PRÀCTICA 1: PREPARACIÓ DE L'ENTORN

Presentació

L'objectiu d'aquestes pràctiques és que l'estudiant sigui capaç d'entendre porcions de codi font d'una versió actual del *kernel* de Linux i d'introduir petites modificacions a les funcionalitats del *kernel*.

Les pràctiques de l'assignatura són dues:

- La primera pràctica descriu com preparar l'entorn de desenvolupament utilitzat a les pràctiques (compilació del *kernel* i l'arrencada d'una màquina utilitzant el *kernel* generat) i com navegar dins de l'estructura de directoris que composen el codi font del *kernel* de Linux. També presenta el mecanisme dels mòduls perquè serà utilitzat a la segona pràctica per introduir noves funcionalitats al codi del *kernel*.
- La segona pràctica proposa realitzar un seguit de modificacions força localitzades al codi del *kernel* de Linux. Per exemple, recórrer l'estructura de taules de pàgines d'un procés, crear un driver per a un nou dispositiu o obtenir informació sobre el sistema de fitxers.

Els coneixements previs necessaris per al desenvolupament d'aquestes pràctiques són:

- Algorísmica.
- Programació en un llenguatge d'alt nivell (preferentment en llenguatge C). Ús de punters.
- Estructures de dades (llistes doblement encadenades, taules hash).
- Conceptes teòrics de l'assignatura Sistemes Operatius.
- Utilització de Linux des de l'intèrpret de comandes.
- Entendre petits fragments de codi escrits en assembler i386.

Aquest document conté l'enunciat de la primera pràctica. La primera pràctica de l'assignatura presenta l'entorn de desenvolupament que permetrà estudiar i modificar el codi font del *kernel* de Linux (concretament, el de la versió 4.19.143). També descriu el mecanisme dels mòduls (*modules*) perquè serà emprat a la segona pràctica per introduir modificacions al codi font del *kernel* de Linux. Finalment, proposa que l'alumne escrigui un mòdul a partir dels mòduls d'exemple.

A les pràctiques, la màquina de l'estudiant (màquina *host*, sigui física o virtual) no arrencarà directament amb els *kernels* generats sinó que serà un emulador de PC (màquina *guest*, executat sobre la màquina *host*) qui arrencarà amb aquests *kernels*. Els motius per treballar d'aquesta forma són els següents:

- Possibilitar que els estudiants utilitzin un entorn el més senzill i similar possible.
- Evitar que sigui necessari haver de reiniciar la màquina *host* després d'algunes proves.
- Garantir que els possibles errors presents al codi escrit pels estudiants no afectin ni l'estabilitat ni la coherència de les dades de la màquina *host*.

De totes formes, també seria possible fer que la màquina *host* arrenqués directament amb els *kernels* generats. Per fer-ho, caldria modificar la configuració del gestor d'arranc instal·lat a la màquina *host*.

Competències

Transversals:

- Capacitat per a adaptar-se a les tecnologies i als futurs entorns actualitzant les competències professionals
- Capacitat per a la comunicació escrita en l'àmbit acadèmic i professional

Específiques:

- Capacitat per a analitzar un problema en el nivell d'abstracció adequat a cada situació i aplicar les habilitats i coneixements adquirits per a abordar-lo i resoldre'l
- Capacitat per a dissenyar i construir aplicacions informàtiques mitjançant técniques de desenvolupament, integració i reutilització

Objectius

Els objectius d'aquesta pràctica són que l'estudiant...

- posi en funcionament l'entorn de desenvolupament de les pràctiques (instal·li el *software* requerit, copiï els fitxers necessaris per desenvolupar la pràctica, els descompacti, compili el *kernel* i comprovi que la màquina *guest* arrenca correctament amb el *kernel* generat).
- conegui els principals directoris en què s'estructura el codi font del *kernel* de Linux.
- sàpiga utilitzar alguna eina que permet navegar dins del codi font del *kernel* de Linux.
- entengui la utilitat del mecanisme dels mòduls.
- instal·li els mòduls d'exemple a la màquina *guest*.
- modifiqui un dels mòduls d'exemple.

Per avaluar l'assoliment dels objectius, els estudiants hauran de contestar *individualment* un seguit de qüestions formulades al final d'aquest document (apartat 4).

Descripció de la Pràctica

1 Preparació de l'entorn de desenvolupament de les pràctiques

1.1 Requisits

En primer terme, l'estudiant ha de comprovar que la seva màquina *host* compleixi els següents requisits:

- El processador de la màquina *host* ha de ser compatible x86.

