Universidad ORT Uruguay

Sistemas Operativo

Entrega Obligatorio

03/12/2023

Bruno Odella - 231665

Diego Pérez - 260186

Angelina Maverino - 280070

Índice

EJERCICIO 1

veterinaria.sh

```
#!/bin/bash

function volver_al_menu(){
  read -p "Presione cualquier tecla para volver al menú "
  menu
}

function validar_cedula(){
while IFS=, read -r cedula_registrada nombre resto; do
  if [[ " $cedula_registrada " =~ " $1 " ]]; then
    echo "El cliente $nombre con la cédula $cedula_registrada ya está registrado"
    return 1
  fi
done < "socios.txt"


}

function registrar_socio(){
  touch socios.txt
  echo "================================="
  read -p "Ingrese cédula del dueño: " cedula
  cedula_a_validar="$cedula"
  temp_socio="$cedula,"
  validar_cedula "$cedula_a_validar"
```

```
valido=$?
if [ "$valido" -eq 1 ]; then # Si retorna 1 entonces hay una cédula duplicada
  volver_al_menu
fi
read -p "Ingrese nombre del dueño: " nombre_d
temp_socio="$temp_socio$nombre_d,"

          echo "==================================="
for ((i = 1; i < 4; i++)); do

  read -p "Ingrese nombre de la mascota: " nombre_m
  temp_socio="$temp_socio$nombre_m,"
  read -p "Ingrese edad de la mascota: " edad_m
  temp_socio="$temp_socio$edad_m,"

  if [ $i -lt 4 ]; then
    read -p "¿Quiere registrar otra mascota? [s/n] " resp
    resp=$(echo "$resp" | tr '[:upper:]' '[:lower:]')
    if [ "$resp" = "s" ] || [ "$resp" = "si" ] || [ "$resp" = "sí" ] || [ "$resp" = "yes" ] || [ "$resp" =
"y" ];then
          echo "==================================="
         else
       for ((j = 4; j > i; j--)); do
           temp_socio+="NULL,NULL,"
          done
```

```
            break
        fi
    fi
done
            echo "================================="
read -p "Ingrese opción de contacto (email o teléfono): " contacto
temp_socio="$temp_socio$contacto"

            echo "================================="
echo "$temp_socio" | tee -a socios.txt
volver_al_menu

}

function agendar_cita(){
  touch citas.txt # Crea el archivo si no existe

  echo "Agendar una nueva cita:"
  read -p "Ingrese cédula del dueño: " cedula_dueno # Cambia 'cedula_dueño' por 'cedula_dueno'
  read -p "Ingrese nombre de la mascota: " nombre_mascota
  echo "Motivos de la cita (por ejemplo: revisión, vacunación, etc.):"
  read -p "Ingrese motivo de la cita: " motivo_cita
  read -p "Ingrese costo de la cita: " costo_cita
```

```
  while true; do
    read -p "Ingrese fecha de la cita (formato AAAA-MM-DD): " fecha_cita
    if [[ $fecha_cita =~ ^[0-9]{4}-[0-9]{2}-[0-9]{2}$ ]]; then
      break
    else
      echo "Fecha no válida. Por favor, use el formato AAAA-MM-DD."
    fi
  done

  while true; do
    read -p "Ingrese hora de la cita (formato HH:MM): " hora_cita
    if [[ $hora_cita =~ ^[0-9]{2}:[0-9]{2}$ ]]; then
      break
    else
      echo "Hora no válida. Por favor, use el formato HH:MM."
    fi
  done

cita="$cedula_dueño,$nombre_mascota,$motivo_cita,$costo_cita,$fecha_cita,$hora_cita"
  echo "$cita" | tee -a citas.txt
  echo "Cita agendada con éxito."
  volver_al_menu
}
```

```
function actualizar_stock(){
  touch articulos.txt # Crea el archivo si no existe

  echo "Actualizar Stock en Tienda:"
  read -p "Ingrese categoría del artículo (ej. medicamentos, accesorios): " categoria
  read -p "Ingrese código del artículo: " codigo
  read -p "Ingrese nombre del artículo: " nombre
  read -p "Ingrese precio del artículo: " precio
  read -p "Ingrese cantidad a agregar al stock: " cantidad

  # Verifica si el artículo ya existe
  articulo_encontrado=false
