Clase 1 : Introduccion a Sistemas Operativos Teoria

Sistema operativo es software,como todo software necesito procesador y memoria para ejecutarse. Son programas que los hicieron un conjunto de personas para poder ser utilizadas, conjunto de instruccion mas conjunto de estructuras de datos(servicios)

Existen dos perspectivas

-Arriba hacia abajo: para la persona que desea utilizar el sistema operativo el SO es una extraccion para ocutar y administrar detalles del sistema operativo. los programas que ejecutamos son clientes del sistema operativo(clientes ya que piden servicios al SO como pedir espacio de memoria para leer un archivo)

-Abajo hacia arriba: Administra recursos de hardaware, en pocas palabras SO administra todos los recursos, brinda conjunto de servicios para que dichos recursos puedan ser utilizados, maneja memoria secundaria, ejecuta simulataneamente procesos y por ultimo multiplexacion en tiempo(CPU) y en espacio(Memoria)

*Multiplexación en el tiempo (CPU): Distribuye el tiempo de CPU entre varios procesos para que parezca que se ejecutan simultáneamente.

*Multiplexación en el espacio (memoria): Distribuye el espacio de la memoria física (RAM) entre diferentes procesos para que puedan coexistir sin interferencias.

objetivos de los sitemas operativos

Comododad: hacer mas facil el uso del hardware.

Eficiencia: hacer uso mas eficiente de los recursos del sistema.

Evolucion: permitir la introduccion de nuevas funciones al sistema sin interferir con funciones anteriores.

Componentes del sistema operativo

Kernel(Componente que esta justo por encima del hardware, lo que hace es

implementar todo lo necesario para poder hacer uso del hardware),

shell(GUI/CUI O CLI), Herramientas(Editores de texto o librerias,...etc)

Kernel:"porcion de codigo", siempre va a estar en memoria principal(porque administra la memoria, administra cpu, traduce cuando quiere usarse archivos... etc)

Servicios de SO

*Multiplexacion de la carga trabajo

*Memoria= ram o principal, discos o otros tipos de memoria es memoria secundaria. implementacion de memoria virtual reflejada en lo fisico

*Proteccion de programas que compiten o se ejecutan concurrentemente.

*Deteccion de errores y respuesta(errores de memoria,cpu,dispositvos,aritmeticos,direcciones no permitidas)

*Interaccion con el usuario por shell

*contabilidad: recoge estadisticas del uso, monitorear parametros de rendimiento, anticipar necesidades de mejoras futuras

si es necesario facturar tiempo de procesamiento.

Clase 2 : Introduccion 2 a Sistemas Operativos Teoria

Problema que un SO debe evitar:

Que un proceso se apropie de la CPU, SO no se esta ejecutando siempre dicho SO tambien necesita cpu, por lo cual tiene que haber manera

de que mientras que se esta ejecutando no acceda a memoria que no debe o tambien debe controlar que no se apropie de la cpu(while true al ejecutarse por ejemplo)

y por ultimo el SO debe evitar que un proceso intente ejecutar instrucciones de E/S por ejemplo.

para poder garantizar esto el SO necesita apoyo del hardware

tres grandes apoyos que el hardware debe proveer

*Modo de ejecuccion: debe tener un bit que muestre el modo que se esta ejecutando, modo kernel o supervisor(toda instruccion que se ejecute es posible de hacerse por ejemplo

Gestion de procesos, Gestion de memoria, Gestion E/S y funciones de soporte) ,

modo usuario: puede acceder solo a su espacio de dirrecion y teniendo instrucciones basicas.trabaja en su propio espacio de direcciones no puede interactuar con el hardware.

ADJUNTO: El kernel o codigo dentro de el se deberia ejecutar en modo supervisor, el resto del SO se deberia ejecutar en modo usuario.

La Cpu puede hacer cambios de modo: hay una unica manera de pasar de modo usuario a modo kernel que es mediante una interrupcion(rutina sistema operativo),

El otro camino de modo kernel a usuario se puede hacer ya que de modo kernel se puede hacer cualquier cossa ademas solamente estas restringiendo.

Acordarme siempre que se vaya a ejecutar un servicio se pasa a modo usuario y cuando sale vuelve a modo kernel.cambio modo constantemente sucediendo.

*Interrupcion de Clock: se debe evitar que un proceso se apropie de la CPU.

