Laboratorio 2: Sistemas Operativos

Profesor: Viktor Tapia
Ayudantes de cátedra: Muryel Constanzo y Nicolás Schiaffino
Ayudantes de Laboratorio: Ian Rossi y Luciano Yevenes

20 de abril de 2025

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 y ejecución de sus programas junto a cualquier indicación que sea necesaria, y un archivo MAKE para poder ejecutar el programa.

2. Tarea

Un conocido profesor del ramo de Sistemas Operativos esta planeando una expedición en solitario al Polo Sur, pero sabe que no podrá soportar el aburrimiento estando semanas solo en la nieve.

A un grupo de estudiantes (incluyendo a ustedes) se les ha asignado la tarea de crear entretenimiento para que el profesor lleve a bordo del *Antarctic Snow Cruiser "Penguin 1"*. Ustedes son los responsables de crear un software para que el profesor pueda jugar contra bots (de esta manera no se requieren mas jugadores, o conexión a internet, algo difícil de conseguir en la Antartida), las veces que quiera, lo cual lo dejará incapaz de sentir aburrimiento.

Pese a que tienen dudas acerca de la finalidad de la expedición, toman con entusiasmo la tarea, decidiendo desarrollar un software para jugar Blackjack.

2.1. Especificaciones

- El juego debe contar con un croupier y 4 jugadores.
- El juego consta de solo un mazo de 52 cartas sin incluir a los jokers.
- Se debe poder interactuar con el juego mediante la terminal, donde el usuario de esta juega como el croupier y los jugadores serán las maquinas creadas con la instrucción **Fork()**.
- Deben ser 5 rondas, o un número determinado por el usuario.
- Cada victoria vale $+1$ punto para cada jugador que le gano al croupier, pero el crupier solo recibe $+1$ punto por ronda si gana la mayoría de los duelos, si gana **todos** los duelos gana $+2$ puntos.

  Ejemplo:

  - Ronda 1:
    4 jugadores pierden $\rightarrow$ Crupier gana 2 punto.

- Ronda 2:
  3 jugadores pierden, 1 gana (jugador que ganó recibe 1 punto) → Crupier gana 1 punto.
- Ronda 3:
  2 ganan (cada jugador que ganó recibe 1 punto), 2 pierden → Empate, crupier no recibe punto.
- Ronda 4:
  3 o más ganan (cada jugador que ganó recibe 1 punto) → crupier no recibe punto.

- **OJO:** el crupier también juega en el Blackjack.

3. Reglas del Blackjack

El objetivo es tener una mano que sume más que la del crupier, pero sin sobrepasar 21.

3.1. Valores de las cartas

- Cartas numéricas (2–10): valen su número.
- Figuras (J, Q, K): valen 10 cada una.
- Ases: pueden valer 1 u 11, según convenga al jugador

3.2. Opciones del jugador

- Pedir: Solicitar una carta adicional.
- Plantarse: Mantener la mano actual y no recibir más cartas.

3.3. Desarrollo de la ronda

- Reparto: Se reparten dos cartas boca arriba a cada jugador y dos al crupier (una boca arriba y otra boca abajo)
- Turno de los jugadores: Comenzando por el primer asiento a la izquierda del crupier, cada jugador decide si "pedir" (recibir una carta adicional) o "plantarse" (mantener su mano actual). La acción se determina según el valor total de su mano:
  - Si el total está en el rango $[..., 11]$, el jugador siempre pedirá una carta.
  - Si el total está en el rango $[12, 18]$, el jugador tendrá un 50 % de probabilidad de pedir.
  - Si el total está en el rango $[19, 21]$, el jugador no pedirá más cartas.
- Turno del crupier: Una vez que todos los jugadores hayan terminado su turno (ya sea porque se plantaron o perdieron), el crupier tomará su turno y mostrara la carta volteada. En esta implementación, el crupier será controlado por el usuario, quien podrá decidir si desea pedir una carta o plantarse. La forma en que se gestionen estas decisiones queda a elección del grupo, siempre y cuando el control se realice a través de la consola.
- Término de la ronda: Una vez que el crupier se plante o se pase de 21, se asignarán los puntos correspondientes a cada jugador según el resultado de la mano. Luego, se iniciará una nueva ronda utilizando nuevamente todas las cartas, y este proceso se repetirá hasta completar las 5 rondas o la cantidad previamente definida por el usuario al inicio del juego.

- **Importante:** Todas las acciones relacionadas con la entrega de cartas deben mostrarse por consola. Esto incluye la repartición inicial (indicando qué carta recibió cada jugador, exceptuando la carta volteada del crupier) y cada vez que un jugador o el crupier solicite una carta. En resumen, toda acción en la que se entregue una carta debe ser registrada y visualizada por consola, indicando explícitamente qué carta fue entregada y a quién.

