

Sistemas Operativos 2/2022

Laboratorio 2

Profesores:

Cristóbal Acosta (cristobal.acosta@usach.cl)
Fernando Rannou (fernando.rannou@usach.cl)

Ayudantes:

Ricardo Hasbun (ricardo.hasbun@usach.cl)
Matias Silva (matias.silva.al@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 un sistema operativo Linux.

II. Objetivos Específicos

1. Validar los parámetros recibidos con `getopt()`.
2. Construir funciones de lectura y escritura de archivos.
3. Comunicación de procesos mediante pipes (`pipe()`).
4. Crear procesos hijos a través del uso de `fork()`.
5. Hacer uso de la familia de funciones `exec()`. Para ejecutar distintos procesos.
6. Utilizar `dup()` o `dup2()` para duplicar descriptores.
7. Conocer y practicar uso de makefile para compilación por partes de programas.

III. Enunciado

III.A. Steam, plataforma digital de distribución de videojuegos

Steam es una plataforma digital de distribución de videojuegos, lanzada al público por Valve por primera vez en el año 2003 pensada originalmente para proveer de actualizaciones de los juegos de Valve hacia sus jugadores. Actualmente se encuentra ampliada para incluir ventas de sus propios juegos y de terceros. Esta plataforma ofrece distintos beneficios para las distribuidoras de juego como lo es: protección contra la piratería, servidores de emparejamiento, transmisiones de vídeo, venta de vídeo juegos, comunidades, etc.



Figure 1. Logo de plataforma Steam.

Steam almacena la información de sus videojuegos en distintas bases de datos. Luego, usando APIs internas, Steam despliega información sobre los distintos videojuegos en su plataforma. Para esto, existen puntos clave como el ID o el nombre del juego, estos son visibles en las mismas URLs de la tienda, por ejemplo: <https://store.steampowered.com/app/1237970/Titanfall2/> la cual tiene la ID de juego 1237970 y el nombre de TitanFall 2.

Para esta experiencia, se ha generado un proceso computacional el cual permite obtener información sobre los videojuegos que existen en la plataforma, gracias a las APIs otorgadas por Steam. Donde, con el uso de la primera API, la de desarrolladores (para más información: <https://steamcommunity.com/dev>) con la cual se pueden obtener las ID'S de cada juego. Teniendo esto, con la API de la tienda de Steam se procede a buscar la información específica de cada juego.

Los datos recolectados por el proceso generan una línea de texto en base a el archivo JSON del respectivo juego. El JSON de cada videojuego se puede ver en el siguiente código la cual es a posterior pasada a formato de texto.

```
1 {
2   "996970": {
3     "data": {
4       "is_free": true,
5       "name": "Far Cry\u00ae New Dawn - HD Texture Pack",
6       "platforms": {
7         "linux": false,
8         "mac": false,
9         "windows": true
10      },
11      "release_date": {
12        "coming_soon": false,
13        "date": "15 Feb, 2019"
14      },
15      "required_age": 0,
16    }
17  }
18 }
```

La siguiente lista es un ejemplo de una pequeña muestra de videojuegos:

```
980830,Spirit Hunter: Death Mark,18,50.0,False,2019,False,Yes,No,No
980880,Twifold,0,60.0,False,2018,False,Yes,No,No
980930,Power Solitaire VR Premium Upgrade,0,20.0,False,2018,False,Yes,No,No
980940,My Little Blacksmith Shop,0,15.0,False,2019,False,Yes,No,No
```

Cada fila tiene la siguiente descripción:

ID, Nombre, Si tiene restricción de edad (≤ 18 años), precio del juego (en dólares), si va a salir próximamente, fecha de lanzamiento, es gratis, disponible para windows, disponible para mac, disponible para linux.

III.B. Cálculos necesarios

Los cálculos necesarios para el desarrollo de este laboratorio son los siguientes:

- (a) Obtener la cantidad de juegos por año
- (b) Obtener el juego más caro, el juego más barato y el promedio de los precios del año.
- (c) Obtener el porcentaje de juegos que tiene cada plataforma por año.
- (d) Obtener todos los juegos gratis de un año.

IV. El programa del lab

IV.A. Lógica de solución

En este laboratorio crearemos un conjunto de procesos que calculen distintos valores antes descritos. Se debe organizar la solución de la siguiente manera.

1. El proceso principal recibirá como argumentos por línea de comando el nombre del archivo de entrada, el nombre del archivo de salida, el año en la que se debe iniciar a buscar los juegos, el precio mínimo para buscar los mismos y 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 principal 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. El broker leerá el archivo de entrada línea por línea y mediante un randomizer elegirá a que worker corresponde dicha línea.
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 cada cálculo necesario para dicha línea los cuales deben ser almacenados en caso de que lo amerite (ej: se encuentra un juego más caro que el anterior para el año 2018). La estructura que se use para almacenar estos resultados queda a elección del desarrollador.
10. Todos los workers, incluso aquellos que no lean ninguna línea (puede ocurrir debido al randomizer) deberán mantener la cantidad de líneas que ha procesado en todo momento. Estas serán impresas en el caso de que se use la bandera -b.
11. 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.
12. 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 -b, 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.
13. Tanto proceso principal, proceso broker y proceso worker debe encontrarse finalizado a la hora de terminar el programa.