*Proteccion de la Memoria: se deben definir limites de memoria a los que puede acceder cada proceso(registro base y limite)

Cosas que deberia saber de primer año :

-flujo normal basico de instrucciones, es buscar y la ejecuta.

-que es lo que rompe ese ciclo: interrupciones como lo hace : las instrucciones estan numeradas por lo cual cuando se interrumpe, se mira en el vector interrupciones segun el numero de interrupcion dado

y se agarra la direccion de memoria donde esta el rutina que atiende esa interrupcion.

Tipos de interrupciones:procesador pasa a modo kernel y le da el poder al sistema operativo resuelva esa interrupcion

Que pasa si en modo kernel tengo una interrupcion:

depende interrupcion si el disco avisa que temrmino hacer algo es una cosa, pero si hay accesos indevido a memoria o division por cero. blue scree of dead(interrupcion por software en modo privilegiado y hay un acceso indevido a memoria,"el que controla se equivoca")

Proteccion de la CPU=tres principales(intrucciones privilegiadas modifican funcionamiento del reloj, se le da tiempo o valor para que se ejecute proceso, se la usa para el calculo de la hora basandose en cantidad) Cada x unidades de tiempo se utiliza interrupciones por clock para poder evitar que un proceso se apropie de la cpu, el kernel le da valor al contado que se decrementa con cada tick de reloj y al llegar a cero puede

expulsar al proceso para ejecutar otro.

Proteccion de la Memoria= Cada proceso tiene delimitado el espacio de memoria en el cual se puede mover, esta delimitacion se ponen en modo kernel y es el kernel el que revisa si la instruccion cuando se esta ejecutando se mueve entre dichos parametros

Sistema operativo brinda servicios a las aplicaciones(leer, escribir, cambiar directorios), las system calls o llamadas al sistema son funciones o procesos que pueden utilizar los programas para hacer uso de herramientas que brinda el SO

cuando uno ejecuta un read,write o open son llamados a una libreria comun que lo que hace es responder a esos llamados, esta libreria esta en modo usuario y lo que hace es invocar una libreria que el sistema operativo brinda, lo que hace eesta libreria del sistema operativo,

es identificar cual es la libreria que llama al sistema, pone un numero en un determinado registro y se ejecuta un trap o una interupcion por software para cambiar a modo privilegiado se ejecuta la atencion de interrupciones,

leeo el numero que deje y desde el modo privilegiado leo el numero y los parametros entonces desde ahi si se puede ejecutar el llamado, vuelvo al modo usuario.

Porque no compila los programas de windows en linux o viceversa ya que los llamados al systema son distintos.

Tipos de kernel

*Kernel Monolitico: Que toda funcionalidad que deberia o debe implementar el sistema operativo se debe ejecutar en modo kernel, cuando un usuario quiere ejecutar un proceso se pasa a modo kernel se resuelve ahi y pasa a modo usuario cuando este se termino.

*Micro kernel: se trata de que el kernel sea lo mas chico posible por lo tanto se deja en modo usuario diferentes componentes para que den apoyo al kernel. reduce al maximo lo que si o si se ejecuta en modo kernel dejando herramientas para que se pueda utilizar lo maximo posible en modo usuario.

Ventaja monolitico: implica menos tiempo en resolucion.

Ventaja Microkernel: Evita problemas ya que esta menos tiempo en modo kernel.

Clase 3 : Procesos 1

Definicion Proceso= Es dinamico,tiene,program counter,ciclo de vida desde que se dispara hasta que se termina.

Entidad de abstraccion(incluye codigo, variables globales o datos y stack elementos temporarios como parametro,variables de retorno...etc)

Debe tener identificaicon del proceso, y del proceso padre.

PCB(process control block)= cada proceso tiene su pcb, es lo primero que se hace al crear un proceso y ultima a destruir al querer terminar o finalizar un proceso

contiene por ejemplo(PID,valores de los registros de la cpu,planificacion(estados etc),ubicacion en memoria,accouting(estadisticas de uso),entrada salida)

Definicion Programa= Es estatico, No tiene program counter, Existe desde que se edita hasta que se borra.

Multiprogramado= varios procesos pueden estar en ejecucion y datos de procesos son totalmente independientes.

Stacks en general se tiene una pila por cada modo de ejecuccion por ejemplo una de modo usuario y una de modo kernel, se ajustan a su tamaño automaticamente.

