PRÁCTICA 1: PREPARACIÓN DEL ENTORNO

Presentación

Estas prácticas pretenden que el estudiante sea capaz de entender porciones de código fuente de una versión actual del kernel de Linux y de introducir pequeñas modificaciones a las funcionalidades del kernel.

Las prácticas de la asignatura son dos:

• La primera práctica describe cómo preparar el entorno de desarrollo utilizado en las prácticas (Compilación del kernel y el arranque de una máquina utilizando el kernel generado) y como navegar dentro de la estructura de directorios que componen el código fuente del kernel de Linux. También presenta el mecanismo de los módulos perqué será utilizado en la segunda práctica para introducir nuevas funcionalidades al código del kernel.

• La segunda práctica propone realizar una serie de modificaciones bastante localizadas en el código del kernel de Linux. Por ejemplo, recorrer la estructura de tablas de páginas de un proceso, crear un driver para un nuevo dispositivo u obtener información sobre el sistema de archivos.

Los conocimientos previos necesarios para el desarrollo de estas prácticas son:

• Algorítmica.
• Programación en un lenguaje de alto nivel (preferentemente en lenguaje C). Uso de punteros.
• Estructuras de datos (listas doblemente encadenadas, tablas hash).
• Conceptos teóricos de la asignatura Sistemas Operativos.
• Utilización de Linux desde el intérprete de pedidos.
• Entender pequeños fragmentos de código escritos en ensamblador i386.

Este documento contiene el enunciado de la primera práctica. La primera práctica de la asignatura presenta el entorno de desarrollo que permitirá estudiar y modificar el código fuente del kernel de Linux (concretamente, el de la versión 4.7.5). También describe el mecanismo de los módulos (módulos) porqué será empleado en la segunda práctica para introducir modificaciones al código fuente del kernel de Linux. Finalmente, propone que el alumno estudie algunas porciones del código del kernel.

En las prácticas, la máquina del estudiante (máquina host, sea física o virtual) no arrancará directamente con los kernels generados sino que será un emulador de PC (máquina guest, ejecutado sobre la máquina host) quien arrancará con estos kernels. Los motivos para trabajar de esta forma son los siguientes:

• Posibilitar que los estudiantes utilicen un entorno lo más sencillo y similar posible.
• Evitar que sea necesario tener que reiniciar la máquina host después de algunas pruebas.
• Garantizar que los posibles errores presentes en el código escrito por los estudiantes no afecten ni la estabilidad ni la coherencia de los datos de la máquina host.

De todas formas, también sería posible hacer que la máquina host arrancara directamente con los kernels generados. Para ello, habría que modificar la configuración del gestor de arranque instalado en la máquina host.

Competencias

Transversales:
• Capacidad para adaptarse a las tecnologías ya los futuros entornos actualizando las competencias profesionales
• Capacidad para la comunicación escrita en el ámbito académico y profesional
Especıficas:
• Capacidad para analizar un problema en el nivel de abstracción adecuado a cada situación y aplicar las habilidades y conocimientos adquiridos para abordarlo y resolverlo
• Capacidad para diseñar y construir aplicaciones informáticas mediante técnicas de desarrollo, integración y reutilización

Objetivos

Los objetivos de esta práctica son que el estudiante ...

• ponga en funcionamiento el entorno de desarrollo de las practicas (instale el software requerido, copiar los archivos necesarios para desarrollar la práctica, los des compactar, compile el kernel y compruebe que la máquina guest arranca correctamente con el kernel generado).
• conozca los principales directorios en qué estructura el código fuente del kernel de Linux.
• Saber utilizar alguna herramienta que permite navegar dentro del código fuente del kernel de Linux.
• entienda la utilidad del mecanismo de los módulos.
• instale los módulos de ejemplo a la máquina guest.
• modifique uno de los módulos de ejemplo.
Para evaluar la consecución de los objetivos, los estudiantes deberán contestar individualmente una serie de cuestiones formuladas al final de este documento (apartado 4).

