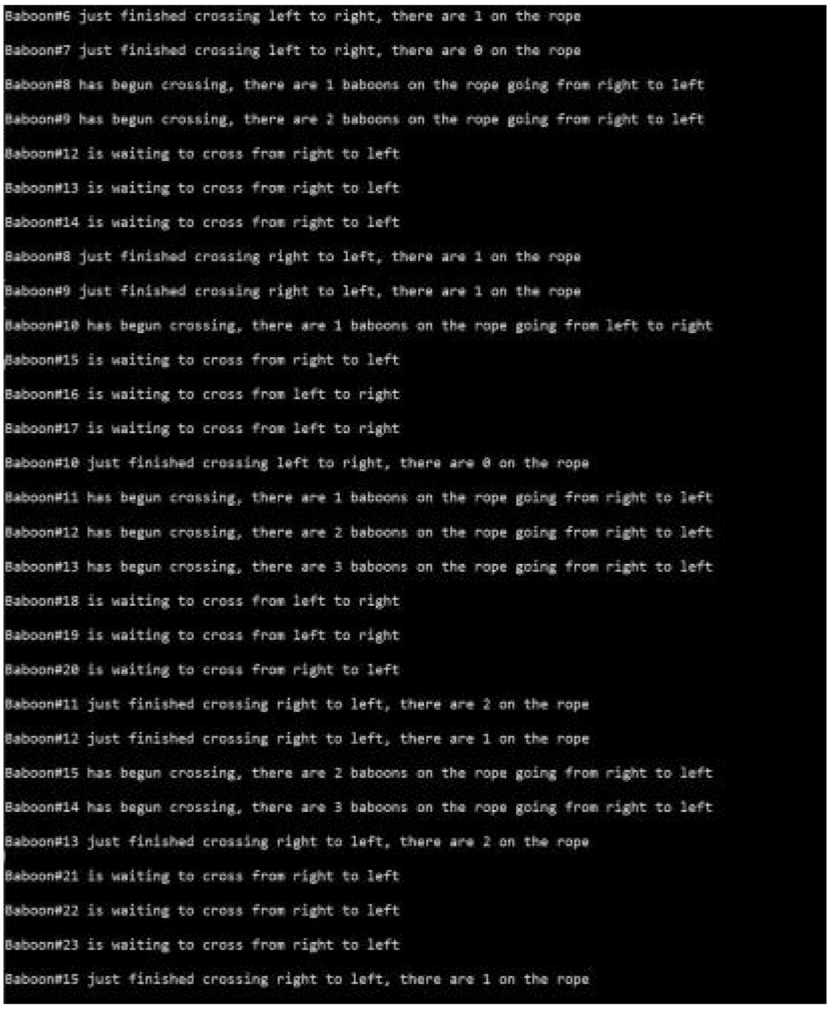
Results

Results



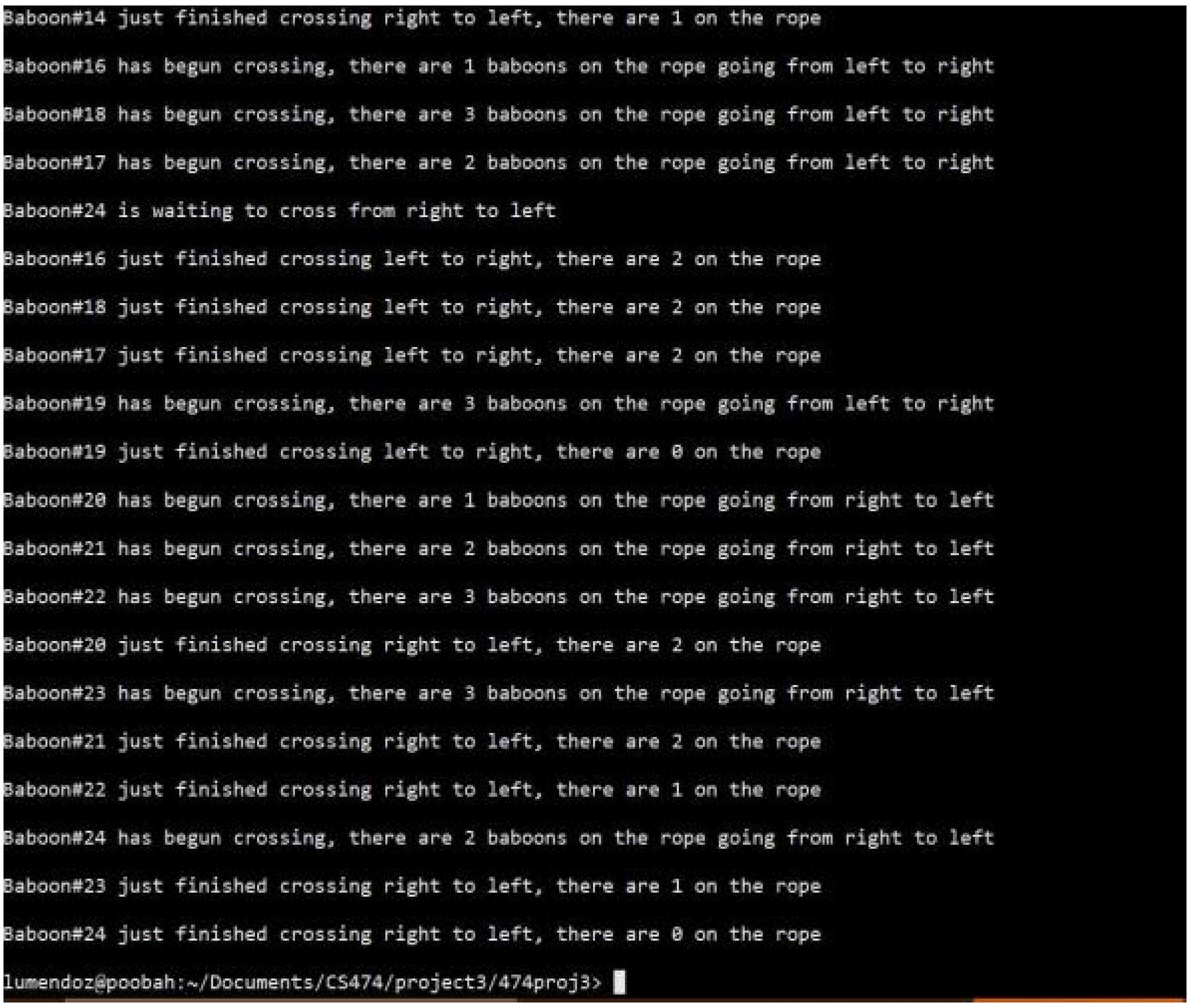
```
lumendoz@poobah:~/Documents/CS474/project3/474proj3> ./baboons input.dat 3
There are 11 baboons waiting to cross from left to right and 14 baboons waiting to cross from right to left
The order of baboons is: L R R L L L L L R R L R R R R R L L L L R R R R R
It will take each baboon 3 seconds to cross
Baboon#0 is waiting to cross from left to right
Baboon#0 has begun crossing, there are 1 baboons on the rope going from left to right
Baboon#1 is waiting to cross from right to left
Baboon#2 is waiting to cross from right to left
Baboon#0 just finished crossing left to right, there are 0 on the rope
Baboon#1 has begun crossing, there are 1 baboons on the rope going from right to left
Baboon#2 has begun crossing, there are 2 baboons on the rope going from right to left
Baboon#3 is waiting to cross from left to right
Baboon#4 is waiting to cross from left to right
Baboon#5 is waiting to cross from left to right
Baboon#1 just finished crossing right to left, there are 1 on the rope
Baboon#2 just finished crossing right to left, there are 1 on the rope
Baboon#3 has begun crossing, there are 1 baboons on the rope going from left to right
Baboon#4 has begun crossing, there are 2 baboons on the rope going from left to right
Baboon#5 has begun crossing, there are 3 baboons on the rope going from left to right
Baboon#6 is waiting to cross from left to right
Baboon#7 is waiting to cross from left to right
Baboon#8 is waiting to cross from right to left
Baboon#4 just finished crossing left to right, there are 2 on the rope
Baboon#6 has begun crossing, there are 3 baboons on the rope going from left to right
Baboon#3 just finished crossing left to right, there are 2 on the rope
Baboon#5 just finished crossing left to right, there are 2 on the rope
Baboon#7 has begun crossing, there are 2 baboons on the rope going from left to right
Baboon#9 is waiting to cross from right to left
Baboon#10 is waiting to cross from left to right
Baboon#11 is waiting to cross from right to left
```

Results



```
Baboon#6 just finished crossing left to right, there are 1 on the rope

Baboon#7 just finished crossing left to right, there are 0 on the rope

Baboon#8 has begun crossing, there are 1 baboons on the rope going from right to left

Baboon#9 has begun crossing, there are 2 baboons on the rope going from right to left

Baboon#12 is waiting to cross from right to left

Baboon#13 is waiting to cross from right to left

Baboon#14 is waiting to cross from right to left

Baboon#8 just finished crossing right to left, there are 1 on the rope

Baboon#9 just finished crossing right to left, there are 1 on the rope

Baboon#10 has begun crossing, there are 1 baboons on the rope going from left to right

Baboon#15 is waiting to cross from right to left

