

Sistemas Operativos 1/2022

Laboratorio 2

Profesores:

Cristóbal Acosta (cristobal.acosta@usach.cl)

Fernando Rannou (fernando.rannou@usach.cl)

Ayudante:

Ricardo Hasbun (ricardo.hasbun@usach.cl)

I. Objetivo General

Este laboratorio tiene como objetivo aplicar algunos conceptos de concurrencia, tales como sección crítica y exclusión mutua para construir una solución multihebras al problema del laboratorio 1. La aplicación debe ser escrita en el lenguaje de programación C con *pthread*, sobre sistema operativo Linux.

II. Objetivos Específicos

1. Diseñar una solución concurrente en base a hebras para el cálculo de propiedades en el plano (u, v)
2. Utilizar recursos de *pthread* para sincronizar las hebras y proteger los recursos compartidos.
3. Practicar técnicas de documentación de programas.
4. Conocer y practicar uso de makefile para compilación de programas.

Las funciones de **threads** que se debe usar en este lab son:

1. `pthread_create()`
2. `pthread_attr_init()`
3. `pthread_join()`
4. `pthread_exit()`
5. `pthread_mutex_init()`
6. `pthread_mutex_lock()`
7. `pthread_mutex_unlock()`

Sin embargo, si usted cree necesario usar, además de éstas, otras funciones, lo puede hacer.

III. Enunciado

III.A. Agujeros negros, discos proto planetarios y ALMA

Hace dos años, los astrónomos lograron reconstruir la primera imagen del horizonte de eventos de un agujero negro. Esta frontera marca el lugar desde donde el cual ya nada puede salir del agujero negro, ni materia ni luz. Es por eso que la imagen muestra radiación fuera de la frontera y dentro de la frontera sólo se ve oscuridad. La figura 1 muestra la primera imagen del agujero negro M87:

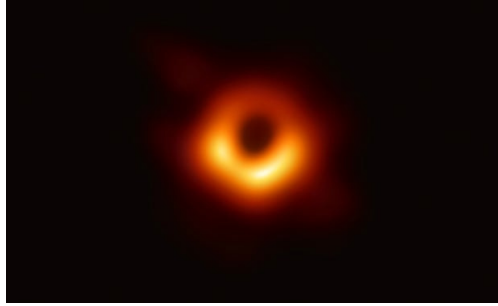


Figure 1. Primera imagen del agujero negro M87.

Esta imagen es el resultado de un proceso computacional llamado síntesis o reconstrucción de imágenes interferométricas. Es decir, las antenas no capturan la imagen que nosotros vemos, sino detectan señales de radio, las cuales son transformadas por un programa computacional en la imagen visual.

En estricto rigor los datos recolectados por las antenas corresponden a muestras del espacio Fourier de la imagen. Estas muestras están repartidas en forma no uniforme e irregular en el plano Fourier. La figura 2 muestra un típico patrón de muestreo del plano Fourier. La imagen de la derecha muestra la imagen reconstruida a partir de los datos. Esta imagen corresponde a un disco protoplanetario llamado HL Tau.

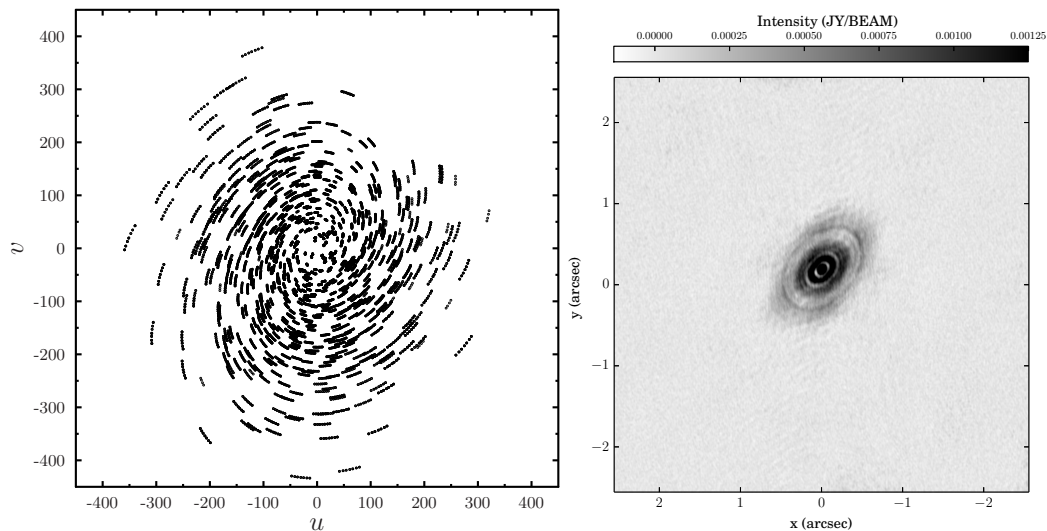


Figure 2. Plano (u, v) e imagen sintetizada de HL Tau.

En cada punto (u, v) se mide una señal llamada Visibilidad. Cada visibilidad es un número complejo, es decir posee una parte real y otra imaginaria. La imagen por otro lado es solo real.