Para implementar esta lógica se deben generar los siguientes 3 programas independientes.

- (a) lab2: Programa principal, el cual obtendrá los argumentos, los validará y los enviará al proceso broker.
- (b) broker: Programa broker, el cual realizará todas las funciones del broker expresadas en los puntos de la Lógica de la solución.
- (c) worker: Programa worker, el cual realizará todas las funciones de los workers expresadas en los puntos de la Lógica de la solución.

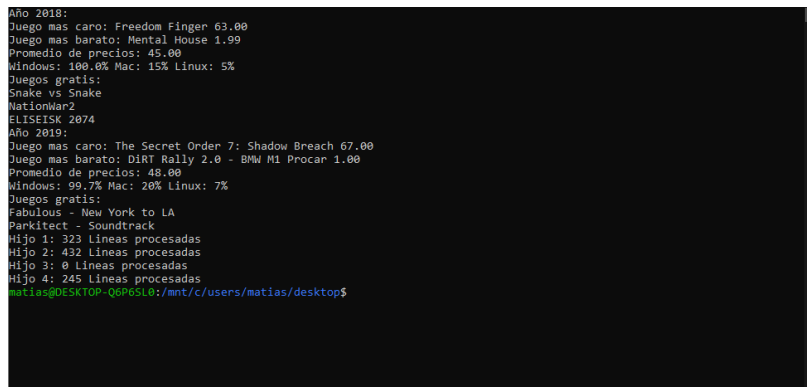
El archivo que debe escribir el proceso padre con las propiedades calculadas por sus workers debe tener el siguiente formato:

En caso de usar la flag -b, por consola se debe mostrar el siguiente formato:



```
salida.txt Bloc de notas
Archivo Edición Formato Ver Ayuda
Año 2018:
Juego mas caro: Freedom Finger 63.00
Juego mas barato: Mental House 1.99
Promedio de precios: 45.00
Windows: 100.0% Mac: 15% Linux: 5%
Juegos gratis:
Snake vs Snake
NationWar2
ELISEISK 2074
Año 2019:
Juego mas caro: The Secret Order 7: Shadow Breach 67.00
Juego mas barato: DiRT Rally 2.0 - BMW M1 Procar 1.00
Promedio de precios: 48.00
Windows: 99.7% Mac: 20% Linux: 7%
Juegos gratis:
Fabulous - New York to LA
Parkitect - Soundtrack
```

Figure 2. Ejemplo archivo de salida.



```
Año 2018:
Juego mas caro: Freedom Finger 63.00
Juego mas barato: Mental House 1.99
Promedio de precios: 45.00
Windows: 100.0% Mac: 15% Linux: 5%
Juegos gratis:
Snake vs Snake
NationWar2
ELISEISK 2074
Año 2019:
Juego mas caro: The Secret Order 7: Shadow Breach 67.00
Juego mas barato: DiRT Rally 2.0 - BMW M1 Procar 1.00
Promedio de precios: 48.00
Windows: 99.7% Mac: 20% Linux: 7%
Juegos gratis:
Fabulous - New York to LA
Parkitect - Soundtrack
Hijo 1: 323 líneas procesadas
Hijo 2: 432 líneas procesadas
Hijo 3: 0 líneas procesadas
Hijo 4: 245 líneas procesadas
matias@DESKTOP-Q6P6SL0:/mnt/c/users/matias/desktop$
```

Figure 3. Ejemplo de salida en la terminal usando el flag -b.

IV.B. Línea de comando

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

```
$ ./lab2 -i datos_juegos.csv -o salida.txt -d 2018 -p 20.90 -n 5 -b
```

- **-i:** nombre del archivo de entrada.
- **-o:** nombre de archivo de salida.
- **-d:** año de inicio del juego.
- **-p:** precio mínimo del juego (está en dólares).
- **-n:** cantidad de workers a generar.
- **-b:** bandera o flag que permite indicar si se quiere ver por consola la cantidad de juegos encontradas por cada proceso worker. Es decir, si se indica el flag, entonces se muestra la salida por consola.

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.
- Los distintos programas deben ser distintos ejecutables. Incluir por ejemplo el proceso broker en lab2 se considerará incorrecto.

V. Entregables

El laboratorio puede ser en parejas o de forma individual. 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 con el código principal del programa, se concentra en obtener y validar los datos entregados como argumentos del programa. Si los datos son validos, debe ejecutar el proceso padre.
- fbroker.c y fbroker.h/fworker.c y fworker.h: Archivo con funciones, 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.
- broker.c: archivo con el código del proceso broker. Debe utilizar las funciones de fbroker y 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.
- worker.c: archivo con el código del proceso worker. Debe utilizar las funciones de fworker y 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.
- 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: 19689333k_186593220.zip

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

NOTA2: Cualquier diferencia en el formato del laboratorio que es entregado en este documento, significara un descuento de puntos.

NOTA3: SOLO UN ESTUDIANTE DEBE SUBIR EL LABORATORIO.

VI. Fecha de entrega

Domingo 30 de Octubre, 2022.