  while IFS=, read -r cat cod nom pre cant; do
    if [ "$codigo" = "$cod" ]; then
      # Actualizar la cantidad y marcar que el artículo fue encontrado
      nueva_cantidad=$((cant + cantidad))
      sed -i "/$cod/d" articulos.txt
      echo "$categoria,$codigo,$nombre,$precio,$nueva_cantidad" | tee -a articulos.txt
      articulo_encontrado=true
      echo "Stock actualizado para el artículo: $nombre."
      break
    fi
  done < articulos.txt
```

```
  # Si el artículo no existe, agregarlo nuevo
  if [ "$articulo_encontrado" = false ]; then
    echo "$categoria,$codigo,$nombre,$precio,$cantidad" | tee -a articulos.txt
    echo "Nuevo artículo agregado: $nombre."
  fi

  volver_al_menu
}

function realizar_venta(){
  touch articulos.txt # Crea el archivo si no existe
  touch ventas.txt # Para registrar las ventas

  echo "Realizar venta de productos:"
  read -p "Ingrese código del artículo a comprar: " codigo_compra
  read -p "Ingrese cantidad a comprar: " cantidad_compra

  articulo_encontrado=false
  while IFS=, read -r categoria codigo nombre precio cantidad; do
    if [ "$codigo" = "$codigo_compra" ]; then
      articulo_encontrado=true
      if [ "$cantidad" -lt "$cantidad_compra" ]; then
        echo "Stock insuficiente. Solo hay $cantidad unidades disponibles."
```

```
      else
        nueva_cantidad=$((cantidad - cantidad_compra))
        sed -i "/$codigo/d" articulos.txt
        echo "$categoria,$codigo,$nombre,$precio,$nueva_cantidad" | tee -a
articulos.txt
        total_venta=$(echo "$precio * $cantidad_compra" | bc)
        echo "Venta realizada: $nombre, Cantidad: $cantidad_compra, Total:
\$${total_venta}"
        echo "$(date +%Y-%m-%d),$codigo,$nombre,$cantidad_compra,\$${total_venta}"
| tee -a ventas.txt
      fi
      break
    fi
  done < articulos.txt

  if [ "$articulo_encontrado" = false ]; then
    echo "Artículo no encontrado."
  fi

  volver_al_menu
}

function generar_informe_mensual(){
  touch ventas.txt # Asegúrate de que el archivo exista

  echo "Generar Informe Mensual:"
```

```
  read -p "Ingrese el año y mes para el informe (formato AAAA-MM): " mes_informe

  total_mes=0
  cantidad_ventas=0

  while IFS=, read -r fecha codigo nombre cantidad total; do
    if [[ $fecha == $mes_informe-* ]]; then
      cantidad_ventas=$((cantidad_ventas + 1))
      # Extrae el monto total de la venta y lo acumula
      monto_venta=$(echo $total | tr -d '$')
      total_mes=$(echo "$total_mes + $monto_venta" | bc)
    fi
  done < ventas.txt

  if [ "$cantidad_ventas" -eq 0 ]; then
    echo "No se encontraron ventas para $mes_informe."
  else
    echo "Informe del mes $mes_informe:"
    echo "Ventas totales: $cantidad_ventas"
    echo "Total recaudado: \$${total_mes}"
  fi

  volver_al_menu
}
```

```
function salir() {
 echo "Saliendo del programa."
 exit 0
}

function menu() {
  while true; do
    clear
    echo "Menú:"
    echo "1. Registrar socio"
    echo "2. Agendar cita"
    echo "3. Actualizar stock en tienda"
    echo "4. Venta de productos"
    echo "5. Informe mensual"
    echo "6. Salir"
    read -p "Seleccione una opción: " opcion


    case $opcion in
      1)
        registrar_socio
        ;;
      2)
            agendar_cita
```

```
        ;;
      3)
          actualizar_stock
        ;;
      4)
        realizar_venta
        ;;
      5)
        generar_informe_mensual
        ;;
      6)
          salir
       ;;
      q)
          salir
          ;;
      *)
        echo "Opción no válida. Por favor, seleccione una opción válida."
        ;;
    esac
  done
}

function main(){
```

