El nucli Linux

Josep Jorba Esteve

PID_00174420

Universitat Oberta de Catalunya
www.uoc.edu

Índex

Introducció

El nucli (en anglès *kernel*) del sistema GNU/Linux (que habitualment anomenarem simplement Linux) [Vasb], és el cor del sistema: s'encarrega d'arrencar-lo i, una vegada aquest ja és utilitzable per les aplicacions i els usuaris, s'encarrega de gestionar els recursos de la màquina, en forma de gestió de la memòria, del sistema de fitxers, de les operacions d'entrada i sortida i dels processos i la seva intercomunicació.

Origen de Linux

El nucli Linux es remunta a l'any 1991, quan Linus Torvalds el va posar a disposició de la comunitat. És un dels pocs sistemes operatius àmpliament usats del qual es disposa el codi font.

L'origen es remunta a l'any 1991, quan a l'agost, un estudiant finlandès anomenat **Linus Torvalds** va anunciar en una llista de notícies que havia creat el seu propi nucli de sistema operatiu, que funcionava conjuntament amb programari GNU, i l'oferia a la comunitat de desenvolupadors perquè el provés i suggerís millores per a fer-lo més utilitzable. Aquest és l'origen del nucli del sistema operatiu que més tard s'anomenaria **GNU/Linux**.

Una de les particularitats de Linux és que, seguint la filosofia de programari lliure, se'ns ofereix el codi font del nucli del sistema operatiu (del *kernel*), de manera que és una eina perfecta per a l'educació en temes d'anàlisi i disseny de sistemes operatius.

L'altre avantatge principal és que, com que disposem dels arxius font, podem recompilar-los, per a adaptar-los millor al nostre sistema, i com veurem en el mòdul "Sintonització, optimització i alta disponibilitat", configurar-los per a donar un millor rendiment al sistema.

En aquest mòdul veurem com gestionar aquest procés de preparació d'un nucli per al nostre sistema: com, partint dels arxius font, podem obtenir una nova versió del nucli adaptada al nostre sistema. Veurem com es configura, es compila posteriorment i es fan proves amb el nou nucli obtingut.

A més, veurem que el nucli ha anat afegint tota una sèrie de característiques al llarg de la seva evolució, que l'han convertit en competitiu davant d'altres sistemes. En especial, observarem algunes característiques de la virtualització que ens ofereixen amb suport des del nucli.

Objectius

En els materials didàctics d'aquest mòdul trobareu els continguts i les eines procedimentals per a aconseguir els objectius següents:

1. Conèixer el funcionament del nucli (*kernel*) i dels processos de configuració associats.
2. Poder configurar el nucli del sistema en les distribucions més habituals.
3. Entendre l'ús dels mòduls del nucli i decidir-ne la integració (o no) dins de la part estàtica del nucli.
4. Conèixer les tècniques de virtualització i, en particular, de les incloses en el nucli.
5. Saber adaptar el nucli a les necessitats particulars de l'usuari.

1. El nucli del sistema GNU/Linux

El nucli o *kernel* és la part bàsica de qualsevol sistema operatiu [Tan87] sobre el qual descansa el codi dels serveis fonamentals per a controlar el sistema complet. Bàsicament, la seva estructura pot separar-se en una sèrie de components, o mòduls de gestió orientats al següent:

- **Gestió de processos**: quines tasques s'executaran i en quin ordre i amb quina prioritat. Un aspecte important és la planificació de la CPU: com s'optimitza el temps de la CPU per a executar les tasques amb el màxim rendiment o interactivitat possible amb els usuaris?

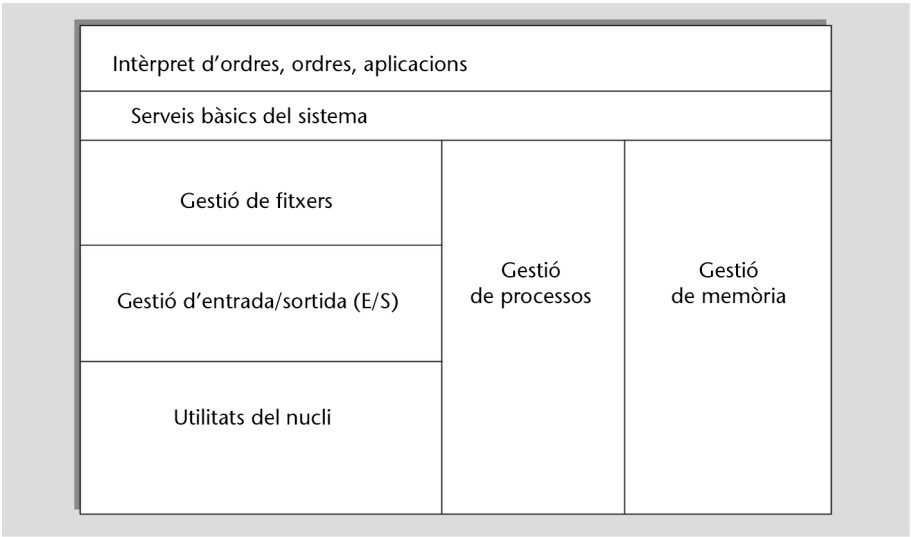
- **Intercomunicació de processos i sincronització**: com es comuniquen tasques entre si, amb quins diferents mecanismes i com poden sincronitzar-se grups de tasques?

- **Gestió d'entrada/sortida (E/S)**: control de perifèrics i gestió de recursos associats.

