PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

2da práctica (tipo a)
(Primer semestre de 2020)

Horario 0781: prof. V. Khlebnikov
Horario 0782: prof. A. Bello R.

Duración: 1 h. 50 min.
Nota: **La presentación, la ortografía y la gramática influirán en la calificación.**
Puntaje total: 20 puntos

Pregunta 1 (6 puntos – 30 min.) Su respuesta debe estar en el Campus Virtual (Documentos del curso, Prácticas, Práctica 2, 0781/0782) **antes de las 11:40**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_21.txt`. Por ejemplo, `20171903_21.txt`.

Encuentre su código de estudiante y la secuencia única de números correspondiente en la siguiente tabla:

```
 1. 20092337: (2, 2, 3, 9)
 2. 20092344: (2, 2, 2, 9)
 3. 20102355: (2, 2, 3, 2)
 4. 20105044: (2, 2, 2, 2)
 5. 20105170: (3, 2, 4, 2)
 6. 20110897: (2, 2, 2, 3)
 7. 20114263: (3, 2, 2, 2)
 8. 20121589: (3, 2, 2, 3)
 9. 20125121: (3, 2, 4, 3)
10. 20125682: (4, 2, 3, 3)
11. 20131994: (3, 2, 2, 4)
12. 20135386: (2, 2, 3, 4)
13. 20140669: (3, 2, 4, 5)
14. 20142910: (4, 2, 2, 5)
15. 20150169: (2, 2, 4, 6)
16. 20150938: (2, 2, 4, 5)
17. 20151887: (2, 2, 4, 4)
18. 20151939: (3, 2, 2, 6)
19. 20152195: (4, 2, 3, 6)
20. 20152892: (3, 2, 4, 6)
21. 20155470: (2, 2, 3, 3)
22. 20155777: (2, 2, 3, 6)
23. 20155800: (2, 2, 2, 6)
24. 20155869: (2, 2, 3, 5)
25. 20156077: (3, 2, 3, 6)
26. 20156106: (4, 2, 3, 5)
27. 20156107: (4, 2, 4, 6)
28. 20156134: (4, 2, 4, 5)
29. 20161008: (2, 2, 2, 7)
30. 20161093: (2, 2, 2, 5)
31. 20161478: (2, 2, 4, 7)
32. 20161496: (3, 2, 4, 7)
33. 20161515: (3, 2, 3, 7)
34. 20161589: (3, 2, 2, 7)
35. 20161677: (4, 2, 3, 7)
36. 20167441: (4, 2, 2, 7)
37. 20167474: (4, 2, 4, 7)
38. 20170516: (2, 2, 4, 8)
39. 20170569: (3, 2, 4, 4)
40. 20170603: (3, 2, 4, 8)
41. 20170824: (2, 2, 3, 8)
42. 20170942: (2, 3, 3, 6)
43. 20171051: (2, 3, 3, 7)
44. 20171702: (2, 3, 3, 8)
45. 20171903: (3, 2, 3, 3)
```

La interpretación de los números en la secuencia es la siguiente:

El 1er número es la cantidad de los procesos lectores, el 2do número es la duración de ejecución de cada lector, el 3er número es la cantidad de los escritores y el 4to número es la duración de ejecución de cada escritor. La duración de ejecución es la duración de ejecución de la función `DB_access()` porque el tiempo de ejecución de la otras sentencias es insignificante.

Por ejemplo, la secuencia `(4, 2, 3, 5)` significa que hay 4 lectores con el tiempo de ejecución de 2 segundos y hay 3 escritores con el tiempo de ejecución de 5 segundos. Los lectores llegan al sistema en los segundos pares: T=0 (1er lector), T=2 (2do lector), T=4 (3er lector), etc. Mientras que los escritores llegan al sistema en los segundos impares: T=1 (1er escritor), T=3 (2do escritor), etc. Se propone el siguiente algoritmo:

```
int a=0, b=0;
semaphore u=1, w=1, x=1, y=1, z=1;

void reader() {
        while (true) {
                down(&x);
                down(&y);
                down(&u);
                a++;
                if (a == 1) down(&z);
                up(&u);
                up(&y);
                up(&x);
                DB_access();
                down(&u);
                a--;
                if (a == 0) up(&z);
                up(&u);
        }
}
void writer() {
        while (true) {
                down(&w);
                b++;
                if (b == 1) down(&y);
                up(&w);
                down(&z);
                DB_access();
                up(&z);
                down(&w);
                b--;
                if (b == 0) up(&y);
                up(&w);
        }
}
```

Entonces, la ejecución de nuestros procesos para la secuencia `(4, 2, 3, 5)` será la siguiente:

Tiempo	Proceso	Estado final	Semáforo u	Semáforo w	Semáforo x	Semáforo y	Semáforo z	a	b
			1	1	1	1	1	0	0
T=0:	R1	en DB_access	1 -> 0 -> 1		1 -> 0 -> 1	1 -> 0 -> 1	1 -> 0	1	0
T=1:	W1	bloqueado en ...		1 -> 0 -> 1		1 -> 0			1
T=2:	R1	terminado	1 -> 0 -> 1				...	0	
	R2	...							
	W1	...							
T=3:	...	...							
...									

