TEC

Sistemas Operativos Avanzados

Apuntes de clase:
20/02/2017

Apuntador:
Albin Andrés Arias Juárez

Indice

Asuntos administrativos

- Se le entrega al profesor los resúmenes de los papers asignados.
- El profesor hace entrega de los resúmenes revisados de la semana anterior, y del primer quiz del curso.
- El profesor explica el primer corte subido al foro, con varios personajes ficticios:
    - **Perfecto:** El estudiante que ha entregado todo y ha obtenido nota perfecta en todo.
    - **Promedio:** El promedio general de todos los estudiantes del curso.
    - **Pasando:** El estudiante que saca lo necesario en todas las asignaciones para al finar aprobar el curso con nota mínima.
    - **Último:** Es el estudiante más vago. No asiste ni entrega asignaciones hasta que tenga que entregar todas las requeridas para pasar con nota mínima.
- El profesor explica, a manera general, el contenido de los cortes, los cuales se suben semanalmente:
    - Resúmenes entregados.
    - Los puntos que faltan para llegar a 67.5.
    - Nota máxima que el estudiante puede obtener.
    - Proyección de la nota final del estudiante.
- Se asignan los papers a resumir y leer para la semana siguiente: 0502, 0503 y 0601.
- Se hacen presentaciones luego del *break*.
    - **PAPER 0201 –** *Exokernel: An Operating System Architecture For Application-Level Resource Management.*
    - **PAPER 0207 –** *Server Operating Systems.*

Repaso de conceptos básicos

Protección

- Se necesitó para garantizar la microprogramación.
- Los programas se deben de correr como si estuvieran corriendo solos.
- Se logra solamente con ayuda del hardware.

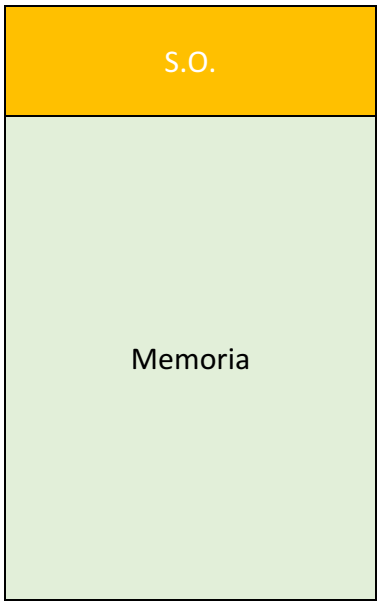
Memoria en ambientes multiprogramados

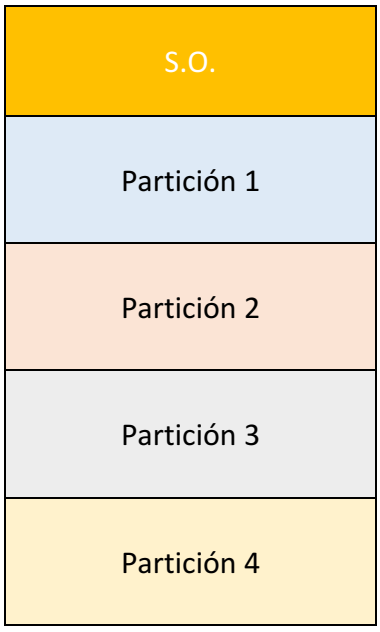
- Hay que dividir la memoria entre los programas multiprogramados.
    - El Sistema Operativo se encarga de esto.

- Al inicio el grado de multiprogramación era muy bajo.
    - Siempre se intenta buscar la solución más simple posible.
- Se busca al inicio, un mecanismo simple administración de memoria.

