Sistemas Operativos 1/2024
Laboratorio 2

Profesores:
Cristóbal Acosta (cristobal.acosta@usach.cl)
Fernando Rannou (fernando.rannou@usach.cl)
Ayudantes:
Ricardo Hasbun (ricardo.hasbun@usach.cl)
Ian Rickmers (ian.rickmers@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 entrada.
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

Para la obtención de imágenes astronómicas de calidad, capturadas por un telescopio, llegan a un laboratorio de procesamiento de datos, donde un equipo de expertos se encarga de analizar y extraer información relevante de cada imagen. Con eso en mente, es que surge la necesidad de desarrollar una aplicación informática especializada en el procesamiento de imágenes astronómicas. Esta aplicación diseñada por el equipo de científicos y programadores de la institución, se convierte en una herramienta para agilizar el proceso de análisis de imágenes.

III.A. Procesamiento de Imágenes

Mediante la construcción de una aplicación simple de procesamiento de imágenes. La aplicación debe ser escrita en lenguaje de programación C sobre sistema operativo Linux. Este laboratorio servirá de base a los siguientes laboratorios, los cuales irán paulatinamente incluyendo el uso de servicios del sistema operativo para realizar la misma función que la aplicación original. Es por eso que es de vital importancia que el diseño y modularización de esta aplicación quede bien definida a partir del laboratorio 1, para que luego sólo nos preocupemos de incluir las nuevas funcionalidades asociadas a procesos, hebras, entre otros.

III.A.1. Los archivos de entrada y de salida

La aplicación consiste en un pipeline de procesamiento de imágenes astronómicas. Cada imagen pasará por tres etapas de procesamiento tal que al final del pipeline se clasifique la imagen como satisfaciendo o no alguna condición a definir. El programa procesará varias imágenes, una a la vez.

Las etapas del pipeline son:

1. Lectura de las imágenes.
2. Saturación de la imagen.
3. Conversión a imagen en escala de grises.
4. Binarización de imagen.
5. Análisis de propiedad.
6. Escritura de resultados.

Donde se destaca, la implementación de los filtros, los cuales van en el siguiente orden, primero la saturación de la imagen, luego, la escala de grises de la imagen y por último la binarización de la imagen.

III.B. Lectura de imágenes

Las imágenes estarán almacenadas en binario con formato bmp.

III.C. Saturación

En una imagen RGB, cada píxel de la imagen está representado por tres números, que representan el componente de color rojo (Red), el de color verde (Green) y el de color azul (Blue). Para saturar a una imagen, se utiliza un factor que multiplique el valor RGB de cada píxel, esto sería de esta manera:

$$R = R * factor \tag{1}$$

$$G = G * factor \tag{2}$$

$$B = B * factor \tag{3}$$

Cabe destacar que el valor RGB de cada píxel no puede superar el valor 255. Este proceso se aprecia en la figura 1, que muestra la saturación de la imagen.

Figure 1. Saturación de imagen.

III.D. Conversión de RGB a escala de grises

En una imagen RGB, cada píxel de la imagen está representado por tres números, que representan el componente de color rojo (Red), el de color verde (Green) y el de color azul (Blue). Para convertir una imagen a escala de grises, se utiliza la siguiente ecuación de luminiscencia:

$$Y = R * 0.3 + G * 0.59 + B * 0.11 \tag{4}$$

donde R, G, y B son los componentes rojo, verde y azul, respectivamente. Así como se muestra en la figura 2.

III.E. Binarización de una imagen

Para binarizar la imagen basta con definir un umbral, el cual define cuáles píxeles deben ser transformados a blanco y cuáles a negro, de la siguiente manera. Para cada píxel de la imagen, hacer:

```
si el pixel > UMBRAL
    p

xel = 1
sino
    p xel = 0
```

Los valores 0 y 1 no son representados por un bit sino son valores enteros (`int`). Al aplicar el algoritmo anterior, obtendremos la imagen binarizada. El valor de umbral tiene que ser entre 0 y 1. Resultando en lo que se ve en la figura 3.

