COMO: Fundamentos de Unix e Internet.

Eric Raymond <esr@thyrsus.com>

Traducción: Camilo Figueroa <cigotete@gmail.com>

Revision History
Revision 2.9
Minor updates.
Revision 1.0
Initial revision.

Este documento describe enun lenguaje no técnico los fundamentos detrabajo de computadores tipo PC, sistemas operativos tipo Unix, y la Internet.

Tabla de contenido

1. Introducción

1.1. Propósito de este documento

Este documento tiene la intención de ayudar a los usuarios de Linux y de Internet quienes "aprenden haciendo" (learning by doing). Aunque esta es una gran manera de adquirir específicas habilidades, algunas veces nos deja problemas particulares en algunos conocimientos fundamentales. Estos problemas pueden dificultarnos la capacidad de pensar creativamente la solución efectiva de algunos problemas debido a la ausencia de un buen modelo mental sobre lo que realmente esta sucediendo.
Intentaré describir con un lenguaje simple y claro como trabaja todo esto. Este documento ha sido adecuado para personas que usan Unix o Linux en hardware de computadoras tipo PC. Aún así usualmente usaré el termino 'Unix' ya que la mayoría de lo que se describirá es consistente a través de las plataformas y las variantes de Linux.
Voy a asumir que usted esta usando un computador tipo PC Intel. Los detalles difieren ligeramente si usted esta usando un Alpha, un PowerPC o algún otro Unix, y que los detalles básico son los mismos.
No deseo repetir las mismas cosas, así que usted debe poner atención, pero esto no significa que debe aprender cada palabra que lea. Es buena idea solo hojear el documento cuando lo lea por primera vez, y regresar y releerlo unas pocas veces después de que haya digerido lo que haya leído.
Este es un documento en constante cambio. Intentaré crear nuevas secciones en la medida que los usuarios lo soliciten, así que regrese y revise periódicamente.

1.2. Nuevas versiones de este documento

Nuevas versiones de este documento (Unix and Internet Fundamentals HOWTO / COMO de Fundamentos de Unix e Internet) serán periódicamente publicadas en comp.os.Linux.help y comp.os.Linux.announce y news.answers. Estos documentos también serán subidos a varios sitios web y servidores FTP, incluyendo la página de inicio de LDP.
También puede visitar la última version de este documento en Internet en la dirección: http://www.tldp.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO/index.html.
Este documento ha sido traducido al Polaco y al Español

1.3. Comentarios, sugerencias y correcciones

Si usted tiene preguntas, comentarios o sugerencias sobre este documento, por favor sientase libre de escribir al email de Eric Raymond, en esr@thyrsus.com. Acepto cualquier sugerencia o critica. Especialmente recibo links donde se encuentren detalladas explicaciones sobre conceptos individuales. SI usted encuentra un error en este documento, por favor hágamelo saber para corregirlo en la siguiente versión. Gracias.
Sobre esta traducción al español, comentarios, sugerencias y correcciones a cigotete@gmail.com.

1.4. Recursos relacionados

Si usted esta leyendo este documento buscando como hackear, usted también debe leer el Como Ser un Hacker, Preguntas y Respuestas / HowTo Become A Hacker FAQ, que contiene links hacia otros útiles

recursos.

inicio

2. Anatomía básica de su computador

Dentro de su computador hay un chip de procesamiento que realiza actualmente el proceso de computación. Él tiene una memoria interna (que la gente de DOS/Windows llama "RAM" y la gente de Unix a menudo llama "Nucleo" (El termino en Unix proviene de una vieja costumbre desde cuando las RAM estaban hechas con formas de donas con núcleo de ferrita). El procesador y la memoria viven en la Tarjeta Madre (MotherBoard), que es el corazón de su computador.
Su computador tiene una pantalla y un teclado. tiene un disco duro y un CD-ROM y quizás un disco blando. Algunos de esos dispositivos funcionan gracias a tarjetas controladoras que se insertan dentro de la tarjeta madre y ayudan al computador a manejar estos dispositivos. Otros dispositivos funcionan directamente sobre la tarjeta madre gracias a chipsets (conjunto de chips. n del t.) especializados, que cumplen la misma función de la tarjetas controladoras. El funcionamiento de su teclado es muy sencillo para necesitar una tarjeta aparte; en este caso el controlador esta construido directamente dentro del chasis del mismo teclado. Más adelante nos adentraremos en algunos de los detalles de como trabajan estos dispositivos. Por ahora, veamos unas pocas cosas básicas para recordar sobre como estos dispositivos trabajan juntos.
Todas las partes dentro de su computador están conectadas por un bus. Físicamente el bus es donde usted inserta las tarjetas controladoras (la tarjeta de video, el controlador de disco o si se tiene, la tarjeta de sonido). El bus es la autopista entre su procesador, su pantalla, su disco y cualquier otra cosa.
(Si usted ha visto referencias a 'ISA', 'PCI', y 'PCMCIA' relacionadas con los computadores y no las ha entendido, estos son tipos de bus. ISA es, excepto por algunas diferencias, el mismo bus que fue usado en los computadores originales IBM en 1980. Actualmente están fuera de uso. PCI (Periferical Component Interconnection / Componente de Interconexión Periférica) es el bus usado en la mayoría de los mas modernos computadores así como en los más modernos computadores Machintosh. PCMCIA es una variedad de el bus ISA con conectores más pequeños, usado en computadores portátiles.
El procesador, quien hace que todo funcione, realmente no puede comunicarse directamente con ninguno de los demás dispositivos, él tiene que hablar directamente con ellos usando el bus. El único otro subsistema que es realmente rápido y con acceso inmediato al procesador es la memoria (el núcleo).
Cuando su computador lee un programa o los datos del disco, lo que realmente sucede es que el procesador usa el bus para enviar al controlador del disco una solicitud de lectura del disco. Un instante después el controlador del disco usa el bus para advertirle al procesador que él (el controlador del disco) ha leído los datos y los ha puesto en algún lugar de la memoria. El procesador entonces puede usar el bus para mirar los datos
El teclado y la pantalla también usan el bus para comunicarse con el procesador, pero de una manera simple. Esto lo discutiremos más adelante. Por ahora usted sabe lo suficiente como para entender que sucede cuando enciende el computador.

inicio

3. Que sucede cuando enciende el computador?

Un computador sin un programa funcionando es solo un trozo inerte de elementos electrónicos. La primera cosa que un computador hace cuando es encendido es iniciar un programa especial llamado Sistema Operativo. EL trabajo del sistema operativo es ayudar a otros programas del computador a trabajar con los difíciles detalles de controlar el hardware del computador.
El proceso de iniciar el sistema operativo se llama Booting/arranque (originalmente el termino fue bootstrapping/'nicio de si mismo' en alusión al proceso de arrancar "comenzando desde los pies a la cabeza"). El computador sabe como iniciar porque la instrucciones del arranque están definidas dentro de uno de sus chips, el chip BIOS (Basic Input Output Sistem / Sistema básico de entrada y salida)
El chip BIOS le indica al computador que mire un un pequeño lugar, usualmente en el disco duro numerado

