Pretende-se um programa multithreaded que preencha um array comum com o máximo de 10000000 entradas.
Cada entrada deve ser preenchida com um valor igual ao seu índice.
Devem ser criados vários threads concorrentes para executar esse preenchimento.
Após o preenchimento, um último thread deverá verificar a correcção desse preenchimento.
O número efectivo de posições do array a preencher e o número efectivo de threads devem ser passados como parâmetros ao programa.

Segue-se uma solução utilizando as funções fill() e verify() para os threads de preenchimento e verificação. Além disso toma-se nota do número de posições preenchidas por cada thread fill().
Os threads partilham globalmente o array a preencher, a posição actual, e o valor de preenchimento actual (que por acaso é igual ao índice actual):

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

#define MAXPOS 10000000 /* nr. max de posições */
#define MAXTHRS 100 /* nr. max de threads */
#define min(a, b) (a)<(b)?(a):(b)

int npos;
pthread_mutex_t mut=PTHREAD_MUTEX_INITIALIZER; /* mutex para a s.c. */
int buf[MAXPOS], pos=0, val=0; /* variáveis partilhadas */

void *fill(void *);
void *verify(void *);

int main(int argc, char *argv[]){
   int k, nthr, count[MAXTHRS]; /* array para contagens */
   pthread_t tidf[MAXTHRS], tidv; /* tid's dos threads */

   if (argc != 3) {
      printf("Usage: fillver <nr_pos> <nr_thrs>\n");
      return 1;
   }

    npos = min(atoi(argv[1]), MAXPOS); /* nr. efectivo de posições */
    nthr = min(atoi(argv[2]), MAXTHRS); /* nr. efectivo de threads */
    for (k=0; k<nthr; k++) {
       count[k] = 0; /* criação dos threads fill() */
       pthread_create(&tidf[k], NULL, fill, &count[k]);
    }
    for (k=0; k<nthr; k++) {
       pthread_join(tidf[k], NULL); /* espera pelos threads fill() */
       printf("count[%d] = %d\n", k, count[k]);
    }
    pthread_create(&tidv, NULL, verify, NULL);
    pthread_join(tidv, NULL); /* thread verificador */
    return 0;
}
```

```
void *fill(void *nr){
   while (1) {
      pthread_mutex_lock(&mut);
      if (pos >= npos) {
         pthread_mutex_unlock(&mut);
         return NULL;
      }
      buf[pos] = val;
      pos++; val++;
      pthread_mutex_unlock(&mut);
      *(int *)nr += 1;
   }
}

void *verify(void *arg){
   int k;
   for (k=0; k<npos; k++){
      if (buf[k] != k) /* escreve se encontrar valores errados */
         printf("buf[%d] = %d\n", k, buf[k]);
   }
   return NULL;
}
```

Se tudo correr bem o programa deverá apenas escrever o número de posições preenchidas por cada thread fill().
Experimentar usando por exemplo:
fillver 1000000 5
fillver 5000000 5
fillver 10000000 5