Alejandro López Sepúlveda

1. Sombrea la respuesta correcta (0,25 c/u).

1) ¿Con qué comando se accede a los procesos en Windows /Linux?
a) Processlist/process
b) Tasklist/ps
c) Tasklist/process
d) Process/ps

2) ¿Qué información nos muestra PID?
a) La ID del proceso padre
b) ID del proceso
c) El procesador que tiene asignado el proceso
d) Ninguna de las anteriores

3) ¿Qué información nos da PPID?

a) **La ID del proceso padre**

b) ID del proceso

c) El procesador que tiene asignado el proceso

d) Ninguna de las anteriores

4) ¿Qué información nos da STIME
a) Tiempo que lleva ejecutándose un proceso
b) Hora a la que empezó a ejecutarse el proceso
c) Nos permite hacer un Set Time
d) Ninguna de las anteriores

2. Explica el estado de los procesos, sus posibles transacciones y haz el gráfico. (1 punto).

Los estados de los procesos son:

Ejecución: Este estado ocurre cuando el proceso está utilizando el procesador para realizar sus tareas.

Bloqueado: En este estado, el proceso no puede funcionar hasta que reciba una entrada de otro proceso.

Listo: Aquí, el proceso está en cola, esperando su oportunidad para usar el procesador y ejecutarse.

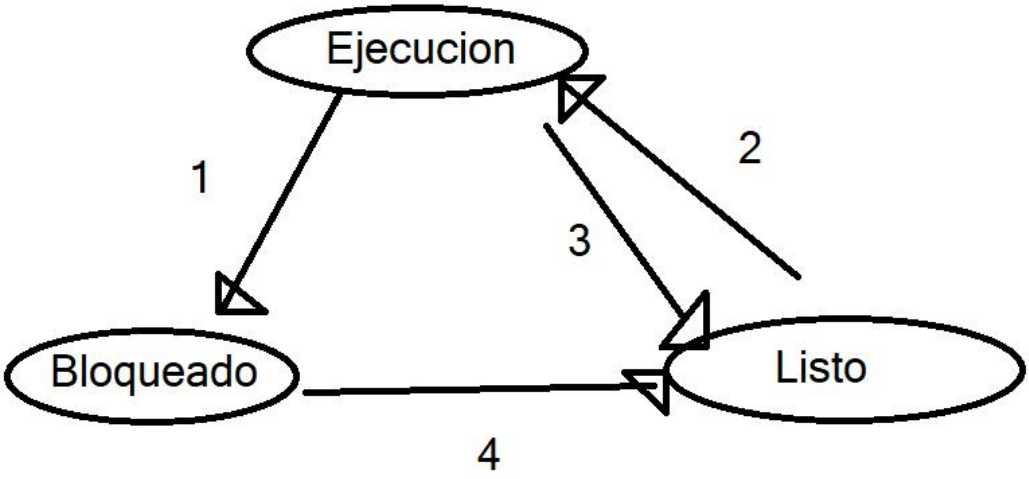
Posibles transacciones:

1. De Ejecución a Bloqueado: Si el proceso esta en ejecución necesita esperar por una orden o evento. Mientras está en este estado, no puede continuar con su ejecución.

2. De Listo a Ejecución: Cuando un proceso esta listo es seleccionado por el sistema operativo, pasa al estado de ejecución, donde utiliza el procesador para llevar a cabo sus tareas.

3. De Ejecución a Listo: Si el proceso en ejecución es interrumpido por el sistema operativo para dar paso a otro proceso, regresa al estado de listo, esperando su próxima oportunidad para utilizar el procesador.

4. De Bloqueado a Listo: Cuando la orden o evento que el proceso estaba esperando se vuelve disponible, pasa del estado bloqueado al estado de listo, y esperará de nuevo su turno para ejecutarse.



3. Escribe un código en Java que permita abrir Chrome. (2 puntos).

```
package examenpsp2_alopez_1examen;

import java.io.IOException;

/*
* Alejandro López Sepúlveda
*/
```

```
public class AbrirChrome {


public static void main(String[] args) {

try {
ProcessBuilder pb = new ProcessBuilder("C:\\Program
Files\\Google\\Chrome\\Application\\chrome.exe");
Process p = pb.start();
} catch (IOException e) {
System.out.println("No se puedo abrir Google Chrome.");
e.printStackTrace();
}
}
}
```