con el menor numero (por si hay otros discos. n. del t.) buscando un programa especial llamado el 'boot loader'/'cargador del arranque' (en Linux, el cargador del arranque es llamado Grub o LILO). EL cargador de arranque es ubicado dentro de la memoria e iniciado. El trabajo del cargador del inicio es es iniciar el sistema operativo.
El cargador inicia esto buscando primero el kernel, cargándolo dentro de la memoria e iniciándolo. Cuando usted inicia Linux y ve en pantalla el texto “LILO” (o “Grub”. n. del .t.) seguido de un grupo de puntos, significa que se esta cargando el kernel (cada punto significa que ha cargado otro 'bloque del disco' del código del kernel.)
¿Usted podría preguntarse porque el BIOS no carga directamente el kernel?, ¿o porque el proceso del 'cargador del inicio' consta de estos dos pasos? Bien, la razón es porque la BIOS no es muy astuta. En efecto es muy estúpida y Linux no usa la BIOS después del momento del arranque. la BIOS fue originalmente escrita para primitivos computadores de 8 bits con diminutos discos, y literalmente no puede acceder al disco lo suficiente como para cargar directamente el kernel. Adicionalmente el proceso de inicio también permite iniciar (y escoger. n del t.) uno de varios sistemas operativos que se encuentran ubicados en diferentes lugares del disco en elevento poco probable de que Unix no sea suficiente para usted.
Una vez el kernel arranca, él tiene que mirar a su alrededor encontrando el resto del hardware y alistándose para hacer funcionar los programas. El kernel no hace esto metiéndose en las posiciones ordinarias de memoria, en vez de ello lo hace en los puertos I/O ('Input/Output'/'Salida/Entrada)(I/O: son lugares especiales en el Bus que probablemente están para que las tarjetas de control de los dispositivos puedan escuchar las peticiones/ordenes del kernel. El kernel no hace esto de una manera desordenada; él tiene una gran cantidad de información incorporada sobre posiblemente donde y como los controladores responderán en el caso de que estén presentes. Este proceso es llamado autoprobing / autodetección.
La mayoría de los mensajes que usted ve en el momento del arranque es el kernel realizando la autodetección de su hardware a través de los puertos I/O, identificando cual esta disponible para cada dispositivo y adaptando dicho dispositivo a su maquina. El kernel Linux es extremadamente bueno en esto, mejor que la mayoría de los otros kernels Unix y mucho mejor que el de DOS o Windows. En efecto, muchos veteranos de Linux piensan que la habilidad de Linux en la detección del proceso dearranque (lo cual hace que sea relativamente fácil para instalar) fue la principal razón para dejar de ser un paquete experimental y libre de Unix y atraer a una masa critica de usuarios.
El que el kernel se encuentre completamente cargado y funcionando no significa que haya terminado el proceso de arranque; este es solo el primer paso del proceso (algunas veces llamado 'run level 1'/'nivel de funcionamiento 1'). Después del primer paso, el kernel toma control de un especial proceso llamado'init' el cual despliega varios procesos de administración.
El primer trabajo del proceso 'init' usualmente es el controlar que la supervisión de los discos este bien realizada. El sistema de archivos del disco tiene cosas muy fragiles; podría estar muy afectado por una falla del hardware o un súbito cambio de voltaje, estas son buenas razones para tomar medidas de recuperación antes de que Unix este de nuevo funcionando. Veremos algo de esto mas en detalle cuando hablemos sobre como puede fallar el sistema de archivos.
El siguiente paso del proceso 'init' es arrancar varios 'demonios'. Un demonio es un programa como la cola de impresión, un detector de llegada de emails o un servidor web que funciona escondido en segundo plano, esperando cosas que hacer. Estos programas especiales a menudo tienen que coordinar varias peticiones que podrían entrar en conflicto. Estos procesos son 'demonios' por que a menudo es fácil escribir un programa que funcione constantemente y conozca sobre todas las peticiones que debería tratar para asegurar que una multitud de copias de procesos (cada copia procesando una solicitud y todas funcionando al mismo tiempo) no se amontonen. La especifica colección de demonios de arranque puede variar, pero casi siempre incluye el proceso de la cola de impresión (un demonio portero para su impresora).
El siguiente paso es prepararse para los usuarios. El proceso Init inicia una copia del programa getty para observar/atender su consola (y quizás mas copias para atender conexiones 'dial-in'/'por marcado remoto' en puerto serial). Este programa es el que despliega el aviso de acceso de su consola. Una vez todos los demonios y procesos getty de cada terminal han sido iniciados, entramos al 'run level 2'/'nivel de funcionamiento 2'. En este nivel usted puede acceder y correr programas.
Pero nosotros aún no hemos alcanzado este segundo paso. El siguiente paso consiste en arrancar varios demonios que dan soporte a redes y otros servicios. Una vez esto se ha hecho estamos en el 'run level 3'/nivel de funcionamiento 3' y el sistema se encuentra totalmente listo para ser usado.

inicio

4. ¿Que pasa cuando usted se registra en el sistema?

Cuando usted se registra (dando un nombre a getty) usted se identifica frente al computador. El corre entonces un programa llamado (naturalmente) login. EL cual toma su clave de acceso y revisa si usted esta autorizado para usar la maquina. Si usted no esta autorizado su intento de acceso es rechazado. Si usted esta autorizado el programa login realiza unas pocas tareas de 'housekeeping'/'administración-mantenimiento' e inicia el interprete de comandos, el 'shell'/'ventana de sesión'. (Si, getty y login pueden ser un programa. Ellos están separados por razones históricas que no son relevantes aquí)
Esto es solo un poco de lo que el sistema hace antes de darle una ventana de sesión (usted necesitará conocer esto mas adelante cuando hablemos sobre los permisos de archivos). Usted se identifica con un nombre de acceso y una clave. Dicho nombre de acceso es revisado en un archivo llamado /etc/passwd, el cual contiene una secuencia de lineas, cada una describiendo una cuenta de usuario.
Uno de esos campos corresponde a una versión encriptada de la clave (algunas veces los campos encriptados son realmente almacenados en un segundo archivo /etc/shadow con ajustados permisos haciendo que sea difícil descubrir la clave). Cuando usted ingresa con una cuenta la clave es encriptada exactamente de la misma forma y el programa login compara si son iguales. La seguridad de este método esta en el hecho que, mientras que es fácil ir de la versión limpia de la clave a la versión encriptada, la forma inversa es muy difícil. De esta manera, aún si alguien puede ver la versión encriptada de la clave no le será posible usarla. (esto también significa que si usted olvida su clave no hay forma de recuperarla, solo funcionará cambiarla a algo que usted escoja.)
Una vez usted se ha registrado satisfactoriamente, usted obtiene todos los privilegios asociados con la especifica cuenta que usted esta usando. Usted también puede ser reconocido como parte de un 'grupo'. Un grupo es una colección de nombres de usuarios establecida por el administrador del sistema. Los grupos pueden tener privilegios independientes de los privilegios de sus miembros. Un usuario puede ser miembro de múltiples grupos. (para detalles sobre como trabajan los privilegios en Unix vaya a la sección de permisos.
Note que aunque usted se refiera normalmente a usuarios y grupos por un nombre, ellos internamente están realmente almacenados internamente en forma de identificadores numéricos (Idś). El archivo de clave compara su nombre de cuenta contra un identificador de usuario; el archivo /etc/group compara los nombres de grupo contra un identificador de grupo. Los comandos que negocian con las cuentas y los grupos realizan automáticamente la traducción.
Su inicio de cuenta también contiene un 'directorio de inicio'. EL lugar es donde el sistema de archivos de Unix ubicará sus archivos personales. Finalmente su cuenta de inicio también ajustará su ventana de sesión, el interprete de comandos que el programa login arrancará para aceptar sus comandos.

inicio

5. ¿Que sucede cuando usted corre programas desde la ventana de sesión?

La ventana de sesión es el interprete para los comandos que usted escribe en ella; es llamada shell (caparazón) porque ella envuelve y oculta el kernel del sistema operativo. Es una importante característica de Unix que el shell y el kernel sean programas separados que se comunican a través de un pequeño conjunto de llamadas del sistema. Esto permite la posibilidad de probar múltiples shells y diferentes sabores de interfaces.
Un shell normalmente mostrará a usted el símbolo '$' que usted ve después de acceder en el sistema (a menos que usted haya ajustado esto para algún otro símbolo). Nosotros no hablaremos sobre la sintaxis de la shell y sobre las sencillas cosas que usted puede ver en la pantalla; en lugar de ello vamos a mirar detrás de la escena de lo que sucede desde el punto de vista del computador.
Desde el momento de arranque y después de que usted corre un programa, usted puede pensar en su computador como un contenedor en el que habita un zoológico de procesos que están esperando algo que hacer. Ellos están en la espera de eventos. Un evento puede ser el presionar una tecla o mover el ratón. O si

su maquina está enganchada a una red, un evento podría ser un paquete de datos entrando en la red.
El kernel es uno de estos procesos. Es especialmente único porque el controla cuando pueden corre otros procesos del usuario, y es normal que sea el único proceso que tiene acceso directo al hardware de la maquina. En efecto, los demás procesos del usuario tienen que hacer una petición al kernel cuando ellos quieran conseguir una entrada del teclado, escribir en la pantalla, leer el disco o escribir en él, o solamente hacer cualquier otra cosa que impliquen bits en la memoria.
Normalmente todas las entradas y salidas (I/O) van a través del kernel así que el puede programar las operaciones y prevenir procesos traslapados. Unos pocos procesos de usuario tienen permitido rodear el kernel, usualmente por que se les ha dado acceso directo a los puertos I/O. Los servidores X (programas que manejan solicitudes de otros programas para hacer gráficas de pantalla en la mayoría de los equipos Unix.
La Shell es solo un proceso de usuario sin nada en particular. El espera la pulsación del teclado, escuchando (a través del kernel) al puerto I/O del teclado. Tal como el kernel los ve, los repite en su pantalla. Cuando el kernel recibe la pulsación de la tecla 'Enter' el pasa su linea de texto al shell. El shell intenta interpretar estas teclas como comandos.
Vamos a decir que usted teclea 'ls' y pulsa la tecla 'Enter' para invocar el listador de directorios de Unix. La shell aplica sus reglas preestablecidas para entender que usted desea correr el comando u orden de ejecución en el archivo /bin/ls. El comando hace una llamada de sistema solicitando al kernel iniciar /bin/ls como un nuevo proceso hijo dándole acceso a la pantalla y el teclado a través del kernel. Entonces la shell duerme esperando que ls termine.
Cuando la orden /bin/ls esta hecha, le avisa al kernel que ha terminado emitiendo al sistema una petición de salida. El kernel entonces despierta la shell diciéndole que puede continuar. La shell emite otro aviso y espera por otra entrada.
Sin embargo otras cosas pueden estar sucediendo mientras su orden 'ls' es ejecutada (tenemos que suponer que usted esta listando un directorio muy largo). Usted puede cambiar a otra consola virtual, registrarse en ella, e iniciar por ejemplo un juego de Quake. O suponiendo que usted esta enganchado a la Internet. Su maquina puede estar enviando o recibiendo email mientras /bin/ls corre.

inicio

6. ¿como trabajan los dispositivos de entrada e interrupción?

