```
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
!
! Compilation with (ADAPTOR + MPICH):
!  adaptor -hpf -dm -free -c loc_tasklib.f90
!  adaptor -hpf -dm -free -o loc_tasklib.exe loc_tasklib.o
!  adaptor -hpf -dm -free -o loc_tasklib.exe loc_tasklib.f90
!
! Running with MPICH:
!  mpdboot -n 21 -f ~/mpd.hosts
!  mpdtrace
!  mpiexec -machinefile ~/machi.dat -np 2 loc_tasklib.exe
!  mpdallexit
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
!
!     Test Program to check the task library functionality in LOCAL
!
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
program  tareas_locales
  implicit none
  integer :: np
  integer, dimension (1:number_of_processors ()) :: ier
  !hpf$ distribute (block) :: ier
  integer errata
  np = number_of_processors ()
  if (np < 2) then
     print *, "application must run on at least 2 processors"
  else
     ier = 0
     print *
     print *, "Test HPF_TASK_LIBRARY (local) on ",np," procs"
     print *, "============================================="
     print *
     call local (ier)
     errata = sum (ier)
     if (errata .eq. 0) then
        print *, "HPFLocalTaskLib NP =", np, ": all tests PASSED"
     else
        print *, "HPFLocalTaskLib NP =", np, ": FAILED"
     end if
  end if
  !
  write (*,*)
  write (*,*)"FINE ... "
  write (*,*)
end program
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
!
!     LOCAL : SPMD code with message passing via TASK library
!
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
extrinsic (HPF_LOCAL) subroutine local (ier)
  implicit none
  use HPF_TASK_LIBRARY
  integer, dimension (:) :: ier
  !hpf$ distribute (block) :: ier
  integer :: siza, rango
  integer :: valor
  !
  call HPF_TASK_INIT ()
  call HPF_TASK_RANK (rank = rango)
  call HPF_TASK_SIZE (size = siza)
  print *, "I am ", rango, " of ", siza
  !
  ! PING PONG Test
  if     (rango == 1) then
     call tarea1 (10)
  elseif (rango == 2) then
     call tarea2 (10, ier)
  end if
```



```
  !
  ! global TEST
  print *, "bcast between processors in local mode"
  valor = 2
  if (rango == 1) valor = 5
  call HPF_BCAST (valor)
  if (valor .ne. 5) ier = 1
  !
  call HPF_TASK_EXIT ()
end subroutine
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
!
!     TASK1 : task that will send data
!
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
subroutine tarea1 (n)
  implicit none
  use HPF_TASK_LIBRARY
  integer, intent (in)   :: n
  integer, dimension (n) :: a, ra, a1
  real   , dimension (n) :: s
  integer, dimension (n,n) :: b
  !
  print *, "send/recv of a scalar"
  call HPF_SEND (data = n, dest = 2)
  !
  print *, "send distributed => recv distributed"
  a = 1
  call HPF_SEND (data = a, dest = 2)
  !
  ! now send an array for comparison
  call random_number (s)
  a = s * 100
  print *, "send A = ", a
  call HPF_SEND (data = a, dest = 2)
  print *, "send_recv "
  call HPF_SEND_RECV (send_data =  a, dest   = 2,  &
                      recv_data = a1, source = 2)
  !
  print *, "send_recv back"
  call HPF_SEND_RECV (send_data = a1, dest   = 2,  &
                      recv_data =  a, source = 2)
  !
  print *, "send/recv two-dimensional"
  b = spread (a, dim = 1, ncopies = n)
  call HPF_SEND (data = b, dest = 2)
  !
  print *, "send/recv two-dimensional with transpose"
  call HPF_SEND (data = b, dest = 2, order = [2, 1])
  !
end subroutine
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
!
!     TASK2 : task that will receive data
!