3.4. Cómo ganar o perder (evaluado luego del turno del crupier)

- El jugador gana si su mano supera la del crupier sin pasarse de 21.
- Un "Blackjack" (21 con las dos primeras cartas) vence automáticamente cualquier otra mano, incluso una del crupier con 21 en más de dos cartas.
- El jugador pierde si su mano supera los 21 puntos ("se pasa") —este es el único caso en que puede perder antes de que el crupier juegue— o si su puntaje es menor al del crupier, siempre que este no se haya pasado.
- En caso de empate ("push"), se considera que el jugador ha ganado la ronda, pero no gana puntos.

4. Termino del juego

Al terminar todas las rondas todos los jugadores deberán comunicar su cantidad de puntos al croupier y deberán terminar sus procesos, cuando el croupier tenga todos los puntajes debera comunicar por consola el jugador que gano (en caso de empate debe decir todos los jugadores que ganaron incluyendo a este mismo si fuese el caso).

5. Consideraciones Específicas

- No es necesario que implementen un sistema de apuestas, solo considerar puntos para los jugadores en base a si ganan o no.
- Deben hacer uso de Make, **no** de CMake.

6. BONUS (20 pts extras)

A medida que avanzan los días en la desolación blanca del Polo Sur, el profesor —cada vez más hábil en el Blackjack— empieza a aburrirse de derrotar a bots ingenuos que solo siguen decisiones básicas. Es entonces cuando surge una necesidad imperiosa: desarrollar una versión avanzada del juego, en la cual los bots implementen una estrategia inteligente basada en conteo de cartas, similar a la usada por jugadores profesionales en casinos reales.

Para este modo adicional, los bots deberán ser capaces de observar las cartas que han salido durante la partida y ajustar su estrategia de juego en consecuencia. A continuación, se detallan las reglas y mecánicas necesarias para incorporar esta funcionalidad:

- **Sistema de conteo:** Se implementará el sistema Hi-Lo, donde:
  - Cartas bajas (2 al 6): se asigna un valor de $\mathbf{+1}$.
  - Cartas medias (7 al 9): se asigna un valor de $\mathbf{0}$.
  - Cartas altas (10, J, Q, K, A): se asigna un valor de $\mathbf{-1}$.
- **Decisiones estratégicas según el conteo:**

  Las decisiones de los jugadores respecto a pedir o plantarse estarán guiadas por la siguiente tabla de referencia:

Tu mano	Conteo acumulado	Decisión recomendada
12–16	Muy positivo ($+4$ o más)	No pedir (alta probabilidad de pasarse)
12–16	Muy negativo (-4 o menos)	Pedir (probabilidad favorable de carta baja)
17–20	Cualquier conteo	Usualmente no pedir

- **Probabilidad de pedir:** Aunque existan recomendaciones, se incluirá una pequeña probabilidad de que el jugador (máquina) decida pedir, la cual disminuirá a medida que el conteo se aleje del equilibrio. La fórmula sugerida es:

$$P(\text{pedir}) = \max\left(0, \frac{1}{2} - \frac{|\text{Conteo}|}{10}\right)$$

  Esto implica, por ejemplo:

- Con un conteo de 0: 50 % de pedir.
- Con un conteo de $+5$ o -5: 0 % de pedir.
- Con un conteo de $+2$: 30 % de pedir.

Esta probabilidad podrá ser utilizada por el jugador controlado por la máquina para tomar decisiones de forma dinámica y realista.

7. README

Debe contener como mínimo:

- Nombre, Rol y Paralelo de los integrantes.
- Especificación de los algoritmos y desarrollo realizado.
- Instrucciones de como compilar y correr el código.
- Supuestos utilizados.

8. Consideraciones Generales

- Se deberá trabajar **OBLIGATORIAMENTE** en parejas.
- Deberá estar subido al Github correspondiente a mas tardar el día Lunes 5 de Mayo a las 23:59 horas.
- Se descontarán 10 puntos por cada hora o fracción de atraso.
- La nota máxima obtenible, si se realiza también el bonus, es de 120.
- 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.
- El código debe ser entregado en forma de **1 solo archivo** `.cpp` o `.c`, nombrado en base al formato `"LAB1_ApellidoIntegrante1_ApellidoIntegrante2"`
- Los códigos serán pasados por un software anti-plagio, en caso de ser detectada copia o uso de una IA para la totalidad del Laboratorio, se pondrá nota 0, hasta que se realice una reunión con los grupos involucrados.
- 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 a través del foro o servidor de Discord. De esta forma los demás grupos pueden verse beneficiados también.
- Si no se entrega README o MAKE, o si su programa no funciona, la nota es 0 hasta la recorrección.
- Se descontarán hasta 50 puntos por:

- Mala implementación del Makefile.
- No respetar el formato de entrega.
- No respetar el formato del README.
- Solicitar edición de código al momento de revisar.
- Código poco prolijo y mal estructurado (ausencia de indentación adecuada, falta de consistencia en el estilo, nombres de variables confusos o poco descriptivos, comentarios insuficientes o irrelevantes).

- **Una vez publicadas las notas tendrán 5 días para apelar con el corrector que les revisó, después de este plazo las notas no tendrán ningún tipo de cambio**.