Su teclado es un dispositivo de entrada muy simple; simple porque genera pequeñas cantidades de datos muy lentamente (por un estándar de computación). Cuando usted presiona o libera una tecla, el evento es enviado por el cable del teclado para provocar 'una interrupción de hardware' / 'hardware interrupt'.
Es trabajo del sistema operativo el atender este tipo de interrupciones. Para cada posible tipo de interrupción habrá un 'manipulador de interrupción' / ' interrupt handler', como parte del sistema operativo que acoge cualquier dato asociado con dicho manipulador de interrupción (como su valor tecleado) hasta que pueda ser procesado.
Lo que el manipulador de interrupciones hace realmente con su teclado es enviar el valor tecleado dentro del área del sistema cerca del inicio de la memoria. Allí, el valor estará disponible para cuando el sistema operativo pase el control a cualquier programa que se supone esta actualmente leyendo el teclado.
Dispositivos de entrada mas complejos como los discos o la tarjetas de red trabajan de similar manera. Anteriormente mi referencia a un controlador de disco que usa el bus para hacer una petición al disco era suficiente. Lo que realmente sucede es que el disco recibe una interrupción. El manipulador de interrupciones del disco entonces copia los datos recibidos dentro de la memoria para su posterior uso por parte del programa que ha hecho la petición.
Cada tipo de interrupción tiene asociado un 'nivel de prioridad' / 'priority level'. Interrupciones con prioridades bajas (como los eventos del teclado) tienen que esperar por/sobre las interrupciones con prioridades altas (como los tictac del reloj o los eventos del disco. Unix esta diseñado para dar prioridad al tipo de eventos que necesitan ser procesados rápidamente para mantener libre de problema a las respuestas que da la maquina.
En los mensajes que aparecen en el momento del arranque, usted puede ver referencias a números IRQ. Note que una de las típicas formas de hacer una incorrecta configuración de hardware es teniendo a dos diferentes dispositivos intentando usar el mismo IRQ, sin entender exactamente por que.
He aquí la respuesta. IRQ es la abreviatura de “Petición de Interrupción” / “Interrupt Request”. El sistema

operativo necesita saber en el momento del arranque que numero de interrupción usara cada dispositivo de hardware, para asociar adecuadamente los manipuladores de interrupción. Si dos diferentes dispositivos tratan de usar el mismo IRQ,las interrupciones serán enviadas al manipulador equivocado. Esto por lo menos inhabilitará el dispositivo y puede algunas veces confundir negativamente al sistema, lo suficiente como para saturarlo o anularlo.

inicio

7. ¿Que hace su computador haga varias cosas a la vez?

Realmente el no lo hace. Los computadores pueden solamente hacer una tarea a la vez. Pero el computador puede cambiar de tareas muy rápidamente y un tonto podría pensar para sus adentros que hace varias cosas a la vez. Esto es llamado timesharing / 'tiempo compartido'
Una de las tareas del kernel es administrar el timesharing. El tiene una parte llamada el programador, el cual mantiene dentro de si información sobre otros procesos (no kernel). Cada 1/60 de segundo un temporizador se apaga en el kernel, generando una interrupción en el reloj. El programador detiene cualquier proceso que actualmente este corriendo, suspendiéndolo en un lugar, y tomando control de otro proceso.
1/60 de segundo puede no sonar a mucho tiempo. Pero para los procesadores de hoy es suficiente para correr decenas de miles de instrucciones de maquina, con las cual puede hacer mucho trabajo. Si aún usted tiene muchos procesadores, cada uno puede completar un poco en cada uno de sus intervalos de tiempo.
En la practica, un programa puede no lograr usar la totalidad de su intervalo de tiempo. Si una interrupción llega desde un dispositivo I/O, el kernel detendrá la tarea actual, iniciará el manipulador de interrupción y retornará a la actual tarea. Una tormenta de interrupciones de alta prioridad puede exprimir el normal procesamiento: esta mala conducta es llamada 'thrashing' y por fortuna es muy difícil de inducir en los modernos Unix
En efecto, la velocidad de los programas es solo muy raramente limitada por la cantidad de tiempo (de maquina) que pueda obtener (hay algunas excepciones a esta regla, como las tarjetas de sonido y las de generación de gráficos 3D). Mucho mas a menudo las demoras suceden cuando el programa tiene que esperar un dato desde una unidad de disco u una conexión de red.
Un sistema operativo que soporte rutinariamente muchos procesos simultáneamente es llamado “multitasking” / “multitarea”. Desde sus cimientos, la familia de sistemas Operativos Unix fueron diseñados para la multitarea y Unix es muy bueno en ello – Mucho mas efectivo que Windows o el viejo Mac OS, los cuales tenían la multitarea atornillada dentro de ellos como una idea de último momento y la realizaban de una manera muy pobre. Una eficiente y confiable multitarea es en gran parte lo que hace a Linux superior para las redes (networking), las comunicaciones y los servicios web (web services).

inicio

8. Que hace que mi computador mantenga procesos de manera conjunta y organizada?

El programador del kernel cuida de dividir los procesos en el tiempo. Su sistema operativo también tiene que dividir estos procesos en el espacio, de tal forma que los procesos no puedan montarse unos con otros. Aun si usted supone que todos los programas intentan trabajar de manera colaborativa, usted no desearía que un bug (error) en uno de ellos sea capaz de afectar a otros. Las operaciones que su sistema operativo realiza para resolver este tipo de problemas se llama 'administración de memoria' / 'memory management'. En su maquina cada proceso necesita su propia área de memoria, como un lugar donde correr su código y mantener sus variables y resultados. Usted puede pensar que este conjunto de cosas esta compuesta de un segmento de código de solo lectura (que contiene las instrucciones de los procesos) y un segmento de datos escribible (conteniendo todas las variables del proceso). El segmento de datos es único para cada proceso, pero si dos procesos están corriendo el mismo código, Unix, automáticamente lo organiza para que estos

procesos compartan el mismo segmento de código, como una medida de eficiencia.

8.1. Memoria virtual: versión simple

La eficiencia es importante por que la memoria tiene un alto costo. Algunas veces usted no tiene la suficiente memoria para mantener todos los programas que la maquina esta corriendo, especialmente si usted usa programas de gran tamaño como un servidor X. Para resolver estos casos Unix usa una técnica llamada Memoria Virtual. El no trata de mantener para un proceso todo el código y datos necesarios. En lugar de ello, Unix mantiene solo un relativamente pequeño espacio de trabajo; el resto de el proceso es dejado en un lugar llamado 'space swap'/'espacio de intercambio' en su disco duro.
Note que en el pasado el termino "algunas veces" fue "casi siempre" debido a que el tamaño de la memoria era generalmente pequeño en relación al tamaño de los programas que corrían, así que el proceso de swapping era frecuente. Ahora la memoria es mucho menos costosa y aun las maquinas low-end (de baja categoría) tienen bastante memoria. Una moderna maquina personal de 64 MB de memoria o mas le es posible correr servidores X y un conjunto de tareas sin nunca necesitar el swap después de que se ha cargado en memoria.

8.2. Memoria virtual. Versión detallada

Realmente en la ultima sección se simplificaron las cosas un poco. Si, los programas ven la mayor parte de la mayoría como un grande y plano banco de direcciones de memoria, que una memoria física, y aquí un disco con 'memoria de intercambio' es usado para mantener esta ilusión. En todo caso su hardware no tiene más que cinco diferentes tipos de memoria dentro de si, y la diferencia entre estos tipos de memoria puede representar una buena decisión cuando los programas tienen que funcionar a máxima velocidad. Para entender realmente que sucede dentro de su computador, usted debe aprender como todas ellas trabaja en conjunto.
Los cinco tipos de memoria son estos: registros del procesador, cache interno (o en chip), cache externo (o fuera de chip), memoria principal, y disco. Y la razón de que sean varios tipos es simple: la velocidad cuesta dinero. He listado estos tipos de memoria en orden decreciente respecto a el tiempo de acceso y en orden creciente respecto al costo. La memoria de registros del procesador es la mas rápida y costosa, y puede ser accedida de manera aleatoria en cerca de un billón de veces por segundo, mientras que la memoria del disco es la mas lenta y barata, y puede realizar cerca de 100 accesos aleatorios en un segundo.
He aquí una lista que refleja velocidades datadas en el inicio del 2000 de una maquina de escritorio común. Mientras que la velocidad y la capacidad crecen los precios van bajando y es posible esperar que este tipo de proporciones se sostengan de manera constante.

Disco: Tamaño 13000 MB. Velocidad de acceso: 100KB/sec.
Memoria principal: Tamaño 256 MB. Velocidad de acceso: 100MB/sec.
Cache Externo: Tamaño 512 MB. Velocidad de acceso: 205MB/sec.
Cache Interno: Tamaño 32KB. Velocidad de acceso: 500MB/sec.
Procesador: Tamaño 28 bytes. Velocidad de acceso: 1000MB/sec.

No es posible construir alguna cosa mas allá de estos rápidos tipos de memoria, Esto requeriría mucho trabajo (y aun si no lo fuera) por que la memoria rápida es volátil. Esto significa que pierde su sustento cuando la energía desaparece. Y aquí hay un gran desequilibrio entre la velocidad del procesador y la velocidad del disco. Los tres niveles de en medio de la jerarquía de memorias (cache interno, cache externo y memoria principal) existen básicamente para cerrar esta brecha.
Linux y otros Unix tienen una característica llamada 'memoria virtual' / 'virtual memory'. Lo que esto significa es que el sistema operativo se comporta como si tuviera mas memoria de la que realmente tiene. Su verdadera memoria física principal se comporta mas como un gran conjunto de ventanas o valijas (caches) que un gran lugar de memoria "virtual", la mayor parte de la cual en un momento dado almacena en disco en una zona especial llamada 'swap area' / 'zona de intercambio'. Fuera del alcance de los programas de usuario, el sistema operativo se encuentra moviendo bloques de datos (llamados "paginas") entre la memoria y el disco para mantener esta ilusión. EL resultado final es que su memoria virtual es mucho mas grande pero no mas lenta que la memoria real.
Que tan lenta sea la memoria virtual depende físicamente de que tanto los algoritmos de swapping encuentren la forma para que los programas usen la memoria virtual. Afortunadamente los datos leídos y escritos por la memoria que están cerca en tiempo también tienden a estar cerca en el espacio de memoria.

