

Sistemas Operativos 1/2022

Laboratorio 1

Profesores:

Cristóbal Acosta (cristobal.acosta@usach.cl)

Fernando Rannou (fernando.rannou@usach.cl)

Ayudante:

Ricardo Hasbun (ricardo.hasbun@usach.cl)

I. Objetivos Generales

Este laboratorio tiene como objetivo aplicar los conceptos de creación de procesos, y comunicación entre estos a través de pipes. La aplicación debe ser escrita en el lenguaje de programación C sobre sistema operativo Linux.

II. Objetivos Específicos

1. Conocer y usar las funcionalidades de `getopt()` como método de recepción de parámetros de entradas.
2. Construir funciones de lectura y escritura de archivos .csv usando `fopen()`, `fread()`, y `fwrite()`.
3. Construir una función que permita la búsqueda de puntos en un radio.
4. Practicar técnicas de documentación de programas.
5. Conocer y practicar uso de makefile para compilación de programas.
6. Crear procesos a través del uso de `fork()`.
7. Ejecutar programas usando alguna de los llamados de la familia `exec()`
8. Comunicar procesos mediante `pipes`.

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:

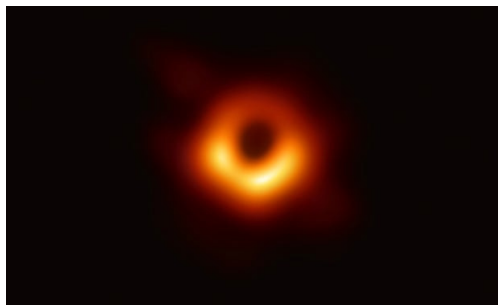


Figure 1. Primera imagen del agujero negro M87.

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 HLTau.

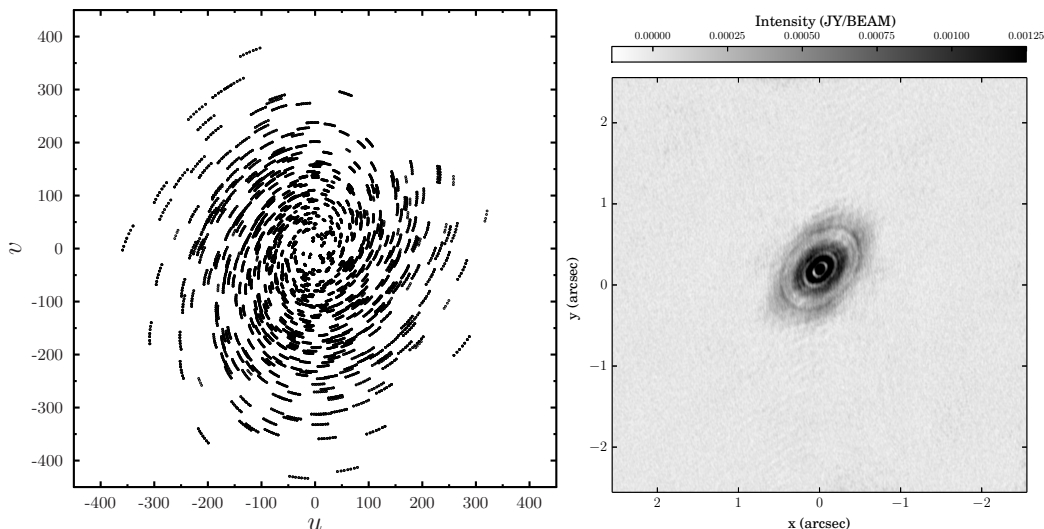


Figure 2. Plano (u, v) e imagen sintetizada de HLTau.

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 esta laboratorio crearemos un conjunto de procesos que calculen las propiedades antes descritas. Usted debe organizar su solución de la siguiente manera.

1. El proceso principal recibirá como argumentos por línea de comando el número de radios en los que se debe dividir el plano de Fourier. Además recibirá el ancho Δ_r de cada intervalo, tal que $R_1 = \Delta_r$, $R_2 = 2\Delta_r$, etc. El proceso también recibe como argumento el nombre del archivo de entrada, y el nombre del archivo de salida.
2. El proceso padre creará tantos pipes e hijos necesarios para establecer comunicación bidireccional con los procesos hijos. El número de procesos a crear es igual al número de discos.
3. El padre leerá línea a línea el archivo de entrada en formato .csv y determinará el disco al que pertenece la visibilidad, y enviará los datos correspondiente al proceso hijo que se encarga de procesar dicho disco.

