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

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

void throw();
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;

int main(int argc, char** argv) {
  rps[0] = "Rock";
  rps[1] = "Paper";
  rps[2] = "Scissors";

  player_one_score = 0;
  player_two_score = 0;

  signal(SIGUSR1, throw);

  const int SIZE = sizeof(int);
  const char* playerOne = "PlayerOne";
  const char* playerTwo = "PlayerTwo";

  // Player One shared memory file descriptor
  int p_one_shm_fd;
  //create player one shared memory object
  p_one_shm_fd = shm_open(playerOne, O_CREAT | O_RDWR, 0666);
  //configure the size of the shared memory object
  ftruncate(p_one_shm_fd, SIZE);
  //memory map the shared memory object
  p_one_ptr = mmap(0, SIZE, PROT_WRITE, MAP_SHARED, p_one_shm_fd, 0);

  // Player Two shared memory file descriptor
  int p_two_shm_fd;
  //create player two shared memory object
  p_two_shm_fd = shm_open(playerTwo, O_CREAT | O_RDWR, 0666);
  //configure the size of the shared memory object
  ftruncate(p_two_shm_fd, SIZE);
  //memory map the shared memory object
  p_two_ptr = mmap(0, SIZE, PROT_WRITE, MAP_SHARED, p_two_shm_fd, 0);
```

```
  int num_rounds = atoi(argv[1]);

  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);
    kill(getpid(), SIGUSR1);
    wait(NULL);
    determine_round_winner();
    printf("---------------------------------------\n");
  }
  determine_winner();
  return 0;
}

void throw() {
    pid_t p_one_id = fork();
    if(!p_one_id) {
      srand(time(NULL) * getpid());
      int choice = rand() % 3 + 1;
      *p_one_ptr = choice;
      exit(0);
    } else {
      wait(NULL);
    }

    pid_t p_two_id = fork();
    if(!p_two_id) {
      srand(time(NULL) * getpid());
      int choice = rand() % 3 + 1;
      *p_two_ptr = choice;
      exit(0);
    } else {
      wait(NULL);
    }
  }

void determine_round_winner() {
  int player_one_choice = *p_one_ptr;
  int player_two_choice = *p_two_ptr;
  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");
  }
}
```