menu

}

clear

main

ventas.txt

2023-05-01,001,Ibuprofeno para perros,2,$30.00

2023-05-02,002,Collar antipulgas,1,$10.00

2023-05-03,003,Alimento Seco para Gatos,3,$60.00

2023-05-04,004,Shampoo para Perros,2,$24.00

temp_socio

1234,

1,

1,

1,

123,

123,

df\n

citas.txt

12345678,Firulais,Vacunación,30.00,2023-05-21,10:00

87654321,Manchas,Revisión,45.00,2023-05-22,11:30

23456789,Bigotes,Esterilización,60.00,2023-06-15,09:00

98765432,Rex,Limpieza Dental,80.00,2023-06-20,14:00

articulos.txt

Medicamento,001,Ibuprofeno para perros,15.00,20

Accesorio,002,Collar antipulgas,10.00,35

Alimento,003,Alimento Seco para Gatos,20.00,50

Higiene,004,Shampoo para Perros,12.00,25

EJERCICIO 2

ejercicio2.c

```
#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <semaphore.h>

#include <unistd.h>


sem_t sE,sG, sP, sI;



// Función que genera una espera aleatoria

void random_wait(){

  int min_wait=0;

  int max_wait=10000;

  srand(time(NULL)); // Randomiza

  int sleep_time = (rand() % (max_wait - min_wait + 1)) + min_wait; //Obtiene un numero al azar entre minimo y maximo

  usleep(sleep_time); //Hace esperar al hilo
```

```
}

//Función para simular la ejecución de procesos y imprimirlos en salida standar
void execute_process(char name){
  random_wait();
  printf("%c", name);
}

void* lA (void * x){
  execute_process('A');
  pthread_exit(NULL);
  return 0;
}


void* lJ (void * x) {
  // Implementa las operaciones necesarias para 'J'
  execute_process('J');
  sem_wait(&sI);
  execute_process('M');
  execute_process('P');
  sem_post(&sP);
  return 0;
}
```

```
void* lk (void * x) {
  execute_process('K');
  pthread_exit(NULL);
  return 0;
}



void* lB (void * x){
  execute_process('B');
  execute_process('D');
  sem_wait(&sE);
  execute_process('F');
  sem_wait(&sG);
  execute_process('Q');
  pthread_t t5, t6;
  pthread_create(&t5, NULL, lJ, NULL);
  pthread_create(&t6, NULL, lk, NULL);
  execute_process('I');
  sem_post(&sI);
  execute_process('L');
  execute_process('N');
  sem_wait(&sP);
```

```
  execute_process('O');
  return 0;
}
void* lC (void * x){
  execute_process('C');
  execute_process('E');
  sem_post(&sE);
  sem_post(&sE);
  return 0;
}

void* lH (void * x){
  execute_process('H');
  sem_wait(&sE);
  execute_process('G');
  sem_post(&sG);
  return 0;
}


int main(){
  sem_init(&sE, 0, 0);
  sem_init(&sG, 0, 0);
  sem_init(&sP, 0, 0);
```

```
  sem_init(&sI, 0, 0);
  pthread_t t1,t2,t3,t4;
  pthread_attr_t attr;
  pthread_attr_init(&attr);
  pthread_create(&t1, &attr, lA, NULL);
  pthread_create(&t2, &attr, lB, NULL);
  pthread_create(&t3, &attr, lC, NULL);
  pthread_create(&t4, &attr, lH, NULL);

  pthread_join(t1, NULL);
  pthread_join(t2, NULL);
  pthread_join(t3, NULL);
  pthread_join(t4, NULL);

  sem_destroy(&sE);
  sem_destroy(&sG);
  sem_destroy(&sP);
  sem_destroy(&sI);

  return 0;
}
```

