Universidad del Valle de Guatemala
Facultad de Ingeniería
CC 3086 Programación de microprocesadores
Samuel Argueta - 211024
Alejandro Martinez - 21430

I. Para este proyecto nos propusimos resolver una tarea simple pero que puede llegar a ser entretenida e incluso educativa. Es difícil saber si un negocio es rentable con distintas inversiones iniciales y cantidades de negocios.

II. Para resolverlo hicimos una simulación muy simple del manejo de gasolineras, cuyas propiedades incluyen: cantidad de carros que las visitan, cantidad de dinero que cada carro gasta, cantidad de gasolina que puede despachar la gasolinera y márgenes de ganancia.

III. Las fortalezas y debilidades de nuestra solución son una y la misma; la simplicidad. Es fácil de entender y con pocos parámetros a modificar, por lo que va directo al grano de ciertas enseñanzas, pero por la misma naturaleza no hay circunstancias extraordinarias, decisiones importantes y muchas otras variables que existirían en la vida real.

IV. Catálogo de funciones:
- int main: crea los hilos de gasStation tras asignarles variables especificadas
- int amountGasStationInput: input de número entre 1 y 3
- int initialInvestmentInput; input de número entre 250 y 15000
- void *gasStation: función en hilos que genera los automóviles.
- void *gasPrice: función en hilos que calcula los cambios de dinero ente las variables.

V. Resultados:
- El programa pide al usuario la cantidad de gasolineras a generar.
- Pide también la inversión inicial por gasolinera
  Genera de forma pseudo aleatoria la cantidad de vehículos que visitan cada gasolinera, los márgenes de ganancia de la gasolinera, y el monto que cada automóvil gasta

  - Retorna los gastos por vehículos en las gasolineras.
  - Retorna las ganancias y gastos totales y por gasolinera.

ANEXO: fase 1

Franquicia Gasolinera (SHELL/TEXACO/PUMA/RINHO)

a. **¿Qué acciones debe poder hacer su programa? Enumérelas.**
   - Ingresar la cantidad de gasolineras que se quiere tener al inicio.
   - Generar un número de compras aleatorias por gasolinera (cuantos carros llenaron gasolina)
   - Generar para cada vehículo un número aleatorio de gasolina comprada
   - Recaudar dinero de las gasolineras al momento de pago.
   - Generar reporte diario de dinero adquirido por gasolinera en todo el día, la suma de todas las gasolineras, y el total en la cuenta.
   - En base al dinero en la cuenta, poder adquirir nuevas gasolineras.

b. **¿Con qué va a trabajar (variables y tipos de datos)?**
   gasolineras: **pthread**, cantidad de dinero: **entero**, cargador de gasolinas: **entero**, gasolina para cada carro: **random, de tipo entero**,

c. **¿Qué información debe pedir al usuario?**
   ¿cuántas gasolineras desea generar inicialmente, entre un rango de 1 a 5?

d. **¿Qué cálculos debe hacer?**
   generación aleatoria de compras realizadas en un dia (20 - 200)
   generación aleatoria de valor de las compras realizadas (Q25 - 400)
   sumatoria de ventas diarias por gasolinera
   sumatoria de ventas diarias totales
   sumatoria de estado de cuenta de dinero
   resta de estado de cuenta de dinero por compra de nueva gasolinera

Diseño:

a. **Diagrama de flujo/pseudocódigo preliminar**

// PROTOTIPOS
Funcion gasolinera **Como** puntero Generico (argumento **Como** puntero Generico)
Funcion costoGasolina **Como** puntero Generico (argumento Como puntero Generico)
Funcion carga gasolinera **Como** puntero Generico (argumento Como puntero Generico)

Inicio Gasolinera
    Definir cantidadDias **Como Entero**
    Definir ganancias **Como** Entero (Global)
    Definir cantidad de Gasolineras **Como** entero
    Definir threads[] Como pthread_t
    Definir camionRecargardor[] **Como** pthread_t
    Definir recargarGasolinera **Como** pthread_mutex_t
    Definir recagargandoGasolinera **Como** pthread_cond_t
    Definir i **Como** Entero

Definir cantidadGasolinaRecargar **Como** Entero <- 10000 *// En quetzales*
Definir gasolinaPorGasolinera **Como** Entero <- 2000 *// En Quetzales*
Definir estadisticas **Como Estructura{**
Definir stationID **Como Entero**
Definir carID **Como Entero**
Definir carSpending **Como Entero**
Definir carArriving **Como Entero**
Definir gasAmount **Como Entero**
}

pthread_mutex_init(&recargarGasolinera)
pthread_cond_init(&recargarGasolinera, NULL)

