Sistemas Operativos 2/2023
Laboratorio 2

Profesores:
Cristóbal Acosta (cristobal.acosta@usach.cl)
Fernando Rannou (fernando.rannou@usach.cl)
Ayudantes:
Ricardo Hasbun (ricardo.hasbun@usach.cl)
Leo Vergara (leo.vergara@usach.cl)

I. Objetivos Generales

Este laboratorio tiene como objetivo aplicar técnicas de programación imperativa mediante lenguaje C, como la recepción de parámetros mediante `getopt` y compilación mediante `Makefile` sobre sistema operativo Linux. Además de aplicar conocimiento adquiridos en cátedra sobre la creación de procesos y la comunicación entre ellos, en un sistema operativo Linux. Para lograr esto, el estudiante debe hacer uso del llamado al sistema `fork()` junto a las funciones `pipe()`, funciones de la familia `exec()` y `dup2()` o `dup()`.

II. Objetivos Específicos

1. Usar las funcionalidades de `getopt()` como método de recepción de parámetros de entradas.
2. Uso del Makefile para compilación por partes de programas.
3. Crear procesos utilizando la función `fork()`.
4. Utilizar `pipe()` para la comunicación entre procesos.
5. Hacer uso de la familia `exec()` para ejecutar procesos.
6. Duplicar descriptores con `dup()` o `dup2()`.
7. Construir funciones de lectura y escritura de archivos.
8. Practicar técnicas de documentación de programas.

III. Enunciado

III.A. El bombardeo de partículas

En este laboratorio, se busca simular la energía depositada en un material por partículas de alta energía que lo impactan (ejemplo: partículas en un satélite). El programa a desarrollar calculará y registrará la energía en Joules que cada partícula va depositando en algún punto del material.

Para facilitar el trabajo, se utilizará un modelo de una dimensión (1D). Entonces, se usará un arreglo para llevar registro de las energías acumuladas.

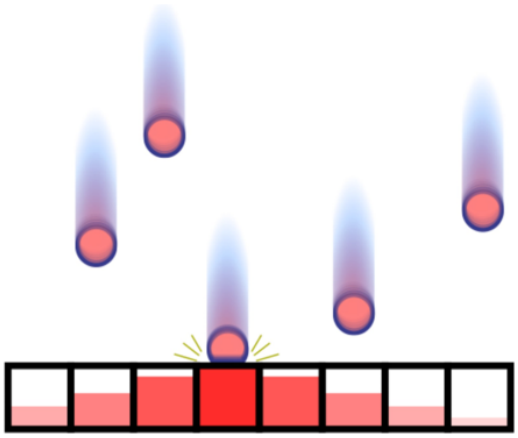

Figure 1. Acumulación de energías por partículas.

Por cada partícula simulada, se entrega su posición de impacto j y la energía potencial E_j que trae. La energía se deposita en la celda j y las celdas vecinas según la siguiente ecuación:

$$E_i = E_i + E_{j,i}$$

donde

$$E_{j,i} = \frac{10^3 \times E_j}{N\sqrt{|j-i|+1}}$$

donde N es el número de celdas del material. Note que es posible que la posición de impacto sea más grande que N. Esto es válido, y significa que aunque la partícula impactó fuera del material medido, su energía igualmente se registra en la zona de interés.

En caso de que la energía depositada sea muy pequeña, esta no se registrará, para lo cual se utilizará un umbral que se calcula como `MIN_ENERGY` $= 10^{-3}/N$, por lo que se deposita energía solo en el caso que $E_{j,i} \geq$ `MIN_ENERGY`.

III.B. Los archivos de entrada y de salida

Un bombardeo de partículas se especifica en un archivo de entrada, el cual contiene el número de partículas, y para cada partícula, la posición de impacto y la energía (E_j). El siguiente es un ejemplo de un archivo de bombardeo:

```
6
4 81
8 10
10 100
7 35
11 12
5 8
```

La primera línea indica la cantidad de partículas y en las siguentes entrega la descripción de cada una de ellas. Es importante destacar que en este archivo `NO` se especifica el número de celdas, ya que este parámetro se ha ingresado por argumento de línea de comando.

Para el archivo de salida, después de haber impactado todas las partículas, se debe escribir en la primera línea la posición del material con máxima energía y la cantidad acumulada. En las siguientes líneas, se debe indicar cada posición en orden con su respectiva energía acumulada (se deben escribir todas las celdas del material). Por ejemplo, haciendo uso de la entrada anterior, teniendo un material con 35 celdas, el resultado sería el siguiente, donde se logra ver en la celda 10 la máxima energía y luego la energía acumulada en todas las posiciones del material. Por ejemplo:

```
10 4732.568359
0  2537.520752
1  2745.225830
2  3027.479004
3  3456.707031
...
```

IV. El programa

IV.A. Lógica de solución

En este laboratorio se simulará el impacto de partículas en un material, que con base en la ecuación (ver sección III) y un archivo de texto, calcule cuál es la celda con mayor energía depositada.

1. El proceso principal (lab2.c) recibirá como argumentos por línea de comando el nombre del archivo de entrada, el nombre del archivo de salida, la cantidad de workers que serán generados y el número de chunks con el cuál se trabajara.

2. El proceso principal validará los datos entregados por pantalla y, una vez validados, ejecutará el proceso broker con `fork()` y algún miembro de la familia `exec()` entregándole los elementos ya validados.

