**TEÓRICO – PRIMERA PARTE**

**Sistema operativo**

Es, a grandes rasgos, el intermediario entre el hardware y el usuario.

Tiene 3 puntos de vista:

* Usuario
* Gestión
* Maquina extendida

Usuario

Conjunto de programas y funciones que ocultan los detalles del hardware ofreciendo al usuario una vía sencilla y flexible de acceso al mismo.

Gestión (recursos)

Administrador de recursos ofrecidos por el hardware para alcanzar un rendimiento eficaz de los mismos. Básicamente propicia una asignación ordenada de los mismos entre los distintos programas que compiten por ellos.

Maquina extendida

Es el encargado de crear buenas abstracciones de alto nivel para lidiar con componentes complejos del hardware.

Qué relación tiene con la seguridad

El sistema operativo es la pieza fundamental del software y se ejecuta en modo kernel. Este modo tiene acceso ilimitado a las funcionalidades del hardware. El resto del software se ejecuta en modo usuario, el cual tiene permisos restringidos a unas pocas instrucciones.

Objetivos del sistema operativo

* Abstracción
* Seguridad

**Monoprogramación**

Llamada así porque se ejecutaba un único programa a la vez.

Originalmente las maquinas las operaba un operador que era el encargado de juntar las tarjetas perforadas, e insertarlas en las maquinas como entrada, para que realizaran la tarea correspondiente.

Como automatizo el trabajo del operador?

Con lo que se llama **monitor residente.** Este era un programa cuya función era la de pasar el control de una tarea a la otra. Este contenía:

* Secuenciador automático de trabajos (determina orden de tareas)
* Intérprete de tarjetas de control
* Controladores de software de entrada y salida

**Off-line**

Cargar la tarjeta es mucho más lento que procesarla.

Aparecen entonces las cintas magnéticas.  
Lo que se hace ahora es que varios lectores de tarjeta, las leen a la vez y después generan cintas que se conjugan en una gran cinta. Esa gran cinta es la que se termina procesando.  
Offline se le denomina al procesamiento de los procesos lentos en paralelo para ahorrar tiempo.

Al procesamiento simultáneo de la información para pasarla a las cintas, que no requería la intervención del operador en ningún momento, se le denominó **proceso batch**.

**Buffering**

Se mete una memoria intermedia donde se le pasa la información de la cinta y la maquina toma los datos desde ahí. De esta manera se logra la lectura y el procesamiento de datos en forma simultánea.

**Spooling**

Las cintas sólo podían escribirse recién después de leerse.  
Aparecen los discos magnéticos, los cuales permiten realizar operaciones de entrada y salida en forma simultánea.  
Ahora mientras un proceso escribe en el disco, otro lee del mismo. Esto permite cierta interacción con la máquina ya que ahora puedo obtener feedback de mis entradas y condicionarlas a las salidas que obtengo.

**Multiprogramación**

Surge gracias al spooling. Ahora tengo la posibilidad de gestionar varios procesos en forma simultánea, posibilitado por el disco magnético.   
Lo que se hace es aprovechar la inactividad del procesador durante la ejecución de una operación de entrada/salida de un proceso x, para atender otro proceso. Los procesos a la espera, se mandan a dormir. Esto genera la noción de simultaneidad en el usuario, cuando en realidad lo que ocurre es que se duermen y despiertan procesos muy rápidamente.  
Obviamente esto trajo algunos problemas como, cómo repartir prioridades, cómo administrar los recursos entre procesos, etc.

**Sistemas de tiempo real**

Son programas que tienen deadline (tiempo de respuesta para que la respuesta me sea útil).

**Tiempo compartido**

Surge con la multiprogramación. El usuario no necesita dar todos los datos de entrada para que el proceso se ejecute, sino que los va dando en la medida que este transcurre, en forma interactiva.  
El trabajo se organiza en sesiones y el sistema suspende procesos dependiendo de la necesidad de cada sesión.