Figure 2. Escala de Grises.

Figure 3. Binarización.

III.F. Clasificación

Se debe crear una función que concluya si la imagen es *nearly black* (casi negra). Esta característica es típica de muchas imágenes astronómicas donde una pequeña fuente puntual de luz está rodeada de vacío u oscuridad. Entonces, si el porcentaje de píxeles negros es mayor o igual a un cierto umbral la imagen es clasificada como nearly black. El valor de umbral tiene que ser entre 0 y 1.

III.G. Resultados

Se deben crear 2 resultados finales:

- Carpeta con las imágenes resultantes con los filtros.
- Archivo CSV con los resultados de la clasificación, este archivo tiene que tener 2 columnas, una con el nombre de cada imagen procesada y otra con su respectiva clasificación, representada con un 1 o un 0 dependiendo de si es clasificada como nearly black o no.

III.H. Línea de comando

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

- -f: cantidad de filtros a aplicar.
- -p: factor de saturación del filtro.
- -u: UMBRAL para binarizar la imagen.
- -v: UMBRAL para clasificación.

- -W: Cantidad de workers a crear.
- -C: nombre de la carpeta resultante con las imágenes, con los filtros aplicados.
- -R: nombre del archivo CSV con las clasificaciones resultantes.

```
$ ./lab2 -N imagen -f 2 -p 20 -C Ejemplo_nombre -R Nombre_resultado -W 5
```

Cabe destacar que, en la cantidad de filtros a aplicar (-f), se refiere a, como se explica anteriormente, el orden en el que se aplican los filtros. Es decir, si se ingresa un -f 2, se deben aplicar solo los dos primeros filtros.

Si no se le da un valor a la bandera -f, se deben aplicar los tres filtros.

Si no se da un valor para cualquiera de los umbrales de la clasificación, este debe tomar el valor 0.5 por defecto.

Las banderas -C y -R son obligatorias, es decir, siempre debe ir un nombre de la carpeta y del archivo resultante.

Si no se da un valor para el factor de saturación, este debe tomar el valor 1.3 por defecto.

Si no se le asigna un valor a los filtros no obligatorios, no se debe escribir la bandera.

III.I. 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.

IV. Lógica de la solución

1. El proceso principal (lab2.c) recibirá como argumentos los datos de la línea de comando, la cual se explica más adelante (mismos parámetros del laboratorio 1). Sumado a la línea de comando se agrega la cantidad de workers que serán generados.
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 utilizará para las tareas necesarias.
4. El broker creará la misma cantidad de workers que la ingresada por pantalla.
5. El broker ejecutará los procesos worker utilizando algún miembro de la familia `exec()` y se comunicará con ellos mediante el uso de `pipes`.
6. Los workers serán los encargados de aplicar los filtros, es decir, aplicar la saturación, conversión a escala de grises y la binarización.
7. Para la aplicación de los filtros por parte de los workers, el broker dividirá usando el módulo (%) la imagen de tamaño NXM, tomando solo la columna, en la cantidad de workers creados, es decir, M% cantidad de workers. Donde el resto de la división se le deberá enviar al último worker.

8. Luego de que los workers terminen su trabajo, estos le enviarán los fragmentos de la imagen al broker, donde este volverá a ensamblar la imagen con los filtros ya aplicados.
9. Finalmente, será el proceso Padre el que realice la evaluación final. Además de juntar las imágenes y generar los archivos de salida.

V. Entregables

El laboratorio es en parejas. Si se elige 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, significará un descuento de puntos.
NOTA 3: SOLO UN ESTUDIANTE DEBE SUBIR EL LABORATORIO.
NOTA 4: En caso de solicitar corrección del lab, esta será en los computadores del Diinf, es decir, si funciona en esos computadores no hay problema.
NOTA 5: Cualquier comprimido que no siga el ejemplo 1, significará un descuento de 1 punto de nota.

VI. Fecha de entrega

Jueves 24 de Junio, 2024.