Sistemas Operativos
2do Semestre 2018
Tarea #3
Semáforos

Wenceslao Palma
<wenceslao.palma@pucv.cl>

Un puente puede soportar como máximo **maxweight** toneladas de peso. El acceso al puente debe ser controlado mediante el uso de dos funciones:

- **entrar_puente(peso)**
- **salir_puente(peso)**

donde **peso** corresponde al peso del vehículo que desea entrar/salir del puente. La solución propuesta debe asegurar que:

- la suma del peso de todos los vehículos presentes en el puente no exceda **maxweight**.
- los vehículos cruzan el puente usando un orden basado en FCFS.

Cada vez que un vehículo desea cruzar el puente se debe crear un proceso hijo usando fork() y **maxweight** debe ser proporcionado al programa como argumento desde la línea de comandos (ver sección 5.2 Kernighan & Ritchie).

ENTRADA
Los datos de los vehículos son proporcionados al programa mediante un archivo de texto llamado **vehiculos.txt** el cual contiene por línea:

- un string de largo 4 que representa la patente del vehículo
- un entero positivo que representa la cantidad de segundos que transcurrieron desde la llegada del vehículo anterior. Dicho valor es 0 para el primer vehículo que desea cruzar el puente.
- un entero positivo que representa el peso en toneladas del vehículo.
- un entero positivo que representa la cantidad de tiempo en segundos que el vehículo demorará en cruzar el puente.

Ejemplo de archivo de entrada:

```
ABGF 0 2 10
ABCD 3 4 20
HGTY 8 6 30
JPOL 7 7 5
KLPO 2 1 15
```

SALIDA
El programa debe mostrar el sgte mensaje cada vez que un vehículo llega al puente, comienza a cruzar el puente y sale del puente:

```
[Vehiculo]: patente ---- [carga actual del puente]: XX toneladas
```

SEMAFOROS
Para trabajar con semáforos se debe seguir la siguiente secuencia: crear semáforo, operaciones sobre el semáforo (wait/signal) y destruir el semáforo. En forma más precisa, usando semáforos POSIX:

- Crear un semáforo.
  int sem_init(sem_t *sem, int pshared, unsigned value); donde **sem** es el semáforo, **pshared** es un argumento cuyo valor es 0 cuando el semáforo es compartido entre los threads de un proceso y **value** es el valor inicial del semáforo.
- Operación wait. **int sem_wait(sem_t *sem);**
- Operación signal. **int sem_post(sem_t *sem);**
- Destruir un semáforo. **int sem_destroy(sem_t *sem);**

```
#include <semaphore.h>

sem_t semaforo;
main(){
    ..............
  sem_init(&semaforo,0,2);
  sem_wait(&semaforo);
  printf("estoy en la sección crítica \n");
  sem_post(&semaforo);
   ..............
}
```

RESTRICCIONES

- La tarea es individual y debe ser codificada utilizando lenguaje ANSI C y semáforos POSIX.
- Sólo se consideran las tareas que cumplan con las especificaciones planteadas.
- La corrección de la tarea incluye una interrogación.
- Sólo se recibirán tareas fuera de plazo dentro de las 24 horas siguientes a la fecha de entrega. Nota máxima es un 5.0

Fecha de entrega : Viernes 30 de Noviembre hasta las 23:59. Enviar código fuente **tarea3-RUTSINDIGITO.c** vía email.
Indique en el código fuente su nombre completo y rut.