- Cal tenir instal·lada alguna distribució Linux a la màquina *host*, **preferentment la facilitada per la UOC aquest curs (la Ubuntu 14.04 de 32 bits)**.
- La màquina *host* ha de disposar de prou espai lliure a disc (uns 2 Gigabytes).
- L'estudiant ha de poder descarregar uns 400 Megabytes per preparar l'entorn de treball.

1.2 Passos a seguir

1. A la màquina *host*, crear un directori on es guardaran tots els fitxers de la pràctica. Entrar-hi.
2. Descarregar de la xarxa els següents fitxers:
   (a) `linux-4.19.143.tar.xz` de [1] (ocupa uns 99 Megabytes): conté el codi font de la versió 4.19.143 del *kernel* de Linux.
   (b) `image.img` de [2] (ocupa uns 300 Megabytes): conté la imatge d'un sistema de fitxers en format ext2; aquest serà el sistema de fitxers accedit per la màquina *guest*.
   (c) `base.zip` de [3] (ocupa uns 60 Kilobytes): conté un seguit de fitxers d'exemple i de *shellscripts*.
3. Descompactar el fitxer `base.zip` utilitzant la comanda `unzip base.zip`
4. Descompactar el fitxer `linux-4.19.143.tar.xz`[1]. Això generarà una estructura de directoris a partir del directori `linux-4.19.143` que ocuparà uns 910 Megabytes.
5. Instal·lar les comandes `bison` i `flex` utilitzant la comanda `sudo apt-get install bison flex`
6. Compilar el *kernel*. Per fer-ho es seguiran els següents passos:
   (a) Situar-se al directori amb el codi font del *kernel* de Linux (comanda `cd linux-4.19.143`).
   (b) Definir la configuració del *kernel*. Per unificar la configuració de tots els estudiants i eliminar aquelles parts del *kernel* no rellevants a aquestes pràctiques, el fitxer `base.zip` conté una d'estàndard (fitxer `.config`) que cal copiar al directori actual (`cp ../.config .`).
   (c) Executar la comanda `make`. Aquesta comanda compila els fitxers font del *kernel* i genera la imatge comprimida del *kernel*. El fitxer generat és `arch/x86/boot/bzImage`. Com a referència, a un ordinador amb un processador i7 a 2.9 GHz, i fent que la màquina *host* també sigui una màquina virtual, el temps de compilació és inferior a quinze minuts.
7. L'estudiant haurà d'utilitzar l'emulador `qemu` [4] per emular la màquina *guest*. En cas de treballar sobre Ubuntu, pot instal·lar-se executant la comanda `sudo apt-get install qemu-kvm`.
8. Fer que la màquina *guest* arrenqui utilitzant com a *kernel* el que acaba de ser compilat i com a sistema de fitxers la imatge que ha estat descarregada de la xarxa. Es farà servir el shellscript `boot.sh`. Un cop arrencada, cal identificar-se com a l'usuari `root`; no cal introduir *password*. Per passar a treballar sobre una altra finestra de la màquina *host* cal prémer *Ctrl Alt* simultàniament.
9. Dins la màquina *guest*, escriure la comanda `halt` per procedir a aturar-la. Quan aparegui el missatge `System halted`, és segur eliminar la finestra creada per `qemu`.

Observacions:

- Arrencar la màquina *guest* és requisit indispensable per poder realitzar les pràctiques.
- Cal diferenciar quines comandes s'han d'executar a la màquina *host* i quines a la màquina *guest*.

[1] Al fitxer `comms.txt`, extret de `base.zip`, estan escrites totes les comandes (amb els seus paràmetres) necessàries per a la preparació de l'entorn de desenvolupament.

2 Navegació pel codi font del *kernel* de Linux

2.1 Directoris principals

El *kernel* de Linux està composat per milers de fitxers font (per exemple, la versió 4.19.143 està formada per 26.118 fitxers amb extensió `.c` i 19.610 fitxers amb extensió `.h` organitzats en 4.599 directoris). A continuació s'enumeren els directoris principals de l'estructura de directoris que emmagatzema el codi font del *kernel* de Linux i quin tipus de codi contenen.

