SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Pabellón de Arteaga

INSTITUTO TECNOLÓGICO® de Pabellón de Arteaga TEC

Sistemas Operativos I

Resumen Lectura 1

Presenta:
Ricardo Montoya Gómez
Docente:
Eduardo Flores Gallegos

19/04/2018

INTRODUCCION

Ha habido cuatro versiones del sistema de tiempo compartido UNIX. La primera (alrededor de 1969-70) corría en las computadoras DEC-PDP 7 y DEC-PDP-9. La segunda versión corría en la desprotegida computadora PDP-11/20. La tercera incorporó la multiprogramación y corría en las computadoras PDP-11/34, /40, /45, /60 y /70; este es la que se describe en la anterior versión publicada de este "paper", y también considerablemente la más usada hoy en día.

Quizás el mayor logro de UNIX es demostrar que un poderoso sistema operativo para uso interactivo no necesita ser caro ni en equipamiento ni en esfuerzo humano

Además de lo propio al sistema operativo, algunos programas disponibles bajo UNIX son

- Compilador C.
- Editor de textos basado en QED.
- Assembler, cargador de vínculos, debugger simbólico.
- Programas de photoypesetting y ecuaciones.
- Docenas de lenguajes incluyendo Fortran 77, Basic, Snobol, APL, Algol 68, M6, TMG, Pascal.

ENTORNO DE HARDWARE Y SOFTWARE

La PDP-11/70, donde el sistema UNIX de Investigación, es instalado es una computadora con palabras de 16 bits (8 bits byte) con 768K bytes de memoria principal; el núcleo ocupa 90K bytes de código dividido en partes iguales y tablas de datos. Este sistema, sin embargo, incluye un gran número de controladores de dispositivos y disfruta de una generosa porción de espacio para buffers de E/S y tablas del sistema; un sistema capaz de correr el software mencionado antes puede requerir como mínimo 96K bytes de núcleo en total.

La preponderancia del software UNIX es escrita en el antes mencionado lenguaje C. Versiones anteriores de este sistema operativo fueron escritas en lenguaje ensamblador (assembly), pero durante el verano de 1973, fue reescrito en C. El tamaño del nuevo sistema fue de alrededor de un tercio más grande que el anterior. Pero dado que el nuevo sistema no solo se volvió más fácil de entender y de modificar, sino que también incluía muchas mejoras funcionales, incluyendo multiprogramación, y la capacidad de compartir código reentrante entre varios programas de usuario, consideramos este incremento de tamaño muy aceptable.

EL SISTEMA DE ARCHIVOS

El rol más importante del sistema es el de proveer un sistema de archivos. Desde el punto de viste del usuario, hay tres tipos de archivos: archivos ordinarios de disco, directorios, y archivos especiales.

- **Archivos Ordinarios:** Un archivo contiene cualquier información que el usuario ponga en él, Un archivo consiste simplemente de una cadena de caracteres, con líneas demarcadas por el carácter de línea nueva. Los programas binarios son secuencias de palabras tal como aparecerán en la memoria principal cuando el programa comience su ejecución.
- **Directorios:** Los directorios proveen un mapeo entre los nombres de archivos y los archivos mismos, y esto induce a una estructura sobre el sistema de archivos como un todo. Un directorio se comporta exactamente como un archivo ordinario, excepto que no puede ser escrito por programas sin privilegios, así que el sistema controla el contenido de los

directorios. Sin embargo, alguien con el permiso apropiado puede leer un directorio como cualquier otro archivo.

Los archivos son nombrados por secuencias de 14 o menos caracteres. Cuando el nombre de un archivo es especificado al sistema, este debe ser de la forma path name, que es una secuencia de nombres de directorios separados por una barra "/", y terminando con un nombre de un fichero. Si la secuencia comienza con una barra, la búsqueda comienza en el directorio root.

Una ruta que no empiece con "/" causa que el sistema comience la búsqueda en el directorio actual en que se encuentra el usuario.

