Víctor Andrés Ruiz Alcalá
76089860-H

SISTEMAS OPERATIVOS
Primera Prueba de Evaluación a Distancia (PED 1)

1.- Explique razonadamente si las siguientes afirmaciones son verdaderas o falsas:

I) El principal inconveniente de los monitores es que solo se pueden utilizar para garantizar la exclusión mutua.

Los monitores, al igual que los semáforos, se pueden utilizar para garantizar la exclusión mutua y para la sincronización de procesos. Por tanto, la afirmación es FALSA.

II) La magnitud del tiempo de conmutación entre procesos depende principalmente de factores asociados al hardware.

Si bien es cierto que la magnitud del tiempo de conmutación no depende exclusivamente del hardware, sí que depende principalmente de factores asociados al hardware. Por tanto, la afirmación es VERDADERA.

III) El algoritmo de planificación SJF minimiza el tiempo de espera medio de un conjunto dado de procesos.

El algoritmo de planificación SJF favorece a los procesos cortos frente a los procesos largos de tal forma que el tiempo de espera de los procesos cortos disminuye más de lo que se aumenta el tiempo de espera en los procesos largos, decrementando así el tiempo de espera medio. Por tanto, la afirmación es VERDADERA.

IV) Las llamadas al sistema requieren siempre de una conmutación hardware de modo usuario a modo supervisor.

Las llamadas al sistema solo las pueden hacer programas que se están ejecutando en modo usuario y que necesitan utilizar alguna instrucción exclusiva del modo supervisor, lo que implica necesariamente una conmutación hardware. Por tanto, la afirmación es VERDADERA.

2.- Determinar cuál debe ser la duración de las ráfagas x de CPU del conjunto de trabajos que se muestran en la siguiente tabla sabiendo que el tiempo de espera de los trabajos T1 y T2 fue de 5 ut, en ambos casos, para el trabajo T3 su tiempo de espera coincide con su tiempo de servicio, y que el tiempo de estancia medio en el sistema fue de 20 ut.

Trabajo	Duración de las ráfagas (ut)
T1	x, 5
T2	3, x, 10
T3	4, x

De los datos proporcionados por la tabla y por el enunciado podemos extraer las siguientes conclusiones acerca del tiempo de respuesta (TR) de cada trabajo:

$TR1 \equiv (x + 5) + 5$, correspondientes a sus ráfagas de CPU y a su tiempo de espera.

$TR2 \equiv (3 + x + 10) + 5$, correspondientes a sus ráfagas de CPU y a su tiempo de espera.

$TR3 \equiv (4 + x) + (4 + x)$, correspondientes a sus ráfagas de CPU y a su tiempo de espera.

Teniendo en cuenta que el tiempo de estancia medio fue de 20 unidades de tiempo y que se realizaron 3 trabajos, entonces el tiempo de respuesta total fue de:

$TR\ Total \equiv TR1 + TR2 + TR3 = 20 \cdot 3 = 60$

Si igualamos las expresiones y despejamos el valor de x:

$TR\ Total \equiv (x + 5) + 5 + (x + 3 + 10) + 5 + (4 + x) + (4 + x) = 60$

$TR\ Total \equiv (4x + 36) = 60 \Rightarrow 4x = 24 \Rightarrow x = 6$

Por tanto, las ráfagas x de CPU duraron 6 unidades de tiempo.

3.- Una persona tiene en su casa una jaula llena de canarios en la que hay un plato de alpiste y un columpio. Todos los canarios quieren primero comer del plato y luego columpiarse, sin embargo sólo tres de ellos pueden comer del plato al mismo tiempo y solo uno de ellos puede columpiarse.

Escribir el pseudocódigo basado en C de un programa que usando un monitor de nombre jaula coordine la actividad de los canarios. Considerar la solución de Hansen en el comportamiento de la operación signal_mon.

```
1   #define P 3 /* Número máximo de Canarios comiendo del Plato */
2   #define C 1 /* Número máximo de Canarios montados en el Columpio */
3
4   monitor jaula
5
6       /* Declaración de las variables locales */
7       int p, c, contadorP, contadorC, plato[P], columpio[C];
8
9       /* Declaración de las variables de condición */
10      condición platoDisponible, columpioDisponible;
11
12      /* Declaración de los procedimientos del monitor */
13      int obtenerPlato (int PID) {
14          int n = 0;
15          if (contadorP == P) wait_mon (platoDisponible);
16          for (n; p < P; n++) {
17              if (plato[p] == -1) {
18              contadorP++;
19              plato[n]=PID;
20              return n;
21              }
22          }
23      }
24
25      void dejarPlato (int numPlato) {
26          contadorP--;
27          plato[numPlato] = -1;
28          signal_mon(platoDisponible);
29      }
30
31      int obtenerColumpio (int PID) {
32          int n = 0;
33          if (contadorC == C) wait_mon (columpioDisponible);
34          for (n; c < C; n++) {
35              if (columpio[c] == -1) {
36              contadorC++;
37              columpio[n]=PID;
38              return n;
39              }
40          }
41      }
42
```

```
43      void dejarColumpio (int numColumpio) {
44          contadorC--;
45          columpio[numColumpio] = -1;
46          signal_mon(columpioDisponible);
47      }
48
49      /* Inicialización del monitor */
50      contadorP = 0
51      contadorC = 0;
52      for(int n = 0; n < P; n++) {
53              plato[n] = -1;
54      }
55      for(int n = 0; n < C; n++) {
56              plato[n] = -1;
57      }
58
59  end monitor
60
61  void canario() {
62      int PID, numPlato, numColumpio;
63
64      PID = obtenerPID();
65      numPlato = jaula.obtenerPlato(PID);
66      comer();
67      jaula.dejarPlato(numPlato);
68      numColumpio = jaula.obtenerColumpio(PID);
69      columpiarse();
70      jaula.dejarColumpio(numColumpio);
71  }
72
73  void main () {
74      ejecucion_concurrente (canario, ... , canario);
75  }
```

Obsérvese que, si bien podría haber simplificado un poco el diseño del columpio, he preferido dejarlo de una forma más general porque así, si en algún momento decido poner uno o más columpios adicionales, bastaría con cambiar el valor de la constante C.