- `arch`: fitxers dependents de l'arquitectura (els de l'arquitectura x86 es troben a `arch/x86`).
- `drivers`: *drivers* dels dispositius
- `fs`: sistema de fitxers
- `include`: fitxers capçalera (*header files*, amb extensió `.h`)
- `init`: inicialització del sistema operatiu
- `ipc`: mecanismes de comunicació entre processos (*interprocess communication*)
- `kernel`: funcionalitats bàsiques de Linux
- `lib`: biblioteca de funcions utilitzada pel *kernel*
- `mm`: gestió de memòria (*memory management*)
- `net`: gestió de xarxa

2.2 Eines que simplifiquen la navegació pel codi font

Tot i que aquestes pràctiques demanaran als estudiants que analitzin un conjunt molt reduït de fitxers, és convenient conèixer eines que permetin buscar informació dins d'aquest munt de fitxers. Per exemple, a quin fitxer està implementada una funció, a quins fitxers s'utilitza, quina és la definició d'una estructura de dades,... Hi ha un seguit d'eines que poden resultar útils per realitzar aquestes tasques:

- `cscope` [5] permet buscar la definició i les referències a variables, funcions o macros.
- La pàgina web Linux Cross Reference [6] permet navegar utilitzant hipertext pel codi font de diverses versions del *kernel* de Linux.

S'aconsella utilitzar `cscope`. A Ubuntu, pot instal·lar-se executant `sudo apt-get install cscope`. `cscope` crea una base de dades amb totes les definicions de símbols existents als fitxers. Com la majoria de fitxers del codi del *kernel* no són rellevants per a aquesta pràctica, és convenient limitar el nombre de fitxers que seran indexats.

Per poder posar en marxa `cscope` cal seguir els següents passos.

1. Situar-se al directori `linux-4.19.143`
2. Copiar-hi el fitxer `cscope.files`, inclòs a `base.zip` (`cp ../cscope.files .`). Aquest fitxer indica quins fitxers font del *kernel* han de ser indexats.

3. Executar la comanda `cscope`. El primer cop que s'executi crearà la base de dades (en aquest cas, ocupa uns 50 Megabytes). Els següents cops, és convenient especificar el paràmetre `-d` perquè no torni a crear la base de dades.

Quan `cscope` està en funcionament, utilitzant les fletxes de cursor és possible triar l'opció de cerca desitjada. A continuació caldrà escriure el símbol a buscar i prémer la tecla *Enter*. Com a resposta, l'eina mostrarà la llista de fitxers font que satisfan la cerca.

Per exemple, la cerca de la *Global definition* de `task_struct` retorna 56 coincidències. Al situar-se a la coincidència detectada en el fitxer `sched.h` i prémer la tecla *Enter*, l'eina permet editar el fitxer en el punt de la definició. Un cop abandonada l'edició, cal prémer la tecla *Tabulador* per poder iniciar una nova cerca. Per finalitzar l'execució de l'eina cal prémer *Ctrl D*.

És possible integrar `cscope` dins de l'editor de text `vim`. Per exemple, si el cursor està situat sobre un nom de rutina, prement una combinació de tecles és possible passar a editar el fitxer font on està implementada la rutina. A [7] hi ha un petit tutorial referent a com combinar `cscope` i `vim`.

3 Mòduls

Linux ofereix el mecanisme dels mòduls per afegir/eliminar codi al *kernel* en temps d'execució (és a dir, dinàmicament) sense haver de recompilar tot el *kernel* i sense haver de reiniciar el sistema. A continuació es presenten tres exemples.

3.1 Exemple 1: mòdul *Hello world!*

El primer exemple és un mòdul que escriu un missatge (mitjançant la rutina del *kernel* `printk`, anàloga a `printf`) al ser instal·lat i un altre al ser desinstal·lat.

3.1.1 Programació del mòdul

El codi del mòdul s'escriu a un fitxer `.c` (`modules/example1/hw.c`). Tot mòdul ha de contenir, com a mínim, dues funcions amb una interfície definida. Una funció s'executarà a l'afegir (instal·lar) el mòdul dins del *kernel* i l'altra s'executarà en eliminar (desinstal·lar) el mòdul. Mitjançant les macros `module_init` i `module_exit` s'especifica quina rutina efectua cada funció.

3.1.2 Compilació del mòdul a la màquina *host*

Per compilar el mòdul s'utilitzarà un fitxer `Makefile` (`modules/example1/Makefile`). La comanda `make` generarà un fitxer amb extensió `.ko` (en aquest exemple, `hw.ko`). Per adaptar aquest `Makefile` a altres fitxers font només cal modificar la línia que defineix la variable `obj-m`.