Esta tendencia es llamada 'Localidad' / ' Locality' o mas familiarmente 'localización de referencia' (y esto es algo bueno). Si las referencias de memoria saltaran alrededor de la memoria virtual de manera aleatoria, generalmente se tendría que leer y escribir del/en disco para cada nueva referencia, y la memoria virtual sería tan lenta como lo es el disco. Pero porque los programas manifiestan realmente una fuerte localidad, el sistema operativo puede hacerlo(leer/escribir) con relativos pocos swaps/intercambios por referencia.
Por experiencia ha sido encontrado que el mas efectivo método para un amplio tipo de patrones de uso de memoria es muy simple: es llamado LRU o el algoritmo del “ultimo recientemente usado”. El sistema de memoria virtual toma los bloques del disco dentro de su conjunto de trabajo en la medida que lo necesite. Cuando él corre fuera de la memoria física al espacio de trabajo, él descarga el bloque 'mas recientemente usado'. Todos los Unix y la mayoría de los otros sistemas operativos con memoria virtual usan pequeñas variaciones del LRU.
La memoria virtual es el primer vinculo en el puente entre las velocidades del disco y el procesador. Esta explicitamente administrada por el sistema operativo. Pero aun hay un gran espacio entre la velocidad de la memoria principal y la velocidad en la cual el procesador puede acceder a su memoria de registro. Las memorias cache internas y externas apuntan aesto usando una técnica similar a la que hemos descrito sobre la memoria virtual.
Así como la memoria principal se comporta como un conjunto de ventanas y valijas en el área de intercambio/swap del disco, el cache externo actúa como una ventana sobre la memoria principal. El cache externo es rápido (250M de velocidad de acceso por segundo, en vez de 100M) y pequeño, El hardware (específicamente el controlador de la memoria de su computador) hace el LRU en el cache externo sobre los bloques de datos traídos desde la memoria principal. Por razones históricas la unidad de almacenamiento swapping es llamada 'una linea' en vez de 'una pagina'.
Pero aun no lo hemos logrado. EL cache interno da el ultimo paso de efectividad en velocidad al almacenar porciones de la cache externa. Esto aun es mas rápido y pequeño. EN efecto esto se sucede en el chip del procesador.
Si realmente queremos hacer que nuestros programas sean realmente rápidos, nos sera muy útil conocer estos detalles. Sus programas logran rapidez cuando tienen una fuertelocalidad, por que esto hace que el caching trabaje mejor. La mas fácil manera de hacer que los programas sean rápidos es, por lo tanto, hacerlos mas pequeños. Si un programa no funciono mas despacio al recibir grandes cantidades de I/O del disco o esperar eventos de red, él usualmente correrá a la velocidad del cache mas pequeño en el que él pueda ajustarse.
Si usted no puede hacer que la totalidad de su programa sea pequeño, enfoque los esfuerzos en ajustar la localidad de las porciones con problemas de velocidad. Detalles sobre las técnicas para hacer esto van mas allá del alcance de este tutorial: En el momento que usted las necesite, usted habrá profundizado lo suficiente con algún compilador para entender por si mismo muchas de las técnicas.

8.3. La unidad de administración de memoria

Aun cuando usted tenga suficiente memoria principal como para evitar el swapping, la parte del sistema operativo llamada 'administrador de memoria'/'memory manager' tiene aun importantes cosas que hacer. El tiene que asegurarse que los programas puedan solamente alterar sus propios segmentos de datos, esto es prevenir que erróneos o maliciosos códigos dentro de un programa dañen los datos de otro. Para lograr esto, él mantiene una tabla de datos y de segmentos de código. La tabla es actualizada siempre que un proceso cualquiera realice una petición de mas memoria o libere memoria (usualmente cuando el programa termina).
Esta tabla es usada para pasar ordenes a una especializada parte esencial del hardware llamada MMU o 'unidad de administración de memoria'. Los chips de procesadores modernos tienen MMUs internamente construidos. La MMU tiene la especial habilidad de colocar barreras alrededor de las áreas de la memoria, de tal forma que una referencia fuera de limite será rechazada y causara una situación de interrupción especial.
Si usted siempre ve un mensaje de Unix que dice “Segmentation fault”/”falla de segmentación”, “core dumped”/”memoria vaciada”, o algo similar, esto es lo que exactamente esta sucediendo: Un programa en ejecución ha hecho una solicitud para acceder a la memoria fuera de su segmento logrando así una interrupción fatal. Esto indica un error en el código del programa: el 'vaciado de memoria' deja tras de si su información de diagnostico para ayudar al programador a detectar el error.
aquí hay otro aspecto para proteger procesos de los otros además de limitar sus accesos de memoria. Usted también necesita ser capaz de controlar sus accesos a archivos para que un malicioso o problemático programa no corrompa piezas criticas del sistema. Esta es la razón del por que Unix tiene permisos de

archivos,tema que discutiremos mas adelante.

inicio

9. Que hace que mi computador guarde cosas en memoria?

Usted probablemente sabe que cualquier cosa en un computador es almacenada como una cadena de bits (dígitos binarios: usted puede pensar en ellos como un gran número de pequeños cambios de encendido y apagado). aquí vamos a explicar como estos bits son usados para representar las letras y números que su computador mastica.
Antes de comenzar con esto, se necesario entender el "tamaño de palabra"/"word size" de su computador. El tamaño de palabra es tamaño preferido de su computadorpara manipular unidades de información: Técnicamente es el ancho del registro de su procesador, área usada por el procesador para hacer cálculos lógicos y matemáticos. Cuando la gente escribesobre computadores que poseen tamañosen bits (llamándolos computadores de "32-bit" o "64-bit"),esto es lo que significa.
La mayoría de computadores (incluyendo computadores 386,486 y Pentium) tienen un ancho de palabra. Los anteriores modelos demaquinas 286 tenían un tamaño de palabra de 16: Las viejas mainframes a menudo usaban palabras de 36-bit. Los AMD opteron, Intel Itaniumy los Alpha desde que estaban en DEC y ahora tiene Compaq tienen 64-bit de palabra.
El computador ve su memoria como una secuencia de palabras numerada desde cero hasta algún alto valor dependiendo del tamaño de su memoria. Ese valor esta limitado por el tamaño de palabra. Esa es la razón de porque los programasen viejas maquinas comolas 286 tenían que funcionar mediante difíciles piruetas para dirigir grandes cantidades de memoria.

9.1. Números

Los números enteros son representados como cualquier otra palabra o pares de palabras, dependiendo de el tamaño de palabra de su procesador. Una maquina con 32-bit de palabra es la mas común representación de entero.
La aritmética del entero es cercana pero no es realmente matemática base 2. Los bit de bajo orden son 1, le sigue el 2, el 4 y así sucesivamente como un puro binario. Pero los números con signo son representados en la notación de "complemento a dos". Los bits de mas alto orden son un bit con signo, lo cual hace que la cantidad sea negativa y cada numero negativo puede obtener su correspondiente positivo valor al invertir todos los bits y sumando uno. Es por esto que los enteros en una maquina de 32-bit tienen el rango de -231 to 231 – 1. Ese 32 avo bit esta siendo usado para el signo: 0 significa un número positivo y 1 significa un numero negativo.
Algunos lenguajes de computación permitenusar la aritmética sin singo, lo que significa base 2 con cero y números positivos solamente.
La mayoría de los procesadores y algunos lenguajes pueden hacer operaciones con 'números de punto flotante'/'Flotating-point number' (esta capacidad esta preestablecida dentro delos chips de procesadores mas recientes). Los números de punto flotante le dan a usted más amplios rangos de valores quelos enteros y permiten ser expresados enfracciones. La forma en la cual esto es realizadovaria y sería muy complicada de discutir ahora en detalle, pero en ideas generales se refiere a la 'notación científica', donde uno podría escribir 1.234 * 1023; la codificación de elnumero es dividida en una mantisa (parte decimal de la fracción. n del t.) y la parte del exponente (23) para la potencia decimal(lo que significa que el número multiplicado resultante podría tener 20 ceros. 23 menos los tres lugares decimales.