- **Archivos Especiales:** Los archivos especiales constituyen la característica más inusual del sistema de archivos UNIX. Cada dispositivo de E/S soportado es asociado con al menos uno de estos archivos. Los archivos especiales son leídos y escritos como un archivo ordinario de disco, pero los pedidos de lectura o escritura se presentan con la activación del dispositivo asociado.

  Hay tres ventajas en tratar los dispositivos de E/S de esta forma: archivos y dispositivos de E/S son tan similares como es posible; los nombres de archivos y de dispositivos tienen la misma sintaxis y el mismo significado, así que los programas que esperan un nombre de archivo como parámetro pueden recibir un nombre de dispositivo; finalmente, los archivos especiales están sujetos al mismo mecanismo de protección que los archivos regulares.
- **Sistemas de Archivos Removibles:** Aunque el root de un sistema de ficheros está siempre cargado en el mismo dispositivo, no es necesario que el sistema jerárquico entero resida en este dispositivo. El efecto de mount es causar referencias al anteriormente archivo ordinario para referirse en cambio al directorio root del sistema de archivos del volumen removible. En efecto, mount reemplaza una hoja del árbol de jerarquía (el archivo ordinario) por un nuevo subárbol (el sistema jerárquico cargado en el volumen removible). Después de mount, virtualmente no hay una distinción entre los archivos en el volumen removible y aquellos en el sistema de archivos permanente.

  Hay sólo una excepción a la regla de tratamiento idéntico de archivos en dispositivos diferentes: ningún enlace puede existir entre una jerarquía de un sistema de archivos y otra.
- **Protección:** Para ilustrar lo esencial de la E/S, algunas de las llamadas básicas se muestran debajo en un lenguaje anónimo que indicará los parámetros requeridos sin entrar en complejidades subyacentes. Cada llamada al sistema puede potencialmente devolver un error, que por simplicidad no es representado en la secuencia de llamadas. Para leer o escribir un archivo asumimos que éste ya existe, debe ser abierto por la siguiente llamada:

  filep = open(name,flag)

  donde name indica el nombre del archivo. Una ruta de acceso arbitraria puede ser dada. El argumento flag indica si el archivo se abrirá para escribir, para ser escrito, o para leer y escribir simultáneamente.

  El valor retornado filep es llamado descriptor de fichero (file descriptor). este es un entero chico usado para identificar el archivo en las llamadas subsiguientes para leer, escribir, o de alguna manera manipular el archivo.

  El sistema de archivos no mantiene ningún candado (lock) visible para el usuario, ni hay ninguna restricción del número de usuarios que pueden tener abierto un archivo para lectura o escritura. Aunque es posible que los contenidos de un archivo aparezcan revueltos si dos usuarios escriben en él simultáneamente, en la práctica no se presentan dificultades.

  Hay, sin embargo, suficientes "inter-candados" (inter-locks) para mantener la consistencia lógica de un sistema de archivos cuando dos usuarios participan simultáneamente en

actividades tales como escribir en el mismo archivo, crear archivos en el mismo directorio, o borrar cada archivo abierto del otro.

IMPLEMENTACION DEL SISTEMA DE ARCHIVOS

Este puntero es un entero llamado i-número (de número índice o "index number") del archivo. Cuando se accede al archivo, su i-número es usado como un índice en una tabla del sistema (i-lista o "i-list") cargada en una parte conocida del dispositivo en el cual reside el directorio. La entrada encontrada de ese modo (el i-nodo del archivo) contiene la descripción del archivo:

- el ID de usuario y de grupo del propietario
- los bits de protección
- la dirección del disco físico o la cinta para los contenidos del archivo
- el tamaño
- fecha de creación, de último uso, y última modificación
- el número de enlaces al archivo, que es el número de veces que aparece en un directorio
- un código indicando si es un directorio, un archivo ordinario, o un archivo especial

El propósito de una llamada de sistema **open** o **create** es cambiar la ruta de acceso dada por el usuario dentro de un i-número buscando explícita o implícitamente los directorios nombrados.

