PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
2da práctica (tipo a)
(Segundo semestre de 2013)

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.) (*Estados de los procesos*) Considere las siguientes transiciones de estados de procesos / hilos:

i)	Null	→	New
ii)	New	→	Ready
iii)	Running	→	Ready
iv)	Ready	→	Running
v)	Blocked	→	Ready
vi)	Running	→	Blocked
vii)	Running	→	Exit
viii)	Ready	→	Exit
ix)	Blocked	→	Exit

a) (2 puntos – 10 min.) ¿Qué descripción corresponde a cada una de las transiciones dadas?

b) (1 punto – 5 min.) Haga un esquema que muestre los estados de los procesos y las transiciones correspondientes.

c) (2 puntos – 10 min.) La CPU siempre buscará instrucciones para ejecutar, en su ciclo "fetch", "execute". Pero dado que los procesos pueden hacer I/O simultánea, hacia el mismo o varios dispositivos, ¿qué ocurre si todos los procesos del usuario están en estado ***blocked***? Plantee lo que debería considerarse en el sistema operativo para esta situación.

Pregunta 2 (5 puntos – 25 min.) (*Common Weakness Enumeration*) This code could be used in an e-commerce application that supports transfers between accounts. It takes the total amount of the transfer, sends it to the new account, and deducts the amount from the original account.

```
1     $transfer_amount = GetTransferAmount();
2     $balance = GetBalanceFromDatabase();
3
4     if ($transfer_amount < 0) { FatalError("Bad Transfer Amount"); }
5
6     $newbalance = $balance - $transfer_amount;
7     if (($balance - $transfer_amount) < 0) { FatalError("Insufficient Funds"); }
8
9     SendNewBalanceToDatabase($newbalance);
10    NotifyUser("Transfer of $transfer_amount succeeded.");
11    NotifyUser("New balance: $newbalance");
```

A race condition could occur between … **(a) where? 1 punto – 5 min.)**

Suppose the balance is initially 100.00. An attack could be constructed as follows:
The attacker makes two simultaneous calls of the program, CALLER-1 and CALLER-2. Both callers are for the same user account.
CALLER-1 (the attacker) is associated with PROGRAM-1 (the instance that handles CALLER-1).
CALLER-2 is associated with PROGRAM-2.
CALLER-1 makes a transfer request of 80.00.
…
CALLER-2 makes a transfer request of 1.00.
…

b) (1 punto – 5 min.) ¿Cuál será el valor final del balance?

c) (3 puntos – 15 min.) Presente el escenario (la secuencia de ejecución y los valores de las variables) para cada valor final posible usando la notación P1.6 (newbalance = ...), por ejemplo, para indicar la ejecución de la línea 6 del Programa 1.

Pregunta 3 (5 puntos – 25 min.) (*M. Raynal*) Dijkstra (1965) generalized Dekker's solution to the case of *n* processes. The variables shared between the *n* processes $P_0, \ldots, P_i, \ldots, P_{n-1}$ are:

```
var flag : array [0 .. n - 1] of (passive, requesting, in-cs);
    turn : 0 .. n - 1;
```

The elements of `flag` are initialized to `passive`, and `turn` takes some arbitrary value. Each process P_i has an integer variable `j`.

```
 1      repeat
 2          flag [i] ← requesting;
 3              while turn ≠ i do if flag [turn] = passive
 4                                then turn ← i
 5                                endif ;
 6                             enddo ;
 7          flag [i] ← in-cs ;
 8          j ← 0;
 9          while (j < n) ∧ (j = i ∨ flag[j] ≠ in-cs) do j ← j + 1 enddo ;
10      until j ≥ n ;
11      < critical section > ;
12      flag [i] ← passive ;
```

a) (4 puntos – 20 min.) Presente la ejecución de tres procesos en una alternancia perfecta hasta que 2 procesos entren en sus secciones críticas.

b) (1 punto – 5 min.) ¿Qué es lo que no garantiza este algoritmo?

Pregunta 4 (5 puntos – 25 min.) (*Semaphores – William Stallings 2ed. 4.14, 4ed. 5.13*) Debe ser posible implementar semáforos generales por medio de semáforos binarios. Se pueden usar las operaciones `WaitB` y `SignalB`, y dos semáforos binarios `espera` y `exmut`. Considérese lo siguiente:

```
procedure Wait(var s: semaphore);
   begin
      WaitB(exmut);
      s:=s-1:
      if s<0 then
         begin
            SignalB(exmut);
            WaitB(espera);
         end
      else
         SignalB(exmut);
   end;

procedure Signal(var s: semaphore);
   begin
      WaitB(exmut);
      s:=s+1;
      if s<=0 then SignalB(espera);
      SignalB(exmut);
   end;
```

a) (2 puntos – 10 min.) Explique el uso de los semáforos `exmut`, y `espera`, en cada uno de los procedimientos.

b) (2 puntos – 10 min.) El programa anterior tiene un defecto. Describa una secuencia de ejecución que lo muestre.

c) (1 punto – 5 min.) ¿Cómo puede solucionarse el defecto?

La práctica ha sido preparada por FS (1,4) y VK(2,3).

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

Pando, 24 de septiembre de 2013