4. Escribe un código en Java que nos muestre por la consola de Eclipse los procesos que tengamos abiertos. (2 puntos).

```
package examenpsp2_alopez_1examen;

import java.io.IOException;
import java.io.InputStream;

public class MostrarProcesos {

public static void main(String[] args) {
// TODO Auto-generated method stub

ProcessBuilder pbCMD = new ProcessBuilder("CMD","/C","tasklist");
Process pCMD = null;
try {
pCMD = pbCMD.start();
} catch (IOException e) {
e.printStackTrace();
}
InputStream is = pCMD.getInputStream();
int c;
try {
while((c=is.read()) != -1) {
System.out.print((char) c);
}
} catch (IOException e) {
e.printStackTrace();
}
try {
is.close();
} catch (IOException e) {
e.printStackTrace();
}
}

}
```

5. Crea el cuadro de Bernstein e indica qué instrucciones son concurrentes. (2 puntos).

Instrucción 1: p1=a*square;
Instrucción 2: p2=b*x;
Instrucción 3: square=x*x;
Instrucción 4: z= m1 + m2;
Instrucción 5: y= z + c;

Instrucciones (I)	Escritura (E)	Lectura (L)
I1	p1	a*square
I2	p2	b*x
I3	square	x*x
I4	z	m1 + m2
I5	y	z + c

Escritura-Lectura	Escritura-Escritura	Lectura-Escritura	¿Es concurrente?
E1 ∩ L2= Ø	E1 ∩ E2= Ø	L1 ∩ E2= Ø	Si
E1 ∩ L3= Ø	E1 ∩ E3= Ø	L1 ∩ E3≠ Ø	No
E1 ∩ L4= Ø	E1 ∩ E4= Ø	L1 ∩ E4= Ø	Si
E1 ∩ L5= Ø	E1 ∩ E5= Ø	L1 ∩ E5= Ø	Si
E2 ∩ L1= Ø	E2 ∩ E1= Ø	L2 ∩ E1= Ø	Si
E2 ∩ L3= Ø	E2 ∩ E3= Ø	L2 ∩ E3= Ø	Si
E2 ∩ L4= Ø	E2 ∩ E4= Ø	L2 ∩ E4= Ø	Si
E2 ∩ L5= Ø	E2 ∩ E5= Ø	L2 ∩ E5= Ø	Si
E3 ∩ L1≠Ø	E3 ∩ E1≠ Ø	L3 ∩ E1= Ø	No
E3 ∩ L2= Ø	E3 ∩ E2= Ø	L3 ∩ E2= Ø	Si
E3 ∩ L4= Ø	E3 ∩ E4= Ø	L3 ∩ E4= Ø	Si
E3 ∩ L5= Ø	E3 ∩ E5= Ø	L3 ∩ E5= Ø	Si
E4 ∩ L1= Ø	E4 ∩ E1= Ø	L4 ∩ E1= Ø	Si
E4 ∩ L2= Ø	E4 ∩ E2= Ø	L4 ∩ E2= Ø	Si
E4 ∩ L3= Ø	E4 ∩ E3= Ø	L4 ∩ E3= Ø	Si

E4 ∩ L5≠ Ø	E4 ∩ E5≠ Ø	L4 ∩ E5= Ø	No
E5 ∩ L1= Ø	E5 ∩ E1= Ø	L5 ∩ E1= Ø	Si
E5 ∩ L2= Ø	E5 ∩ E2= Ø	L5 ∩ E2= Ø	Si
E5 ∩ L3= Ø	E5 ∩ E3= Ø	L5 ∩ E3= Ø	Si
E5 ∩ L4= Ø	E5 ∩ E4= Ø	L5 ∩ E4≠ Ø	No

6. Desarrolla las ventajas de la programación concurrente en los monoprocesadores. (1 punto).

Mejora la Capacidad de Respuesta: Permite al sistema manejar múltiples tareas simultáneamente.

Utilización Eficiente del CPU: Asegura que el CPU esté activo incluso cuando un proceso está esperando.

Escalabilidad: Un diseño concurrente puede adaptarse fácilmente a sistemas multiprocesador en el futuro.

Optimiza el Tiempo de Procesamiento: Permite que otros procesos continúen mientras uno está en espera.

7. Desarrolla los dos problemas inherentes a la programación concurrente. (1 punto).

1. Condición de carrera

Se produce cuando varios hilos o procesos interactúan con un recurso común al mismo tiempo, y al menos uno de ellos realiza una operación de escritura. La secuencia en que se realizan estos accesos determina el resultado, lo cual puede ocasionar comportamientos inesperados en el sistema.

2. Deadlock

Sucede cuando dos o más hilos o procesos se bloquean indefinidamente, esperando que el otro libere un recurso. Cada uno posee un recurso que el otro necesita para continuar, y ninguno está dispuesto a ceder.