3.1.3 Transferir el mòdul al sistema de fitxers de la màquina *guest*

El següent pas és transferir el fitxer `.ko` al sistema de fitxers de la màquina *guest*. Assumint que el fitxer es troba a `/home/user/aso/modules/example1/hw.ko`, és possible fer-ho de dues formes:

- Utilitzant la comanda `scp` des de la màquina *guest*. Per fer-ho cal que el servidor de *Secure Shell* (`sshd`) estigui en execució a la màquina *host* (a Ubuntu, podeu instal·lar-lo i posar-lo en marxa amb la comanda `sudo apt-get install openssh-server`). La màquina *host* estarà identificada amb l'adreça IP 10.0.2.2. Cal executar `scp user@10.0.2.2:aso/modules/example1/hw.ko .`
- Muntant la imatge del sistema de fitxers. Aquesta opció requereix no tenir en funcionament la màquina *guest*. Cal executar a la màquina *host* les comandes:
`./mount_image.sh; sudo cp modules/example1/hw.ko image/root; ./mount_image.sh -u`

3.1.4 Comandes per gestionar mòduls

Un cop copiat el fitxer `.ko` al sistema de fitxers de la màquina *guest*, des de l'interpret de comandes de la màquina *guest* és possible utilitzar un seguit de comandes per gestionar els mòduls.

- `lsmod`: llista els mòduls que actualment estan instal·lats.
- `insmod`: instal·la un mòdul; està parametritzada amb el nom del mòdul a instal·lar.
- `rmmod`: desinstal·la el mòdul especificat com a paràmetre.

Al cas de l'exemple, serà precís instal·lar el mòdul a la màquina *guest* (`insmod hw.ko`); això farà que aparegui el missatge *Hello world!*. També és possible que aparegui el missatge *loading out-of-tree module taints kernel*; ignoreu aquest missatge.

Posteriorment, serà possible desinstal·lar el mòdul (comanda `rmmod hw.ko`) amb el que apareixerà el missatge *Bye world!*.

3.2 Exemple 2: mòdul que permeti la interacció amb l'usuari

Per poder realitzar mòduls més útils que l'anterior és convenient que els mòduls puguin interaccionar amb l'usuari. És a dir, és necessari algun mecanisme de comunicació que permeti que l'usuari aporti informació al mòdul i que el mòdul retorni resultats a l'usuari.

En aquest exemple, el problema a resoldre pel mòdul és aportar informació sobre un procés (identificador del procés pare, identificador de l'usuari propietari,...). Caldrà comunicar-se amb el mòdul per indicar-li l'identificador de procés del que es vol informació i perquè retorni la informació del procés.

3.2.1 Comunicació amb el mòdul: el pseudo-sistema de fitxers `/proc`

Existeixen diferents alternatives per establir comunicació amb el mòdul. Una de les més senzilles és la interfície oferta pel pseudo-sistema de fitxers `/proc`. Utilitzant una rutina del *kernel* (`proc_create`), és possible crear noves entrades en aquest sistema de fitxers i associar rutines a les entrades per tal de capturar les escriptures i respondre a les lectures sobre aquestes entrades.

A l'exemple (`modules/example2`), el codi d'inicialització del mòdul crea una entrada anomenada `procdemo` al directori `/proc`. Les lectures i escriptures sobre aquest fitxer seran servides per les rutines `read_proc` i `write_proc` respectivament.

Per establir la comunicació amb el mòdul, el primer pas serà escriure al fitxer l'identificador del procés sobre el que es vulgui obtenir informació. Això provoca que s'executi la rutina `write_proc` del mòdul i es

rebi el *pid* del procés. El segon pas serà llegir el contingut del fitxer. Això provoca que el mòdul executi la rutina `read_proc` i retorni la informació.

El codi de desinstal·lació del mòdul elimina l'entrada de `/proc`.

Un cop recompilat el mòdul, transferit i instal·lat a la màquina *guest*, la comanda `echo 5 > /proc/procdemo` indica al mòdul que es vol obtenir informació sobre el procés amb identificador igual a 5. L'execució de la comanda `cat /proc/procdemo` provocarà que el mòdul retorni la informació disponible. El mòdul obtindrà aquesta informació accedint a l'estructura de dades `struct task_struct` associada al procés; aquesta estructura conté el PCB (*Process Control Block*) del procés.

3.3 Exemple 3: mòdul que captura una crida al sistema

El tercer exemple (`modules/example3`) mostra com es pot capturar una crida al sistema.

Les crides al sistema tenen un identificador numèric (definits a `arch/x86/include/generated/uapi/asm/unistd_32.h`). El *kernel* inicialitza la taula `sys_call_table` de forma que a la posició i de la taula s'emmagatzema el punter a la rutina del *kernel* que implementa la crida al sistema amb identificador i. Per exemple, a la posició 4, corresponent a la crida al sistema `write`, trobem el punter a la rutina `sys_write`, que és la rutina del *kernel* que implementa la crida al sistema `write`.

Aquest exemple captura les invocacions a la crida al sistema `open` de forma que, abans d'invocar la rutina del *kernel* `sys_open`, el mòdul fa un `printk` amb el nom del fitxer a obrir.