4. Cada proceso hijo leerá del pipe las visibilidades que le envía el padre y las usará para calcular las propiedades.
5. Cuando el padre termine de leer y enviar todas las visibilidades, enviará un mensaje de FIN a los hijos, para que estos salgan de su ciclo de lectura y procesamiento.
6. Una vez que llega el aviso de FIN, cada proceso hijo enviará al proceso padre los resultados de sus cálculos, y terminará su ejecución.
7. El padre lee los resultados de cada pipe e imprime en el archivo de salida los resultados por disco.
8. El proceso padre cierra todos los archivos y termina su ejecución.

Para implementar esta lógica se debe crear dos programas. El primero será **lab1** el cual será el proceso padre y por lo tanto se encargará de las funciones que le corresponde (previamente descritas), como por ejemplo crear los procesos hijos mediante la función **fork()**.

Por otro lado, el segundo programa será **vis** el cual procesa las visibilidades. Este proceso debe ser ejecutado por los procesos hijos, mediante el llamado al sistema **execve()**. Es importante recalcar que ambos programas deben ser compilados independientemente, es decir **vis** no contiene funciones invocadas desde **lab1**. En estricto rigor, sería posible ejecutar ambos programas en forma independiente.

El archivo que debe escribir el proceso padre con las propiedades calculadas por sus hijos, debe tener el siguiente formato:



```
propiedades.txt: Bloc de notas
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.

En donde el nombre del archivo es el nombre entregado como argumento (en este caso: propiedades.txt), además se incluye todas las propiedades que cada hijo calculó en su disco asignado según el ancho y el número de discos también ingresados como argumentos (en este caso: 3 discos).

IV.B. Línea de comando

El programa se ejecutará usando los siguientes argumentos (ejemplo):

```
$ ./lab1 -i visibilidades.csv -o propiedades.txt -d ancho -n ndiscos -b
```

- **-i**: nombre de archivo con visibilidades
- **-o**: nombre de archivo de salida
- **-n**: cantidad de discos
- **-d**: ancho de cada disco.

- **-b**: bandera o flag que permite indicar si se quiere ver por consola la cantidad de visibilidades encontradas por cada proceso hijo. Es decir, si se indica el flag, entonces se muestra la salida por consola.

Ejemplo de compilación y ejecución:

```
>> gcc lab1.c -o lab1
>> gcc vis.c -o vis
>> ./lab1 -i uvplaneco65.csv -o propiedades.txt -d 100 -n 4 -b
```

En el primer caso, se ejecuta el programa entregando como nombre de archivo de entrada **uvplaneco65.csv**, además el proceso principal deberá crear 4 hijos y cada hijo recibirá los datos que le envíe el proceso padre. Las visibilidades se repartirán entre los siguientes rangos: [0, 100), [100, 200), [200, 300), [300,). Finalmente, debido a que el primer caso considera el flag **-b**, cada hijo debe identificarse y mostrar la cantidad de visibilidades que procesó. Ejemplo:

```
Soy el hijo de pid 7502, procesé 10000 visibilidades
Soy el hijo de pid 7503, procesé 14000 visibilidades
...
```

Como requerimientos no funcionales, se exige lo siguiente:

- Debe funcionar en sistemas operativos con kernel Linux.
- Debe ser implementado en lenguaje de programación C.
- Se debe utilizar un archivo Makefile para compilar cada programa o un solo archivo Makefile con dos targets.
- 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.

V. Entregables

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

- **Makefile**: Archivo para compilar los programas o **makefile1** y **makefile2** para compilar individualmente.
- **lab1.c**: archivo con el código del proceso padre. Puede incluir otros archivos fuente. Recuerde que todas las funciones deben estar comentadas, explicadas de forma entendible especificando sus entradas, funcionamiento y salida. Si una función no está explicada se bajará puntaje. Se deben comentar todas las funciones de la siguiente forma:

```
//Entradas: explicar qué se recibe
//Salidas: explicar qué se retorna
//Descripción: explicar qué hace
```

- **Vis.c**: archivo con el código del proceso hijo. Puede incluir otros archivos fuente. Recuerde que todas las funciones deben estar comentadas, explicadas de forma entendible especificando sus entradas, funcionamiento y salida. Si una función no está explicada se bajará puntaje.
- 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

NOTA: Si los laboratorios son en parejas, pueden ser de distintas secciones. Si es así, por favor avisar al ayudante.

VI. Fecha de entrega

Jueves 05 de Mayo, 2022.