Descripción de la Práctica

1 Preparación del entorno de desarrollo de las prácticas

1.1 Requisitos

En primer término, el estudiante debe comprobar que su hardware cumpla los siguientes requisitos:

• El procesador de la máquina host debe ser compatible x86.
• Hay que tener instalada alguna distribución Linux a la máquina host, preferentemente la facilitada por la UOC este curso (la Ubuntu 14.04 de 32 bits).
• La máquina host debe disponer de suficiente espacio libre en disco (unos 2 Gigabytes).
• El estudiante debe poder descargar unos 400 Megabytes para preparar el entorno de trabajo.

1.2 Pasos a seguir

1. En la máquina host, crear un directorio donde se guardarán todos los archivos de la práctica. Entrar.
2. Descargar de la red los siguientes archivos:
    (a) linux-4.7.5.tar.xz de [1] (Ocupa unos 90 Megabytes): contiene el código fuente de la versión 4.7.5 del kernel de Linux.
    (b) image.img de [2] (Ocupa unos 300 Megabytes): contiene la imagen de un sistema de ficheros en formato ext2; este ser`a el sistema de ficheros accedido por la máquina guest.
    (c) base.zip de [3] (Ocupa unos 2 Megabytes): contiene una serie de ficheros de ejemplo y de shellscripts.
3. Des compactar el archivo base.zip usando el comando unzip base.zip
4. Des compactar el archivo linux-4.7.5.tar.xz . Esto generará una estructura de directorios a partir del directorio linux-4.7.5 que ocupará unos 740 Megabytes.
5. Compilar el kernel. Para ello se seguirán los siguientes pasos:
    (a) Situarse en el directorio con el código fuente del kernel de Linux (pedido cd linux-4.7.5).
    (b) Definir la configuración del kernel. Para unificar la configuración de todos los estudiantes y eliminar aquellas partes del kernel no relevantes a estas prácticas, el fichero base.zip contiene una de estándar (fichero .config) que hay que copiar en el directorio actual (cp ../.config.).
    (c) Ejecutar el pedido make. Este comando compila los archivos fuente del kernel y genera la imagen comprimida del kernel. El archivo generado es arch / x86 / boot / bzImage. Como referencia, a un ordenador con un procesador i7 a 2.9 GHz, y haciendo que la máquina host también sea una máquina virtual, el tiempo de compilación es inferior a diez minutos.
6. El estudiante deberà de utilizar el emulador qemu [4] Para emular la máquina guest. En caso de trabajar sobre Ubuntu, puede instalar a ejecutando el comando sudo apt-get install qemu-kvm.

7. Hacer que la máquina guest arranque utilizando como kernel lo que acaba de ser compilado y como sistema de ficheros la imagen que ha sido descargada de la red. Se hará servir un shellscript boot.sh. Una vez arranque , hay que identificarse como usuario root; no es necesario introducir password. Por pasar a trabajar sobre una ventana de la máquina host pulse Ctrl Alt simultáneamente.
8. Dentro de la máquina guest, escribir el comando halt para proceder a detenerla. Cuando aparezca el mensaje System halted, es seguro eliminar la ventana creada por qemu.

Observaciones:

• Arrancar la máquina guest es requisito indispensable para poder realizar las prácticas.
• Hay que diferenciar qué comandos a ejecutar en la máquina host y cuáles a la máquina guest.

2 Navegación por el código fuente del kernel de Linux

2.1 Directorios principales

El kernel de Linux está compuesto por miles de archivos fuente (por ejemplo, la versión 4.7.5 está formada por 23.100 ficheros con extensión .cy 18.312 ficheros con extensión .h organizados en 4.107 directorios). A continuación se enumeran los directorios principales de la estructura de directorios que almacena el código fuente del kernel de Linux y qué tipo de código contienen.

