Introduccion a los Sistemas Operativos

Teoria 1 – 20/8/24:

Sistema Operativo:

* Es un software que esta pensado para administrar otro software.
* Requiere un procesador y memoria
* El SO oculta el HW y presenta a los programas abstracciones mas simples de manejar.
* Usuario 🡪 SO 🡨 Hardware
* Administra los recursos de HW de uno o mas procesos.
* Ejecución simultanea de procesos.

Interrupciones:

* Por Hardware: Son asincrónicas
* Por Software: Se causan por algo que se esta haciendo mal. Por ejemlo: Acceso a memoria indebido, división por 0, buffer overflow. Se causa por el ciclo de execute.

Teoría - 27/8/24:

System Calls:

* Es la forma en que los programas de usuario acceden a los servicios del SO
* Se ejecutan en modo kernel o supervisor
* Los parámetros asociados a las llamadas pueden pasarse de carias maneras: por registros, bloques o tablas en memoria o la pila.
* Ej: count=read(file, buffer, nbytes);

Los programas no funcionan en los dos con el mismo código ya que las System Calls en cada sistema operativo es diferente.

Los compiladores no son los mismos entre sistemas operativos. Entre 32 y 64 bits es diferente, RM e INTEL también son diferentes, etc.

* El codigo: read(file, buffer, nbytes); se traduce en:
  + Push file
  + Push buffer
  + Push nbytes
  + Call

Se usa como una “api” para llegar del mundo usuario al mundo kernel, que hace:

Se busca forzar una interrupción

Para buscar que llamada al sistema busco de todas las que hay

En la llamada a sistema hay un vector de direcciones de funciones

Una vez que estoy en el vector de llamadas al sistema, el sys call handler, es el que sabe a donde avanzar y ejecutando la llamada a sistema.

SYSCALL:

Para programar el clásico “hello world” se necesitan mínimo 2 llamadas al sistema:

* SYSCALL WRITE
* SYSCALL EXIT

El comando man es para ayuda, “man write” en consola te dice que hace el write.

En arquitectura de 32 bits, Linux tiene 3 archivos standard:

* Entrada standard pos 0
* Salida standard pos 1
* Error standard pos 2

En arquitectura de 64 bits

Proceso:

* Es un programa en ejecución
* Para nosotros serán sinónimos: tarea, job y proceso.

Diferencia entre programa y proceso:

Programa:

* Es estático
* No tiene program counter
* Existe desde que se edita hasta que se borra

Proceso:

* Es dinámico
* Tiene program counter
* Su ciclo de vida comprende desde que se solicita ejecutar hasta que termina.

El programa es el “.exe”, cuando vos ejecutas el programa, se crea un proceso que ejecuta el programa. Lo que se ejecuta es el proceso que ejecuta el programa.

El programa nunca cambia, lo que cambia es el proceso ya que cada vez que se abre se crea un proceso que ejecuta este programa, y cuando se cierra se termina la ejecución del proceso.

Los procesos pueden coexistir en un SO, pero si solo tenemos una CPU, se puede ejecutar 1 por vez

Atributos de un proceso:

* Identificación del proceso y del proceso padre.
* Identificación del susuario que lo “disparo”
* Si hay estructura de gruspo, grupo que lo disparo.
* En ambientes multiusuario, desde que terminal y quien lo ejecuta.

Process control block (PCB):

* Estructura de datos asociada al proceso (abstracción)
* Existe 1 por proceso
* Es lo primero que se crea cuando se crea un proceso y lo ultimo que se borra cuando termina.
* Contiene la información asociada con cada proceso:
  + PID, PPID, etc.
  + Valores de los registros de la CPU (PC, AC, etc.)
  + Planificación (estado, prioridad, tiempo consumido, etc.)
  + Ubicación (representación) en memoria.
  + Accouting
  + Entrada salida (estado, pendientes, etc.)

Practica – 29/8/24:

En /dev hay archivos que identifican dispositivos.

En /etc hay archivos de configuración.

UEFI:

hace como un escribano

cuando el so bootea, va a la parte del uefi y checkea si el software esta certificado y quien lo certifico. Lo mismo ocurre con el navegador.

Practica 1 fin hasta UEFI.