Cuando un nuevo archivo es creado, un i-nodo es alojado para éste y una entrada de directorio es hecha y contiene el nombre del archivo y su número de i-nodo. Hacer un enlace a un archivo existente implica crear una entrada de directorio con el nuevo nombre, copiar el i-número de la entrada del archivo original, e incrementar el campo de números de enlaces del i-nodo.

La noción de i-lista es una característica inusual de UNIX. En la práctica, este método de organización del sistema de archivos tiene demostrada mucha confiabilidad y fácil manejo. Para el sistema mismo, uno de sus puntos fuertes es que cada archivo tiene un nombre corto y sin ambigüedades, relacionado de una manera simple con la protección, la dirección, y otra información necesaria para acceder al archivo. Esto también permite un algoritmo muy simple y rápido para chequear la consistencia del sistema de archivos, por ejemplo, la verificación de qué porciones de cada dispositivo contienen información útil y aquellas que están libres para ser alojadas son disjuntas, y juntas completan el espacio del dispositivo.

PROCESOS E IMÁGENES

Una imagen es un ambiente de ejecución de computadora. Incluye una imagen de memoria, valores de registros generales, estado de archivos abiertos, directorio actual y demás. Una imagen es el estado actual de una pseudo-computadora.

Un proceso es la ejecución de una imagen. Mientras el procesador está ejecutando un proceso, la imagen debe residir en la memoria principal; durante la ejecución de otros procesos permanece en la memoria principal hasta la aparición de un proceso activo y de prioridad más alta que lo fuerza a ser intercambiado al disco (swap).

- **Procesos:** Excepto cuando el sistema se está autocargando para operar, un nuevo proceso sólo puede empezar a existir por el uso de la llamada a sistema fork:

processid = fork ()

- **Tuberías (pipes):** Los procesos deben relacionarse con sus familiares usando las mismas llamadas a sistema read y write que son usadas por el sistema de archivos de entrada y salida. La llamada:

  filep = pipe()

- **Ejecución de programas:** Otra primitiva de sistema trascendental es invocada por

  execute(file, arg1, arg2, ... , argn)

- **Sincronización de procesos:** Otra llamada a sistema de control de procesos:

  processid = wait(status)

- Terminación, Por último:

  Exit (status)

termina un proceso, destruye su imagen, cierra sus archivos abiertos, y generalmente lo hace desaparecer. El padre es informado mediante la llamada wait, y se le deja disponible status. Los procesos también pueden terminar como resultado de varias acciones ilegales o señales generadas por usuarios.

Perspectiva

Tal vez paradójicamente, el éxito del sistema UNIX se debe en gran medida al hecho de que no fue diseñado para cumplir con los objetivos predefinidos. La primera versión fue escrita cuando uno de nosotros (Thompson), dis satisfecho con las instalaciones de computadoras disponibles, descubrió un poco utilizado PDP-7 y se propuso crear una más ambiente hospitalario.

Primero: porque somos programadores, diseñamos naturalmente el sistema para hacer fácil escribir, prueba, y ejecutar programas. La expresión más importante de nuestro deseo de conveniencia de programación fue que el sistema fue arreglado para el uso interactivo, aunque la versión original apoyó solamente a un usuario. Que cree que un sistema interactivo adecuadamente diseñado es mucho más productivo y satisfactorio de usar que un sistema ' ' Batch ' '. Además, un sistema de este tipo es fácilmente adaptable al uso no interactivo, mientras que el versículo no es cierto.

Segundo: siempre ha habido restricciones de tamaño bastante graves en el sistema y su software. Dado los deseos parcialmente antagónicos de eficiencia razonable y poder expresivo, la restricción de tamaño tiene animó no sólo la economía, pero también una cierta elegancia del diseño.

Tercero: casi desde el principio, el sistema fue capaz de, y lo hizo, mantenerse. Este hecho es más importan de lo que podría parecer. Si los diseñadores de un sistema se ven obligados a utilizar ese sistema, rápidamente se hacen conscientes de sus deficiencias funcionales y superficiales y están fuertemente motivados para corregirlos antes de que sea demasiado tarde