• arch: archivos dependientes de la arquitectura (los de la arquitectura x86 se encuentran en arch / x86).
• drivers: drivers de los dispositivos
• fs: sistema de ficheros
• include: ficheros cabecera (header filas, con extensión .h)
• init: inicialización del sistema operativo
• ipc: mecanismos de comunicación entre procesos (interprocess communication)
• kernel: funcionalidades basicas de Linux
• lib: biblioteca de funciones utilizada por el kernel
• mm: gestión de memoria (memory management)
• neto: gestión de red

2.2 Herramientas que simplifican la navegación por el código fuente

Aunque estas prácticas pedirán a los estudiantes que analicen un conjunto muy reducido de ficheros, es conveniente conocer herramientas que permitan buscar información dentro de este montón de archivos. Por ejemplo, en qué fichero está

implementada una función, a qué archivos se utiliza, cuál es la definición de una estructura de datos, ... Hay una serie de herramientas que pueden resultar útiles para realizar estas tareas:

• cscope [5] Permite buscar la definición y las referencias a variables, funciones o macros.
• La página web Linux Cross Reference [6] Permite navegar utilizando hipertexto por el código fuente de varias versiones del kernel de Linux.
Se aconseja utilizar cscope. En Ubuntu, puede instalar a ejecutando sudo apt-get install cscope. cscope crea una base de datos con todas las definiciones de símbolos existentes en los ficheros. Como la mayoría de archivos del código del kernel no son relevantes para esta práctica, es conveniente limitar el número de ficheros que serán indexados.

Para poder poner en marcha cscope hay que seguir los siguientes pasos.

1. Situarse en el directorio linux-4.7.5
2. Copiar en ella el archivo cscope.files, incluido a base.zip (cp ../cscope.files.). Este archivo indica qué archivos fuente del kernel deben ser indexados.
3. Ejecutar el pedido cscope. La primera vez que se ejecute creará la base de datos (en este caso, ocupa unos 48 Megabytes). Los siguientes golpes, es conveniente especificar el parámetro -d por qué no vuelva a crear la base de datos.

Cuando cscope está en funcionamiento, utilizando las flechas de cursor es posible elegir la opción de búsqueda deseada. A continuación deberemos escribir el símbolo buscar y pulsar la tecla Enter. Como respuesta, la herramienta mostrará la lista de archivos fuente que satisfacen la búsqueda.

Por ejemplo, la búsqueda de la Global definition de task struct devuelve 46 coincidencias. Al situarse en la segunda coincidencia detectada en el fichero sched.h i pulsar la tecla Enter, la herramienta permite editar el archivo en el punto de la definición. Una vez abandonada la edición, hay que pulsar la tecla Tab para poder iniciar una nueva búsqueda. Para finalizar la ejecución de la herramienta pulse Ctrl D.

Es posible integrar cscope dentro del editor de texto vim. Por ejemplo, si el cursor está situado sobre un nombre de rutina, pulsando una combinación de teclas es posible pasar a editar el archivo fuente donde está implementada la rutina. A [7] Hay un pequeño tutorial sobre cómo combinar cscope y vim.

3 Módulos

Linux ofrece el mecanismo de los módulos para añadir / eliminar código al kernel en tiempo de ejecución (es decir, dinámicamente) sin tener que recompilar todo el kernel y sin tener que reiniciar el sistema. A continuación se presentan tres ejemplos.

3.1 Ejemplo 1: módulo Hello world!

El primer ejemplo es un módulo que escribe un mensaje al ser instalado y otro al ser desinstalado.

3.1.1 Programación del modulo

El código del módulo escribe a un fichero .c (modules / example1 / hw.c). Todo módulo debe contener, como mínimo, dos funciones con una interfaz definida. Una función se ejecutará al añadir (instalar) el modulo dentro del kernel y la otra se ejecutará al eliminar (desinstalar) el módulo. Mediante las macros module init y módulo exit específica qué rutina efectúa cada función.

3.1.2 Compilación del módulo la máquina host

Para compilar el módulo se utilizará un Makefile (modules / example1 / Makefile). El pedido make generará un fichero con extensión .ko (en este ejemplo, hw.ko). Para adaptar este Makefile a otros ficheros fuente basta con modificar la línea que define la variable obj-m.

3.1.3 Transferir el módulo al sistema de archivos de la máquina guest

El siguiente paso es transferir el archivo .ko al sistema de archivos de la máquina guest. Asumiendo que el archivo se encuentra en /home/user/aso/modules/example1/hw.ko, es posible hacerlo de dos formas:

• Utilizando el pedido scp desde la máquina guest. Para ello es necesario que el servidor de Secure Shell (sshd) esté en ejecución en la máquina host (a Ubuntu, puede instalar y ponerlo en marcha con el comando sudo apt-get install openssh-server). La máquina host estará identificada con la dirección IP 10.0.2.2. Hay que ejecutar scp user@10.0.2.2: aso / modules / example1 / hw.ko.