Particiones estáticas

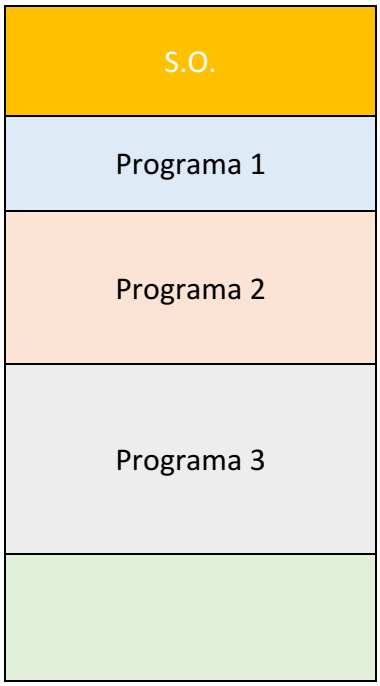
- Se hizo históricamente.
- El concepto era de: dividir la memoria *a priori* en particiones de tamaño fijo.
    - Lo anterior se hacía basado en datos históricos, justicia, etc…
    - Hay que conocer lo que se está corriendo para asignarle un espacio.
- Una vez que un espacio se asignaba, este no cambiaba.
- Cualquier cosa dinámica que se inventó luego, ocurría dentro de ese espacio.
- El SO fijaba y conocía estas particiones.
- Habían tamaños variables, pero eran tamaños fijos y estáticos.
- Era un esquema simple para el SO:
    - Menos probable que esté malo.
    - Sencillo de implementar.
    - Cualquier solución simple es beneficiosa para todo el mundo.
- ¿Protección?
    - Se fijaban registros de frontera de acuerdo a quien estuviera corriendo en un momento dado.
- Cuando el programa termina de ejecutarse, se libera ese espacio de memoria.
- Ventajas:
    - Simple.
    - Apropiada para la época.
    - Se podía saber la cantidad de programas corriendo en un momento dado.
- Desventajas:
    - Grado de multiprogramación era muy bajo.
    - Desperdicio.
        - Se podía asignar mucho espacio a un programa que no usara mucho.
    - Los programas podían no caber en una partición aunque sí en la memoria total.





Particiones dinámicas

- Conforme van llegando los programas, se les asignan recursos.
  - No son definidas *a priori*.
- Cada vez que llega un programa nuevo, se le da solo la memoria que necesita.
- Cuando terminan, liberan la memoria que tenían asignada.
- Al llegar otro programa, se le da un espacio de memoria libre de tamaño suficiente.
- El SO lleva control de los espacios libres y ocupados.
- El SO puede deducir cuanta memoria va a utilizar ese programa.
- ¿Cómo se selecciona el lugar donde va el nuevo programa?
  - First Fit.
  - Best Fit.
  - Worst Fit.
  - Buddy System.
- Ventajas:
  - Menos desperdicio.
  - Varias particiones = varios programas en ejecución.
- Desventajas:
  - Overhead.
  - Difícil de programar.
    - Liberación de pedazos y buscar pedazos.
  - Fragmentación.



First Fit

- Primer hueco libre donde quepa el programa entrante.

Best Fit

- El hueco libre más pequeño que sea más grande que lo que estoy pidiendo.
  - Tiende a que queden “hendijas” de memoria que no se puedan usar.

Worst Fit

- El espacio más grande y ya.
  - Asigna memoria rápidamente.

Buddy System

- El sistema divide la memoria en bloques de ciertos tamaños.

- Las medidas de tamaños permitidos son usualmente potencias de 2, o de la forma de la secuencia de Fibonacci.
- Se crean bloques de tal manera que todos, excepto el más pequeño, puedan ser divididos entre 2.

