```
/* Disciplina: Computacao Concorrente */
/* Prof.: Silvana Rossetto */
/* Descricao: implementa  o problema dos leitores/escritores usando
variaveis de condicao da biblioteca Pthread
*/

#include<pthread.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

#define L 4 //numero de threads leitoras
#define E 2 //numero de threads escritoras

//variaveis do problema
int leit=0; //contador de threads lendo
int escr=0; //contador de threads escrevendo

//variaveis para sincronizacao
pthread_mutex_t mutex;
pthread_cond_t cond_leit, cond_escr;

//entrada leitura
void InicLeit (int id) {
   pthread_mutex_lock(&mutex);
   printf("L[%d] quer ler\n", id);
   while(escr > 0) {
     printf("L[%d] bloqueou\n", id);
     pthread_cond_wait(&cond_leit, &mutex);
     printf("L[%d] desbloqueou\n", id);
   }
   leit++;
   pthread_mutex_unlock(&mutex);
}

//saida leitura
void FimLeit (int id) {
   pthread_mutex_lock(&mutex);
   printf("L[%d] terminou de ler\n", id);
   leit--;
   if(leit==0) pthread_cond_signal(&cond_escr);
   pthread_mutex_unlock(&mutex);
}

//entrada escrita
void InicEscr (int id) {
   pthread_mutex_lock(&mutex);
   printf("E[%d] quer escrever\n", id);
   while((leit>0) || (escr>0)) {
     printf("E[%d] bloqueou\n", id);
     pthread_cond_wait(&cond_escr, &mutex);
     printf("E[%d] desbloqueou\n", id);
   }
   escr++;
   pthread_mutex_unlock(&mutex);
}
```

```
//saida escrita
void FimEscr (int id) {
   pthread_mutex_lock(&mutex);
   printf("E[%d] terminou de escrever\n", id);
   escr--;
   pthread_cond_signal(&cond_escr);
   pthread_cond_broadcast(&cond_leit);
   pthread_mutex_unlock(&mutex);
}

//thread leitora
void * leitor (void * arg) {
  int *id = (int *) arg;
  while(1) {
    InicLeit(*id);
    printf("Leitora %d esta lendo\n", *id);
    FimLeit(*id);
    sleep(1);
  }
  free(arg);
  pthread_exit(NULL);
}

//thread leitora
void * escritor (void * arg) {
  int *id = (int *) arg;
  while(1) {
    InicEscr(*id);
    printf("Escritora %d esta escrevendo\n", *id);
    FimEscr(*id);
    sleep(1);
  }
  free(arg);
  pthread_exit(NULL);
}

//funcao principal
int main(void) {
  //identificadores das threads
  pthread_t tid[L+E];
  int id[L+E];

  //inicializa as variaveis de sincronizacao
  pthread_mutex_init(&mutex, NULL);
  pthread_cond_init(&cond_leit, NULL);
  pthread_cond_init(&cond_escr, NULL);

  //cria as threads leitoras
  for(int i=0; i<L; i++) {
    id[i] = i+1;
    if(pthread_create(&tid[i], NULL, leitor, (void *) &id[i]))
exit(-1);
  }

  //cria as threads escritoras
  for(int i=0; i<E; i++) {
```

```
    id[i+L] = i+1;
    if(pthread_create(&tid[i+L], NULL, escritor, (void *) &id[i+L]))
exit(-1);
  }

  pthread_exit(NULL);
  return 0;
}
```