Complete y presente esta tabla para todos los procesos indicando el estado final: `en DB_access`, `bloqueado en ...` (indique el nombre del semáforo) o `terminado`. También indique los valores finales de semáforos o sus cambios si los valores se cambian o los cambios suceden. El formato de tabla pueden no mantener, suficiente separar las columnas con ';', por ejemplo:

```
T=0: R1; en DB_access; u=1,0,1; x=1,0,1; y=1,0,1; z=1,0; a=1
T=1: W1; bloqueado en ...; w=1,0,1; y=1,0; b=1
...
```

Pregunta 2 (6 puntos – 30 min.) Su respuesta debe estar en el Campus Virtual (Documentos del curso, Prácticas, Práctica 2, 0781/0782) **antes de las 12:20**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>_22.txt. Por ejemplo, 20171903_22.txt.

Se tiene cinco procesos (P1, P2, P3, P4, P5) que llegan de forma simultánea para ser planificados, cada uno de ellos presenta la siguiente distribución de tiempos: 1000 udt. de CPU, 10 udt. de disco y finalmente 2000 udt. de CPU. Si el planificador aplica el algoritmo Round Robin con un *quantum* de 10 udt., ¿cuál es el tiempo de espera de cada uno de los procesos? Usted debe justificar su respuesta.

Nota: udt. significa unidad de tiempo.

Pregunta 3 (8 puntos – 30 min.) Su respuesta debe estar en el Campus Virtual (Documentos del curso, Prácticas, Práctica 2, 0781/0782) **antes de las 13:00**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>_23.txt. Por ejemplo, 20171903_23.txt.

a) (4 puntos) Se tiene dos procedimientos. El primero se denomina *izqDer* que imprime: `----->` y el otro procedimiento denominado *derIzq* que imprime: `<-----`. Usted debe desarrollar el código del **monitor** *Crossing* con los procedimientos *izqDer* y *derIzq* sincronizando el total de hilos, para ejecutar ambos procedimientos con alternancia estricta. Usted no conoce el número de hilos creados, solo sabe que ambos procedimientos serán ejecutados de forma concurrente, por igual número de hilos.

Como un ejemplo de la estructura de un monitor, se le proporciona el código presentado en el libro de texto (Figure 2-34):

```
monitor ProducerConsumer
    condition full, empty;
    integer count;

    procedure insert(item: integer);
    begin
        if count = N then wait(full);  /* The buffer has N slots. */
        insert_item(item);
        count := count + 1;
        if count = 1 then signal(empty)
    end;

    function remove: integer;
    begin
        if count = 0 then wait(empty);
        remove = remove_item;
        count := count - 1;
        if count = N - 1 then signal(full)
    end;

    count := 0;
end monitor
```

b) (4 puntos) This figure shows one way in which message passing can be used to enforce mutual exclusion:

```
/* program mutualexclusion */

const int n = /* number of process */

void
P(int i)
{
        message msg;
        while (true)
                receive (box, msg);
                /* critical section */
                send (box, msg);
                /* remainder */
        }
}
void
main()
{
        create mailbox (box);
        send (box, null);
        parbegin (P(1), P(2), …, P(n));
}
```

We assume the use of the blocking `receive` primitive and the nonblocking `send` primitive. A set of concurrent processes share a mailbox `box`, which can be used by all processes to send and receive.

The mailbox is initialized to contain a single message with null content. A process wishing to enter its critical section first attempts to receive a message. If the mailbox is empty, then the process is blocked. Once a process has acquired the message, it performs its critical section and then places the message back into the mailbox. Thus, the message functions as a token that is passed from process to process.

The preceding solution assumes that if more than one process performs the receive operation concurrently, then:

- If …
- If …

Complete el texto.

La práctica ha sido preparada por AB (2,3a) y VK (1,3b)
con LibreOffice Writer en Linux Mint 19.3 Tricia.

Profesores del curso: (0781) V. Khlebnikov
(0782) A. Bello R.

Lima, 29 de mayo de 2020