#1	64 KB Libres			
#2	8 KB Asignados	8 KB Libres	16 KB Libres	32 KB Libres

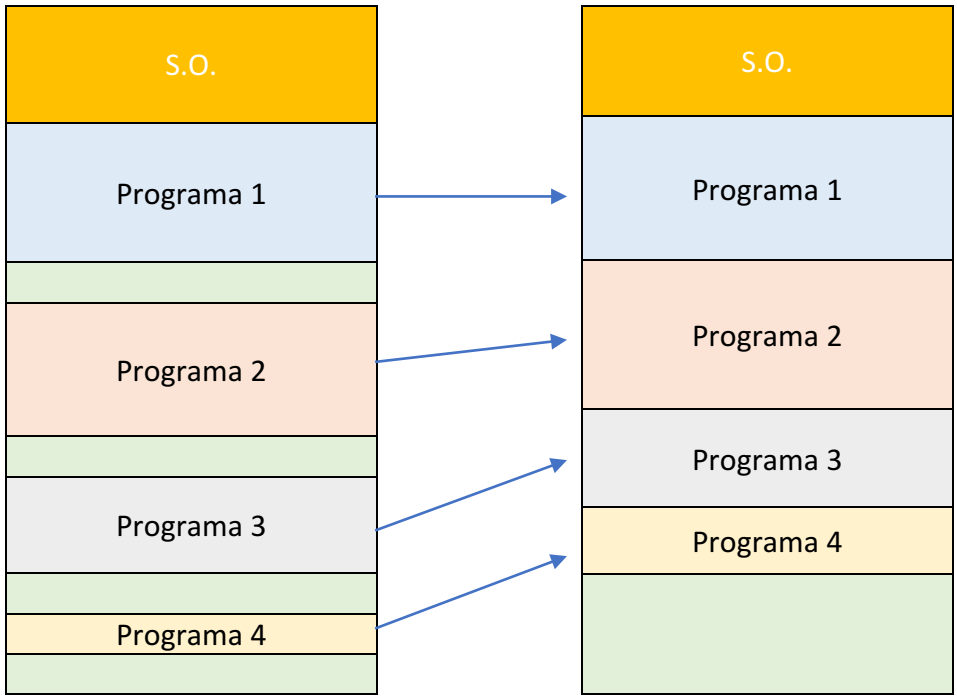
La anterior tabla, muestra en la fila #1, la memoria libre en su totalidad, y vemos que, en la fila #2, se asigna un bloque de 8 KB a un programa, el siguiente bloque libre es de 8 KB, y el siguiente es de 8 KB * 2, osea 16 KB, y el último es de 16 KB * 2, osea 32 KB.

Fragmentación

- Muchos espacios pequeños en la memoria que no pueden ser usados.
- Es un problema cuando un programa es rechazado por no tener memoria suficiente, cuando si se suman los espacios libres y se ve que puede correr.
- El programa que no "cabe" es rechazado por falta de memoria, pese a que la suma de los espacios disponibles alcanza para que el programa se ejecute.
- Se combate con **comptactación.**

Compactación

- Se trata de mover los espacios para que estén seguidos y no tengan "hendijas".
- ***Burping*** (erupto).
- Ventajas:
    - Combate la fragmentación.
    - Aprovechamiento de recursos.
    - Sube grado de multiprogramación.
    - Evita rechazar programas.
- Desventajas:
    - Overhead.
    - El usuario esperaba a que se hiciera la compactación
        - Sube el tiempo de ejecución final.
    - Problemas con programas en ejecución.
        - Tienen direcciones adentro que deben ser cambiadas.
        - Las referencias internas en memoria se echan a perder.



Re localización

- Trata sobre corregir las referencias a memoria que dependían de la posición inicial en memoria.
- Es difícil de hacer.
- El programa cree que empieza en A, y ahora va a ser trasladado a B.
- El problema es saber a dónde se deben hacer esas correcciones. Lugares sensibles.
    - Depende de la instrucción.
- Requiere un **diccionario de relocalización** para saber qué lugares hay que re localizar.
    - Es una estructura de datos.
    - Bitmap.
    - Indicaba qué lugares eran sensibles a la ubicación.
    - Esto lo creaba el compilador.
        - Es fácil saber qué cosas son sensibles a la ubicación.
        - Es pequeño.
- Defectos:
    - Overhead.
    - Afecta duración de los programas.
        - Es malo, pues nos gusta que sea predecible para poder planificar.
    - No se conoce la duración de los procesos.
- ¿Cuándo se hace la re localización?
    - Cuando terminan los programas.

- Cuando se va a ejecutar un programa.
- Cuando se detectan muchos segmentos.
- … Es variable…