**TEÓRICO – SEGUNDA PARTE**

**Proceso**

Programa en ejecución, junto con su entorno asociado (variables, registros, pila, etc).  
El concepto nace con la multiprogramación. Ahora tenemos a procesos dormidos y procesos en ejecución, mientras que antes los procesos se llevaban a cabo uno a la vez.

**Diferencia entre proceso y programa**

Si hiciéramos una piza, el programa sería la receta y el proceso sería la ejecución de la receta.  
Mientras que un programa implica una secuencia de instrucciones a aplicar, un proceso es la aplicación de dicha secuencia.

**PCB (Process Control Block)**

Describe información del proceso.  
Estado, número, program counter, registros, límites de memoria, procesador, descendientes, prioridad.

**Estados de un proceso**

*En ejecución* – Usando la CPU en ese momento.  
*Listo* – Ejecutable. Se detuvo temporalmente para dejar ejecutar otro proceso.  
*Bloqueado* – No puede ejecutarse hasta que ocurra cierto evento. Por ejemplo a la espera de un dispositivo de entrada/salida.

A veces la cola de listos es demasiado grande, entonces a mí me puede interesar bloquear algunos procesos de la cola. A estos se les llama *suspendidos* y son enviados al disco duro.

**Hilo**

Son tareas dentro de un proceso, podría decirse que son subprocesos.  
La razón por la cual son necesarios, es que permiten hacer todo sin tener que detener el proceso. De lo contrario debería suspenderse la ejecución de un proceso para realizar otro y así. Estos hilos van a compartir un mismo contexto de recursos, variables, que es el proceso en sí.

**Condición de carrera**

Cuando el mismo código puede dar dos valores distintos, dependiendo de cuál de los hilos concurrentes llega primero al procesador, esto se llama condición de carrera.

**Scheduler o Planificador**

Tiene por función repartir la capacidad del procesador entre los distintos procesos que se están ejecutando. Es el encargado de asignar las prioridades entre los distintos procesos.

**RSR (Recursos Serialmente Reusables)**

Son recursos que podemos reutilizar pero que sólo los puede usar un proceso a la vez, ej una impresora.

**TEÓRICO – TERCERA PARTE**

**Evolución algoritmo de Alice y Bob**

En principio un proceso tiene que esperar por el otro para ejecutarse. Esto no es bueno.  
Entonces se le agrega una variable booleana dentro del while para que un perro pueda pasear dos veces seguidas. Esto no resulta bien ya que ambos procesos entran en la zona crítica.  
La tercera solución fue sacar la var booleana para afuera del while, pero termina en deadlock.  
Quienes terminan por resolver este problema son Dekker y posterior a él, Peterson.

**Ventajas del semáforo**

Es un TAD de bajo nivel  
Evita que exista el busy waiting  
Es extensible a la cantidad de procesos que compitan por recursos que se desee

**PRÁCTICO**

**TAD – SEMÁFORO**

Init (s,v)  
s = v

P(s)  
if (s = 0) then  
Bloquear proceso  
Else  
s--;  
  
V(s)  
s++  
Despertar proceso bloqueado

**ESTRUCTURA BUFFER**

Buffer : array [0…N-1] of T  
p\_a\_i, p\_a\_e : 0…N-1 Inicializo en 0  
cant : 0…N Inicializo en 0

Insertar\_al\_final (x : T)  
Precondicion: cant < N  
Buffer[p\_a\_i] = x  
p\_a\_i = (p\_a\_i + 1) mod N  
cant++  
Fin Insertar\_al\_final  
  
T Extraer\_primero()  
Precondicion: cant > 0  
x : T  
Buffer[p\_a\_e] = x  
p\_a\_e = (p\_a\_e + 1) mod N  
cant--  
Fin Extraer\_primero

**OTROS PROCESOS**

Proc Main  
B1:Buffer  
s:Semaforo  
init(s,1)  
cobegin  
 P1  
 P2  
coend

Proc P1  
Variables  
x : int  
Begin  
While true do  
…  
end while  
end Proc