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
subroutine tarea2 (n, ier)
  implicit none
  use HPF_TASK_LIBRARY
  integer, dimension (1)   :: ier
  integer                  :: n
  integer, dimension (n)   :: a, ra, z
  integer, dimension (n,n) :: b, res2, res2t
  integer :: errata = 0 , n1
  !
  a = 0
  call HPF_TASK_INIT ()
  call HPF_RECV (data = n1, source = 1)
  if (n1 .ne. n) then
```

```
    print *, "received illegal scalar val N"
    errata = errata + 1
  end if
  call HPF_RECV (data = a, source = 1)
  if (sum (a) .ne. n) then
    print *, "received illegal integer array A (1)"
    errata = errata + 1
  end if
  ! receive an array for comparison
  call HPF_RECV (data = z, source = 1)
  print *, "received A as Z = ", z
  print *, "send_recv "
  a = 0
  call HPF_SEND_RECV (send_data = z, dest = 1,  &
                      recv_data = a, source = 1)
  if (any (a .ne. z)) then
    print *, "received illegal integer array A (send_recv 1)"
    errata = errata + 1
  end if
  !
  call HPF_SEND_RECV (send_data = a, dest = 1,  &
                      recv_data = z, source = 1)
  if (any (a .ne. z)) then
    print *, "received illegal integer array A (send_recv 2)"
    errata = errata + 1
  end if
  !
  ! print *, "send/recv two-dimensional"
  res2 = spread (z, dim=1, ncopies = n)
  call HPF_RECV (data = B, source = 1)
  if (any (b .ne. res2)) then
    print *, "received illegal integer array B"
    print *, "received: ", b
    print *, "expected: ", res2
    errata = errata + 1
  end if
  !
  print *, "send/recv two-dimensional with transpose"
  call HPF_RECV (data = b, source = 1)
  res2t = transpose (res2)
  if (any (b .ne. res2t)) then
    print *, "received illegal integer array B (transpose)"
    print *, "received: ", b
    print *, "expected: ", res2t
    errata = errata + 1
  end if
  print *, "TASK2 : errors = ", errata
  ier (1) = errata
  !
end subroutine
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
```

```
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
! Compilation with (ADAPTOR + MPICH):
!  adaptor -hpf -dm -free -c tasklib.f90
!  adaptor -hpf -dm -free -o tasklib.exe tasklib.o
!  adaptor -hpf -dm -free -o tasklib.exe tasklib.f90
!
! Running with MPICH:
!  mpdboot -n 21 -f ~/mpd.hosts
!  mpdtrace
!  mpiexec -machinefile ~/machi.dat -np 2 tasklib.exe
!  mpdallexit
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
!
!     Test Program to check the task library functionality
!
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
program comunica_tasks
  implicit none
  integer np, n1, n2, errata
  np = number_of_processors ()
  if (np < 2) then
     print *, "application must run on at least 2 processors"
     stop
  end if
  n1 = np/2
  n2 = np - n1
  print *
  print *, "Test HPF_TASK_LIBRARY on ",np," procs"
  print *, "==================================="
  print *
  call tareas (np, n1, n2, errata)
  if (errata .eq. 0) then
     print *, "HPFtaskLib NP =", np, ": all tests PASSED"
    else
     print *, "HPFtaskLib NP =", np, ": at least one test FAILED"
  end if
  !
  write (*,*)
  write (*,*)"FINE ... "
  write (*,*)
end program
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
!
!     TASKING: starts two concurrent data parallel tasks
!
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
subroutine  tareas (p, p1, p2, errata)
  implicit none
  ! runs two data parallel tasks on P1 and P2 processors
  integer, intent (in)   :: p, p1, p2, errata
  integer, parameter     :: n = 4             ! problem size
  integer, dimension (1) :: ier
  !hpf$ distribute ier (block) onto proc (p:p)
  !hpf$ processors proc (P)
  print *, "run TASK1 on ", 1,    " - ", p1
  print *, "run TASK2 on ", p1+1," - ", p
  ! define a task region, otherwise home will be ignored
  !hpf$ independent task_region
  !hpf$   on (proc(1:p1))   , resident
            call  tarea1 (n)
  !hpf$   on (proc(p1+1:p)), resident
            call  tarea2 (n, ier)
  !hpf$ end task_region
  errata = ier (1)
end subroutine
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
!
!     TASK1 : task that will send data
!