Practica 2 empieza:

Las particiones se identifican con letras desde la C en Windows.

Las particiones en Linux están basadas en los buses:

* /dev/hda: configurado como Master en el 1º bus IDE
* /dev/hdb: configurado como Slave en el 1º bus IDE
* /dev/hdc: configurado como Master en el 2º bus IDE
* /dev/hdd: configurado como Slave en el 2º bus IDE

Después surgieron otros nuevos:

* /dev/sda
* /dev/sdb
* /dev/sdc
* /dev/sdd

Si se cambian los discos de lugar en los puertos de la mother, no va a bootear ya que se tiene guardado donde esta el /boot y no lo encontrara.

Usar VIM

Los archivos mas importantes de la parte de USUARIO son:

* /etc/passwd
* /etc/group
* /etc/shadow

Comandos para manejo de usuarios:

* Useradd
* Passwd
* Usermod
* Userdel
* Groupdel

Permisos:

RWX

R: READ

W: WRITE

X: EXECUTION

Sobre los usuarios se aplican:

UGO:

1. Permisos del dueño
2. Permisos del grupo
3. Permisos de otros usuarios

Para cambiar los permisos se usa CHMOD:

1 1 1  
R W X  
  
Esto en binario es 7, para que tenga todos los permisos se debe poner 7.

1 0 1  
R W X

Esto en binario es 5, que te da permisos de lectura y ejecución, el mas normal.

Para tener todos los permisos de dueño y el grpo y otros usuarios tener solo lectura y ejecución, se debe poner:

CHMOD 755 /directorio/

7 5 5  
U G O

El proceso de arranque:

1. Se empieza a ejecutar el codigo del BIOS
2. El BIOS ejecuta el POST
3. El BIOS lee el sector de arranque (MBR)
4. Se carga el gestor de arranque (MBC)
5. El bootloader carga el kernel y el initrd
6. Se monta el initrd como sistema de archivos raız y se inicializan componentes esenciales (ej.: scheduler)
7. El Kernel ejecuta el proceso init y se desmonta el initrd
8. Se lee el /etc/inittab
9. Se ejecutan los scripts apuntados por el runlevel 1
10. El final del runlevel 1 le indica que vaya al runlevel por defecto
11. Se ejecutan los scripts apuntados por el runlevel por defecto
12. El sistema esta listo para usarse

Comandos:

El pipe “|” concatena comandos

Wc: cuenta líneas

“algo” > miarchivo modifica el archivo entero.

“algo mas” >> miarchivo agrega al final del archivo.

Teoría – 30/8/24:

Que es el espacio de direcciones en un proceso?

* Es el conjunto de direcciones de memoria que ocupa el proceso:
  + Stack
  + Text
  + Datos
* No incluye su PCB o tablas asociadas.
* faltan

El bus de direcciones le da el espacio de direcciones del proceso

El contexto de un proceso:

* Incluye toda la información que SO necesita para adminstrar el proceso, y la CPU para ejecutarlo correctamente.
* Son parte del contetto, los registros de cpu, inclusive el contador de program, prioridad del proceso si tiene E/S pendientes, etc.

Cambio de contexto (Content Switch):

* Se produce cuando la CPU cambia de un proceso a otro.
* Se debe resguardar el contexto del proceso saliente, que pasa a espera y retornara después a la CPU.
* Se debe cargar el contexto del nuevo proceso y comenzar desde la instrucción siguiente a la última ejecutada en dicho contexto.
* Es tiempo no productivo de CPU.
* El tiempo que consume depende del soporte de HW.

Cada interrupción (que es la q despierta al kernel) va a involucrar un Content Switch

Este modelo plantea meter el kernel dentro de los procesos.

El kernel “dentro” del proceso:

* El “código” del kernel se encuentra dentro del espacio de direcciones de cada proceso
* El kernel se ejecuta en el mismo contexto que algún proceso de usuario
* El kernel se puede ver como una colección de rutinas que el proceso utiliza

Las PCB son estructuras de datos que tienen que estar en la ram.

El stack/pila tiene:

Stack modo kernel

* Si el proceso pisa alguna dirección de retorno del kernel, cuando vuelva al modo usuario se arma quilombo, por eso se tiene que administrar bien los stack pointer.

Stack modo usuario