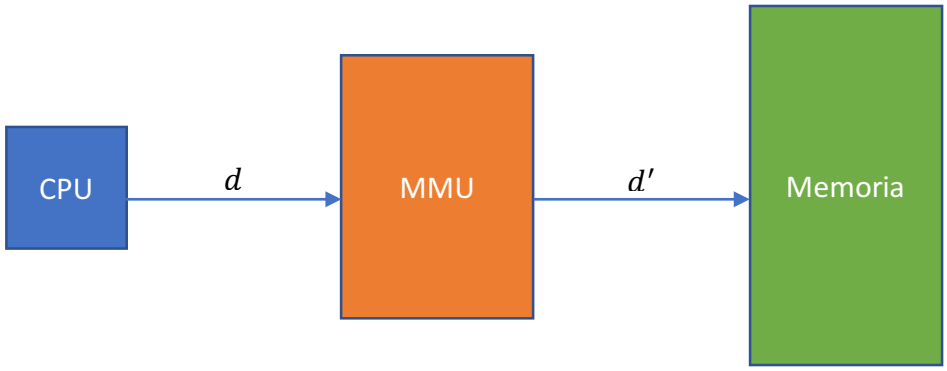
Paginación

- Queremos eliminar que sea requisito que el programa sea continuo.
- La memoria se imagina como una cuadrícula, con pedazos de tamaño fijo.
  - Cada pedazo es una página.
- Se evalúa a cuántas páginas corresponde un requerimiento.
  - Se busca dónde hay páginas libres para el programa.
    - P1 ocupa 3. Se buscan y se aplican.
    - P2 ocupa 4. Se buscan y se aplican.
- Se solucionó con hardware. Diferencia entre espacio lógico y físico de direcciones.
  - Lógico: # de página.
  - Físico: ubicado en memoria.
  - M.M.U.

0 $P_2 1$	1 $P_1 0$	2	3
4	5	6 $P_2 0$	7 $P_2 3$
8	9 $P_2 2$	10 $P_1 2$	11
12 $P_1 1$	13	14 $P_2 4$	15

Memory Management Unit (MMU)

- Componente de hardware.
- Se encuentra entre el CPU y la memoria.
- Las direcciones vienen del CPU y se corrigen.
- Esa dirección corregida llega a la memoria.
- Las direcciones vienen en potencias de 2.
- Tiene una tabla de páginas.
- Cada programa tiene una *page table*.
    - El SO pone al registro del MMU a señalar el *page table* del proceso en ejecución.
- El *page table* es privado para cada proceso.
- Como entrada:
    - Recibe una dirección lógica del CPU.
    - Con la *page table* se hace un mapeo de las páginas lógicas y físicas.
- La dirección generada por el CPU es una dirección lógica o virtual.
    - Tiene la página. Con lo que la MMU sabrá cuál es la página lógica y con esto saber la página física.
    - Offset.
- Funciona y es barato.
- Hay cierta pérdida de rendimiento al tener que pasar por el MMU.
- La *page table* vive en la RAM.
    - Ahora hay dos accesos a memoria: 1 para ver la tabla, y otro para ver las cosas.
    - Esto se soluciona con la **caché**.



El CPU envía una dirección lógica d a la Memoria, la cual antes pasa por la MMU, que la convierte a una dirección física d'.

Caché

- Recuerda las traducciones más recientes.
- Cuando llega una dirección se busca en el caché y listo ☺.
- Si no está en el caché, hay que buscarla en memoria ☹.
- **TLB. Translation Lookaside Buffer.**
    - Cache que está dentro de MMU.
    - Contiene las últimas traducciones que se hicieron.
    - TLB-hit. Cuando encuentra en la caché una dirección ya traducida.
    - TLB-miss. Cuando no encuentra en la caché una dirección buscada.
    - **Principio de localidad.**
    - Flush. TLB borra las traducciones viejas.
    - Es una memoria asociativa: compara todos al mismo tiempo.
    - TLB Cold. Menos hits (se enfría con cambio de contexto).
    - TLB Warm. Más hits.
- Tagged TLB. Identificador para un proceso.
    - Cuando se llena la cache, hay algo que quita entradas viejas.
    - El tagged también tiene su process ID.
    - Más fácil de buscar.
    - Se evita hacer flush.
    - Tiene que ser más grande (tanto horizontal, como vertical).

Tiempo compartido

- Falta de interactividad.
    - Permitió idear mecanismos para tener múltiples usuarios interactuando con un computador.
- Periféricos y consola.
- El usuario vuelve a manejar la computadora.

Tipos especiales de sistemas operativos

- Aplicaciones en tiempo real.
- Cada vez se hacían más especializaciones.
- El sistema operativo era más complejo.
    - Atiende más aplicaciones y más usuarios.