```



```
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
subroutine  tarea1 (n)
  implicit none
  use HPF_TASK_LIBRARY
  integer, intent (in)       :: n
  integer, dimension (n)     :: a, ra, a1
  real   , dimension (n)     :: r
  !hpf$ distribute (block)   :: a, a1
  integer, dimension (n,n)   :: b
  !hpf$ distribute (*,block) :: b
  integer :: kontrol
  !
  call HPF_TASK_INIT ()
  !
  print *, "send/recv of a scalar"
  call HPF_SEND (data = n, dest = 2)
  !
  a = 1
  print *, "send distributed => recv distributed"
  call HPF_SEND (data = a, dest = 2)
  !
  ! now send an array for comparison
  call random_number (r)
  a = r * 100
  print *, "send A = ", a
  call HPF_SEND (data = a, dest = 2)
  !
  print *, "send replicated => recv distributed"
  ra = a
  call HPF_SEND (data = ra, dest = 2)
  !
  print *, "send distributed => recv replicated"
  call HPF_SEND (data =  a, dest = 2)
  !
  print *, "send replicate => recv replicated"
  call HPF_SEND (data = ra, dest = 2)
  !
  print *, "send/recv with REQUEST"
  call HPF_SEND_INIT (data = a, dest = 2, request = kontrol)
  call HPF_TASK_COMM (kontrol)
  !
  print *, "send_recv "
  call HPF_SEND_RECV (send_data =  a, dest   = 2,  &
                      recv_data = a1, source = 2)
  !
  print *, "send_recv back"
  call HPF_SEND_RECV (send_data = a1, dest   = 2,  &
                      recv_data =  a, source = 2)
  !
  print *, "send/recv two-dimensional"
  b = spread (a, dim=1, ncopies=n)
  call HPF_SEND (data = b, dest = 2)
  !
  print *, "send/recv two-dimensional with transpose"
  call HPF_SEND (data = b, dest = 2, order = [2, 1])
  !
  call HPF_TASK_EXIT ()
end subroutine
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
!
!     TASK2 : task that will receive data
!
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
subroutine tarea2 (n, ier)
  implicit none
  use HPF_TASK_LIBRARY
  integer, dimension (1)     :: ier
  integer                    :: n
  !hpf$ inherit               :: ier
```

```
   integer, dimension (n)     :: a, ra, z
   !hpf$ distribute (block)   :: a, z
   integer, dimension (n,n)   :: b, res2, res2t
   !hpf$ distribute (*,block) :: b, res2, res2t
   integer errata = 0, kontrol, n1
   !
   a = 0
   call HPF_TASK_INIT ()
   !
   call HPF_RECV (data = n1, source = 1)
   if (n1 .ne. n) then
       print *, "received illegal scalar val n"
       errata = errata + 1
   end if
   !
   call HPF_RECV (data = a, source = 1)
   if (sum (a) .ne. n) then
       print *, "received illegal integer array A (1)"
       errata = errata + 1
   end if
   !
   ! receive an array for comparison
   call HPF_RECV (data = z, source = 1)
   print *, "received A as Z = ", z
   !
   ! print *, "send replicated => recv distributed"
   call HPF_RECV (data = a, source = 1)
   if (any (a .ne. z)) then
       print *, "received some illegal integer array A (2)"
       errata = errata + 1
   end if
   !
   ! print *, "send distributed => recv replicated"
   ra = 0
   call HPF_RECV (data = ra, source = 1)
   if (any (ra .ne. z)) then
       print *, "received some illegal integer array RA (1)"
       errata = errata + 1
   end if
   !
   ! print *, "send replicated => recv replicated"
   ra = 0
   call HPF_RECV (data = ra, source = 1)
   if (any (ra .ne. z)) then
       print *, "received some illegal integer array RA (2)"
       errata = errata + 1
   end if
   !
   ! print *, "send/recv with REQUEST"
   a = 0
   call HPF_RECV_INIT (data = a, source = 1, request = kontrol)
   call HPF_TASK_COMM (kontrol)
   if (any (a .ne. z)) then
      print *, "received some illegal integer array A (request)"
      errata = errata + 1
   end if
   !
   print *, "send_recv "
   a = 0
   call HPF_SEND_RECV (send_data = z, dest   = 1,  &
                       recv_data = a, source = 1)
   if (any (a .ne. z)) then
      print *, "received some illegal integer array A (send_recv 1)"
      errata = errata + 1
   end if
   !
   call HPF_SEND_RECV (send_data = a, dest   = 1,  &
                       recv_data = z, source = 1)
   if (any (a .ne. z)) then
```



```
      print *, "received illegal integer array A (send_recv 2)"
      errata = errata + 1
   end if
   !
