```
**********************
    tabla_mult.c
**********************
/* tabla_mult.c imprime las 10 tablas de multiplicar usando
   una UNICA VARIABLE DE CONDICION. Es una solucion viable
   pero poco eficiente porque avisa a todos los hilos
   bloqueados en dicha variable de condicion y algunos de
   esos hilos no deberian desbloquearse y han de bloquearse de nuevo
*/

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

#define DIM 10

pthread_mutex_t mutex;
pthread_cond_t  turno_imprimir;
int turno_tabla=1;
int v_indice[DIM+1];

// En el programa usamos vecotres de DIM+1 componentes porque en C
los
// componentes de un vector empiezan a numerarse por 0. Pero
trataremos con
// las componentes 1 hasta 10 de esos vectores para que la
implementacion
// resulte mas sencilla y legible.

void *f_tabla(void *arg)
{
  int i;
  int n_tabla;
  n_tabla=*(int *)arg;

  for (i=1; i<(DIM+1); i++)
  {  pthread_mutex_lock(&mutex);
     while ( turno_tabla != n_tabla)
        pthread_cond_wait(&turno_imprimir,&mutex);

     printf("%d x %d = %d  \t",n_tabla,i,n_tabla*i);
     turno_tabla++;
     if (turno_tabla == (DIM+1)) { turno_tabla=1; printf("\n");}

     pthread_cond_broadcast(&turno_imprimir);
     pthread_mutex_unlock(&mutex);
  }
  pthread_exit(0);
}

main()
{
  int i;
```

```
  pthread_t hilo[DIM+1];

  pthread_mutex_init(&mutex, NULL);
  pthread_cond_init(&turno_imprimir,NULL);

  for (i=1; i<(DIM+1); i++)
  { v_indice[i]=i;
    pthread_create(&hilo[i], NULL,  f_tabla, (void *)&v_indice[i]);
  }
  for (i=1; i<(DIM+1); i++) pthread_join(hilo[i], NULL);

  pthread_cond_destroy(&turno_imprimir);
  pthread_mutex_destroy(&mutex);

  pthread_exit(0);
}



**********************
    tabla_mult2.c
**********************
/* tabla_mult2.c imprime las 10 tablas de multiplicar usando
   un VECOTR DE VARIABLES DE CONDICION. Esta implementacion
   avisa (desbloqea) directamente al hilo que debe imprimir
   a continuacion su parte de la tabla.  */

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


#define DIM 10

pthread_mutex_t mutex;
pthread_cond_t  turno_imprimir[DIM+1];
int turno_tabla=1;

void *f_tabla(void *arg)
{
  int i;
  int n_tabla;
  n_tabla=(int *)arg;

  for (i=1; i<(DIM+1); i++)
  {  pthread_mutex_lock(&mutex);
     while ( turno_tabla != n_tabla)
        pthread_cond_wait(&turno_imprimir[n_tabla],&mutex);

     printf("%d x %d = %d  \t",n_tabla,i,n_tabla*i);
     turno_tabla++;
     if (turno_tabla == (DIM+1)) { turno_tabla=1; printf("\n");}
```

```
     pthread_cond_signal (&turno_imprimir[turno_tabla]);
     pthread_mutex_unlock(&mutex);
  }
  pthread_exit(0);
}

main()
{
  int i;
  pthread_t hilo[DIM+1];

  pthread_mutex_init(&mutex, NULL);
  for (i=1; i<(DIM+1); i++)
pthread_cond_init(&turno_imprimir[i],NULL);


  for (i=1; i<(DIM+1); i++) pthread_create(&hilo[i], NULL,  f_tabla,
(void *)i);
  for (i=1; i<(DIM+1); i++) pthread_join(hilo[i], NULL);

  for (i=1; i<(DIM+1); i++)
pthread_cond_destroy(&turno_imprimir[i]);
  pthread_mutex_destroy(&mutex);

  pthread_exit(0);
}
```