9.2. Caracteres

Los caracteres son normalmente representados como cadenas de siete bits cada carácter en una codificación llamada ASCII (American Standard Code for Information Interchange / EstandardAmericano de Código para el Intercambio de Información): en las maquinas modernas, cada uno de los 128 caracteres ASCII son los últimos (los mas bajos) siete bits de un octeto o byte de8 bits: los octetos son empaquetados dentro de las palabras de memoria, así que (por ejemplo) una cadena de seis caracteres solo puede tomar dos palabras

de memoria. Para ver un listado con los códigos ASCII escriba 'man ascii' en su ventana de Unix.
El anterior párrafo estuvo equivocado en dos formas. La menor de ellas es que el termino 'octeto' es formalmente correcto pero raramente es verdaderamente usado: la mayoría de las personas se refieren a un octeto como un byte y espera que tengan 8 bits de longitud. Estrictamente hablando, el termino 'byte' es más general: es usado para, por ejemplo maquinas de 36-bits como 9-bit de bytes (aunque esto probablemente nunca sucederá de nuevo)
La mayor de ellas es que no esta permitido que todas las palabras usen ASCII. En efecto, muchas de las palabras no pueden ser ASCII. Esto puede no ser importante par el Ingles Americano, pero los acentos y también otros caracteres especiales son necesarios paralos usuarios que usan otros lenguajes. Inclusive el Ingles Británico tiene problemas con la ausencia del símbolo usado para el símbolo de la moneda Libra. han existido varios intentos de corregir este problema. Todo uso de bits de mayor tamaño que no esta en ASCII, haciéndolo en la mitad inferior del conjunto de 256 caracteres. El mayor uso de esta técnica es llamado 'conjunto de caracteres Latin-1 (mas familiarmente llamado ISO 8859-1). Este es el conjunto de caracteres por defecto para Linux, HTML y X. Microsoft Windows usa una versión cambiada de Latin-1 adicionándole caracteres como las dobles comillas derecha e izquierda en lugares donde el propio Latin-1 ha dejado sin asignar por históricas razones (para profundizar sobre los problemas que esto causa visite la pagina demoroniser)
Los soportes para lenguajes de Europa Occidental, incluyendo el Ingles, francés, Alemán, Español, Italiano, Alemán, Noruego, Suizo, y el Danes. Sin embargo esto no es suficiente, y como resultado ahora hay series del latin-2 por medio de -9 conjuntos de caracteres para apoyar caracteres del griego, Árabe, Hebreo, Esperanto y serbo-croacia. Para mayores detalles visite la pagina Sopa del alfabeto ISO .
La última solución es un gran estandard llamado Unicode (y su gemelo idéntico ISO/IEC 10646-1:1993). Unicode es idéntico al Latin-1 en los 256 casilla mas bajos. Sobre estos hay un espacio de 16 bits que incluye el Griego, Cirilico, Armenio, Hebreo, Arabe, Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayalam, Thai, Lao, Georgian, Tibetan, Kana japones, el completo conjunto de caracteres del Hangul Coreano y el unificado conjunto de caracteres para los ideogramas Chinos/Japoneses/Koreanos (CJK). Para mayores detalles visite la pagina de Unicode.

inicio

10. ¿Que hace mi computador cuando almacena cosas en el disco?

Cuando usted mira un disco duro bajo Unix, usted observa un árbol de directorios y archivos con nombre. Normalmente usted no necesita profundizar en su estructura, pero él se ha convertido en algo muy útil para saber que sucedió en detalle si usted tiene un disco dañado y necesita recuperar los archivos. Desafortunadamente, no es una buena forma describir la organización del disco desde el nivel inferior del mismo, así que lo describiremos desde el mismo hardware hacia arriba.

10.1. Nivel inferior del disco y el sistema de archivos

El área de superficie de su disco, donde él almacena la información, esta dividida en algo similar a un tablero de dardos – dentro de circulares pistas las cuales como las partes de un pastel. Debido a que las pistas cercanas a la parte exterior tienen mas área que aquellas cercanas al centro del disco, las pistas del exterior están divididas en mas sectores que las pistas del interior. Cada sector (o bloque de disco) tiene el mismo tamaño, el cual bajo los modernos Unix es generalmente 1 KB binario (1024 palabras de 8 bits). Cada bloque de disco tiene su única dirección o 'disk block number'/'numero de bloque del disco'.
Unix divide el disco dentro de 'particiones de disco'. Cada partición es un continuo campo de bloques que son usados separadamente de cualquier otra partición, cada uno como un sistema de archivos o espacio swap aparte. La razón original para tener particiones es la de recuperar los daños en un mundo de discos muy lentos y susceptibles a errores; los limites entre estas particiones reducen la fracción de su disco que probablemente lo haría inaccesible o corrupto debido a un mala lectura del disco. Hoy en día es más importante que las particiones puedan ser declaradas como 'solo-lectura' (previniendo a un intruso el modificar archivos críticos del sistema o compartirlas sobre una red de varias maneras que no discutiremos aquí. La partición con el número más bajo es a menudo tratada de manera especial como la partición de

arranque'/'boot partition' donde usted puede colocar el kernel para que sea iniciado.
Cada partición es implementada como espacio swap (usado para implementar la memoria virtual) o un sistema de archivos para almacenar archivos. Las particiones swap corresponden a una secuencia lineal de bloques. De otra mano, el sistema de archivos necesita una forma de mapear(relacionar) los nombres de los archivos a secuencias de bloques del disco. Debido a que los archivos crecen o disminuyen de tamaño y/o cambian con el tiempo, un bloque de datos de archivos no sería una secuencia lineal de estos sino mas bien todos estarían de manera dispersa a lo largo de su partición. (desde donde quiera que el sistema operativo pueda encontrar un bloque libre cuando lo necesite). Este efecto de dispersión se llama 'fragmentación'.

10.2. Nombres de archivos y directorios

Dentro de cada sistema de archivos, el proceso de mapeo de nombres a bloques es manipulado a través de una estructura llamada 'i-node'. Hay un conjunto de ellos cerca de la parte inferior (los números más bajos de bloque) de cada sistema de archivos (los más bajos de ellos son usados para proteger y etiquetar propósitos que no describiremos aquí). Cada I-node describe un solo archivo. Los bloques de datos de archivos (incluyendo directorios) se encuentran sobre los i-nodes (los números más altos del bloque)
Cada I-node contiene una lista de números de bloque del disco en el archivo que él describe (Actualmente esto no es del todo cierto, solo es correcto para archivos pequeños, pero el resto de los detalles no es importante conocerlos aquí).Note que cada i-node no contiene el nombre del archivo.
Los nombres de los archivos se encuentran en la 'estructura de directorios'. Una 'estructura de directorios' solo mapea los nombres hacia los números i-node. Esto sucede porque, en Unix, un archivo puede tener múltiples nombres y todos verdaderos (a esto se le llama Links Duros): Estos son solo múltiples entradas de directorios que apuntan al mismo i-node.

10.3. Puntos de montaje

En la más simple situación, su entrada en el sistema de archivos de Unix se encuentra en una sola partición. Esto es inusual aun cuando usted vea esto en algunos pequeños sistemas personales Unix. Es más típico que estén regados a lo largo de varias particiones, posiblemente en diferentes discos físicos. Así que, por ejemplo, su sistema puede tener una pequeña partición donde se encuentra el kernel, una partición ligeramente mayor donde se encuentran las utilidades del sistema operativo, y una partición mucho más grande donde se encuentran ubicados los archivos del usuario.
La única partición a la cual se tiene acceso inmediatamente después de arrancar el sistema se llama la 'partición raiz'/'root partition', la cual es (la mayoría de las veces) la única desde la cual usted puede arrancar. En ella se encuentra le directorio raíz del sistema de archivos, el nodo superior bajo cual cualquier otra cosa se encuentra.
Las otras particiones en el sistema tienen que ser adjuntadas a la partición root para que la totalidad del sistema de múltiples particiones pueda ser accesible. Él montará cada una de las particiones sobre un directorio en la partición root.
Por ejemplo, si usted tiene un directorio Unix llamado /usr, es probable que él sea un punto de montaje que apunta a una partición que contiene muchos programas que han sido instalados con el Unix pero que no son requeridos durante el proceso de arranque.

10.4. Como un archivo logra ser visto

Ahora podemos dar una mirada al sistema de archivos desde arriba. Cuando usted abre un archivo (por ejemplo: /home/esr/WWW/ldp/fundamentals.xml) esto es lo que sucede.
El kernel arranca en la raíz el sistema de archivos de Unix (en la partición raíz). Él busca un directorio llamado 'home'. Usualmente 'home' es un punto de montaje para la partición de usuario que esta ubicada en cualquier otro lugar, así que él (el kernel) va hacia ese lugar. En el nivel superior de la estructura del directorio de dicha partición de usuario el kernel busca una entada llamada 'esr' y extrae un número i-node. El kernel entonces se dirige a dicho i-node, advirtiendo que dicho i-node esta asociado a un bloque de datos de archivos correspondiente a una estructura de archivos. Aquí busca 'WWW', extrae el número i-node y se dirige al correspondiente subdirectorio, buscando ahora 'lpd'. Aquí hay otro i-node de directorio, abriéndolo él encuentra el numero i-node para 'fundamentals-xml'. Ese i-node no es un directorio, en lugar de ello dicho i-node contiene la lista de bloques del disco asociados con el archivo.