Usaremos la siguiente notación para describir los datos:

- $V(u, v)$ es la visibilidad en la coordenada (u, v) del plano Fourier
- $V(u, v).r$ es la parte real de la visibilidad
- $V(u, v).i$ es la parte imaginaria de la visibilidad
- $V(u, v).w$ es el ruido en la coordenada (u, v)

La siguiente lista es un ejemplo de una pequeña muestra de visibilidades

```
46.75563,-160.447845,-0.014992,0.005196,0.005011
119.08387,-30.927526,0.016286,-0.001052,0.005888
-132.906616,58.374054,-0.009442,-0.001208,0.003696
-180.271179,-43.749226,-0.011654,0.001075,0.003039
```

-30.299767,-126.668739,0.015222,-0.004145,0.007097
-18.289482,28.76403,0.025931,0.001565,0.004362

La primera columna es la posición en el eje u ; la segunda es la posición en el eje v ; la tercera columna es el valor real de la visibilidad; la cuarta, el valor imaginario y finalmente, la quinta es el ruido. Estos datos corresponden a datos reales de un disco proto planetario captadas por el observatorio ALMA.

El número de visibilidades depende de cada captura de datos y del telescopio usado, pero puede variar entre varios cientos de miles de puntos hasta cientos de millones. Por lo tanto, el procesamiento de las visibilidades es una tarea computacionalmente costosa y generalmente se usa paralelismo para acelerar los procesos. En este lab, simularemos este proceso usando varios procesos que cooperan para hacer cálculo a las visibilidades. Por ahora, no haremos síntesis de imágenes.

III.B. Cálculo de propiedades

En ciertas aplicaciones, antes de sintetizar la imagen, es necesario y útil extraer características del plano Fourier. Uno de estos preprocesamiento, agrupa las visibilidades en anillos concéntricos con centro en el $(0,0)$ y de radios crecientes. Por ejemplo, suponga que definimos un radio r , el cual divide las visibilidades en dos anillos, aquellas visibilidades cuya distancia al centro es menor o igual a r , y aquellas cuya distancia al centro es mayor a r . La distancia de una visibilidad al centro es simplemente:

$$d(u,v) = (u^2 + v^2)^{1/2}$$

Si definiéramos dos radios R_1 y R_2 ($R_1 < R_2$), las visibilidades se dividirían en tres discos: $0 \leq d(u,v) < R_1$, $R_1 \leq d(u,v) < R_2$, y $R_2 \leq d(u,v)$.

Para este lab, las propiedades que se calcularán por disco son

1. Media real : $\frac{1}{N} \sum V(u,v).r$, donde N es el número de visibilidades en el disco
2. Media imaginaria : $\frac{1}{N} \sum V(u,v).i$, donde N es el número de visibilidades en el disco
3. Potencia: $\sum (V(u,v).r)^2 + (V(u,v).i)^2)^{1/2}$
4. Ruido total: $\sum V(u,v).w$

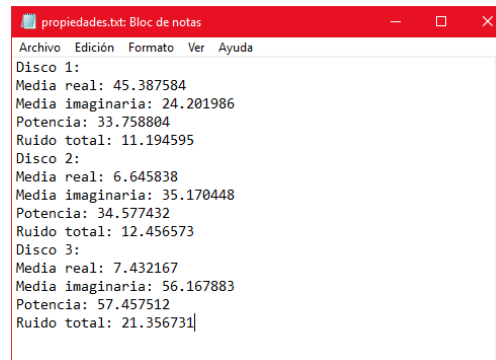
IV. El programa del lab

IV.A. Lógica de solución

En este laboratorio crearemos un proceso multihebreado que calcule las propiedades antes descritas. Usted debe organizar su solución de la siguiente manera.

1. El proceso principal (hebra madre) recibirá argumentos por línea de comando el número de radios en los que se debe dividir el plano de Fourier y el número de hebras hijas que realizarán el cálculo. Además recibirá el ancho Δ_r de cada intervalo, tal que $R_1 = \Delta_r$, $R_2 = 2\Delta_r$, etc. Este proceso también recibe como argumento el nombre del archivo de entrada, el nombre del archivo de salida y el número de líneas que cada hebra lee por vez del archivo de entrada. A esta última variable le llamamos **chunk**.
2. Note que el número de hebras hijas o trabajadoras no necesariamente es igual al número de anillos.
3. El archivo de entrada debe ser declarado como un descriptor global (fuera del main) de tal forma que sea un archivo compartido por todas las hebras hijas. También se puede declarar este archivo como un FILE stream global, en vez de un descriptor.
4. Antes de crear a las hebras hijas, la hebra madre debe abrir el archivo de entrada, pero no debe leer nada de él.
5. La hebra madre debe crear a las hebras hijas, y una vez creadas las hebras hijas, la hebra madre debe esperar a que todas ellas terminen su trabajo.