Escribir “Ingrese la cantidad de gasolinas con las cuales desea iniciar para su franquicia”
Leer cantidad de Gasolineras

Para i<- 0 **Hasta** cantidad de Gasolinas **Con paso** 1, **hacer**:
Si pthread_create(&threads[gasolinera recurrente], NULL, NULL) es diferente de 0 **hacer**:
Escribir “Failed to create the thread”
Fin condicional
Si pthread_create(&camion Recargador[gasolinera recurrente], NULL,&cargar gasolinera ,NULL) es diferente de 0, **hacer:**
Escribir “Failed to create the thread”
Fin Si
Fin Para

Para i <0 **Hasta** cantidad de Gasolinera **Con paso** 1, **hacer**:
Si pthread_join(threads[gasolinera recurrente], (void**)&) **hacer:**
Escribir “Failed to join the current thread”
Fin Condicional
Si pthread_join(camionRecargador[camion recargador], NULL) es diferente de 0, **hacer:**
Escribir “Failed to join the current thread”
Fin Condicional
Fin Para
pthread_mutex_destroy(&recargarGasolinera)
pthread_cond_destroy(&recargarGasolinera)
Fin Gasolinera

*/*FUNCION PARA CREAR SIMULAR LA FUNCIONALIDAD DE LAS GASOLINERAS*
** Este hilo, es una gasolinera, donde se generaran vehiculos de manera pseudoaleatoria.*
** Estos vehiculos, seran hilo, y se iran al metodo costoGasolina, donde se genera el precio*
** o cantidad de gasolina adquirida (en quetzales) por cada vehiculo*
**/*
Funcion gasolinera **Como** puntero Generico (argumento **Como** puntero Generico)

Definir cantidad de carros **Como** valor aleatorio de tipo Entero
Definir threads[] **Como** pthread_t
Definir gananciaPorCarro **Como** Entero <- (Entero*)locación de memoria(tamaño de(Entero*))
*gananciaPorCarro = 0;
Definir i **Como** Entero
Para i<-0 **Hasta** cantidad de carros **Con paso** 1, **hacer:**
Si pthread_create(&threads[carro recurrente), NULL, &costoGasolina, gananciaPorCarro) es diferente de cero, **hacer:**
Escribir "Failed to create the thread"
Fin condicional
Fin Para
Fin Funcion

/ FUNCION PARA GENERAR LAS GANANCIAS DE CADA CARRO*
** Esta funcion, obtiene la adquisición de gasolina del carro actual con respecto a su gasolinera.*
** Devuelve su cantidad al metodo gasolinera*
/
Funcion costoGasolina **Como** puntero Generico (argumento Como puntero Generico)
Definir gasolinaAdquirida **Como Entero** <- *(Entero*)argumento // Quetzales
Definir gasolinaPorCarro **Como aleatorio de tipo Entero**
Definir gasolinaAdquirida **Como** Entero (Valor aleatorio)
pthread_mutex_lock(&recargarGasolinera) *// Esta variable, se crear una asi, diferente, para no confundirse*
gasolinaPorGasolinera <- gasolinaPorGasolinera - gasolinaPorCarro
pthread_mutex_unlock(&recargarGasolinera)
pthread_cond_broadcast(&recargandoGasolinera)
cantidadGasolina <- cantidadGasolina - gasolinaAdquirida

(Entero)argumento <- gasolinaPorCarro
retornar argumento

Fin Funcion

/ FUNCION PARA RECARGAR LA GASOLINERA*
** Funcion donde se recarga la gasolinera, en caso de quedarse sin gasolina, simula la llegada de un camion*
** para llegar y recargar la gasolinera*
/
Funcion carga gasolinera **Como** puntero Generico (argumento Como puntero Generico)
pthread_mutex_lock(&recargarGasolinera)
Mientras gasolinaPorGasolinera **mayor a** 0 **hacer:**
pthread_cond_wait(&recargargandoGasolinera,&recargarGasolinera)
Fin mientras

gasolinaPorGasolinera <- gasolinaPorGasolinera + cantidadGasolinaRecargar
pthread_mutex_unlock(&recargarGasolinera)

Fin Funcion

VIDEOS

Avances, código base, sin variables de condición.

https://youtu.be/jjfvnmnMSZY