Program 6: Process Synchronization Using File Locks

Code:

```
// Score reader and updater (Shows runs and wickets)
// Reads and writes score into a file

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<fcntl.h>
#include<unistd.h>

int runs, wickets;

void exit_if(int result, const char *msg) {
      if (result == -1) {
            perror(msg);
            exit(EXIT_FAILURE);
      }
}

void update_score(int fd, int runs, int wickets) {
      int rp = 0, wp = 0;
      ssize_t nrd;
      char buf[32];
      struct flock writelk, compet;
      writelk.l_type = F_WRLCK;
      writelk.l_whence = SEEK_SET;
      writelk.l_start = 0;
      writelk.l_len = 0;

      if(fcntl(fd, F_SETLK, &writelk) == -1) {
            fcntl(fd, F_GETLK, &compet);
            printf("Lock held by pid %ld, ", (long)compet.l_pid);
            printf("try again\n");
```

```
            return;
        }

        // seek to beginning
        lseek(fd, 0, SEEK_SET);

        // read score
        exit_if((nrd = read(fd, buf, 31)), "read score");
        buf[nrd] = 0;
        if (nrd != 0) sscanf(buf, "%d-%d", &rp, &wp);

        printf("Updating score...\n");

        // write score
        lseek(fd, 0, SEEK_SET);
        sprintf(buf, "%d-%d", rp+runs, wp+wickets);
        write(fd, buf, strlen(buf)+1);

        // sleep(10) for taking OS Lab screenshot with 2 terminals
        sleep(10);

        // Unlock file
        writelk.l_type = F_UNLCK;
        writelk.l_whence = SEEK_SET;
        writelk.l_start = 0;
        writelk.l_len = 0;
        exit_if(fcntl(fd, F_SETLK, &writelk), "fcntl unlock");
        return;
    }

    void read_score(int fd) {
        char buf[32];
        struct flock readlk, compet;
        readlk.l_type = F_RDLCK;
        readlk.l_whence = SEEK_SET;
        readlk.l_start = 0;
```

```
	readlk.l_len = 0;

	if(fcntl(fd, F_SETLK, &readlk) == -1) {
		fcntl(fd, F_GETLK, &compet);
		printf("Lock held by pid %d, ", (int)compet.l_pid);
		printf("try again\n");
		return;
	}

	// seek to beginning
	lseek(fd, 0, SEEK_SET);

	// read and print score
	read(fd, buf, 31);
	buf[31] = '\0';
	printf("\e[31m%s\e[0m\n", buf);

	// Unlock file
	readlk.l_type = F_UNLCK;
	readlk.l_whence = SEEK_SET;
	readlk.l_start = 0;
	readlk.l_len = 0;
	exit_if(fcntl(fd,F_SETLK,&readlk), "fcntl unlock");
	return;
}

int main (int argc, char *argv[]) {
	int fd, runs = 0, wickets = 0;
	char buf[128];

	// Open file
	exit_if((fd = open(".scores", O_RDWR | O_CREAT,
		S_IRUSR | S_IWUSR)),
		"Open file .scores");
	while(1) {
		int n, r;
```

```
            printf("1. Read Score  2. Add Runs  3. Add A Wicket");
            printf("  0. Exit\n");
            scanf("%d", &n);
            switch(n) {
            case 0: return 0;
            case 1: read_score(fd);
                  break;
            case 2: printf("Num of runs: ");
                  scanf("%d", &r);
                  update_score(fd, r, 0);
                  break;
            case 3: update_score(fd, 0, 1);
                  break;
            default: printf("Invalid choice\n");
            }
      }
      return 0;
}
```

Output: