



Práctico 2

Procesos e Hilos

2.1. ¿Qué entendés por proceso? ¿Existe un mapeo uno-a-uno entre programas y procesos? Por ej., si se ejecuta tres veces el mismo programa, ¿cuántos procesos se crean? ¿3 ó 1?

2.2. Describir el concepto de pseudoparalelismo.

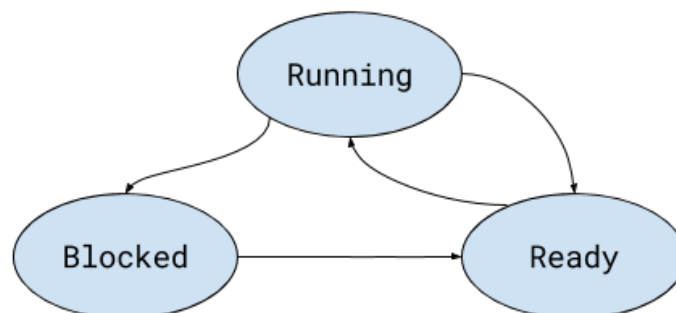
2.2. ¿Cuáles son los principales componentes de un proceso? Describir el contenido de cada uno de dichos componentes.

2.3. Definir el concepto de demonio (*daemon*).

2.4. Enunciar al menos cuatro eventos que provoquen la creación de procesos.

2.5. Enunciar dos condiciones en las que un proceso puede terminar de manera voluntaria y dos en las que se produzca de forma involuntaria.

2.6. Considerando el siguiente diagrama de estados, resolver los siguientes ejercicios en el orden propuesto:



- Agregar los estados *New* y *Exit*.
- Describir cada uno de los estados.
- Describir la razón de cada transición.
- Dividir el estado *Running* en dos nuevos estados: uno denominado *Running-Kernel-Mode* y el otro *Running-User-Mode*. Describir estos nuevos estados y las nuevas transiciones.

- e) Agregar los estados *Blocked/Suspend* y *Ready/Suspend*, los cuales se describen de la siguiente forma:
 - i) *Blocked/Suspend*: el proceso está en memoria secundaria y a la espera de un evento.
 - ii) *Ready/Suspend*: el proceso está en memoria secundaria pero está disponible para ejecutarse tan pronto como sea cargado en memoria principal.

2.7. Describir las diferencias entre los planificadores de corto, mediano y largo plazo.

2.8. Especificar las acciones llevadas a cabo por el kernel al realizar un cambio de contexto entre procesos.

2.9. *Process Control Block (PCB)*.

- a) ¿Cuál es el objetivo de la estructura de datos conocida como *Process Control Block (PCB)*?
- b) ¿De qué manera se caracteriza unívocamente a un proceso en el PCB?
- c) Usualmente el PCB está en el espacio de direccionamiento del SO. Asumamos que ponemos el PCB en el espacio de direccionamiento del proceso: ¿qué problemas podría causar?
- d) ¿Todos los sistemas operativos tienen el mismo PCB?

2.10. ¿Qué entendés por proceso multihilo?

2.11. ¿Cuál es la diferencia entre paralelismo y concurrencia? ¿Es posible tener concurrencia sin paralelismo?

2.12. Mencionar tres ejemplos de programación en los cuales una solución multihilo proporciona un mejor rendimiento que una solución de un solo hilo.

2.13. ¿Cuál es la diferencia entre los hilos a nivel de usuario (ULTs) y los hilos a nivel de *kernel* (KLTs)?

2.14. Nombre ventajas y desventajas de los hilos a nivel de *kernel* (KLTs).

2.15. Nombre ventajas y desventajas de los hilos a nivel de usuario (ULTs).

2.16. Describir las acciones realizadas por el *kernel* en un cambio de contexto entre hilos a nivel de *kernel*.

2.17. ¿Cuáles de los siguientes componentes son compartidos entre los hilos de un proceso multihilo?

- a) Registros.
- b) Memoria dinámica.
- c) Variables globales.
- d) Pila.

2.18. ¿Es posible que una solución multi-hilada usando múltiples hilos a nivel usuario produzca un mejor rendimiento sobre un sistema multiprocesador que sobre un sistema de un solo procesador?

2.19. Considerar un sistema multiprocesador y un programa multi-hilado escrito utilizando el modelo muchos-a-muchos, en el cual el número de hilos a nivel de usuario en el programa es mayor que el número de procesadores en el sistema. Discutir las implicaciones en cuanto a rendimiento para los siguientes escenarios:

- a) El número de hilos a nivel de *kernel* asociados al programa es menor que el número de procesadores.
- b) El número de hilos a nivel de *kernel* asociados al programa es igual al número de procesadores.
- c) El número de hilos a nivel de *kernel* asociados al programa es mayor que el número de procesadores, pero es menor que el número de hilos a nivel de usuario.