script_pruebas_ejercicio2.sh

```
#!/bin/bash
```

```
# Número d veces que se ejecutará el programa
if [ $# -eq 0 ]; then
  num_executions=50
else
  num_executions="$1"
fi
output="salida.txt"
bad_results_file="casos_fallidos.txt"
echo "Batería de pruebas corrida el $(date)" > "$bad_results_file"
failed_cases=0

mensaje_errores() {
 if [ "$failed_cases" -gt 0 ]; then
  echo "Fallaron $failed_cases casos. Estan en el archivo $bad_results_file"
 fi

}

mensaje_salida() {
 if [ "$failed_cases" -eq 0 ]; then
   echo "Todas las $num_executions ejecuciones salieron en el órden correcto"
 else
  mensaje_errores
```

```
 fi
 rm "$output"
}

sigint_handler() {
  echo ""
  echo "Proceso interrumpido con Ctrl+C."
  if [ $failed_cases -eq 0 ]; then
    echo "Hasta ahora ninguna ejecución falló"
  else
    mensaje_errores
  fi
  exit 0
}

# Registra el manejador de señales para SIGINT
trap 'sigint_handler' SIGINT



check_output(){

  temp_output="temp.txt"

  cat "$output" | grep --color=always 'D.*B\|E.*C\|G.*H' | tee -a "$temp_output"
```

```
cat "$output" | grep --color=always 'F.*D\|F.*E\|G.*E\|G.*H' | tee -a "$temp_output"
cat "$output" | grep --color=always 'Q.*F\|Q.*G' | tee -a "$temp_output"
cat "$output" | grep --color=always 'I.*Q\|J.*Q\|K.*Q' | tee -a "$temp_output"
cat "$output" | grep --color=always 'L.*I' | tee -a "$temp_output"
cat "$output" | grep --color=always 'M.*I\|M.*J' | tee -a "$temp_output"
cat "$output" | grep --color=always 'N.*L' | tee -a "$temp_output"
cat "$output" | grep --color=always 'O.*P' | tee -a "$temp_output"
cat "$output" | grep --color=always 'P.*M' | tee -a "$temp_output"

if [ -s "$temp_output" ]; then
  ((failed_cases++))
  echo "============================" >> "$bad_results_file"
  cat "$temp_output" >> "$bad_results_file"
  echo -e "RESULTADO \e[31mFALLIDO\e[0m" | tee -a "$bad_results_file"
  echo "============================" >> "$bad_results_file"
  rm "$temp_output"
  return 1
else
  echo -e "\e[34m$(cat $output)\e[0m"
  echo -e "RESULTADO \e[32mOK\e[0m"
  rm "$temp_output"
  return 0
fi
```

```
}

# Compila el programa
gcc -pthread -o ejercicio2 ejercicio2.c

# Verifica si la compilación tuvo éxito
if [ $? -eq 0 ]; then
  echo "Compilación exitosa."
  # Bucle para ejecutar el programa 200 veces
  for ((i = 1; i <= $num_executions; i++)); do
    # Ejecuta el programa y guarda la salida estándar en el archivo
    ./ejercicio2 > $output
    echo "============================="
    # Verifica si la ejecución tuvo éxito
    if [ $? -eq 0 ]; then
      echo "Ejecución $i"
      check_output
    else
      echo "Error en la ejecución $i."
    fi
    echo "============================="
  done
  mensaje_salida
else
```

```
  echo "Error de compilación."
fi
```

EJERCICIO 3

main.adb

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with Ada.Command_Line;
with Ada.Strings;
with Ada.Task_Identification; use Ada.Task_Identification;
with ada.numerics.discrete_random;

procedure Main is

  type Par_De_Pisos is record
    Floor_From, Floor_To: Integer;
  end record;

  Task type Semaforo is
    Entry preguntarEstado(state : OUT Integer);
    Entry wait;
    Entry signal;
  End Semaforo;

  Task body Semaforo is
```

```
  estado: integer := 1;
begin
  Loop
    select
      accept signal;
      estado:=estado + 1;
    or
      accept preguntarEstado(state : OUT integer) do
        state:=estado;
      End;
    or
      accept wait;
      estado:=estado - 1;
    or
      terminate;
    end select;
  End loop;
end Semaforo;



Suno : Semaforo;
Sdos : Semaforo;

Task type irAlFrom is
```

```
    entry ir(w : in integer; x : in integer ; y : out integer);
    entry empezar(w : IN integer);
  end irAlFrom;
  Task body irAlFrom is
    z : integer;
    anterior : integer := 0;
    ir1 : integer ;
    ir2 : integer;
  begin
    accept empezar(w : IN integer) do
      z := w;
    end empezar;
    loop
      accept ir(w : in integer; x : in integer ; y : out integer) do
        ir1 := w;
        ir2 := x;
        y := ir2;
      end ir;
      if z = 1 then
        Suno.wait;
      else
        Sdos.wait;
      end if;
      Put_Line ("- " & z'Image & " Elevator moving from floor" & anterior'Image & " to" &
ir1'Image);
```

```
        delay 1.0;
        Put_Line ("- " & z'Image & " Elevator moving from floor" & ir1'Image & " to" &
ir2'Image);
        anterior := ir2;
        delay 1.0;
        if z = 1 then
          Suno.signal;
        else
          Sdos.signal;
        end if;
      end loop;
    end irAlFrom;

    uno : irAlFrom;
    dos : irAlFrom;


    Task type trabajo is
      entry agregar(w : in Par_De_Pisos);
      entry empezar(w : in integer; y : in integer);
    end trabajo;
    Task body trabajo is

      dondeEstaUno : integer := 0;
      dondeEstaDos : integer := 0;
```

```
  estado1 : integer := 1;
  estado2 : integer := 1;

  ir1 : integer;
  ir2 : integer;

  diferencia1 : integer;
  diferencia2 : integer;

begin
  accept empezar(w : in integer; y : in integer) do
    uno.empezar(w);
    dos.empezar(y);
  end empezar;
  loop
    accept agregar(w : in Par_De_Pisos) do


      ir1 := w.Floor_From;
      ir2 := w.Floor_To;