6. El cálculo de las propiedades del plano Fourier es realizado por las hebras hijas, en forma concurrente.
7. Cuando una hebra está libre, intenta leer **chunk** líneas del archivo de entrada. Este acceso debe ser exclusivo, es decir, solo una hebra hija puede estar leyendo el archivo a la vez.
8. Una vez que lee las líneas, la hebra actualiza las propiedades para cada anillo correspondiente, para lo cual existirá una o varias estructuras compartidas en la cual se registren las propiedades de cada anillo.
9. Note que la hebra lee puntos (u, v) que muy probablemente pertenezcan a diferentes anillos del plano, por lo que debe actualizar las sumas correspondientes a cada anillo. Es decir, cada hebra debe determinar a qué anillo pertenece cada punto que ha leído.
10. El acceso a estas estructuras compartidas también debe ser exclusivo. Note que es muy probable que varias hebras estén actualizando estas estructuras a la vez.
11. Además note que la actualización de propiedades de diferentes anillos no es exclusiva. Su solución debe permitir máxima concurrencia en la actualización de las propiedades.
12. Una vez que una hebra termine de actualizar las propiedades para los puntos leídos, vuelve a intentar leer **chunk** líneas del archivo de entrada. Esto lo realiza hasta que se hayan leído todas las líneas.
13. Como el número de hebras no necesariamente divide al número de líneas del archivo de entrada, se cumplirá el caso en que una hebra leerá menos líneas que **chunk** cuando llegue al final del archivo. También es posible que la hebra intente leer y se encuentre con fin de archivo. Usted debe crear un mecanismo para que cuando se alcance esta situación, todas la hebras hijas dejen de trabajar.
14. Una vez que todas las hebras hijas han finalizado, la hebra madre debe terminar de calcular las propiedades de cada anillo, dividiendo los valores calculados por los números de visibilidades encontradas en cada anillo. Este último valor también se calcula concurrentemente por las hebras hijas.
15. Finalmente, la hebra madre escribe los resultados con el mismo formato que el lab1.



```
Archivo  Edición  Formato  Ver  Ayuda
Disco 1:
Media real: 45.387584
Media imaginaria: 24.201986
Potencia: 33.758804
Ruido total: 11.194595
Disco 2:
Media real: 6.645838
Media imaginaria: 35.170448
Potencia: 34.577432
Ruido total: 12.456573
Disco 3:
Media real: 7.432167
Media imaginaria: 56.167883
Potencia: 57.457512
Ruido total: 21.356731
```

Figure 3. Ejemplo archivo de salida.

IV.B. Línea de comando

Los argumentos de línea de comando deben ser procesados usando `getopt()`. El programa se ejecutará usando los siguientes argumentos (ejemplo):

```
$ ./lab2 -i visibilidades.csv -o propiedades.txt -d ancho -n ndiscos -h numerohebras -c chunk -b
```

- **-i**: nombre de archivo con visibilidades
- **-o**: nombre de archivo de salida
- **-n**: cantidad de discos

- **-d**: ancho de cada disco.
- **-h**: número de hebras trabajadoras
- **-c: chunk**, número de líneas que la hebra lee por vez
- **-b**: bandera que permite indicar si se quiere ver por consola las propiedades calculadas. Es decir, si aparece la opción, entonces se muestra la salida por consola.

Ejemplo de ejecución:

```
>> ./lab2 -i uvplaneco65.csv -o propiedades.txt -d 100 -n 4 -h 4 -c 100
```

Usted debe además implementar chequeo de los argumentos tal que el programa siempre entregue mensajes de error cuando algunos o varios de los argumentos no son válidos.

Como requerimientos no funcionales, se exige lo siguiente:

- Debe funcionar en sistemas operativos con kernel Linux.
- Debe ser implementado en lenguaje de programación C y pthreads.
- Se debe utilizar un archivo Makefile para su compilación.
- Realizar el programa utilizando buenas prácticas, dado que este laboratorio no contiene manual de usuario ni informe, es necesario que todo esté debidamente comentado.
- Que el programa principal esté desacoplado, es decir, que se desarrollen las funciones correspondientes en otro archivo .c para mayor entendimiento de la ejecución.
- Se deben verificar y validar los parámetros de entrada.

V. Entregables

El laboratorio es en parejas o en forma individual y se descontará 1 punto por día de atraso. Se debe subir en un archivo comprimido a usachvirtual los siguientes entregables:

- **Makefile**: Archivo para make que compila el programa.
- **archivos .c y .h** Se debe tener como mínimo un archivo lab2.c principal con el main del programa. Se debe tener mínimo un archivo .h que tenga cabeceras de funciones, estructuras o datos globales. Se deben comentar todas las funciones de la forma:

```
//Entradas: explicar que se recibe
//Funcionamiento: explicar que hace
//Salidas: explicar que se retorna
```

- Trabajos con códigos que hayan sido copiados de un trabajo de otro grupo serán calificados con la nota mínima.

El archivo comprimido debe llamarse: RUTESTUDIANTE1_RUTESTUDIANTE2.zip

Ejemplo: 19689333k_186593220.zip

VI. Fecha de entrega

Jueves 30 de Junio.