Cebrian Serrano, Guillermo
Gil Valverde, Alberto

Memoria antenas MPI:

Cambios realizados en el main:

Añadimos las variables rank y size.

//Iniciamos MPI:

```
MPI_Init( &nargs, &vargs );
MPI_Comm_rank( MPI_COMM_WORLD, &rank );
MPI_Comm_size( MPI_COMM_WORLD, &rank );
```

//Comprobamos que el numero de procesos sea mayor que dos para que el codigo no sea secuencial con if(size != 2).

//Creamos el tipo derivado MPI_Antena:

```
MPI_Datatype MPI_Antena;

Antena antena;
```

//Direcciones de los campos:

```
MPI_Aint address_antena;
MPI_Aint address_y;
MPI_Aint address_x;

MPI_Get_address(&antena, &address_antena);
MPI_Get_address(&antena.y, &address_y);
MPI_Get_address(&antena.x, &address_x);
```

Para el rank = 0:

Leemos los argumentos de entrada, reservamos memoria para las antenas, leemos las antenas iniciales las colocamos y creamos el mapa con malloc y lo inicializamos con el valor MAX_INT.

Una vez dentro del bucle para colocar las nuevas antenas en el proceso rank 0 particionamos la matriz por filas y las enviamos a los procesos distintos de rank 0:

```
int particiones = rows;
int p;
for(p=0;p<particiones;p++){
```

```
                        //Calculamos la particion
                        //Filas de la matriz
                        int p_ini = p*cols/particiones;
                        int p_fin = ((p+1) * cols/particiones)-1;
                        int tam = p_fin - p_ini + 1;
                        // Enviamos la particion
                        MPI_Send(&mapa[p_ini],tam,MPI_INT,p+1,999,MPI_COMM_WORLD);
                        // Recibimos el maximo
                       MPI_Recv(&max,1,MPI_INT,0,888,MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
 } // for
                    // Salimos si ya hemos cumplido el maximo
                    if (max <= distMax) break;
                    // Incrementamos el contador
                    nuevas++;

                    Antena antena = nueva_antena(mapa, rows, cols, max);
                    actualizar(mapa,rows,cols,antena);
```

El resto de procesos:

```
int * mapa;
mapa = malloc((size_t) cols * sizeof(int));
// Recibimos la particion de la matriz
MPI_Recv(mapa,cols,MPI_INT,0,999,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
// Calcular el máximo
int max = calcular_max(mapa, 0, cols,size);
// Enviamos el maximo
MPI_Send(&max,1,MPI_INT,0,888,MPI_COMM_WORLD);
```

Finalizamos MPI:

```
MPI_Finalize();
return EXIT_SUCCESS;
```

Cambios en la funcion calcular_max:
Añadimos el argumento size.

creamos un array de maximos:

```
int * maximo = malloc((size_t) tam * sizeof(int));
```

hacemos un MPI_Reduce(mapa, maximo, tam, MPI_INT, MPI_MAX, 0,
MPI_COMM_WORLD);

calculamos el maximo de el array y devolvemos el valor max:

```
for(i=0; i<rows; i++){

            if(maximo[i]>max){
                  max = maximo[i];
            }
      } // i
      return max;
```