Baboon#16 is waiting to cross from left to right

Baboon#17 is waiting to cross from left to right

Baboon#10 just finished crossing left to right, there are 0 on the rope

Baboon#11 has begun crossing, there are 1 baboons on the rope going from right to left

Baboon#12 has begun crossing, there are 2 baboons on the rope going from right to left

Baboon#13 has begun crossing, there are 3 baboons on the rope going from right to left

Baboon#18 is waiting to cross from left to right

Baboon#19 is waiting to cross from left to right

Baboon#20 is waiting to cross from right to left

Baboon#11 just finished crossing right to left, there are 2 on the rope

Baboon#12 just finished crossing right to left, there are 1 on the rope

Baboon#15 has begun crossing, there are 2 baboons on the rope going from right to left

Baboon#14 has begun crossing, there are 3 baboons on the rope going from right to left

Baboon#13 just finished crossing right to left, there are 2 on the rope

Baboon#21 is waiting to cross from right to left

Baboon#22 is waiting to cross from right to left

Baboon#23 is waiting to cross from right to left

Baboon#15 just finished crossing right to left, there are 1 on the rope
```

Results



```
Baboon#14 just finished crossing right to left, there are 1 on the rope
Baboon#16 has begun crossing, there are 1 baboons on the rope going from left to right
Baboon#18 has begun crossing, there are 3 baboons on the rope going from left to right
Baboon#17 has begun crossing, there are 2 baboons on the rope going from left to right
Baboon#24 is waiting to cross from right to left
Baboon#16 just finished crossing left to right, there are 2 on the rope
Baboon#18 just finished crossing left to right, there are 2 on the rope
Baboon#17 just finished crossing left to right, there are 2 on the rope
Baboon#19 has begun crossing, there are 3 baboons on the rope going from left to right
Baboon#19 just finished crossing left to right, there are 0 on the rope
Baboon#20 has begun crossing, there are 1 baboons on the rope going from right to left
Baboon#21 has begun crossing, there are 2 baboons on the rope going from right to left
Baboon#22 has begun crossing, there are 3 baboons on the rope going from right to left
Baboon#20 just finished crossing right to left, there are 2 on the rope
Baboon#23 has begun crossing, there are 3 baboons on the rope going from right to left
Baboon#21 just finished crossing right to left, there are 2 on the rope
Baboon#22 just finished crossing right to left, there are 1 on the rope
Baboon#24 has begun crossing, there are 2 baboons on the rope going from right to left
Baboon#23 just finished crossing right to left, there are 1 on the rope
Baboon#24 just finished crossing right to left, there are 0 on the rope
lumendoz@poobah:~/Documents/CS474/project3/474proj3>
```

Code

```