   ! print *, "send/recv two-dimensional"
   res2 = spread (z, dim = 1, ncopies = n)
   call HPF_RECV (data = b, source = 1)
   if (any (b .ne. res2)) then
      print *, "received illegal integer array B"
      print *, "received: ", b
      print *, "expected: ", res2
      errata = errata + 1
   end if
   !
   print *, "send/recv two-dimensional with transpose"
   call HPF_RECV (data = b, source = 1)
   res2t = transpose (res2)
   if (any (b .ne. res2t)) then
      print *, "received illegal integer array B (transpose)"
      print *, "received: ", b
      print *, "expected: ", res2t
      errata = errata + 1
   end if
   print *, "TASK2 : errors = ", errata
   ier (1) = errata
   call HPF_TASK_EXIT ()
   !
end subroutine
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
```

```
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
! Problema del Juego de la Vida
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
! Compilation with INTEL:
!  ifort -vms -c vida.f90
!  ifort -vms -o vida.exe1 *.o
!
! Compilation with GNU:
!  g95 -c circulo.f90
!  g95 -o vida.exe2 *.o
!
! Compilation with (ADAPTOR + MPICH):
!  adaptor -hpf -dm -free -c vida.f90
!  adaptor -hpf -dm -free -o vida.exe3 *.o
!
! Running with MPICH:
!  mpdboot -n 4 -f ~/mpd.hosts
!  mpdtrace
!  mpdringtest
!  mpiexec -machinefile ~/machi.dat -np 4 vida.exe3
!  mpiexec -machinefile ~/machi.dat -1 -np 4 vida.exe3
!  mpdallexit
!
! Para ver alguna "instantanea", e.g. la #10
!    xv -expand 0.5 gvida.010.pgm &
!
! Opcional:
!    a) Convertir las "instantaneas" *.pgm en *.gif con:
!       for f in *.pgm
!        do echo convert $f `basename $f /pgm`.gif ;
!                convert $f `basename $f /pgm`.gif ;
!       done
!    b) Crear el unico archivo de animacion (de pequenio tamanio
!       porque solo guarda las diferencias entre las fotos):
!       anim.pl -p gvida -t
!       anim.pl -p gvida
!    c) Ejecutar la animacion con: gifview gvida_anim.gif &
!
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
! Programa HPF para simular el juego de la vida en un tablero 2D
! Animacion con xv -expand 10 -wait 0.1 -rw *.pgm
! Ref. Liverpool HPF course, Marshall A.C., 1997.
! ------------------------------------------------------------------
program vida
  implicit none
  !
  !parametros del "experimento biologico"
  integer, parameter :: n_generaciones = 100
  integer, parameter :: n = 64 ! tablero 2D
  integer, parameter :: h = 16 ! patron inicial
  !
  !eventual impresion txt (mejor evitarla pues es MUY LENTA !!)
  integer, parameter :: cmax =  70 ! max col.impr.
  logical, parameter :: imprime = .false.
  !
  !declara y distribuye datos
  integer, dimension (n,n) :: tablero
  integer, dimension (n,n) :: vecinos
  !hpf$ distribute (cyclic,cyclic) :: tablero, vecinos
  !
  integer        :: nx, ny, ntasks, nvivas
  integer        :: i, j, m, p
  character (16) :: pic, txt
  character (1)  :: s
  !
  !variables para cronometro
  real   , dimension (3) :: time
  integer                :: h1, h2, h3
  real   , parameter     :: cero = 0.0
```



```
  !
  !titulos
  ntasks = number_of_processors ()
  write (*,*)
  write (*,*)' juego de la vida con HPF y con salida a xv'
  write (*,*)' number_of_processors = ', ntasks
  !
  !inicia cronometro;  t1 = second ()
  time = cero
  call system_clock (count = h1, count_rate = h3)
  !