• Montando la imagen del sistema de archivos. Esta opción requiere no tener en funcionamiento la máquina guest. Hay que ejecutar la máquina host los pedidos: ./mount image.sh; sudo cp modules / example1 / hw.ko image / root; ./mount image.sh –u

3.1.4 Pedidos para gestionar módulos

Una vez copiado el archivo .ko al sistema de archivos de la máquina guest, desde el intérprete de comandos de la máquina guest es posible utilizar una serie de pedidos para gestionar los módulos.

• lsmod: lista los módulos que actualmente están instalados.
• insmod: instala un módulo; está parametrizada con el nombre del módulo instalar.
• rmmod: desinstalado instala el módulo especificado como parámetro.

El caso del ejemplo, será preciso instalar el módulo la máquina guest (insmod hw.ko); esto hará que aparezca el mensaje Hello world !. Posteriormente, será posible desinstalar él (pedido rmmod hw.ko) con lo que aparecerá el mensaje Bye world !.

3.2 Ejemplo 2: módulo que permita la interacción con el usuario

Para poder realizar módulos más útiles que la anterior es conveniente que los módulos puedan interaccionar con el usuario. Es decir, es necesario algún mecanismo de comunicación que permita que el usuario aporte información al módulo y que el módulo devuelva resultados al usuario.
En este ejemplo, el problema a resolver por módulo es aportar información sobre un proceso (identificador del proceso padre, identificador del usuario propietario,..). Deberá comunicarse con el módulo para indicarle el identificador de proceso del que se quiere información y porqué devuelva la información del proceso.

3.2.1 Comunicación con el módulo: el pseudo-sistema de archivos / proc

Existen diferentes alternativas para establecer comunicación con el módulo. Una de las más sencillas es la interfaz ofrecida por pseudo-sistema de archivos / proc. Utilizando una rutina del kernel (proc create), es posible crear nuevas entradas en este sistema de ficheros y rutinas a las entradas para capturar las escrituras y responder a las lecturas sobre estas entradas.

En el ejemplo (modules / example2), el código de inicialización del m`odul crea una entrada llamada procdemo el directorio / proc. Las lecturas y escrituras sobre este archivo serán servidas por las rutinas read proc y write proc respectivamente.

Para establecer la comunicación con el módulo, el primer paso será escribir en el archivo el identificador del proceso sobre el que se quiera obtener información. Esto provoca que se ejecute la rutina write proc del módulo y se reciba el pid del proceso. El segundo paso será leer el contenido del archivo. Aix`o provoca que el módulo ejecute la rutina read proc y devuelva la información.
El código de desinstalación del módulo elimina la entrada de / proc.

Al compilar este módulo aparece un warning indicando que el sımbolo find task by pid ns no está definido. Esto se debe a que algunos símbolos (nombres de rutinas y variables, ...) del kernel se pueden utilizar desde un módulo únicamente si aparecen en un listado, y esta rutina no aparece. Para solucionar este problema, hay que añadir las dos líneas siguientes al final del archivo arch / x86 / kernel / i386 ksyms 32.ci recompilar el kernel (en la máquina de referencia, tarda menos de un minuto).

```
#include <linux / sched.h>
EXPORT_SYMBOL (find_task_by_pid_ns);
```

Una vez recompilado el módulo (sin que aparezca el warning), transferido e instalado en la máquina guest, la pedido echo 5> / proc / procdemo indica al módulo que se quiere obtener información sobre el proceso con identificador igual a 5. La ejecución del pedido cat / proc / procdemo provocará que el módulo devuelva la información disponible. El módulo obtendrá esta información accediendo a la estructura de datos struct task struct asociada al proceso; esta estructura contiene el PCB (Process Control Block) del proceso.

3.3 Ejemplo 3: módulo que capture las llamadas al sistema

El tercer ejemplo (modules / example3) es más completo porqué el módulo modifica el comportamiento de una porción del kernel. Además, combina código escrito en lenguaje C (modules / example3 / syscalls.c) y en lenguaje ensamblador (modules / example3 / syscall entry.S).
El objetivo del módulo es contabilizar cuántas veces se invoca cada llamada al sistema. Para ello, el módulo modificará la rutina de atención a la interrupción 0x80, que es la utilizada para implementar la entrada al sistema causada por las llamadas al sistema.