Sistema Operativo de propósito general

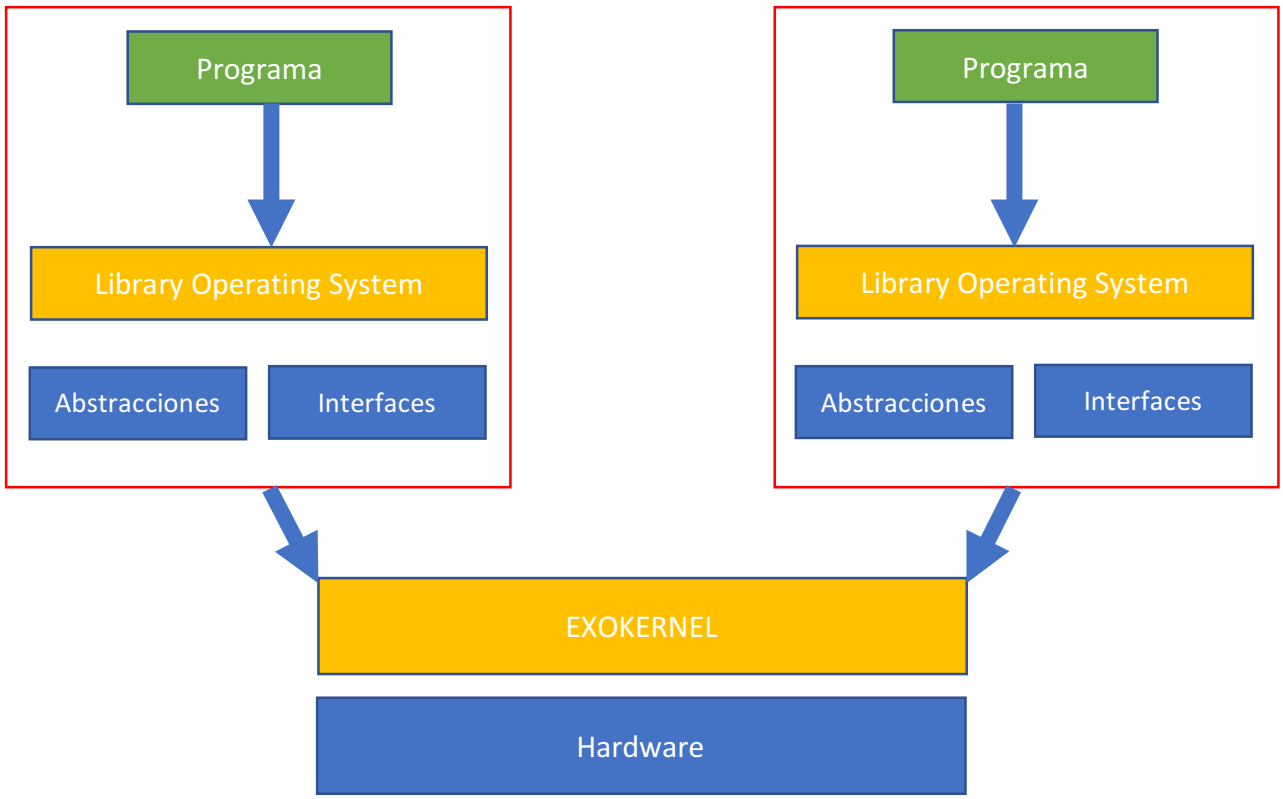
- Sirve para cualquier cosa.
- Al ser tan grande, se hace “mantecoso” y lento.
- Es mejor hacer cosas sin intervención del SO.
- El SO es como la municipalidad.
    - Si se quema la luz del poste que hay por mi casa, yo no debería solucionarlo… Debería llamar al gobierno local para que vengan a arreglar el poste.
- Funciones y módulos. El SO administra:
    - CPU.
    - Memoria.
    - I/O.
    - Archivos (información).
    - Comunicaciones.
    - Bookkeeping.

	CPU	**MEMORIA**	**I/O**	**INFORMACIÓN**	**COMUNICACIÓN**
BOOKKEEPING					
POLÍTICAS					
ASIGNAR					
DES-ASIGNAR					

Si se quiere hacer un sistema operativo, hay que responder y “llenar” la anterior tabla.
Políticas: ¿Qué vamos a hacer? Respecto a CPU, Memoria, I/O, Información y Comunicaciones.
Mecanismos: ¿Cómo?