Espacio de direcciones de un proceso= conjunto de direcciones de memoria que ocupa el proceso,no incluye su PCB o tablas asociadas,modo usuario en estos limites, modo kernel se pueden romper limites.

Contexto proceso= toda informacion que necesita el sistema operativo para administrar proceso.

Cambio contexto= se produce cuando la cpu cambia de un proceso a otro, debiendo resguardar el contexto saliente, que pasa a espera y retorna despues a la cpu, es tiempo no productivo de cpu, el tiempo depende del hardaware.

Kernel no es un proceso, esto se puede explicar desde dos ennfoques:

-kernel como entidad independiente: kernel se ejecuta fuera de todo proceso independiente o a la par de todos los procesos, cada vez que ocurre una interrupciion debo guardar contexto que se esta ejecutando, luego pasa al kernel(contextext switch). kernel tiene su propia region de memoria,

tiene su propia stack, finalizada su actividad le devuelve contro a otro proceso.

-kernel dentro del proceso: el kernel se encuentra dentro del espacio de direcciones de cada proceso. el kernel se ejecuta en el mismo contexto que algun proceso de usuario,desventaja el espacio es fijo lo cual disminuye espacio de proceso. por lo cual en cada proceso se deberia tener mas de una pila para poder resguardar informacion ante hackeo.

Proceso tiene un ciclo de vida o estados=

-estado nuevo o cero donde se incializa estructura de datos, luego de estar completo se carga en memoria.

-pasa a estar en estado listo o ready= tiene todo lo necesario para ejecutarse pero necesita CPU.

-cuando se selecciona para a estar en estado de ejecucion= se ejecuta contextet switch y pasa a estado de ejecucion,puede pasar que se interrupa y no finalice o esperar a un dispositivo de entrada salida(pasa a waiting)

-termina pasa a esta listo, pasa a estado de desincronizador borrando o eliminado lo que ya no se necesita(pcb por ejemplo)

-waiting: espara que inicie tarea de entrada salida y espera a que dicho proceso se complete y vuelve a ready.

Colas de planificacion= Son pcb que se enlazan, a esperar acciones o estado, cada accion tiene una o mas colas por ejemplo puede haber tres colas de waiting una de finalizacion etc.

Modulos de planificacion= pedazos de software que realizan tareas asociadas a la planificacion que nacen cuando se realiza alguna accion por ejemplo al borar algun proceso.

Los principales modulos que pueden aparer es:

*Scheduler de long term: es el que realiza la actividad de admitir proceso de nuevo a estado listo, mira cual pasa a estado listo despues de haberse creado. y looder ayuda y se encarga de pasar a ready.

*Scheduler de short term: es el que realiza la tarea de llevar a esta listo a running

*Scheduler de medium term: hay veces que es necesario bajar el grado de multiprogramacion, baja proceso de memoria llevandolo al area de swap(nuevo espacio de memoria) despues se encarga cuando quiera pasa a ready desde suspendido.

Diagrama de transiciones UNIX= fork(crea un proceso en estado de creado o nuevo) puede sucede que haya suficiente memoria como para cargarlo o que no haya suficiente memoria. entoces se pasa a ready to run o a suspend ready to run

esto hace medium term mover entre estos dos estados, short term se encarga de que el proceso arranque o se ponga en ejecuccion y arranca en modo kernel una vez cargado se larga a modo de ejecuccion pero en modo usuario

si kernel detecta que ya termino, pasa a modo zombien o exit. puede haber dos formas de entrar a la cpu, modo normal que entra porque esta listo o por modo de apropiacion

Clase 4: Procesos 2

que es planificar la cpu? necesidad de determinar cual de todos los procesos que estan listo para ejecutarse, se ejecutara.

Algoritmos apropiativos= hacen que el proceso en ejecucion sea expulsado de la cpu

Algoritmo no apropiativos= los procesos se ejecutan hasta que el mismo abandone la cpu(bloquea E/S o finaliza y no hay decision de planificacion durante la interrupciones de reloj)

Categorias de los algoritmos de planificacion= metas posibles(equidad(parte justa de la cpu a da proceso) y balance mantener ocupa todas las partes del sistema))

Procesos batch= no existen usuarios que esperen una respuesta en una terminal, se utilizan algoritmos no apropiativos y tiene ventajas como rendimiento, tiempo de retorno)

planificadores de procesos batch ejemplos FCFS(firsr come first served) y SJF(shortest job first)

Procesos interactivos= No solo interaccionan con el usuario(ejemplo un reproductor de musica), son necesarios algotirmo apropiativos para evitar que un proceso acapare la cpu.

