# Laboratorio 2:

**Profesor:** Viktor Tapia
**Ayudante de cátedra:** Tomás Rebolledo y Nicolás Schiaffino
**Ayudante de Tarea:** Javiera Cárdenas y Nicolás Toro

Septiembre 2024

## 1 Reglas Generales

Para la siguiente tarea se debe realizar un código programado en lenguaje C o C++. Se exigirá que los archivos se presenten de la forma más limpia y legible posible. Deberá incluir un archivo README con las instrucciones de uso de sus programas junto a cualquier indicación que sea necesaria, y un archivo MAKE para poder ejecutar el programa.

## 2 Tarea

Se solicita recrear el juego de mesa llamado "Uno", para esto se les pide que desarrollen el juego y además de que un participante puede jugar a través de la consola, se creen 3 jugadores extra para que el juego quede de 4 jugadores en total. Los jugadores extras deben ser creados con la función fork().

## 3 Objetivo del Juego

El objetivo es ser el primer jugador en quedarse sin cartas en la mano y asi ser el ganador.

## 4 Preparación

- Cada jugador tiene su mano, además esta la pila de descarte y el mazo.
- Se reparten 7 cartas a cada jugador.
- El tablero y el estado del juego se manejan mediante memoria compartida, accesible por todos los procesos.

## 5 Tipos de Cartas

- **Cartas Numéricas**: Con números del 0 al 9 en colores rojo, amarillo, verde y azul, siendo 19 cartas por color (El 0 no se repite).
- **Cartas Especiales**:
  - *Cambio de Sentido*: Cambia la dirección del juego (8 cartas en total, 2 por cada color).
  - *Salta*: El siguiente jugador pierde su turno (8 cartas en total, 2 por cada color).
  - *Roba Dos (+2)*: El siguiente jugador debe robar dos cartas y pierde su turno (8 cartas en total, 2 por cada color).
  - *Comodín*: Permite al jugador elegir el color para el siguiente turno (4 cartas en total).
  - *Comodín Roba Cuatro (+4)*: Permite al jugador elegir el color y obliga al siguiente jugador a robar cuatro cartas y perder su turno (4 cartas en total).

## 6 Desarrollo del Juego

- **Turnos**: Los turnos se manejan mediante procesos paralelos (*forks*). Cada proceso representa a un jugador y accede al tablero utilizando memoria compartida.
- **Jugando Cartas**: En su turno, un jugador debe jugar una carta que coincida en color o número con la carta superior de la pila de descarte. Si no puede, debe robar una carta del mazo. Si la carta robada es jugable, puede ser jugada inmediatamente; de lo contrario, el turno pasa al siguiente jugador.

## 7 Termino del Juego

El juego termina cuando un jugador se queda sin cartas en su mano. Se debe mostrar por pantalla la cantidad de cartas restantes que tiene cada jugador.

## 8 Jugadores y procesos

El juego se compone de 4 jugadores (siendo usted uno de ellos). Cada jugador debe ser un proceso distinto, por lo que en total su programa estará ejecutando 5 procesos (un padre y 4 hijos). El orden sera el jugador seguido por los 3 bots. Recuerde que para finalizar el programa, primero deben finalizar los procesos hijos, y luego el proceso padre deberá liberar la memoria compartida utilizada por el tablero para finalizar la ejecución del programa.

## 9 Presentación Aleatoria

Para cada tarea, se seleccionarán grupos al azar para presentar su tarea frente a ayudantes y eventualmente profesor, recibiendo una ponderación del 80% y 20% entre tarea y presentación respectivamente. Si su grupo presentó en una tarea, no volverá a salir nuevamente. Se comunicará días antes que grupos presentarán.

## 10 README

Debe contener como mínimo:

- Nombre, RoI y Paralelo de los integrantes.
- Especificación de los nombres de los archivos.
- Instrucciones generales de compilación y uso.

## 11 Consideraciones Generales

- Se deberá trabajar de a pares. Se deberá entregar en Github a mas tardar el día 4 de Octubre de 2024 a las 23:59 horas. Se descontarán 10 puntos por cada hora o fracción de atraso. Las copias serán evaluadas con nota 0 en el promedio de las tareas.
- La tarea debe ser hecha en el lenguaje C o C++. Se asume que usted sabe programar en este lenguaje, ha tenido vivencias con el, o que aprende con rapidez.
- Pueden crear todas las funciones auxiliares que deseen, siempre y cuando estén debidamente comentadas.
- Las tareas serán ejecutadas en **Linux**, cualquier tarea que no se pueda ejecutar en dicho sistema operativo, partirá de nota máxima 60.
- Las preguntas deben ser hechas por Aula. De esta forma los demás grupos pueden beneficiarse en base a la pregunta.
- Si no se entrega README o MAKE, o si su programa no funciona, la nota es 0 hasta la recorrección.
- Se descontarán 50 puntos por:
  - Mala implementación del Makefile.
  - No respetar el formato de entrega.
  - Solicitar edición de codigo para las rutas.