El código de inicialización del módulo obtiene cuál es la rutina de atención (handler) a la interrupción 0x80 y la sustituye por una nueva rutina (new syscall handler). Este nuevo handler invoca una rutina que hace el conteo (do added syscall handler) e invoca el antiguo handler. Para estas prácticas no es necesario entender el código de las rutinas capture syscall handler y restore syscall handler.

Tenga en cuenta que la rutina do added syscall handler rebr`a como parámetro, a través del registro eax, el identificador de llamada al sistema que ha sido invocada. Para ofrecer los datos estadísticas al usuario, el módulo crea la entrada syscall en el directorio / proc. Leer de este fichero devuelve la lista de llamadas al sistema que han sido utilizadas así como el número de invocaciones a cada una de ellas.

Además, escribir un entero sobre el fichero / proc / syscall provoca que se pongan a cero los contadores y se pase a contabilizar las llamadas al sistema realizadas únicamente por el proceso con el pid especificado.

El código de desinstalación del módulo restaura la rutina de atención a la interrupción de entrada al sistema y elimina la entrada / proc / syscall.

Para probar este ejemplo es necesario que, de forma análoga al ejemplo anterior, agregue el símbolo idt table al final del archivo arch / x86 / kernel / i386 ksyms 32.cy que compilar el kernel.

4 Actividades

A continuación se plantean una serie de actividades que debe contestar individualmente para demostrar el logro de los objetivos de la práctica.

1. (10%) Adjuntar una captura de pantalla (screenshot) del escritorio de su máquina host donde se muestren dos ventanas: la correspondiente a la máquina guest y la correspondiente a un navegador web que muestre su página de inicio en el campus de la UOC.

2. Analizar cómo se determina la dirección de memoria donde se encuentra la rutina de atención a una llamada al sistema y el paso de parámetros entre el programa de usuario y el núcleo del sistema operativo.

2.1. Entrada al núcleo del sistema operativo.
Las interrupciones software hacen posible que un programa de usuario provoque la ejecución de código propio del núcleo del sistema operativo. La instrucción de Lenguaje máquina Intel x86 que provoca una interrupción software es int 3 . La instrucción int tiene como parámetro un número entero de 8 bits (el identificador de interrupción software). Ejecutar int cambia el modo de ejecución del procesador y, mediante el vector de interrupciones, transfiere el control a la rutina del núcleo del sistema operativo que da servicio a la interrupción software solicitado.
Como Linux ofrece más de 256 llamadas al sistema, no es posible asociar una interrupción software cada llamada al sistema. Linux soluciona este problema utilizando una única interrupción software (la 0x80) para implementar la entrada al sistema de todas las llamadas al sistema.
Esto provoca que, además de los parámetros de la llamada al sistema, el código de usuario debe transferir en el núcleo del sistema operativo un parámetro adicional, el identificador de llamada al sistema. Por ejemplo, el identificador de la llamada write está definido en la constante NR write.
El paso de parámetros se realiza utilizando registros del procesador. Para deducir qué registros utilizan, analizaréis el código ensamblador del programa auxiliar / parameters / writeread.c presente en base.zip. Os adjuntamos el código ensamblador de este archivo a una distribución Ubuntu 10.4 ya una Ubuntu 4.12.

- (5%) Utilizando cscope, qué fichero está definida la constante NR execve? qué valor tiene?

- (10%) ¿Cuál es el convenio de paso de parámetros entre el código de usuario y el núcleo del sistema operativo Ubuntu 10.04? Pista: edite el fichero dis de Ubuntu 10.04, y buscar el código correspondiente a la rutina main; desde aquı, vaya al código de libc write e identifique donde se está haciendo la llamada al sistema.
- (5%) A qué cree que se debe la diferencia entre Ubuntu 10.04 y 12.04?

