Sistemas Operativos

Práctica 3

Curso

2018-2019

Procesos e hilos: planificación y sincronización

Profs. Sistemas Operativos

Índice

Introducción

- Objetivos:
  - Comprender y evaluar los mecanismos de sincronización
- Como objetivo instrumental, el alumno se familiarizará con el uso de:
  - POSIX Threads
  - Semáforos
  - Mutexes y variables condición

Introducción

- Recordad: Procesos vs. Hilos
  - Dos procesos (padre - hijo) *no comparten nada*: se duplica toda la imagen de memoria
  - Dos hilos (mismo proceso) *comparten todo* menos la pila
- Haced y estudiad los ejercicios y ejemplos
  - Ayudan a comprender la materia. . . .
  - . . . y suelen acabar en las cuestiones

Índice

Ejemplo de uso

- Si llega un coche y puente ocupado por otros coches en dirección contraria: esperar al puente esté vacío.
- Si llega un coche y hay menos de tres coches en su mismo sentido, deberá de cruzar inmediatamente.
- Si llega un coche y hay tres coches en su mismo sentido, tiene que esperar a que el primero de ellos abandone el puente para cruzar.
- Puente de tamaño **LENGTH=3**: hasta tres vehículos en el puente y que se tarda en cruzarlo 3 ciclos de reloj.

Ejemplo de uso

Terminal #1

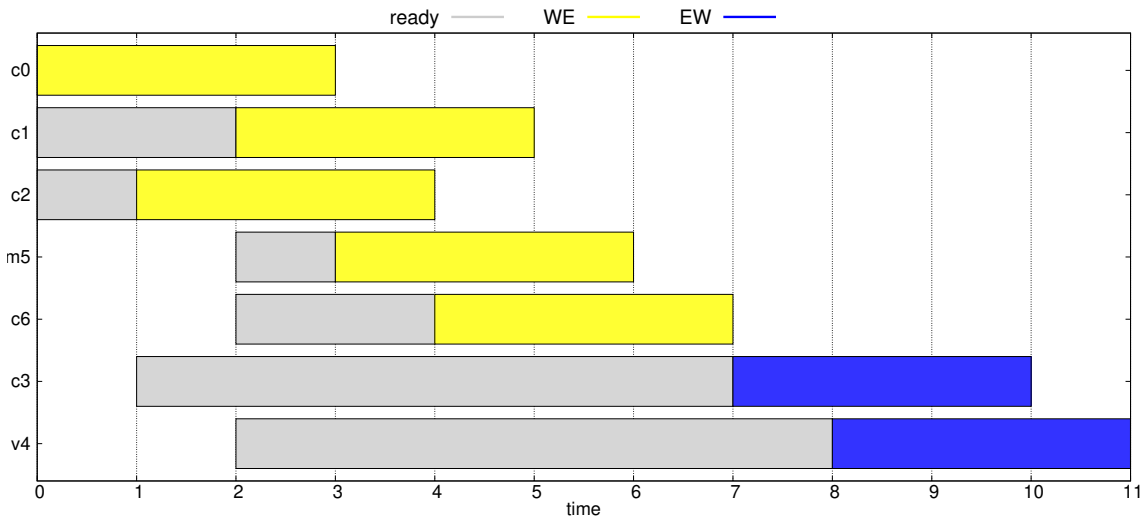
```
$ ./sim examples/example1.txt example1.log
local_time(0)=0
local_time(2)=1
local_time(1)=2
local_time(5)=3
local_time(6)=4
local_time_extra(6)=7
local_time(4)=7
local_time(3)=8

$ cd ../gantt
$ ./generate_gantt_chart ../sim/example1.log
```

Ejemplo de uso

Terminal #1

```
$ cd ../gantt
$ ./generate_gantt_chart ../sim/example1.log
```



Sintaxis de ficheros de vehículo

- En la carpeta *examples* se incluyen varios ejemplos
- Es sencillo construir nuevos ejemplos siguiendo la sintaxis

```
$ cat examples/example1.txt
0 c 0 1
1 c 0 1
2 c 0 1
3 c 1 0
4 v 2 0
5 m 2 1
6 c 2 1
```

- Columna 1: id del vehículo
- Columna 2: tipo de vehículo
- Columna 3: tiempo de comienzo
- Columna 4: sentido del vehículo

Índice

Características generales

- Simulador diseñado emular comportamiento del puente
  - Si llega un coche y puente ocupado por otros coches en dirección contraria: esperar al puente esté vacío.
  - Si llega un coche y hay menos de tres coches en su mismo sentido, deberá de cruzar inmediatamente.
  - Si llega un coche y hay tres coches en su mismo sentido, tiene que esperar a que el primero de ellos abandone el puente para cruzar.
    - En un *tick* se considera que se cruza el puente.

Índice

Estructura tcar

tcar

```
typedef struct {
  int id;
  char type; // C=car, M=minivan, V=van
  int my_direction; // 0, 1
  int t_arrival;
  int t_cross_in;
  int t_cross_out;
} tcar;
```

la estructura *tcar* cuyos campos están relacionados con:

- *id*: identificador del vehículo, descrito con un número entero único.
- *type*: tipo de vehículo descrito con un carácter (c: coche, v: furgoneta y m: monovolumen)
- *my_direction*: dirección del vehículo descrita con un 0: este/oeste o 1: oeste/este.
- *t_arrival*: momento de llegada al puente.
- *t_cross_in*: momento en el que comienza a cruzar el puente.
- *t_cross_out*: momento en el que acaba de a cruzar el puente.

Estructura tbridge

tbridge

```
#define EMPTY -1

typedef struct {
  pthread_mutex_t mtx;
  pthread_cond_t VCs[2];
  int cars_on_bridge;
  int cur_direction;
  int cars_waiting[2];
} tbridge;
```

la estructura *tbridge* cuyos campos corresponden a:

- *mtx*: cerrojo que permite acceder a los campos de la estructura de forma exclusiva.
- *VCs*: dos variables condicionales que implementan las colas donde los vehículos esperan cuando el puente está ocupado (una para la dirección este/oeste y otra para la oeste/este).
- *cars_on_bridge*: número de coches que están accediendo al puente, es decir cruzándolo.
- *cur_direction*: sentido de circulación actual del puente: este/oeste o viceversa.
- *cars_waiting*: número de coches esperando en ambos sentidos.

Índice

Parte obligatoria

Cambios en el código C

- Desarrollar la funcionalidad del puente, rellenando las funciones *bridge_in(tcar *dcar)* y *bridge_out(tcar *dcar)*.
- Implementar una barrera de sincronización no reentrante usando cerrojos y VC
  - Completar el fichero `barrier.c` (funciones `sys_barrier_init()`, `sys_barrier_destroy()` y `sys_barrier_wait()` de la rama `#else`)
  - Cuidado con los despertares espurios de la llamada `pthread_cond_wait` (usar `while`)
  - Modificar el *Makefile* para evitar que se declare la macro `POSIX_BARRIER`

Script shell

- No recibirá argumentos de entrada pero solicitará al usuario:
  - Nombre del fichero de ejemplo que se desea simular
- Se simulará el ejemplo y se generarán las gráficas correspondientes