Ejemplos de procesos intereactivos:

* Round Robin(mas clasico y utilizado)= la cola de planificacion, la diferencia que tiene es que se le da tiempo para estar en la cpu esto se llama quantums, entonces lo que hace SO le da ese lazo de tiempo, mediante la interrupcion por clock ve cuanto tiempo hace que esta en ejecucion

cuando llega el momento saca el proceso de la cpu y mete otro, se tiene que tener en cuenta que un quantum muy chico hay sobrecarga de cambio de contexto por mas que sea rapido.

* Prioridad = cada proceso tiene una prioridad entonces se puede ejecutar en ese orden, esta ordenado por algun criterio genera inanicion(procesos con muy baja prioridad no se ejecutaran nunca)

* Colas multinivel = todos los procesos tienen una prioridad y por cada prioridad genero una cola donde aplico round robin a cada cola. la prioridad es variable para que no ocurra esto de que no se ejecute nunca.

* SRTF Shortes remaining time first = es apropiativo, el que menos tiempo tarde en ejecutar adentro de la cpu va a ser el que se ejecuta, requiere cpu para poder calcular cuanto va a tardar cada proceso y ordenarlo, desperdicia tiempo en el algotirmo y no en los procesos es el problema.

Politica versus mecanismo= lo que suele cambiar en las versiones de sistemas operativos son las politicas no los mecanismos ya que dependiendo la cantidad de actividades a realizar a la par es el cuatum que se le va a dar.

Creacion de procesos= un proceso siempre es creado por otro proceso, el proceso nace ya por el sistema al arrancar y desde ahi se crean todos los sub procesos.

Cuando se crea un proceso= se crea PCB, asigna PID(Process Idetication) unico,asiga memorai para regiones, crea estruturas de datos asociadas.

relacion entre procesos padre e hijo= cuando se crea proceso hijo se pueden utilizar por igual padre e hijo y otra manera es si un proceso crea otro debe esperar a que otro temrine para seguir.

unix= El hijo es un duplicado del padre.

fork=crea nuevo proceso; padre e hijo iguales sino lo que se copio tambien los valores de refistros son iguales, pc igual en ambos casos, para saber donde estoy parado tengo que preguntar si estoy en padre o hijo("fork retorna dos valores(contexto padre positivo el valor que me retorno process ID de hijo)(menor a cero error por lo cual nunca se ejecuto entonces no se creo el hijo)(igual a cero exito")

execve()=carga nuevo programa en espacio de direcciones.

Windows= crea proceso y se le carga adentro el programa.

CreateProcess()= crea un nuevo proceso y carga el programa para ejecucion.

Procesos cooperativos e independientes

independiente: el proceso no afecta ni puede ser afectado por la ejecucion de otros procesos.

cooperativo: afecta o es afectado por otros procesos en el sistema

para que sirven los procesos cooperativos? compartir informacion,para acelerar el computo(separa una tarea en sub-tareas que cooperan ejecutandose paralelamente),planificar tareas a ejecutar en paralelo.

Anexo Algoritmo Apropiativos

Algoritmos no apropiativos= proceso deja el estado de ejecucion solo cuando: termina,se bloquea voluntariamente o solicita una operacion de E/S

ALgoritmos apropiativos= round robin(se termina su quatum), Algoritmo prioridades apropiativo(llega a la cola de listos un proceso de mayor prioridad) y Algoritmo SRTF(llega a la cola de listos un proceso con menor tiempo de ejecucion que el que se esta ejecutando)

Clase Introduccion a administracion de Memoria

Objetivos= responsabilidad del kernel, soporte brindado por Hardaware, manejo optimo de recursos

Mediante hardware y etructuras de datos se crea un intermediario que pase se los procesos logicos a procesos fisicos en memoria ram

Clase Administración de Memoria

1. Importancia de la Memoria Principal:

Los programas y datos deben residir en la memoria principal para poder ejecutarse y ser referenciados directamente. La administración de esta memoria es crucial para garantizar un funcionamiento eficiente de los sistemas operativos (SO).