      Suno.preguntarEstado(estado1);
      Sdos.preguntarEstado(estado2);
```

```
  while estado1 = 0 and estado2 = 0 loop
    delay 1.0;
    Suno.preguntarEstado(estado1);
    Sdos.preguntarEstado(estado2);
  end loop;

  if estado1 = 0 and estado2>0 then
    dos.ir(ir1, ir2 , dondeEstaDos);

  elsif estado2 = 0 and estado1>0 then
    uno.ir(ir1, ir2 ,  dondeEstaUno);

  elsif estado1 > 0 and estado2 > 0 then
    diferencia1 := dondeEstaUno - ir1;
    diferencia2 := dondeEstaDos - ir1;

    if abs(diferencia2) < abs(diferencia1) then
      dos.ir(ir1, ir2 , dondeEstaDos);
    else
      uno.ir(ir1, ir2 ,  dondeEstaUno);
    end if;
  end if;

end agregar;
```

```
      end loop;
   end trabajo;



   -- prueba : Par_De_Pisos := (Floor_From => 0, Floor_To => 5);
   --prueba1 : Par_De_Pisos := (Floor_From => 0, Floor_To => 3);
   -- prueba2 : Par_De_Pisos := (Floor_From => 6, Floor_To => 2);
   -- prueba3 : Par_De_Pisos := (Floor_From => 3, Floor_To => 8);
   -- prueba4 : Par_De_Pisos := (Floor_From => 1, Floor_To => 9);
   -- prueba5 : Par_De_Pisos := (Floor_From => 2, Floor_To => 3);
   -- prueba6 : Par_De_Pisos := (Floor_From => 0, Floor_To => 3);
   -- prueba7 : Par_De_Pisos := (Floor_From => 1, Floor_To => 5);
   -- prueba8 : Par_De_Pisos := (Floor_From => 10, Floor_To => 0);
   -- prueba9 : Par_De_Pisos := (Floor_From => 4, Floor_To => 0);


   S : trabajo;


   type randRange is new Integer range 0..10;  -- genero numeros aleatorios en este
rango
   package Rand_Int is new ada.numerics.discrete_random(randRange);
   use Rand_Int;
   gen : Generator;
   numM : randRange;   -- para guardar el numero aleatorio generado para poder
mostrarlo del Main
```

```
   numN : randRange;
   Input_Number : Integer;



begin
   Put_Line ("Bienvenido al edificio!");
   S.empezar(1,2);
   Put_Line("Ingrese numero de pedidos: ");
   Get(Input_Number);
   while Input_Number /= 0 loop
      reset(gen);
      numM :=random(gen);
      numN := random(gen);
      S.agregar((Floor_From => Integer (numM), Floor_To => Integer (numN)));
      Input_Number :=  Input_Number - 1;
   end loop;


   null;
end Main;
```

EJERCICIO 4

docker-compose.yml

```
version: '3.8'
```

```
services:
  app:
    cap_add:
      - NET_ADMIN
    entrypoint:
      - /app/entrypoint.sh
    build:
      context: ./app
    volumes:
      - ./app/instance/:/app/instance
    networks:
      - app-network
    environment:
      - FLASK_APP=flaskr
   # Otras configuraciones necesarias para tu aplicación (volumenes, variables de entorno, etc.)


  nginx:
    image: nginx:latest
    ports:
      - "80:80"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
    networks:
```

```
    - app-network
  depends_on:
    - app

networks:
  app-network:
```

.env

```
FLASK_APP=flaskr
FLASK_ENV=development
```

nginx.conf

```
events { }
http {
  server {
    listen 80;

    location / {
        proxy_pass http://app:8080; # Redirige al puerto donde se ejecuta tu app (puerto 8080 en este caso)
    }
  }
}
```

Dockerfile

```
FROM python:3.11.5

# Directorio de trabajo en el contenedor
```

```
WORKDIR /app



# Copia los archivos de la aplicación al contenedor
COPY . .

# Crea el entorno virtual e instala las dependencias de la aplicación
RUN python3 -m venv venv \
    && . venv/bin/activate \
    && pip install -e .

RUN chmod +x entrypoint.sh
```

entrypoint.sh

```
#!/bin/sh

cd /app



if [ -f /app/instance/*.sqlite  ]; then
  echo "La base de datos ya fue inicializada"
else
  echo "Inicializando base de datos"
  flask init-db
fi
```

```
flask run --host=0.0.0.0 --port=8080
```