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 procesos se pueden encontrar en tres estados:

- En ejecución: Si se está ejecutando, es decir, usando el procesador.

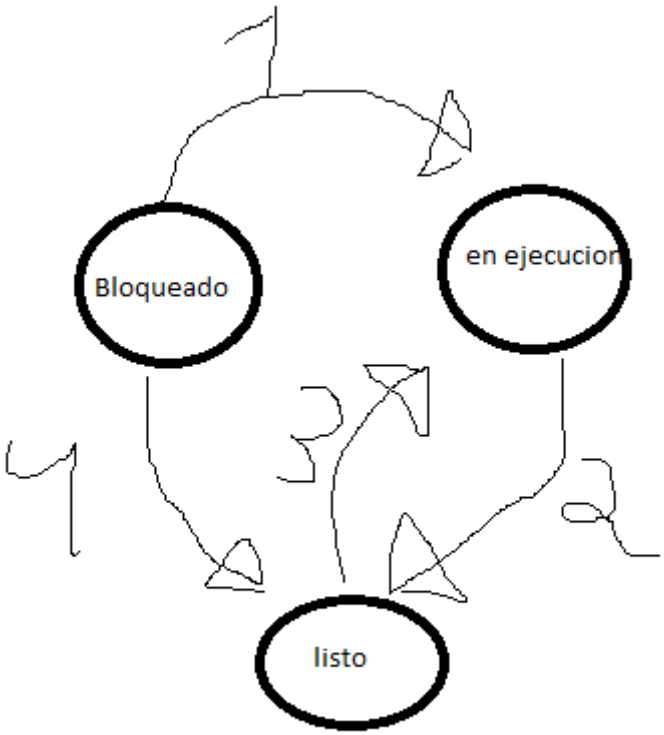
- Bloqueado: No puede hacer nada hasta que otro proceso le dé la entrada.

- Listo: El proceso está esperando su turno en el procesador para ejecutarse.

Bloqueado--2-->en ejecucion--4-->listo

en ejecucion<-3--listo

Bloqueado---1--->listo



Las transacciones entre los estados son las siguientes:

De Ejecución a Bloqueado: El proceso espera que ocurra un evento.

De Bloqueado a Listo: Ocurre el evento que el proceso esperaba.

De Listo a Ejecución: El S.O le otorga tiempo de CPU.

De Ejecución a Listo: Se le ha acabado el tiempo asignado por el S.O.

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

Se puede hacer de estas dos formas (una la comento)

```
public static void main(String[] args) {
//     ProcessBuilder          pb=new          ProcessBuilder          ("C:\\Program
Files\\Google\\Chrome\\Application\\chrome.exe");
ProcessBuilder pb=new ProcessBuilder ("CMD","/C","start chrome");


try {
Process p=pb.start();
} catch (IOException e) {

e.printStackTrace();
}
```

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

```
public static void main(String[] args) {

ProcessBuilder pb =new ProcessBuilder("CMD","/C","tasklist");
try {
Process process=pb.start();
InputStream is= process.getInputStream();
int c;
while((c=is.read())!=-1) {
System.out.print((char)c);
}


} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}


}


}
```

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

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

instrucciones	Lectura y escritura	escritura y lectura	escritura y escritura
1 y 2	a,square ∩ p2= ø	p1∩ b,x= ø	p1∩ p2= ø
1 y 3	a,square ∩ square= ø	p1∩ x,x= ø	p1∩ square= ø
1 y 4	a,square ∩ z= ø	p1∩ m1,m2= ø	p1∩ z= ø

1 y 5	a,square ∩ y= ø	p1 ∩ z,c= ø	p1∩ y= ø
2 y 3	b,x ∩ square= ø	p2 ∩ x,x= ø	p2∩ square= ø
2 y 4	b,x ∩ z= ø	p2 ∩ m1,m2= ø	p2∩ z= ø
2 y 5	b,x ∩ y= ø	p2∩ z,c= ø	p2∩ y= ø
3 y 4	x,x ∩ z= ø	square ∩ m1,m2= ø	square ∩ z= ø
3 y 5	x,x∩y= ø	square ∩ z,c= ø	square ∩ y= ø
4 y 5	m1,m2∩ y= ø	z ∩ z,c= ø	z y= ø

1 y 3 no son concurrentes

4 y 5 no son concurrentes

Todos los demas si lo son

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

Permite optimizar recursos

La programación concurrente en monoprocesadores mejora la eficiencia al aprovechar el tiempo de CPU de manera más efectiva, permitiendo que otros hilos se ejecuten mientras uno espera por operaciones de entrada/salida.

Facilita el diseño orientado a objetos:

Facilita un diseño más modular y comprensible al permitir la ejecución simultánea de unidades independientes, lo que favorece la creación de clases y objetos en un enfoque orientado a objetos.

Mejora el aprovechamiento de entrada/salida:

Contribuye a un mejor rendimiento en situaciones de entrada/salida al permitir que otros hilos se ejecuten mientras uno está inactivo, mejorando así el aprovechamiento de la CPU.

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

Exclusion mutua, es comun que varios procesos intenten acceder a la misma variable

Condicion de sincronizacion un proceso p1 espera a que otro proceso p2 le pase informacion para que continue el proceso