```
#define MAX_NODES 16

THREADED
esclave(
        /* Endroit ou mettre la sync slot pour ma tache. */
        SPTR *GLOBAL ma_tache,
        /* Endroit ou mettre la sync slot pour ma fibre de finalisation. */
        SPTR *GLOBAL ma_fin,
        /* La barriere globale. */
        SPTR la_barriere,
        /* Sync slot a signaler lorsque tout est fini. */
        SPTR fin
        )
{
  int k = 0;

  /* On transmet les sync slots pour les fibres locales. */
  PUT_SYNC( TO_SPTR(MA_TACHE), ma_tache, la_barriere );
  PUT_SYNC( TO_SPTR(MA_FIN),   ma_fin,   la_barriere );

  /* Tache activee apres que toutes aient synchronise la barriere. */
  FIBER MA_TACHE <* 1, 1 *> {
    printf( "k = %d\n", k++ );
    SYNC( la_barriere );
  }

  /* Fibre de terminaison du programme. */
  FIBER MA_FIN <* 1 *> {
    SYNC( fin );
    TERMINATE;
  }
}

THREADED
coordonnateur(
              /* Nombre d'iterations a effectuer pour chacune des taches. */
              int nb_iterations,
              SPTR fin )
{
  int i;
  SPTR taches[MAX_NODES], fins[MAX_NODES];

  /* On cree les esclaves en specifiant la barriere
     et la sync slot de terminaison.  */
  for( i = 0; i < NUM_NODES; i++ )
   INVOKE( i, esclave,
           TO_GLOBAL(&taches[i]), TO_GLOBAL(&fins[i]),
           TO_SPTR(BARRIERE), TO_SPTR(FIN) );

  /* La barriere: initialement, on doit recevoir 2 signaux de chacun
     des esclaves (1 signal pour la tache et un autre pour la fibre
     de terminaison.  Par la suite, un seul signal est requis de chacun
     des esclaves. */
  FIBER BARRIERE <* 2*NUM_NODES, NUM_NODES *> {
    nb_iterations--;
    if (nb_iterations > 0)
      /* Il reste d'autres iterations: on resignale chacune des taches. */
      for( i = 0; i < NUM_NODES; i++ ) SYNC( taches[i] );
    else
      /* Fin des iterations: on signale terminaison globale. */
      for( i = 0; i < NUM_NODES; i++ ) SYNC( fins[i] );
  }

  FIBER FIN <* NUM_NODES *> {
    /* Tous les esclaves ont termine: on peut donc terminer. */
    SYNC( fin );
    TERMINATE;
  }
}
```



```
void print_usage_and_die()
{
  printf( "usage:\n" );
  printf( "  barriere n\n" );
  printf( "  (n > 0)\n" );
}

THREADED MAIN( int argc, char* argv[] )
{
  int n;

  /* On lit le nombre d'iterations a effectuer. */
  if ( argc <= 1 )
    print_usage_and_die();
  n = atoi( argv[1] );
  if ( n <= 0 )
    print_usage_and_die();

  INVOKE( 0, coordonnateur, n, TO_SPTR(FIN) );

  FIBER FIN <* 1 *> {
    TERMINATE;
  }
}
```