Al compilar aquest mòdul apareix un *warning* indicant que el símbol `sys_call_table` no està definit. Això és degut a que els símbols (noms de rutines i variables, ...) del *kernel* es poden utilitzar des d'un mòdul únicament si apareixen a un llistat, i aquesta taula no hi apareix. Per solucionar aquest problema, cal afegir la següent línia

```
EXPORT_SYMBOL(sys_call_table);
```

al fitxer `arch/x86/um/sys_call_table_32.c` després de la declaració de `sys_call_table`. Anàlogament, cal exportar el símbol `ia32_sys_call_table` al fitxer `arch/x86/entry/syscall_32.c`. Finalment, cal recompilar el *kernel* (a la màquina de referència, triga menys de dos minuts).

Un cop recompilat el mòdul (sense que aparegui el *warning*), transferit i instal·lat a la màquina *guest*, podreu probar el mòdul.

4 Activitats

A continuació es plantegen un seguit d'activitats que heu de contestar *individualment* per demostrar l'assoliment dels objectius de la pràctica.

1. (10%) Adjunteu una captura de pantalla (*screenshot*) de l'escriptori de la vostra màquina *host* on es mostrin dues finestres: la corresponent a la màquina *guest* i la corresponent a un navegador web que mostri la vostra pàgina d'inici al campus de la UOC.
2. (10%) Adjunteu una captura de pantalla que mostri que heu aconseguit fer funcionar l'exemple 1.
3. (10%) Adjunteu una captura de pantalla que mostri que heu aconseguit fer funcionar l'exemple 2.
4. (10%) Adjunteu una captura de pantalla que mostri que heu aconseguit fer funcionar l'exemple 3.
5. (10%) Utilitzant `cscope`, responeu les següents preguntes:
   (a) A quin fitxer està implementada la rutina `do_execve`?
   (b) Quin és el valor de la constant `__NR_execve`?
6. (50%) A partir del codi dels exemples, escriviu un mòdul que monitoritzi les invocacions a la crida al sistema `exit`. La crida al sistema `exit` té un únic paràmetre denominat codi d'acabament, un enter que pren valors entre 0 i 255. El mòdul ha de capturar les crides a `exit` i ha d'indicar, a través del fitxer `/proc/traceexit`, quants cops s'ha invocat la crida `exit` amb cada codi d'acabament. Per fer proves, disposeu del programa `modules/traceexit/exit_code.c` amb el que, un cop compilat a la màquina *host* i transferit a la màquina *guest*, podeu invocar la crida `exit` amb el codi d'acabament que desitjeu. S'adjunta un possible exemple de l'execució del mòdul.

```
root@virtual:~/modules/traceexit#
root@virtual:~/modules/traceexit# insmod traceexit.ko
exit captured
Correctly installed
 Compiled at Sep 10 2020 16:31:11
root@virtual:~/modules/traceexit# ./exit_code 45
root@virtual:~/modules/traceexit# ./exit_code 45
root@virtual:~/modules/traceexit# ./exit_code 32
root@virtual:~/modules/traceexit# cat /proc/traceexit
Code Times
  32     1
  45     2
root@virtual:~/modules/traceexit# ./exit_code 32
root@virtual:~/modules/traceexit# ./exit_code 32
root@virtual:~/modules/traceexit# ./exit_code 156
root@virtual:~/modules/traceexit# cat /proc/traceexit
Code Times
  32     3
  45     2
 156     1
root@virtual:~/modules/traceexit# rmmod traceexit
exit restored
root@virtual:~/modules/traceexit# _
```

Recursos

Recursos Bàsics

[1] Accés directe al codi font de la versió 4.19.143 del kernel de Linux.
URL http://www.kernel.org/pub/linux/kernel/v4.x/linux-4.19.143.tar.xz

[2] Imatge del sistema de fitxers utilitzat a la màquina *guest*.
URL http://einfmlinux1.uoc.edu/aso/image.img

[3] Fitxers d'exemple i shellscripts per a la primera pràctica.
URL http://einfmlinux1.uoc.edu/aso/base.zip

Recursos Complementaris

[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

Criteris d'avaluació

El pes de cada resposta està indicat a l'enunciat. Cal que les respostes estiguin justificades.

Format i data de lliurament

El format de presentació de les respostes és un únic fitxer `zip` o `tar` que serà el resultat de compactar un fitxer `pdf` amb les respostes a les activitats i els fitxers font que hagueu escrit. Aquest fitxer ha lliurar-se al registre d'avaluació continuada abans de la data límit establerta al pla docent (24:00 del 4 de novembre).