OpenMP - CAR 2017
Práctica a realizar en aula informática

1. Se trata de recoger de nuevo de la web de la asignatura CAR_2017_VG.tgz, deshacerlo y compilar:

```
# tar zxvf CAR_2017_VG.tgz
# cd CAR_2016_VG
```

... comprobar los flags de compilación en Makefile: poner en el flag de optimización: "–O2"

```
# make pruVGo
```

Declarar la variable para encontrar las librerías y ampliar el límite del tamaño de pila permitido:

```
# export LD_LIBRARY_PATH=/opt/intel/lib/intel64
# ulimit -s unlimited
```

Ejecutar 'pruVGo' con la imagen "grande", anotar tiempo empleado y verificar el resultado:

```
# pruVGo –f ./Im1_IronMan.png
# eog Im3_Final.png
```

El tiempo empleado será algo superior a 1 segundo. A continuación, incorporar a `img_ngo.c` las líneas necesarias para paralelizar los bucles externos definidos en la subrutina Filtro y en la subrutina `Comp` y modificar adecuadamente el fichero Makefile (con *icc*: *-qopenmp*):

```
for (i=0; i<h; i++)         // Subrutina Filtro
  for (j=0; j<w; j++)
    *(im_o->imagen+i*w+j) = FilPixel(im_i, i, j, MF);

for (i=0; i<h; i++) {       // Subrutina Comp
  for (j=0; j<w; j++) {
    desp = i*w + j;
    difes += abs(*(i1->imagen + desp) - *(i2->imagen + desp));
  }
}
```

Dado que se están haciendo pruebas y no se trata de una versión final, interesará que se pueda definir en tiempo de ejecución tanto el número de *threads* como el tipo de planificación, para lo cual será necesario declarar en el código que la planificación es "*runtime*" y definir algunas variables de entorno, por ejemplo:

```
# export OMP_NUM_THREADS=8
# export OMP_SCHEDULE=guided,100
# export OMP_DISPLAY_ENV=true
```

Se trataría de anotar tiempos con distintas opciones en cuanto a planificación y número de *threads*. Para continuar las pruebas, será conveniente recompilar el código de *pruVGo* eliminando las optimizaciones del compilador, es decir, poniendo "–O0" y usar una imagen más pequeña:

```
# make clean
# make pruVGo
# pruVGo –f ./Im1_drone.png
```

En este caso se trata de comprobar los efectos asociados a declarar las variables del bucle paralelizado como privadas o no hacerlo, en particular, el tiempo de ejecución y ver en ambos casos si el resultado obtenido es o no correcto.

Finalmente puede realizar el mismo tipo de pruebas, ajustando qué variables tendrán que definirse como privadas, con el programa *mandelbrot.c* que genera una representación del conjunto de Mandelbrot que puede ser compilada, ejecutarse y visualizar el resultado con:

```
# /opt/intel/bin/icc -Wall -g -O2 -qopenmp mandelbrot.c -o m
# ./m
# eog mandelbrot.ppm
```