CC4302 Sistemas Operativos – Tarea 1 – Semestre Primavera 2023 – Prof.: Luis Mateu

La función *integral* incluida en *test-integral.c* calcula secuencialmente la integral de una función que se recibe como parámetro. El encabezado de la función es:

```
typedef double (*Funcion)(void *ptr, double x);
double integral(Funcion f, void *ptr, double xi, double
xf, int n);
```

Integral calcula numéricamente $\int_{xi}^{xf} f(q,x)\,dx$ usando el método de los trapecios, en donde *n* es el número de trapecios usados para aproximar el área bajo la curva. Para ello se emplea la siguiente fórmula:

$$\int_{xi}^{xf} f(q,x)\,dx \approx h\cdot\left[\frac{f(q,xi)+f(q,xf)}{2}+\sum_{k=1}^{n-1} f(q,xi+k\cdot h)\right]$$

con $h=\frac{xf-xi}{n}$. El puntero *q* se usa para pasar parámetros adicionales a la función en caso de necesidad. A modo de ejemplo suponga que Ud. dispone de la función *g(x, y)*. En el código de más abajo la función *integral_g_dx* usa *integral* para calcular numéricamente $\int_{xi}^{xf} g(x,y)\,dx$. Observe que como *g* no posee el tipo requerido por *integral*, se introduce *g_aux* que sí posee el tipo requerido.

```
double g_aux(void *ptr, double x) {
  double y= *(double *)ptr;
  return g(x, y);
}

double integral_g_dx(double xi, double xf, double y, int
n) {
  return integral(g_aux, &y, xi, xf, n);
}
```

Programe la función integral_par con el siguiente encabezado:

```
double integral_par(Funcion f, void *ptr,
                    double xi, double xf, int n, int p);
```

Integral_par debe calcular la misma integral que la función *integral* de más arriba, pero en paralelo usando *p* threads. Para lograrlo, descomponga el intervalo [*xi*, *xf*] en *p* subintervalos. Llame a *integral* (secuencial) para calcular la integral en cada uno de los subintervalos, pero usando *n/p* trapecios.

El programa de prueba incluido en *test-integral.c* calcula: $\int_{yi}^{yf}\int_{xi}^{xf} \sin(x+2y)\,dx\,dy$ usando 10000 subintervalos para *x* y 20000 para *y*.

Instrucciones

Baje *t1.zip* de U-cursos y descomprímalo. El directorio *T1* contiene los archivos *test-integral.c*, *Makefile*, *integral.h* (con los encabezados requeridos) y otros archivos. Ud. debe reprogramar en el archivo *integral.c* la función *integral_par*. Defina otras funciones si las necesita. **Se descontarán 5 décimas si su solución no usa la indentación de Kernighan**.

Pruebe su tarea bajo Debian 11 de 64 bits. Ejecute el comando *make* sin parámetros. Le mostrará las opciones que tiene para compilar su tarea. Estos son los requerimientos para aprobar su tarea:

- *make run* debe felicitarlo por aprobar este modo de ejecución. El *speed up* reportado debe ser de al menos 1.5.
- *make run-g* debe felicitarlo.
- *make run-san* debe felicitarlo y no reportar ningún incidente en el manejo de memoria.

Cuando pruebe su tarea con *make run* en su notebook asegúrese de que posea al menos 2 cores, que esté configurado en modo alto rendimiento y que no estén corriendo otros procesos intensivos en uso de CPU al mismo tiempo. De otro modo podría no lograr el *speed up* solicitado.

Invoque el comando *make zip* para ejecutar todos los tests y generar un archivo *integral.zip* que contiene *integral.c*, con su solución, y *resultados.txt*, con la salida de *make run*, *make run-g* y make *run-san*.

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *integral.zip* generado por *make zip*. Recuerde descargar el archivo que subió, descargar nuevamente los archivos adjuntos y volver a probar la tarea tal cual como la subió a U-cursos. Solo así estará seguro de no haber entregado archivos incorrectos. No se consideran los días de receso, sábados, domingos o festivos.