10.5. Propietario, permisos y seguridad de archivo

para mantener a los programas alejados de accidentes o malas situaciones al tomar datos que no debe usar, Unix tiene características de permisos. Estas características fueron originalmente diseñadas para soportar el timesharing (compartición del tiempo) alproteger en la misma maquina a múltiples usuarios de entre si. en la época en la que Unix funcionaba principalmente en costosos minicomputadores compartidos.
Para entender el concepto de archivos depermisos es necesario recordar la descripción de usuarios y grupos en la sección ¿Que pasa cuando usted se registra en el sistema?. Cada archivo tiene un usuario propietario y un grupo propietario. Inicialmente estos permisos son los del creador del archivo: ellos pueden ser cambiados con los programas chown(1) y chgrp(1).
Los permisos básicos que pueden ser asociados con un archivo son 'lectura'/'read' (permiso para leer sus datos), 'escribir/'write' (permisos para modificar lo) y 'ejecutar'/'execute' (permisos para correr como un programa). Cada archivo tiene tres conjuntos de permisos: uno para su usuario propietario, uno para su grupo propietario, y uno para los demás. Los 'privilegios' que usted logra cuando se registra en el sistema son solo la capacidad de lectura, escritura y ejecución de aquellos archivos para los cuales los bits de permisos están asignados a su 'identificador de usuario'/'user ID' o a uno de los grupos a los que pertenece, o archivos que han sido adecuados como accesibles para todo el mundo.
Para observar como estos permisos interactuan y comoUnix los despliega vamos a mirar una lista de archivos en un hipotético sistema Unix.

```
snark:~$ ls -l notes
-rw-r--r-- 1 esr users 2993 Jun 17 11:00 notes
```

Este es un archivo ordinario de datos. La lista nos indica que él pertenece al usuario 'esr' y fue creado dentro de su grupo propietario 'users'. Probablemente la maquina en la que estamos ubico por defecto en ese grupo a cada usuario ordinario: otros grupos de usuario que comúnmente usted podrá ver son 'staff', 'admin', 'wheel' (por obvias razones los grupos no son muy importantes en computadores personales o estaciones de trabajo unipersonales). Su sistema Unix puede usar un grupo por defecto distinto. Quizás alguno nombrado después de su identificador de Usuario.
La cadena '-rw-r—r--' representa los bits de permisos para el archivo. El primer bit es la posición para el bit del directorio: él puede mostrar 'd' si el archivo es un directorio, o mostrar 'l' si el archivo es un link simbólico. Los tres siguientes bit son los permisos del usuario, los tres contiguos son los permisos de grupo, y los tres últimos son los permisos para los demás (a menudo llamados 'permisos para todo el mundo'). En este archivo el usuario propietario 'esr' puede leer o escribir el archivo, otras personas en el grupo 'users' pueden leer el archivo y cualquier otra persona en el mundo puede leerlo. Estos son los típicos conjuntos de permisos para un archivo ordinario de datos.
Ahora vamos a mirar un archivo con unos permisos muy diferentes. Este archivo es el GCC (el compilador GNU del lenguaje de programación C)

```
snark:~$ ls -l /usr/bin/gcc
-rwxr-xr-x 3 root bin 64796 Mar 21 16:41 /usr/bin/gcc
```

Este archivo pertenece al usuario llamado 'root' y a un grupo llamado 'bin': este archivo puedeser escrito (modificado) solo por el usuario 'root',pero leído o ejecutado por cualquier otro. Esta es una típica titularidad y configuración de permisos para un comando pre-instalado en el sistema. El grupo 'bin' existe en algunos Unix para agrupar juntos a los comandos del sistema (el nombre es una reliquia histórica, es la abreviatura para 'binario'). En vez de el grupo 'usr' el sistema Unix podría usar el grupo 'root' (no es exactamente lo mismo que el usuario 'root').
El usuario 'root' es el nombre convencional para el usuario con el numero de identificación 0, que es una especial y privilegiada cuenta que puede sobrescribir todos los privilegios. El acceso a la raíz es útil pero peligrosa: un error al teclear mientras se esta como 'root' adentro del sistema puede alterar archivos críticos del sistema que un usuario ordinario no podría tocar usando el mismo comando.
Debido a que la cuenta root es muy poderosa, el acceso a ella debe ser protegido cuidadosamente. La clave del usuario 'root' es la pieza más crítica de seguridad en su sistema y es lo que muchos crackers e intrusos siempre desean conseguir.
Sobre claves: No la deje escrita y no escoja claves que fácilmente puedan ser adivinadas, como el primer nombre de su novio/a esposo/a. Esta es una mala practica muy común que ayuda a los crackers a no parar. En general no escoja ninguna palabra del diccionario: hay programas llamados 'diccionarios de crackers' que funcionan buscando claves probables a través de listas de comunes elecciones. Una buena técnica e

escoger una combinación consistente en una palabra, un dígito, y otra palabra, como por ejemplo 'shark6cider' o 'jump3joy': esto hará que el tiempo necesario de búsqueda para un diccionario de búsqueda sea muy largo. No use estos ejemplos, se puede esperar que los crackers después de leer este documento incorporen estos ejemplos en sus diccionarios.
Ahora veamos un tercer caso:

```
snark:~$ ls -ld ~
drwxr-xr-x 89 esr users 9216 Jun 27 11:29 /home2/esr
snark:~$
```

Este archivo es un directorio (note el 'd' en el primer espacio de permisos). Observamos que el archivo puede ser escrito por esr pero leído y ejecutado por cualquier otro.
El permiso de lectura sobre un directorio le da la habilidad de listar el directorio – esto es mirar los nombres de los archivos y directorios que él contiene. El permiso de escritura da la habilidad de crear y borrar archivos en el directorio. Estas reglas tienen sentido si usted recuerda que el directorio incluye una lista de nombres de los archivos y subdirectorios que contiene.
Los permisos de ejecución en un directorio significa que es posible entrar en el directorio para abrir los archivos y los directorios contenidos en él. En efecto, él le da permisos para acceder a los i-nodes en el directorio. Un directorio sin ningún permiso de ejecución puede ser inútil.
Ocasionalmente usted podría ver un directorio que es ejecutable para los usuarios 'world' pero no es legible para estos mismos usuarios: Esto significa que un usuario aleatorio pude acceder a los archivos y directorios inferiores pero solo conociendo sus nombres exactos (el directorio no puede ser listado).
Es importante recordar que leer, escribir o ejecutar permisos en un directorio es independiente de los permisos en los archivos y directorios que contiene. En particular, la posibilidad de escritura en un directorio no da automáticamente acceso a los archivos existentes.
Finalmente vamos a ver los permisos del programa login.

```
snark:~$ ls -l /bin/login
-rwsr-xr-x 1 root bin 20164 Apr 17 12:57 /bin/login
```

Este programa tiene los permisos que podríamos esperar para un comando de sistema – excepto por la 's' donde el permiso de ejecución para el propietario debería estar. Esta es la manifestación visible de un permiso especial llamado el 'set-user-id' o 'setuid bit'
El bit setuid es normalmente adjuntado a programas que necesitan dar privilegios de root a usuarios ordinarios, pero de una forma controlada. Cuando este permiso es definido en un programa ejecutable usted logra los privilegios de el propietario de el archivo del programa mientras el programa corra por parte suya, sean o no de su propiedad.
Como la cuenta root misma, los programas setuid son útiles pero peligrosos. Cualquiera quien pueda subvertir o modificar un programa setuid propiedad del root puede usarlo para generar una ventana de sesión con privilegios de usuario. Por esta razón abrir un archivo en la mayoría de los Unix para escribir hace que dicho archivo desactive su bit setuid. Muchos ataques a la seguridad de Unix tratan de explotar bugs en los programas setuid buscando aprovecharse de ellos. Por eso administradores de seguridad de sistemas están consientes de estos programas tomando cuidadosas medidas y reacios a instalar nuevos.
Debemos resaltar un par de importantes detalles de lo que hemos visto ahora: A saber, como el grupo propietario y sus permisos son asignados cuando un archivo o directorio es creado por primera vez. El tema de grupos es de cuidado porque los usuarios pueden ser miembros de múltiples grupos, pero uno de ellos (especificado en la entrada del usuario en /etc/passwd) es el grupo por defecto para el usuario y normalmente será el propietario de los archivos creados por el usuario.
La historia detrás de los bits de permisos con los que inicia un archivo es un poco mas complicada. Un programa que crea un archivo normalmente especificará los permisos con los que dicho archivo comenzará. Pero estos permisos serán modificados por una variable de entorno del usuario llamada el 'umask'. El umask especifica cuales bit de permisos serán desactivados cuando es creado un archivo: El valor más común, y que esta establecido por defecto en la mayoría de sistemas es -------w- or 002, el cual desactiva el bit de la escritura para todo el mundo. Para mayores detalles mire la documentación del comando umask en su ventana de sesión.
El grupo inicial para un directorio es también un poco mas complicado. En algunos sistemas Unix un nuevo directorio obtiene el grupo por defecto del usuario creador (esta es una convención del sistema V): en otros, él obtiene el grupo propietario del directorio padre en el cual ha sido creado (esta es la convención BSD). En algunos modernos Unix, incluyendo Linux, la conducta definitiva puede ser seleccionada al configurar

el set-group-ID en el directorio (chmod g+s).