#include <fcntl.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

////Create semaphores////
sem_t rope; // The rope sempahore blocks baboons from entering the rope from both
sides
sem_t right_to_left_mutex; // The right_to_left_mutex ensures that the queue
        // stays in order while crossing right to left
sem_t left_to_right_mutex; // The left_to_right_mutex ensures that the queue
        // stays in order while crossing left to right
sem_t mutex; // The mutex semaphore prevents one side from holding the rope
indefinently
sem_t capacity; // The capacity semaphore prevents more than 3 baboons from being
```

```
on the rope at once

int left = 0; //The left and right variables are used to check if the rope is free
int right = 0; // to control if the rope semaphore needs to be posted or wait
int cross_time; // Time to wait while baboons cross

//The function to run when a baboon is crossing from the left side to the right
side
void *left_to_right(void *baboonnum ) {
        //Wait until mutex and left_to_right_mutex are available
        sem_wait( &mutex );
        sem_wait( &left_to_right_mutex );
        //Increment the value in left
        left++;
        //If the value in left is 1, that means there are no
        // baboons going from left_to_right currently, so the function
        // must check that the rope is not being used by the other side
        if ( left == 1 ) {
                sem_wait( &rope );
        }
        // Create variables to hold the baboon number and
        // how many baboons are currently on the rope
        int *num = (int*) baboonnum;
        int numonrope;
        //Post to the mutex semaphores to allow another thread to start
        // however because the rope semaphore was not posted, the other side
        // cannot start until this side finishes
        sem_post( &left_to_right_mutex );
        sem_post( &mutex );
        //Check that capacity is not empty, which keeps the rope
        // from having more than 3 baboons on at once
        sem_wait( &capacity );
        sem_getvalue( &capacity, &numonrope );
        //The print statement simulates the crossing beginning
        printf( "Baboon#%d has begun crossing, there are %d baboons on the rope
going from left to right\n\n",
                *num, 3 - numonrope );
        //Sleep for the given amount time provided by the user
        sleep( cross_time );
        sem_getvalue( &capacity, &numonrope );
        //Print once the baboon has finished crossing
        printf( "Baboon#%d just finished crossing left to right, there are %d on
the "
                "rope\n\n",
                *num, 2 - numonrope);
        //Post back to the capacity once a baboon leaves the rope
        sem_post( &capacity );
        //Decrement the value in left
        sem_wait( &left_to_right_mutex );
        left--;
