PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
2da práctica (tipo a)
(Segundo semestre de 2018)

Horario 0781: prof. V. Khlebnikov
Horario 0782: prof. F. Solari A.

Duración: 1 h. 50 min.
Nota: No se puede usar ningún material de consulta.
La presentación, la ortografía y la gramática influirán en la calificación.
Puntaje total: 20 puntos

Pregunta 1 (5 puntos – 25 min.) Se pretende la organización del trabajo intercalado de 2 hilos creados con los identificadores `th_flip` y `th_flop`, ejecutando las funciones respectivas `thread_flip()` y `thread_flop()`. Para sincronización se usan las funciones `thr_continue()` y `thr_suspend()`:

`thr_suspend()` immediately suspends the execution of the thread specified by target_thread.
On successful return from `thr_suspend()`, the suspended thread is no longer executing.
Once a thread is suspended, subsequent calls to `thr_suspend()` have no effect.
Signals cannot awaken the suspended thread; they remain pending until the thread resumes execution.

`thr_continue()` resumes the execution of a suspended thread.
Once a suspended thread is continued, subsequent calls to `thr_continue()` have no effect.

Al terminar su parte de trabajo, cada hilo despierta al otro y suspende a sí mismo:

```
...
thread_t th_flip, th_flop;
...

void * thread_flip(void *arg) {
    ...
    while (1) {
        ...
        thr_continue(th_flop);
        thr_suspend(th_flip);
    }
}
```

```
void * thread_flop(void *arg) {
    ...
    while (1) {
        thr_suspend(th_flop);
        ...
        thr_continue(th_flip);
    }
}
```

Encuentre *race condition* en este código.

Pregunta 2 (5 puntos – 25 min.) (*Busy Waiting with Hardware Instructions help*) Los procesadores Intel, de tipo IA-32 (antigua iAPX86, conocidos también como x86 y x86_64), no tienen exactamente una instrucción TSL, pero si una instrucción XCHG que permite intercambiar el contenido de un registro con una posición de memoria. Entonces, se plantea el siguiente código para una función `void AcquireLock (int *lock)()` y la respectiva `void ReleaseLock (int *lock)()`:

```
void AcquireLock (int *lock) {
  asm {
    L1: MOV  eax,#00000001h     ; colocar valor 1 en el registro 'eax' de 32 bits del CPU
        XCHG eax,[lock] ; intercambiar contenido del registro eax con contenido de variable apuntada por lock
        CMP  eax,#0h            ; comparar contenido del registro 'eax' con valor 0
        JNE  L1                 ; saltar si no fue igual a 0 a L1
        RET                     ; retorno de la función
  }
}
```

a) (1 punto) Escriba el código para `void ReleaseLock (int *lock)()` explicando el significado.

b) (2 puntos) Explique qué ocurre cuando dos procesos o hilos que comparten el acceso a *lock* invocan *AcquireLock()* concurrentemente, mediante alguna secuencia que evidencie si se cumple el propósito de solución o no.

c) (1 punto) Haga un esquema en el tiempo (gráfica de Gantt) que muestre la secuencia de ejecución de dos procesos o hilos que utilizan *AcquireLock()* y *ReleaseLock()* **concurrentemente** como en **b)**. Considere que el tiempo de ejecución de la *sección crítica* protegida por estas es relativamente más largo que el tiempo máximo de CPU asignado al proceso o hilo.

d) (1 punto) ¿Cómo podría modificarse el código *AcquireLock()* de manera que no se haga espera activa o *busy-wait*?

Pregunta 3 **(5 puntos – 25 min.)** Binary semaphore properties. Assumes only values 0 or 1. Wait blocks if semaphore = 0. Signal (up operation) either wakes up a waiting process, if there is one, or sets value to 1 (if value is already 1, signal is "wasted"). How can we implement a counting semaphore by using binary semaphores?

Implementing a counting semaphore `S` with binary semaphores `S1` and `S2` (user space):

```
binary-semaphore S1 = 1, S2 = 0
counting-semaphore S
S.value = 1

    down(S):                          up(S):

        down(S1);                         down(S1);
        S.value--;                        S.value++;
        if (S.value < 0) {                if (S.value <= 0)
            L1: up(S1);                       up(S2);
            L2: down(S2);                 up(S1);
        } else
            up(S1);
```

This code does not work.
1. Processes P1–P4 perform `down(S)`, P2–P4 are preempted between lines L1 and L2: the value of the counting semaphore is now -3.
2. Processes P5–P7 now perform `up(S)`: the value of the counting semaphore is now 0.
3. Now, P2–P4 wake-up in turn and …
Complete el desarrollo.

Pregunta 4 **(5 puntos – 25 min.)** (El siguiente código corresponde a una función que será invocada desde un programa principal para ser ejecutada por dos hilos concurrentes.

```
void thread_thing(int *pnum_times) {
  int i, j, x;

  for (i = 0;  i < 4; i++) {
            printf("doing one thing\n");
            for (j = 0; j < 10000; j++) x = x + i;
            (*pnum_times)++;
          }
}
```

a) (1 punto – 5 min.) Para el caso en que a cada hilo se le indica como argumento la dirección de una variable global diferente, digamos "&r1" y "&r2", siendo r1=r2=0 inicialmente. ¿Cuál sería el resultado de r1+r2? Justifique su respuesta.

b) (2 puntos – 10 min.) Analice, ¿que ocurriría si en el programa principal se indica a ambos hilos la misma dirección, digamos "&r" de una variable global? ¿Cuál debería ser el resultado correcto de "r" ? Justifique su respuesta.

Se le pide a algún estudiante del curso que agregue lo necesario para que el resultado al utilizarse la misma variable global en ambos hilos, sea el resultado correcto, presentando la siguiente solución:

```
void thread_thing(int *pnum_times) {
  int i, j, x;

  pthread_mutex_lock(s);
  for (i = 0;  i < 4; i++) {
            printf("doing one thing\n");
            for (j = 0; j < 10000; j++) x = x + i;
            (*pnum_times)++;
          }
  pthread_mutex_unlock(s);
}
```

c) (1 punto – 5 min.) ¿El resultado será el esperado? ¿Cómo será la ejecución concurrente de los hilos en este caso?
d) (1 punto – 5 min.) ¿Qué cambios en el código deben hacerse para que cumpla los criterios de una buena solución?

La práctica ha sido preparada por FS(2,4) y VK(1,3)
con LibreOffice Writer en Linux Mint 19 Tara.

Profesor del curso: (0781) V. Khlebnikov
(0782) F. Solari A.

Pando, 28 de septiembre de 2018