10.6. Como las cosas pueden andar mal

Anteriormente se aludía que los sistemas de archivos eran frágiles. Ahora sabemos que para obtener un archivo usted tiene que saltar a través de lo que podría ser una arbitraria y larga cadena de directorios y referencias de i-node. Ahora supongamos que su disco duro produce un mal salto.
Si usted tiene suerte, solo desechará algunos datos de archivo. Si no tiene suerte, se podría corromper una estructura de directorio o un numero i-node y dejar en el limbo a la totalidad del árbol de un subdirectorio, o peor aún, resultar en una estructura corrompida que apunta múltiples rutas a un mismo bloque de disco o i-node. Tal daño puede ser propagado como una normal operación de archivo, desechando datos que no estaban en el lugar donde inicio el salto equivocado.
Afortunadamente, este tipo de contingencias se han convertido en algo poco comúnasí como el hardware de los discos se han vuelto más confiables. Aun más, esto significa que Unix buscará revisar periódicamente la integridad del sistema de archivos para asegurarse que todo funcione correctamente. Los Unix modernos realizan rápidamente dicha revisión de integridad en cada partición en el momento del arranque, justo antes de montarlas. Cada pocos re inicios sucederá la revisión de integridad de manera mas detallada, tomándose unos pocos minutos de duración.
Si todo esto suena a que Unix es terriblemente complejo y susceptible de fallas, entonces puede sonar tranquilizador saber que estas revisiones en el momento del arranque comúnmente detectan y corrigen problemas normales antes de que ellos se conviertan en problemas realmente desastrosos. Otros sistemas operativos no tienen estas facilidades, lo cualaligera un poco el tiempo de arranque pero puede dejarlo a usted mas seriamente afectado cuando intente recuperarlo a mano (inclusive asumiendo que usted tiene una copia de las utilidades de Norton o cualquier otra cosa que este a primera mano)
Una de las tendencias en los diseños de los actuales Unix es el 'registro diario del sistema de archivos'/'journalling file systems'. Este sistema organiza el tráfico para el disco garantizando que el disco se encuentre en un consistente estado al cual pueda regresar. Esto hace que la revisión detallada sea mas rápida en el momento del arranque.

inicio

11. ¿Como trabajan los lenguajes de programación?

Ya hemos discutido sobre como los programas son corridos. En ultimas cada programa tiene que ser ejecutado como un flujo de datos que son instrucciones en el lenguaje maquina de su computador. Pero los seres humanos no manejan muy bien el lenguaje demaquina: hacer esto se ha convertido en un raro y oscuro arte aun entre hackers.
hoy en día casi todo el código de Unix esta escrito en un 'lenguaje de alto nivel'/'high-level language' excepto una pequeña cantidad de interfaces directas de hardware soportadas en el kernel mismo (el termino 'alto nivel' es un viejo termino usado paradistinguir los 'lenguajes ensamblador'/'assemblerlanguajes' de 'bajo nivel'/"low-level, los cuales son básicamentepequeños envoltorios alrededor del código de maquina.
Existen varios diferentes tipos de lenguajes de alto nivel. Para hablar acerca de ellos es útil tener en mente que el 'código fuente'/'source code' de un programa (la versión editable creada por un ser humano) tiene que ir a través de algún tipo de traducción al código maquina que la maquina pueda realmente correr.

11.1. Lenguajes Compilados

La mayoría de los tipos de lenguaje convencionales son lenguajes compilados. Los lenguajes compilados son traducidos a archivos ejecutables de maquina (en código binario) por un programa especial llamado (lógicamente) el compilador. Una vez ha sido generado el binario, usted puede hacerlocorrer directamente sin tener que mirar de nuevo el código fuente (la mayoría del software es entregado en binarios compilados hechos desde código que usted no ve).
Los lenguajes compilados tienden a dar un excelente desempeño y a tener un mas completo acceso al sistema Operativo, pero también hay dificultad en programar en él.

C, el lenguaje en el que Unix esta escrito es delejos es el mas importante de ellos (con su variante C++). FORTRAN es otro lenguaje compilado queaún es usado entre ingenieros y científicos pero con años de haber sido creado y mucho mas primitivo. En el mundo Unix estos son los lenguajes compilados que principalmente se usan: Fuera de ellos, COBOL es ampliamente usado para software de finanzas y negocios.
Antes eran usados muchos otros lenguajes compilados, pero la mayoría de ellos se han extinto o son estrictamente usados en herramientas de investigación. Si usted es un nuevo desarrollador de Unix usando un lenguaje compilado, estaráprobablemente usando C o C++.

11.2. Lenguajes Interpretados

Un lenguaje interpretado depende de un programa interprete que lea el código fuente y lo traduzca inmediatamente en cómputos y llamadas del sistema. El código fuente debe ser reinterpretado (y el interprete debe esta presente) cada vez que el código es ejecutado.
Los lenguajes interpretados tienden a ser mas lentos que los lenguajes compilados, y a menudo tienen limitado acceso al hardware y a partes importantes del sistema operativo. De otra parte estos lenguajes tienden a ser fáciles para programar y menos estrictos que los lenguajes compilados
Muchas utilidades de Unix, inlcuyendo la shell, bc, sed y awk son de hecho pequeños interpretes de lenguajes. BASICs es usualmente interpretado, de igual manera Tcl. Históricamente, el más importante lenguaje interpretado ha sido LISP (con mayores mejoras sobre la mayoría de sus sucesores). Hoy, las shells de Unix y el Lisp que se encuentra dentro del editor Emacsson probablemente los mas importantes lenguajes interpretados puros.

11.3. Lenguajes Pseudocódigo

Desde 1990 ha tomado creciente importancia un tipo de lenguaje híbrido que usa la compilación y la interpretación. los lenguajes P-code son como los lenguajes compilados por que la fuente es traducida a un binario compacto que es lo que usted realmente ejecuta, pero no es un código maquina. En lugar de ellos es un pseudocódigo (o código-p) que usualmente es bastante mas simple pero mas poderoso que un lenguaje real de máquina. Cuando usted corre el programa se interpreta el p-código.
El pseudocódigo puede correr casi tan rápido como un binario compilado (los interpretes de seudocódigo pueden ser simples sencillos y rápidos). Los lenguajes seudocódigo pueden mantener la flexibilidad y poder de un lenguaje interpretado.
Importantes lenguajes seudocódigo son Python, Perl y Java.

inicio

12. Que hace que Internet trabaje?

Para ayudar a entender como Internet trabaja vamos a mirar las cosas que suceden cuando usted hace una tarea normal en Internet – Llegar con el navegador hasta la página de inicio de este documento en el sitio web del Proyecto de documentación de Linux (Linux Documentation Project - LDP). Este documento es:

http://www.tldp.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO/index.html

Lo que significa que el se encuentra en el archivo HOWTO/Unix-and-Internet-Fundamentals-HOWTO/index.html accesible en la telaraña mundial (World Wide Web – WWW) en el host www.tldp.org

12.1. Nombres y Ubicaciones

La primer cosa que su navegador tiene que hacer es establecer una conexión con la maquina donde el documento se encuentra (el host. n. del t.), Para hacer esto el primero debe buscar en la red la ubicación de el host www.tldp.org ('host' se usa para abreviar 'maquina anfitriona'/'host machine' o 'red anfitriona'/'network host': www.tlpd.org es un común nombre de host). Su ubicación corresponde realmente

a un número llamado 'dirección IP'/'IP adress' (explicaremos que significa IP más adelante).
Para hacer esto el navegador interroga a un programa llamado 'Servidor de nombres'/'name server'. El servidor de nombres puede encontrarse en su maquina, pero es mas probable que funcione en una maquina de servicios a la cual el navegador establece la conexión. Cuando usted ingresa con un ISP (Proveedor de servicios de internet / Internet Service Provider . ISP. n del t.), parte del procedimiento de configuración la mayoría de las veces involucrará indicarle al navegador la dirección IP del servidor de nombres en la red.
Los servidores de nombres ubicados en diferentes maquinas hablan entre si, intercambiando información y manteniendo actualizada toda la información que necesitan para resolver los nombres de host (relacionarlos con las direcciones IP). El servidor de nombres que usted usa, en el proceso de resolver la ubicación de www.tldp.org puede consultar a tres o cuatro diferentes sitios a través de la red, pero esto usualmente sucede muy rápido (en menos de un segundo). Veremos en detalle a los servidores de nombres en la próxima sección.
El servidor de nombres le dirá a su navegador que la dirección IP del host www.tldp.org es 152.19.254.81: conociendo esto el navegador nuestra maquina será capaz de intercambiar bits directamente con www.tldp.org.