        //If left is 0, it means there are no baboons currently crossing from
left_to_right
        // and that the rope can be incremented to allow the other side to start
crossing
```

```
        if ( left == 0 )
                sem_post( &rope );
        sem_post( &left_to_right_mutex );
        pthread_exit( NULL );
}
//This function is identical to the left_to_right function except
// going the other direction
void *right_to_left( void *baboonnum ) {

        sem_wait( &mutex );
        sem_wait( &right_to_left_mutex );
        right++;
        if ( right == 1 ) {
                sem_wait( &rope );
        }
        int *num = (int*) baboonnum;
        int numonrope;
        sem_post( &right_to_left_mutex );
        sem_post( &mutex );
        sem_wait( &capacity );
        sem_getvalue( &capacity, &numonrope );
        printf( "Baboon#%d has begun crossing, there are %d baboons on the rope
going from right to left\n\n",
                *num, 3 - numonrope );
        sleep( cross_time );
        sem_getvalue( &capacity, &numonrope );
        printf( "Baboon#%d just finished crossing right to left, there are %d on
the "
                "rope\n\n",
                *num, 2 - numonrope);
        sem_post( &capacity );
        sem_wait( &right_to_left_mutex );
        right--;
        if ( right == 0 )
                sem_post( &rope );
        sem_post( &right_to_left_mutex );
        pthread_exit( NULL );
}

int main( int argc, char **argv ) {
        FILE *file; // File to read from
        int *timetocross = ( int * )malloc(
                sizeof( int ) ); // Used to set time taken for baboons to cross
        char baboons[100]; // Used to hold queue of baboons
        int babooncount = 0;
        int lcount = 0; // Used to determine how many baboons are going in each
                // direction in the queue
        int rcount = 0;
        char direction; //Used to hold the value of the character being read from
the file
        // Check that arguments are provided
        if ( argc < 3 ) {
                printf( "Please enter a file name followed by an integer 1-10
which "
```

```
                        "represents the time taken to cross the rope\n" );
                free( timetocross );
                return -1;
        }
        file = fopen( argv[1], "r" ); // Open file to read order of baboons from
        int gettime = atoi( argv[2] );
        cross_time = gettime;

        // Fill the queue of baboons based on the order provided from the file
        while ( ( fscanf( file, "%c", &direction ) != EOF ) ) {
                if ( direction == ',' )
                        continue;
                if ( direction == 'L' ) {
                        lcount = lcount + 1;
                        baboons[babooncount] = 'L';
                        babooncount = babooncount + 1;
                }
                if ( direction == 'R' ) {