  ! inicializa tablero
  nx = size (tablero,1)
  ny = size (tablero,2)
  !hpf$  independent
  forall (i=1:nx, j=1:ny) tablero (i,j) = 0
  m = n / 2
  !hpf$  independent
  forall (i = m - h/2 + 1 : m + h/2 )  tablero (i,m) = 1
  !
  !tablero: impresion para "xv"
  write (pic,100) 0
  open  (10, file = pic, status = 'unknown')
  write (10,110)  nx, ny, 1
  write (10,120)  tablero
  close (10, status = 'keep')
  !
  !tablero: eventual impresion txt (mejor evitarlo)
  if ( imprime .and. n .le. cmax ) then
    write (txt,90) 0
    open  (11, file = txt, status = 'unknown')
    do  i = 1, nx
    do  j = 1, ny
    s = '.'
    if ( tablero (i,j) .eq. 1 )  s = '*'
    write (11,80,advance='no') s
    end do ! j
    write (11,*)
    end do ! i
    close (11, status='keep')
  end if
  !
  !inicia cronometro
  call system_clock (count = h2)      ! t2 = second ()
  time (1) = h2 - h1                  ! time (1) = t2 - t1
  call system_clock (count = h1)      ! t1 = second ()
  !
  !lazo sobre las generaciones
  generacion : do  p = 1, n_generaciones
    !vecinos
    vecinos =      tablero                          &
        + cshift ( tablero, shift = -1, dim = 1)  &
        + cshift ( tablero, shift = +1, dim = 1)
    vecinos =      vecinos                          &
        + cshift ( vecinos, shift = -1, dim = 2)  &
        + cshift ( vecinos, shift = +1, dim = 2)
    vecinos = vecinos - tablero
    !
    ! actualiza tablero segun reglas de juego
    where ( vecinos < 2 .or. vecinos > 3 )  tablero = 0
    where ( vecinos == 3 )                  tablero = 1
    !
    !tablero: impresion para xv
    write (pic,100) p
    open  (10, file = pic, status = 'unknown')
    write (10,110)  nx, ny, 1
    write (10,120)  tablero
    close (10, status = 'keep')
    !
```

```
    !tablero: eventual impresion txt (evitarlo para n grande)
    if  (imprime .and. n .le. cmax) then
      write (txt,90) p
      open  (11, file = txt, status = 'unknown')
      write (11,'(i5)') p
      do  i = 1, nx
      do  j = 1, ny
      s = '.'
      if ( tablero (i,j) .eq. 1 )  s = '*'
      write (11,80,advance='no') s
      end do ! j
      write (11,*)
      end do ! i
      close (11, status='keep')
    end if
  end do generacion
  !
  !cronometro
  call system_clock (count = h2)     ! t2 = second ()
  time (2) = h2 - h1                 ! time (2) = t2 - t1
  time (3) = time (1) + time (2)     ! total_time
  time = time / h3                   ! pasa de ciclos a seg
  !
  !numero de celulas vivas remanentes
  nvivas = sum (tablero, mask = tablero .eq. 1)
  write (*,*)
  write (*,*)' nro de celulas vivas     ; nvivas = ', nvivas
  write (*,*)' al cabo de la generacion ;      p = ', p
  !
  !imprime
  write (*,*)
  write (*,*)' tpo de inicial.  [s] ; t1 = ', real (time(1))
  write (*,*)' tpo de barrido   [s] ; t2 = ', real (time(2))
  write (*,*)' tiempo_total     [s] ; t3 = ', real (time(3))
  write (*,*)
  !
  !imprime en archivo historico
  open  (12, file = 'vida.tim', status = 'unknown')
  write (12,60)  ntasks, time (1:3)
  close (12, status = 'keep')
  !
  !formatos
  60  format (1x, i4, 3 (1x, e19.9) )
  70  format (i1)
  80  format (a1)
  90  format ('tvida', i3.3, '.txt')
  !formatos p/archivo xv. Usar xv -expand 10 -wait 0.1 -rw *.pgm
  100 format ('gvida.', i3.3, '.pgm')
  110 format ('P2' / i4 , 2x , i4 / i4)
  120 format ( 35 (1x, i1) )
end program
! ====================================================================
```