12.2. El Sistema de Nombres de Dominio

La totalidad de los programas y bases de datos de red que cooperan en resolver los nombres de host a direcciones IP son llamados 'DNS' (Sistema de Nombres de Dominio). Cuando usted ve referencias a un 'servidor DNS'/'DNS Server' esto significa que estamos hablando de un servidor de dominios. Ahora vamos a ver como funciona todo esto.
En internet los nombres de host están compuestos de partes separadas por puntos. Un 'dominio'/'domain' es una colección de maquinas que comparten un sufijo como nombre en común. Los Dominios pueden estar dentro de otros dominios. Por ejemplo, la maquina www.tldp.org se encuentra en la maquina .tldp.org subdomino del dominio .org.
Cada dominio esta definido por un servidor de nombres autorizado que conoce la dirección IP de las otras maquinas en el dominio. El servidor autorizado (o primario) de nombres puede tener backups en caso de alguna falla: si usted ve referencias a un 'servidor de nombres secundario'/'sencondary name server' (o 'secundario DNS'/'secondary DNS') es que se esta hablando de uno de esos bakups. Estos servidores secundarios a menudo refrescan su información desde sus servidores primarios cada pocas horas, así, un cambio hecho en el servidor primario a la relación nombre de host-IP será automáticamente propagado.
Ahora, aquí esta la parte importante. Los servidores de nombres de un dominio no tienen que saber la ubicación de todas las maquinas en otros dominios (incluyendo sus propios subdominios): él solo tiene que saber la ubicación de los servidores de nombres. En nuestro ejemplo, el servidor de nombres autorizado para administrar el dominio .org sabe la dirección IP de el servidor de nombres para .tlpd.org pero no la dirección de todas las demás maquinas en el dominio .tlpd.org.
Los dominios en el sistema DNS están organizados como un gran árbol invertido. En la cima se encuentran los 'servidores raíz'/'root servers' (administran los dominios .org .com. net, etc. n del t): ellos están conectados a su software DNS. El servidor raíz sabe las direcciones IP de los servidores de dominio para los 'dominios de nivel superior'/'top-levels domains' (tlpd.org, example.org, etc. n. del t.). Cada servidor de dominios de nivel superior conoce donde se encuentran los servidores de dominios directamente bajo ellos, y así sucesivamente.
DNS esta cuidadosamente diseñada para que cada maquina pueda funcionar con independencia con la mínima cantidad de conocimientos que necesite sobre la forma del árbol y para que los cambios locales a los arboles inferiores puedan ser hechos de manera simple al cambiar en la base de datos de un servidor de nombres autorizado las relaciones 'nombre de host'-'dirección IP'.
Cuando su navegador hace la consulta sobre la dirección IP de www.tlpd.org lo que sucede realmente es: primero su servidor de nombres (quizás el servidor del ISP de su cuenta de acceso a Internet. n. del t.) pregunta al servidor raíz para saber donde puede ubicar los servidores de nombres para los dominios .org. Una vez conoce esto, él entonces pregunta al servidor de nombres del dominio .org para saber donde la dirección IP de el servidor de nombres de .tldp.org. Una vez se ha hecho esto él pregunta al servidor de nombres .tldp.org para saber la dirección del host www.tldp.org.
La mayoría de las veces, su servidor de nombres no tiene que trabajar tan duro. Los servidores de nombres realizan mucho cache: cuando su servidor resuelve un nombre de host, el mantiene en memoria por un momento la asociación con la dirección IP resultante. Esto sucede por que, cuando usted navega un nuevo sitio web usualmente usted ve un mensaje de su navegador que indica 'buscando'/'looking up' el host para la primera página que usted encontró. Eventualmente el proceso de mapeo nombre-a-dirección expira y su

DNS tiene que solicitar nuevamente este proceso – esto es importante para que no se conserve la antigua (e invalida) información cuando un nombre de host cambie de dirección. La dirección IP cacheada de un sitio web no será desplegada si el host es inalcanzable.

12.3. Paquetes y enrutadores

Lo que el navegador desea hacer es enviar un comando al servidor web en www.tldp.org se parece a esto.

```
GET /LDP/HOWTO/Fundamentals.html HTTP/1.0
```

Esto es lo que sucede. El comando es preparado dentro de un 'paquete'/'packet', un bloque de bits como un telegrama que es empaquetado con tres importantes cosas: La dirección origen (la dirección de su maquina), la dirección destino (152.19.254.81), y un número de servicio o número de puerto (80 en este caso) que indica que es una solicitud hecha para la telaraña mundial World Wide Web.
Su maquina entonces envía el paquete por medio del cable (la conexión de su ISP o red local) hasta que llega a una maquina especializada llamada 'enrutador'/'router'. El enrutador tiene un mapa de Internet en su memoria – no siempre completo, pero describe por completo el vecindario de su red y sabe como conseguir los enrutadores de otros vecindarios en la Internet.
Su paquete puede pasar a través de varios enrutadores en la dirección de su destino. Los enrutadores son inteligentes. Ellos observan cuanto tardan otros enrutadores en avisar que han recibido un paquete. Ellos también usan esa información para dirigir el tráfico sobre conexiones rápidas. Los enrutadores usan estas conexiones para advertir cuando otro enrutador (o un cable) ha caído en la red y compensar dicha caída encontrando otro enrutador.
Hay una leyenda urbana que dice que la Internet fue diseñada para sobrevivir a una guerra nuclear. Esto no es cierto, pero el diseño de la Internet es extremadamente bueno en conseguir un desempeño confiable sobre hardware problemáticos y condiciones inciertas. Esto es principalmente debido al hecho de que su inteligencia esta distribuida a través de miles de enrutadores en lugar de estar concentrada en unos pocos gigantes interruptores (como en la red telefónica). Esto significa que las fallas tienden a ser bien localizadas y la red puede pasar al rededor de estas fallas.
Una vez que su paquete logra llegar a la maquina destino, esta maquina usa el numero de servicio para introducir el paquete al servidor web. El servidor web puede saber a donde responder al revisar en el paquete la orden con la dirección IP origen. Cuando el servidor web devuelve este documento (el que estas leyendo. n del t.), este documento sera dividido en paquetes que han sido numerados. El tamaño de los paquetes variará según el medio de transmisión en la red y el tipo de servicio.

12.4. TCP e IP

Para entender como sucede la transmisión de múltiples paquetes, usted necesita saber que la Internet actualmente usa dos protocolos (maneras de negociar e intercambiar la información n.del t.), organizados uno sobre otro.
En el nivel inferior, el IP ('Protocolo de Internet' /'Internet Protocol'), es el responsable de etiquetar individualmente los paquetes con las direcciones fuente y destino de dos computadores que intercambian información sobre una red. Por ejemplo, cuando usted accede a http://www.tldp.org el paquete que usted envió tiene la dirección IP de su computador, como 192.168.0.101 y la dirección 152.19.254.81 del computador de www.tldp.org. Estas direcciones trabajan de la misma manera como la dirección de su casa trabaja cuando alguien envía a usted una carta. La oficina postal puede leer la dirección y determinar donde usted se encuentra y la mejor ruta para hacerle llegar su carta, como lo hace un enrutador con el trafico de Internet.
En el nivel superior, TCP (Protocolo de Control de Transmisión'/'Transmission Control Protocol') da la confiabilidad. Cuando dos maquinas negocian una conexión TCP (la cual hacen usando el protocdo IP), el receptor sabe que debe enviar acuses de recibo de el paquete que recibirá del remitente. Si el remitente no ve un acuse de recibo del paquete en un tiempo de espera determinado, él reenvía el paquete.
Adicionalmente, el remitente da a cada paquete TCP una secuencia de números, la cual el receptor puede usar para rearmar los paquetes en caso de que ellos lleguen en desorden (esto fácilmente puede suceder si la velocidad de red varía durante la conexión).
Los paquetes IP también contienen un checksum (procedimiento de comprobacióno suma de comprobación. n. del t.) para facilitar la detección de datos corruptos debido a problemas en la red (El checksum es calculado de el resto del paquete para saber si el resto del paqueteo el checksum están

corruptos; rehaciendo el calculo y comparando el checksum recibido es muy probable reconocer un error). Así, desde el punto de vista de cualquiera que usa TPC/IP y los servidores de nombre, esta parece una confiable manera de enviar flujos de bytes usando parejas de nombre-de-host/numero-de-servicio. Las personas que escriben protocolos de red casi nunca tienen que pensar sobre la definición del tamaño del paquete, como se rearma el paquete, la revisión de errores, el procedimiento checksum y la retransmisión que sucede bajo su nivel.

12.5. HTTP, un protocolo de aplicación

Ahora vamos a regresar a nuestro ejemplo. Los navegadores y servidores web hablan un 'protocolo de aplicación'/'application protocol' que funciona encima de TCP/IP, usándolo simplemente como una forma de enviar y recibir flujos de bytes. Este protocolo se llama HTTP ('Protocolo de Transferencia de Hipertexto'/'Hypertext Transfer Protocol') y ya hemos visto un comando de él – el GET mostrado anteriormente.
Cuando el comando GET interactua con es servidor web de www.tldp.org con el servicio número 80, él será despachado a un 'servidor demonio'/'server daemon' que escucha en el puerto 80. La mayoría de los servicios de Internet son implementados por servidores demonio que no hacen más que escuchar en puertos, esperando por la llegada de un comando de ejecución.
Si el diseño de la Internet tiene una regla por sobre todas, esta es que todas las partes deben ser tan simples y accesibles a seres humanos como sea posible. HTTP y sus parientes (como el Simple Mail Transfer Protocol SMTP, que es usado para mover electrónica mente mails entre host) tienden a usar simples comandos de texto de impresión que terminan con un a entrada de retorno de linea.
Esto es ligeramente ineficiente: en algunas circunstancias usted puede lograr más velocidad al usar un protocolo de código estrictamente binario. Pero la experiencia ha mostrado que los beneficios de tener comandos fáciles de describir y entender por seres humanos sobrepasa cualquier beneficio en eficiencia que usted puede lograr con el costo de hacer las cosas difíciles y opacas.
De esta manera lo que el servidor demonio envía de regreso a usted vía TCP/IP también es texto. El comienzo de la respuesta se parecerá a algo como esto:

```
HTTP/1.1 200 OK
Date: Sat, 10 Oct 1998 18:43:35 GMT
Server: Apache/1.2.6 Red Hat
Last-Modified: Thu, 27 Aug 1998 17:55:15 GMT
Content-Length: 2982
Content-Type: text/html

```

Estas cabeceras será seguida por una línea en blanco y el texto de la página web (después de lo cual la conexión es terminada). Su navegador solo desplegará la pagina. Las cabeceras le indicarán al navegador como (en particular, la cabecera Content-Type indica que el dato regresado es realmente html.

inicio

13. Para aprender más

Esta es una lista de lecturas de HOWTO que contiene libros que usted puede leer para aprender más sobre los tópicos que hemos tocado aquí. Quizás también quiera leer el documento Como convertirse en un Hacker.