```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>
#include "lock.h"

#define SIZE 100000000

typedef struct {
  Lock *lock;
  int count;
  int array[SIZE];
} Environment;


Environment *make_environment ()
{
  int i;

  Environment *env = malloc (sizeof (Environment));
  env->count = 0;
  for (i=0; i<SIZE; i++) {
    env->array[i] = 0;
  }
  env->lock = make_lock ();

  return env;
}

void fill_array (Environment *env) {
  int i = 0;
  double x;
  int filled = 0;

  while (1) {
    acquire (env->lock);

    i = env->count;
    if (env->count >= SIZE) {
      release (env->lock);
      break;
    }
    env->count++;

    env->array[i]++;
    release (env->lock);


    filled++;
  }
  printf ("I filled in %d cells\n", filled);
}

void check_array (int *array)
{
  int i;
  for (i=0; i<SIZE; i++) {
    if (array[i] != 1) {
      printf ("element %d is %d\n", i, array[i]);
    }
  }
  printf ("done\n");
}
```



```c
void *entry (void *arg)
{
  // take the argument and cast it to be an Env pointer
  Environment *env = (Environment *) arg;

  fill_array (env);

  pthread_exit (NULL);
}

main ()
{
  int i, ret;
  pthread_t child;
  Environment *env;

  // create the shared environment
  env = make_environment ();

  // create a child
  ret = pthread_create (&child, NULL, entry, (void *) env);
  if (ret < 0) {
    perror ("pthread_create failed");
    exit (-1);
  }

  fill_array (env);

  pthread_join(child, NULL);

  check_array (env->array);

  exit (0);
}
```