Exokernel

- Separar protección de administración.
- Permitir que las aplicaciones de usuarios administren sus recursos.
- Capa de software mínima.
- Protección.
- Library Operating Systems.
- Delgada capa de software que multiplea y exporta los recursos de hardware de manera segura.
- Simplicidad permite eficiencia.
- Entre más bajo sea el nivel de un primitivo, más eficiencia.
- Se puede tener Windows y Linux al mismo tiempo.



Microkernel

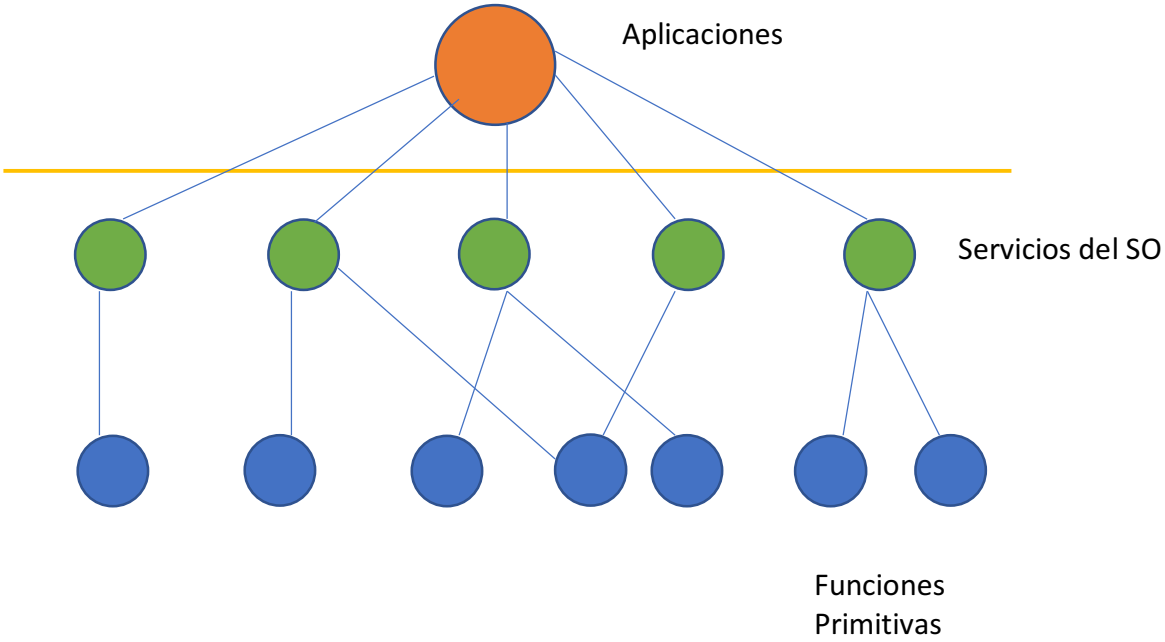
- Estructura monolítica.
- No se quiere cruzar la frontera del kernel.
- Ventajas.
    - Es factible la creación de un SO así.
    - Modularidad.
    - Flexibilidad y extensibilidad.
    - Seguridad.

- Servidores independientes de hardware.
- Correctitud.
  - Más fácil verificar un kernel pequeño.

– Desventajas.
  - Las primeras implementaciones eran lentas o erróneas.
    - Liedtke refutó esto. Dijo que no se investigó bien para la implementación de ellas.
  - Más eficiente comunicar 2 módulos dentro del SO monolítico.

– ¿Qué debe ir en el microkernel?
  - Abstracciones primitivas.
  - IPC.
  - Criterios.
    - Algo debe ir en el microkernel únicamente si ponerlo afuera provoca que el sistema no funcione bien.
    - Si algo puede ser implementado de más de una forma, probablemente no deba ir en el microkernel.

– Rendimiento.
  - Cambio de frontera de manera eficiente.
  - IPC.
  - Liedtke comprueba que el sistema Mach (no macOS de Apple) hacía una mala utilización de caché.
  - Cambiar 2 capas.
  - Optimizador de un compilador.



SPIN

- Servicios del SO.
- Escrito en Modula-3.
    - Protección del lenguaje.
- Linking dinámico.
- Dominios de protección.