2. Funciones del Sistema Operativo:

El kernel del sistema operativo es el encargado de gestionar la memoria principal o RAM. Las principales funciones del SO en este contexto son:

- Llevar un registro de las partes de memoria que están en uso y aquellas que están libres.

- Asignar y liberar memoria de manera eficiente.

- Abstraer al programador de la asignación de memoria, garantizando la seguridad entre procesos (evitando que un proceso acceda a la memoria de otro) y ofreciendo espacios de código en común.

3. Requisitos de la Administración de Memoria RAM:

Los sistemas operativos deben cumplir con varios requisitos para gestionar la RAM de manera efectiva:

- Reubicación: El programador debe abstraerse del comportamiento del proceso durante su ejecución. Si un proceso se intercambia (swap) con otro, las direcciones pueden cambiar, por lo que es esencial mantener actualizadas las referencias a la memoria.

- Protección: Los procesos no deben acceder a ubicaciones de memoria fuera de sus límites permitidos.

- Compartición: Los procesos deben poder compartir espacios de memoria comunes para optimizar el uso de memoria y permitir el trabajo conjunto.

4. Espacio de Direcciones:

El espacio de direcciones es el rango de direcciones posibles que un proceso puede utilizar para direccionar sus instrucciones y datos. Este tamaño depende de la arquitectura del sistema.

- Direcciones Lógicas: Representan una dirección en el espacio de direcciones del proceso.

- Direcciones Físicas:Se refieren a una ubicación específica en la memoria física (dirección absoluta).

5. Conversión de Direcciones Lógicas a Físicas:

Cuando se utilizan direcciones lógicas, es necesario convertirlas a direcciones físicas mediante registros auxiliares:

- Registro Base: Indica el inicio de la zona de memoria asignada al proceso.

- Registro Límite: Define el tamaño de la zona de memoria que se puede utilizar.

El proceso de conversión implica sumar la dirección lógica al registro base y verificar que la suma no exceda el registro límite. Si la comparación falla, se genera una interrupción por software (trap) debido a un acceso de memoria inválido.

6. Resolución de Direcciones en Tiempo de Ejecución:

Cada vez que se accede a una dirección lógica, debe traducirse a una dirección física, un proceso que debe ser lo suficientemente rápido para no afectar el rendimiento de la CPU. La Unidad de Manejo de Memoria (MMU) realiza esta traducción y determina si el acceso a la RAM es válido.

7. Mecanismos de Asignación de Memoria:

Los métodos de asignación de memoria incluyen:

- Particiones Fijas: El SO divide la memoria RAM en particiones de tamaño predefinido que se asignan a los procesos.

- Particiones Dinámicas: No hay predivisión; la memoria se asigna a medida que los procesos lo requieren.

8. Fragmentación:

La fragmentación se refiere a la falta de contigüidad en la memoria disponible:

- Fragmentación Interna: Se produce cuando hay espacio no utilizado dentro de una partición asignada, y no tiene solución.

- Fragmentación Externa: Se refiere a la falta de espacio contiguo que puede solucionarse mediante la compactación.

9. Técnicas de Administración de Memoria:

- Paginación: Consiste en dividir el espacio físico en pequeños bloques (marcos) y el espacio lógico en páginas de igual tamaño. El SO gestiona la correspondencia entre páginas y marcos a través de una tabla de páginas, permitiendo el uso no contiguo de la memoria.

- Segmentación: Se basa en dividir el espacio de direcciones en segmentos que pueden ser cargados de manera no continua. Cada dirección lógica en segmentación se compone de un

número de segmento y un desplazamiento. Esto permite una mejor compartición y protección de la memoria.

- Segmentación Paginada: Combina las ventajas de la segmentación y la paginación, proporcionando un primer nivel de resolución mediante la segmentación y un segundo nivel mediante la paginación. Cada segmento puede estar asociado a un conjunto de páginas, optimizando la gestión de memoria.

10. Consideraciones Finales:

La administración de memoria es una tarea crítica que debe ser manejada eficientemente por el sistema operativo, aprovechando las capacidades del hardware y asegurando que los procesos tengan acceso seguro y controlado a la memoria. Esto incluye la protección de los procesos y la posibilidad de compartir espacios de memoria, permitiendo así un funcionamiento eficaz de los sistemas computacionales.