Sistemas Operativos

Alumnos:

Nombre	No. de cuenta
Membrilla Ramos Isaias Iñaki	315293728
Miramón Pérez Jocelyn	320246881

Grupo: 6

Profesor:

Ing. Gunnar Eyal Wolf Iszaevich

Tarea 1

Semestre 2025-2

Fecha de entrega: 27 de marzo del 2025

El CRUCE DEL RÍO

1. Problema

1.1. Planteamiento

- Para llegar a un encuentro de desarrolladores de sistemas operativos, hace falta cruzar un río en balsa.
- Los desarrolladores podrían pelearse entre sí, hay que cuidar que vayan con un balance adecuado.

1.2. Reglas

- En la balsa caben cuatro (y sólo cuatro) personas
  - La balsa es demasiado ligera, y con menos de cuatro puede volcar.
- Al encuentro están invitados hackers (desarrolladores de Linux) y serfs (desarrolladores de Microsoft).
  - Para evitar peleas, debe mantenerse un buen balance: No debes permitir que aborden tres hackers y un serf, o tres serfs y un hacker. Pueden subir cuatro del mismo bando, o dos y dos.
- Hay sólo una balsa
- No se preocupen por devolver la balsa (está programada para volver sola).

Posibles casos:

Lo ilustraremos, pero con ratas y luchadores, nada personal, solo amamos las ratas y la lucha libre.

Figure 1: Participantes

Figure 2: 2 ratas y dos luchadores

Figure 3: 4 luchadores

Figure 4: 4 ratas

En el caso de que no se cumplan las condiciones, será una lucha a muerte. Sino son cuatro, morirán ahogados.

2. Lenguaje y entorno

El lenguaje que empleamos fue pyhton, por lo que, usamos los módulos threading, random time y pygame.

Listing 1: Importación de bibliotecas

```
1 import threading
2 import random
3 import time
4 import pygame
```

Para ejecutarlo, es necesario tener la instalación de python 3. Lo cual, dependiendo del sistema operativo puede realizarse en la línea de comandos una instrucción parecida a esta:

```
python --version
```

También se necesita tener instalada la biblioteca pygame, la cual se instala de la siguiente forma:

```
pip3 install pygame
```

Para poder ejecutar el programa, las imágenes tux.png y clippy.png deben estar en la misma carpeta que el código.

3. Estrategia de sincronización

Para la implementación del código visualizamos primero los casos en la sección Problema, podemos notar que mientras hayan dos de cada grupo, o 4 del mismo grupo no habrá problemas, por lo tanto, es que si suben en parejas del mismo grupo hasta completar la cantidad máxima de 4 pasajeros en la balsa, siempre se cumplirá la regla.

La estrategia que se utilizó fue *Queue*, la cual consiste en usar los semáforos como una cola, una para los hackers y otra para los serfs, con el fin de hacer que se bloqueen hasta que otro del mismo tipo llegue; al hacerlo, libera al primero de la cola. Además de dos variables globales que sirven de contador y permiten llevar el control de los hackers esperando para ser emparejados. También se utiliza un mutex para bloquear las variables globales, como total y las variables auxiliares que usamos para generar las parejas. Se utilizó solo un mutex en lugar de tres debido a la experiencia del usuario; si usábamos tres, no se mostraba adecuadamente cómo subían a la balsa. Cada que se completa un par de hackers o serfs, se llama a subir (). La primera vez, si total está en 1, se decrementa a 0. Aún no se va la balsa. La segunda vez que un par llama a subir(), total ya está en 0, por lo que dos parejas zarparon completando así el grupo de 4.

Ejecutando el programa:

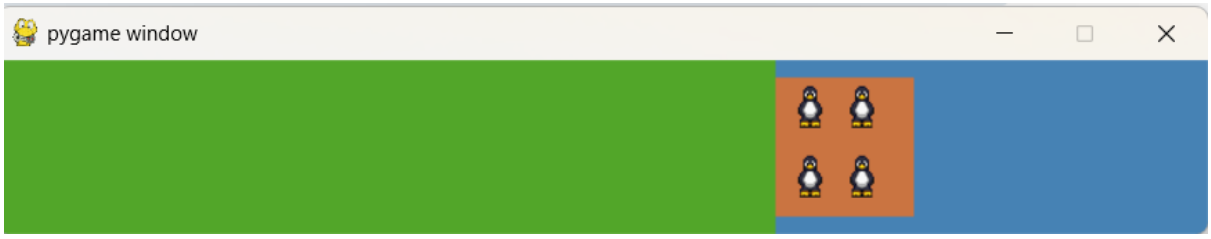

Figure 5: 1er caso, 4 hackers.

Figure 6: 2do caso, 4 serfs

Figure 7: 3er caso, 2 hackers y 2 serfs