```c
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <string.h>
#include <fcntl.h>
#include <signal.h>
#include <stdlib.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/syscall.h>
#include <stdbool.h>

#define ROCK 1
#define PAPER 2
#define SCISSORS 3

#define SIGREQ 98
#define SIGREADY 99

void throw_one();
void throw_two();
void determine_round_winner();
void determine_winner();

//pointer to player one shared memory object
int* p_one_ptr;
//pointer to player two shared memory object
int* p_two_ptr;
//strings that represent rock, paper, scissors in array
const char *rps[3];
//scores for the game
int player_one_score;
int player_two_score;
//thread identifier for each thread created
pthread_t tid[2];
//array that stores rolls from threads
int choice[2];
//array that stores each thread state
int command[2];
//array for each threads mutex
pthread_mutex_t lock[2];
//array for each threads signal/wait
pthread_cond_t cond[2];
int num_rounds;
int player_one_thread_id;
int player_two_thread_id;

int main(int argc, char** argv) {
  num_rounds = atoi(argv[1]);

  int parent_id = getpid();

  pthread_mutex_init(&lock[0], NULL);
  pthread_mutex_init(&lock[1], NULL);

  rps[0] = "Rock";
  rps[1] = "Paper";
  rps[2] = "Scissors";

  player_one_score = 0;
```

```
  player_two_score = 0;

  pthread_create(&tid[0], NULL, throw_one, NULL);
  pthread_create(&tid[1], NULL, throw_two, NULL);

  if(getpid() == parent_id) {
    printf("Child 1 tid: %d\n", player_one_thread_id);
    printf("Child 2 tid: %d\n", player_two_thread_id);
    printf("Beginning %d Rounds...\n", num_rounds);
    printf("Fight\n");
    printf("---------------------------------------\n");

    for(int i=0; i<num_rounds; i++) {
      printf("Round %d:\n", i+1);
      pthread_mutex_lock(&lock[0]);
      pthread_mutex_lock(&lock[1]);
      command[0] = SIGREQ;
      command[1] = SIGREQ;
      pthread_cond_signal(&cond[0]);
      pthread_cond_signal(&cond[1]);
      while(command[0] == SIGREQ || command[1] == SIGREQ){
      }
      pthread_mutex_unlock(&lock[0]);
      pthread_mutex_unlock(&lock[1]);
      determine_round_winner();
      printf("---------------------------------------\n");
    }
    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);
    determine_winner();
  }
  return 0;
}

void throw_one(int signum) {
  player_one_thread_id = syscall(SYS_gettid);
  for(int i=0; i<num_rounds; i++) {
    while(command[0] != SIGREQ) {
    }
    srand(time(NULL) * syscall(SYS_gettid) * (i + 100));
    choice[0] = rand() % 3 + 1;
    command[0] = SIGREADY;
  }
}

void throw_two(int signum) {
  player_two_thread_id = syscall(SYS_gettid);
  for(int i=0; i<num_rounds; i++) {
    while(command[1] != SIGREQ) {
    }
      srand(time(NULL) * syscall(SYS_gettid) * (i + 100));
      choice[1] = rand() % 3 + 1;
      command[1] = SIGREADY;
  }
}

void determine_round_winner() {
  int player_one_choice = choice[0];
  int player_two_choice = choice[1];
  bool player_one_win = false;
  bool player_two_win = false;
```

```
  if(player_one_choice != player_two_choice) {
    if(player_one_choice == ROCK && player_two_choice == PAPER) {
      player_two_win = true;
    } else if(player_one_choice == ROCK && player_two_choice == SCISSORS) {
      player_one_win = true;
    } else if(player_one_choice == PAPER && player_two_choice == ROCK) {
      player_one_win = true;
    } else if(player_one_choice == PAPER && player_two_choice == SCISSORS) {
      player_two_win = true;
    } else if(player_one_choice == SCISSORS && player_two_choice == ROCK) {
      player_two_win = true;
    } else {
      player_one_win = true;
    }
  }

  printf("Child 1 throws %s!\n", rps[player_one_choice - 1]);
  printf("Child 2 throws %s!\n", rps[player_two_choice - 1]);
  if(player_one_win) {
    printf("Child 1 Wins!\n");
    player_one_score++;
  } else if(player_two_win) {
    printf("Child 2 Wins!\n");
    player_two_score++;
  } else {
    printf("Draw!\n");
  }
}

void determine_winner() {
  printf("---------------------------------------\n");
  printf("Results:\n");
  printf("Child 1: %d\n", player_one_score);
  printf("Child 2: %d\n", player_two_score);

  if(player_one_score > player_two_score) {
    printf("Child 1 Wins!\n");
  } else if(player_two_score > player_one_score) {
    printf("Child 2 Wins!\n");
  } else {
    printf("Draw!\n");
  }
}
```