Sistemas Operativos 1/2025
Laboratorio 2: Desafío de Comunicación y Elección de Líderes

Profesores:
Cristóbal Acosta (cristobal.acosta@usach.cl)
Fernando Rannou (fernando.rannou@usach.cl)
Miguel Cárcamo (miguel.carcamo@usach.cl)
Ayudantes:
Ricardo Hasbun (ricardo.hasbun@usach.cl)
Daniel Calderón (daniel.calderon.r@usach.cl)

May 7, 2025

1 Introducción

En el desafío anterior se implementó la comunicación entre procesos por medio de señales en Linux. Ahora, para esta segunda experiencia, se utilizará el mismo contexto, pero usando todas las herramientas vistas en cátedra relacionadas a procesos, es decir, `fork()`, `pipe()`, funciones de la familia de `exec()` y `dup()`/`dup2()`.

Para esta segunda versión del desafío, los estudiantes deberán volver a implementar funcionalidades de `getopt()`, como método de recepción de parámetros de entrada, hacer uso de archivo Makefile para la compilación por partes de los programas, creación de procesos usando la función `fork()`. Sin embargo, deberán emplear `pipe()` para establecer la comunicación entre los procesos.

Hacer uso de algún miembro de la familia de funciones `exec()` para ejecutar procesos. Duplicar los descriptores ya sea con `dup()` o `dup2()`. Y finalmente practicar técnicas de documentación de programas.

2 Descripción del Desafío

El desafío del laboratorio consiste en implementar un programa en C donde:

1. El proceso principal crea un conjunto de procesos hijos organizados en forma de anillo, donde se especificará la cantidad de hijos a crear por medio de la línea de comando con la opción `-p`.

2. Se inicializa un **token** (un número entero) con un valor especificado desde la línea de comandos mediante la opción `-t`.

3. El primer proceso en el anillo, inicia el desafío decrementando el **token** en un valor aleatorio (obtenido en el rango de 0 a $M-1$, donde M se pasa como argumento con la opción `-M`).

4. Cada proceso, al recibir el **token**, realiza lo siguiente:

   (a) Resta un valor aleatorio (entre 0 y $M-1$) al **token**.

   (b) Si el **token** resultante es mayor o igual a cero, lo envía al siguiente proceso activo mediante el uso de pipes.

   (c) Si el **token** resultante es negativo, el proceso debe terminar su ejecución mediante un **exit()**. Tener en cuenta que, al momento de quitar un proceso del anillo, se deben eliminar los pipes que conectan al proceso, y en caso de tener 2 o más procesos en juego, redirigir el canal de comunicación de pipes, manteniendo el flujo de información dentro del anillo, sin tomar en cuenta el proceso eliminado.

5. Al momento de eliminar un proceso, se activa un mecanismo de elección de líder entre los procesos restantes. El proceso a eliminar, envía un aviso al proceso siguiente de que ha salido del juego; este a su vez, debe enviar el aviso al siguiente, y así hasta que todos los procesos restantes sepan de la eliminación. El nuevo líder (por ejemplo, el proceso con el mayor identificador) reinicializa el **token** y reanuda el desafío. Cabe mencionar que luego de la eliminación de un proceso, todos los procesos sobrevivientes son candidatos a ser el nuevo líder.

6. El desafío continúa hasta que sólo queda un proceso, el cual es declarado ganador.

7. Finalmente, el proceso que sobrevive es quien informa al proceso Padre del término del juego, también informando que ha sido el ganador. Luego, proceso Padre mostrará el flujo del juego por stdout.

3 Restricciones y Consideraciones

- **Comunicación vía pipes:** La transmisión del **token** y la notificación de eventos (como la eliminación de un proceso) deben realizarse exclusivamente mediante pipes. No se permite el uso de otros mecanismos como señales o memoria compartida.

- **Generación de números aleatorios:** Se puede utilizar la función `rand()` de `stdlib.h` para generar números, asegurándose de inicializar la semilla con `srand()`.

- **Sincronización:** Es fundamental prestar atención a la sincronización de los procesos y al manejo de condiciones de carrera al recibir los mensajes por medio de los pipes.

4 Ejemplo de Ejecución

El flujo de ejecución **debe** ser el siguiente:

```
$ ./desafio1 -t 10 -M 10 -p 4
Proceso 0 ; Token recibido: 10 ; Token resultante: 8
Proceso 1 ; Token recibido: 8 ; Token resultante: 5
```

```
Proceso 2 ; Token recibido: 5 ; Token resultante: 2
Proceso 3 ; Token recibido: 2 ; Token resultante: -1 (Proceso 3 es
   eliminado)
Proceso 0 ; Token recibido: 10 ; Token resultante: 9
Proceso 1 ; Token recibido: 9 ; Token resultante: 4
Proceso 2 ; Token recibido: 4 ; Token resultante: -2 (Proceso 2 es
   eliminado)
Proceso 0 ; Token recibido: 10 ; Token resultante: 7
Proceso 1 ; Token recibido: 7 ; Token resultante: 1
Proceso 0 ; Token recibido: 1 ; Token resultante: -3 (Proceso 0 es
   eliminado)
Proceso 1 es el ganador
```

Listing 1: Ejemplo de ejecución del programa. Cuando es incluido el flag -D y cuando no es incluido.

5 Entregables

El laboratorio es en parejas. Si se elige una pareja, ésta no podrá ser cambiada durante el semestre. Se descontará 1 punto (de nota) por cada 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:

Comprima y entregue los siguientes archivos:

1. `Makefile`: Archivo para compilar el programa.
2. `desafio2.c`: Archivo principal en C con el main del código fuente.
3. `funciones.c`: Archivo con las funciones del programa.
4. `funciones.h`: Archivo con las cabeceras de las funciones del programa.
5. `README.txt`: Instrucciones para compilar y ejecutar el programa.
6. 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 Laboratorio N°2".
- **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 laboratorio, 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.

Fecha de entrega: 29-05-2025, antes de las 23:59 hrs.