                        rcount = rcount + 1;
                        baboons[babooncount] = 'R';
                        babooncount = babooncount + 1;
                }
        }
        // Print how many baboons are on each side to cross
        printf("\nThere are %d baboons waiting to cross from left to right"
           " and %d baboons waiting to cross from right to left\n\n",lcount,
rcount);
        printf("The order of baboons is: ");
        for(int i = 0; i < babooncount; i++){
            printf("%c ", baboons[i]);
        }
        printf("\n\nIt will take each baboon %d seconds to cross\n\n",
cross_time);
        // Initialize correct number of threads for each side
        pthread_t l_to_r_thread[lcount];
        int left_thread_count = 0;
        pthread_t r_to_l_thread[rcount];
        int right_thread_count = 0;
        // Initialize semaphores
        sem_init( &rope, 0, 1 );
        sem_init( &right_to_left_mutex, 0, 1 );
        sem_init( &left_to_right_mutex, 0, 1 );
        sem_init( &mutex, 0, 1 );
        sem_init( &capacity, 0, 3 );
        // Create threads based on order of queue
        for ( int i = 0; i < babooncount; i++ ) {
                int *tharg = (int*)malloc(sizeof(*tharg));
                *tharg = i;
                sleep(1);
                if ( baboons[i] == 'L' ) {
                        // printf("Left baboon\n");
                        printf( "Baboon#%d is waiting to cross from left to
right\n\n", i );
                        pthread_create( &l_to_r_thread[left_thread_count], NULL,
                                        left_to_right, tharg);
                        left_thread_count++;
                } else if ( baboons[i] == 'R' ) {
                        printf( "Baboon#%d is waiting to cross from right to
left\n\n", i );
                        pthread_create( &r_to_l_thread[right_thread_count], NULL,
                                        right_to_left, tharg);
                        right_thread_count++;
                }
        }
        // Wait for threads to finish
        for ( int j = 0; j < left_thread_count; j++ ) {
                pthread_join( l_to_r_thread[j], NULL );
        }
        for ( int j = 0; j < right_thread_count; j++ ) {
                pthread_join( r_to_l_thread[j], NULL );
        }
        free( timetocross );
        // Destroy semaphores
        sem_destroy( &rope );
        sem_destroy( &right_to_left_mutex );
        sem_destroy( &left_to_right_mutex );
        sem_destroy( &mutex );
        sem_destroy( &capacity );
        return 0;
}
```

Conclusion:

This assignment is the perfect analogy of how threads are able to be synchronized with the use of semaphores. By using threads to represent each baboon, we showed how resources can be shared among different threads running in parallel while maintaining order and synchronization.