2.2. Rutina de atención a la interrupción software 0x80 y rutina de atención a la llamada al sistema.

La rutina de atención a la interrupción software 0x80 llama entry INT80 32. Esta rutina utiliza el identificador de llamada al sistema para, mediante una tabla, determinar cuál es la rutina del núcleo del sistema operativo que da servicio a esta llamada al sistema.

- (5%) A qué fichero está implementada la rutina entry INT80 32?
- (5%) Analizando el código de la rutina, indique como se llama la tabla que contiene las direcciones de las rutinas de atención a las llamadas al sistema.

2.3. Paso de parámetros entre rutinas del núcleo del sistema operativo.

El convenio de paso de parámetros utilizado dentro del núcleo del SO es el siguiente: el primer parámetro se pasa utilizando el registro eax, el segundo utilizando edx y el tercero utilizando ECX.

Si una rutina tiene más de tres parámetros, el resto los recibirá mediante la pila.

- (10%) Queremos invocar una rutina del núcleo (escrita en lenguaje C) con la interfaz int rutina (int par1, int par2) ;. Si los parámetros se encuentran en los registros eax y ECX, qué será necesario hacer antes de invocar la rutina?

3. (50%) (Se debe haber hecho la actividad anterior) Modificar el módulo de ejemplo 3 de forma que monitorice las invocaciones a la llamada al sistema execve y muestre cuales son los 5 últimos ficheros ejecutables que han estado invocados en el sistema. Se adjunta un posible ejemplo de la ejecución del módulo.

```
root@virtual:~/modules/exec# insmod ksyscalls.ko
Old syscall handler at 0xc11da04c
New syscall handler at 0xc890114c
Correctly installed
 Compiled at Sep 27 2016 16:16:40
root@virtual:~/modules/exec# date
Tue Sep 27 16:17:41 CEST 2016
root@virtual:~/modules/exec# cat /proc/lastcomm
/bin/date
/bin/cat
root@virtual:~/modules/exec# ps
  PID TTY          TIME CMD
  810 tty1     00:00:00 login
  816 tty1     00:00:00 bash
  861 tty1     00:00:00 ps
root@virtual:~/modules/exec# who
root     tty1         Sep 27 16:11
root@virtual:~/modules/exec# ls
Makefile         ksyscalls.ko     modules.order    syscalls_entry.S
Module.symvers   ksyscalls.mod.c  syscalls.c       syscalls_entry.o
exec.zip         ksyscalls.mod.o  syscalls.c.orig
init_names.h     ksyscalls.o      syscalls.o
root@virtual:~/modules/exec# cat /proc/lastcomm
/bin/cat
/bin/ps
/usr/bin/who
/bin/ls
/bin/cat
root@virtual:~/modules/exec#
```

Pista: Habéis de modificar el ejemplo 3 de forma que también paséis como parámetro a la rutina do added syscall handler el primer parámetro de les llamadas al sistema (el ejercicio anterior os ha de servir para saber dónde está). Leer el comentario que aparece en el fichero syscalls entry.S relativo al convenio de paso de parámetros utilizado en el kernel de Linux.

Observaciones:

- En algunas preguntas se debe examinar código ensamblador x86. De las diversas sintaxis existentes, en estas prácticas utilizaremos la AT&T. A [8] podéis encontrar una introducción a esta sintaxis.

Recursos

Recursos Básicos

[1] Acceso directo al código fuente de la versión 4.7.5 del kernel de Linux.
URL http://www.kernel.org/pub/linux/kernel/v4.x/linux-4.7.5.tar.xz
[2] Imagen del sistema de ficheros utilizado en la maquina guest.
URL http://einfmlinux1.uoc.edu/aso/image.img
[3] Ficheros de ejemplo i shellscripts para la primera práctica.
URL http://einfmlinux1.uoc.edu/aso/base.zip

Recursos Complementarios

[4] QEMU Home Page.
URL http://wiki.qemu.org/Index.html
[5] CScope Home Page.
URL http://cscope.sourceforge.net/
[6] Linux Cross Reference.
URL http://lxr.free-electrons.com/
[7] The Vim/Cscope tutorial.
URL http://cscope.sourceforge.net/cscope_vim_tutorial.html
[8] AT&T Assembly Syntax.
URL http://asm.sourceforge.net/articles/linasm.html#Syntax