3. El broker recibirá los argumentos del proceso padre y los utilizara para las tareas necesarias.

4. El broker creará la misma cantidad de workers que la ingresada por pantalla.

5. El broker ejecutara los procesos worker utilizando algún miembro de la familia `exec()` y se comunicará con ellos mediante el uso de `pipes`

6. El broker leerá el archivo de entrada por *chunks* líneas y mediante un randomizer elegirá a que worker corresponden dichas líneas.

7. Cuando no queden líneas por leer, el broker avisará a cada uno de los workers con la palabra "FIN", indicando que no debe esperar más líneas del broker.

8. Cada worker se encargará de obtener y procesar las líneas que entregue el broker.

9. Cuando un worker recibe una línea este se encargará de realizar el `calculo necesario` para dicha línea los cuales deben ser almacenados en caso de que lo amerite. La estructura que se use para almacenar estos resultados queda a elección del desarrollador.

10. Cuando un worker lea la palabra clave "FIN" este deberá dejar de esperar líneas del broker y enviar a este último la cantidad de líneas que procesó durante su ejecución. Luego, se finaliza su proceso.

11. Una vez que todos los workers hayan terminado sus cálculos, el broker retomará el control y escribirá en el archivo de salida los resultados siguiendo el formato entregado. Si se utiliza la

bandera -D, estos resultados deberán además aparecer en la salida estándar (stdout) del terminal del Sistema Operativo incluyendo con ellos la cantidad de líneas que procesó cada worker (basta con mostrar el PID del worker o con una ID arbitraria asignada al mismo). Con esto, el padre finaliza su ejecución.

12. Tanto proceso padre, proceso broker y proceso worker debe encontrarse finalizado a la hora de terminar el programa.

IV.B. Línea de comando

La ejecución del programa tendrá los siguientes parámetros que deben ser procesados por `getopt()`:

- `-N`: es el número de celdas
- `-P`: es el número de procesos (workers)
- `-i`: es el archivo con el bombardeo (archivo de entrada)
- `-o`: es el archivo de salida
- `-c`: es el número de chuck, es decir, cantidad de líneas a leer por chunk.
- `-D`: bandera o flag que permite indicar si se quiere ver por consola la cantidad de celdas de material.

Por ejemplo:

```
$ ./lab2 -N 5 -P 3 -i input.txt -o output.txt -c 10 -D
```

En el caso que se utilice la flag -D se debe imprimir por consola un gráfico simple y normalizado con las energías. Por ejemplo, utilizando un N=10 y la flag -D presente, se obtiene:

```
0   8881.3223 |oooooooooooooooooooooooooooooooooooooooooooooooooo
1   9608.2910 |ooooooooooooooooooooooooooooooooooooooooooooooooooo
2  10596.1777 |ooooooooooooooooooooooooooooooooooooooooooooooooooooo
3  12098.4746 |oooooooooooooooooooooooooooooooooooooooooooooooooooooooo
4  15066.8076 |ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
5  13584.3311 |oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
6  13256.4805 |ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
7  14255.6436 |ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
8  13870.8066 |ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
9  14156.3018 |ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
```

IV.C. Requerimientos

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 los distintos 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.
- Los programas se encuentren desacoplados, es decir, que se desarrollen las funciones correspondientes en otro archivo .c para mayor entendimiento de la ejecución.
- La solución debe implementar `fork()`, `excec()`, `dup2()` y `pipe()`. De lo contrario se considerara invalido.

V. Entregables

El laboratorio es en parejas. Si se elije una pareja está no podrá ser cambiada durante el semestre. Se descontará 1 punto (de nota) por día de atraso con un máximo de tres días, a contar del cuarto se evaluará con nota mínima. Debe subir en un archivo comprimido ZIP (una carpeta) a USACH virtual con los siguientes entregables:

- `Makefile`: Archivo para compilar los programas.
- `lab2.c`: Archivo principal del laboratorio, contiene el main.
- `broker.c`: Archivo que contiene el main del proceso broker.
- `worker.c`: Archivo que contiene el main del proceso worker.
- `fworker.c y fworker.h`: Archivo con funciones para el worker, en el .c el desarrollo de la función y en el .h las cabeceras. 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.
- `fbroker.c y fbroker.h`: Archivo con funciones para el broker, en el .c el desarrollo de la función y en el .h las cabeceras. 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.
- `Otros:` Cualquier otro archivo fuente que se vea necesario para la realización del lab. Pueden ser archivos con funciones, estructuras de datos, etc.
- Trabajos con códigos que hayan sido copiados de un trabajo de otro grupo serán calificados con la nota mínima.

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
```

El archivo comprimido (al igual que la carpeta) debe llamarse: RUTESTUDIANTE1_RUTESTUDIANTE2.zip

Ejemplo 1: 19689333k_186593220.zip

NOTA 1: El archivo debe ser subido a uvirtual en el apartado "Entrega Lab2".

NOTA 2: Cualquier diferencia en el formato del laboratorio que es entregado en este documento, significara un descuento de puntos.

NOTA 3: SOLO UN ESTUDIANTE DEBE SUBIR EL LABORATORIO.

NOTA 4: En caso de solicitar corrección del lab está será en los computadores del Diinf, es decir, si funciona en eso computadores no hay problema.

NOTA 5: Cualquier comprimido que no siga el ejemplo 1, significara un descuento de 1 punto de nota.

VI. Fecha de entrega

Jueves 09 de Noviembre, 2023.