- **Gestió de memòria**: optimització de l'us de la memòria, sistema de paginació i memòria virtual.

- **Gestió de fitxers**: com el sistema controla i organitza els fitxers presents en el sistema, i com hi accedeix.

Figura 1. Funcions bàsiques d'un nucli respecte a les aplicacions i les ordres executades



En els sistemes privatius, el nucli, o *kernel*, està perfectament "ocult" sota les capes del programari del sistema operatiu, i l'usuari final no té una perspectiva clara de què és aquest nucli ni té tampoc cap possibilitat de canviar-lo o optimitzar-lo, si no és per l'ús d'esotèrics editors de "registres" interns, o programes especialitzats de tercers (normalment d'alt cost). A més, el nucli sol ser únic, és el que proporciona el fabricant, el qual es reserva el dret d'introduir-hi les modificacions que vulgui i quan vulgui, i tractar els errors que apareguin en terminis no estipulats, mitjançant actualitzacions que ens ofereix com a "pedaços" d'errors (o grups de pedaços anomenats comunament *service packs*).

Un dels principals problemes d'aquesta aproximació és precisament la disponibilitat d'aquests pedaços: disposar de les actualitzacions dels errors a temps i, si es tracta de problemes de seguretat, encara amb més raó, ja que fins que no estiguin corregits no podem garantir la seguretat del sistema per a problemes ja coneguts. Moltes organitzacions, grans empreses, governs, institucions científiques i militars no poden dependre dels capricis d'un fabricant per a solucionar els problemes de les seves aplicacions crítiques.

En aquest cas, el nucli Linux ofereix una solució de codi obert, amb els consegüents permisos de modificació, correcció i possibilitat que qualsevol que vulgui i tingui els coneixements adequats en generi noves versions i actualitzacions de manera ràpida. Això permet als usuaris crítics controlar-ne millor les aplicacions i el mateix sistema, i poder muntar sistemes amb el sistema operatiu "a la carta", personalitzat al gust de cadascú. També permet disposar, al seu torn, d'un sistema operatiu amb codi obert, desenvolupat per una comunitat de programadors coordinats mitjançant Internet i accessible, ja sigui per a educació, per a disposar del codi font i abundant documentació, o per a la producció final dels sistemes GNU/Linux adaptats a necessitats individuals o d'un determinat col·lectiu.

En disposar del codi font, es poden aplicar millores i solucions de manera immediata, a diferència del programari privatiu, on hem d'esperar les actualitzacions del fabricant. Podem, a més, personalitzar el nucli tant com necessitem, requisit essencial, per exemple, en aplicacions d'alt rendiment, crítiques en el temps o en solucions amb sistemes encastats (com ara dispositius mòbils).

A continuació repassem una mica la història del nucli [Kera, Kerb]. El nucli Linux el va començar a desenvolupar un estudiant finès anomenat Linus Torvalds, el 1991, amb la intenció de fer una versió semblant a MINIX [Tan87] (versió per a PC d'UNIX [Bac86]) per al processador 386 d'Intel. La primera versió publicada oficialment va ser la de Linux 1.0 el març de 1994, en la qual s'incloïa només l'execució per a l'arquitectura i386 i suportava màquines d'un sol processador. Linux 1.2 va ser publicat el març de 1995 i va ser la primera versió que donava cobertura a diferents arquitectures, com ara Alpha, Sparc i Mips. Linux 2.0, el juny de 1996, va afegir més arquitectures i va ser la primera versió a incorporar suport multiprocessador (SMP) [Tum]. Amb Linux 2.2, del

MINIX

El nucli té l'origen en el sistema MINIX, desenvolupat per Andrew Tanenbaum, com a clon d'UNIX per a PC.

gener de 1999, es van incrementar les prestacions de suport SMP de manera significativa, i es van afegir controladors per a gran quantitat de maquinari. En la 2.4, el gener de 2001, es va millorar el suport SMP, es van incorporar noves arquitectures i es van integrar controladors per a dispositius USB, PC Card (PCMCIA dels portàtils), part de PnP (*plug and play*), suport de RAID i volums, etc. En la branca 2.6 del nucli (desembre de 2003), es va millorar sensiblement el suport SMP, es va introduir una millor resposta del sistema de planificació de CPU, l'ús de fils (*threads*) en el nucli, millor suport d'arquitectures de 64 bits, suport de virtualització i una millor adaptació a dispositius mòbils.

Respecte al procés de desenvolupament, des de la seva creació per Linus Torvalds el 1991 (versió 0.01), ell mateix ha continuat mantenint el nucli, però a mesura que el seu treball li ho permetia i a mesura que el nucli madurava (i creixia), s'ha vist obligat a mantenir les diferents versions estables del nucli gràcies a diferents col·laboradors, mentre que Linus continua (en la mesura possible) desenvolupant i recopilant aportacions per a l'última versió de desenvolupament del nucli. Els col·laboradors principals en aquestes versions han estat [Lkm]:

- 2.0 David Weinehall.
- 2.2 Alan Cox (també desenvolupa i publica pedaços per a la majoria de versions).
- 2.4 Marcelo Tosatti.
- 2.5 Linus Torvalds.
- 2.6 Greg Kroah-Hartman (versions estables) / Linus Torvalds, Andrew Morton (*releases* de desenvolupament).

Complexitat del nucli

El nucli avui en dia ha assolit uns graus de maduresa i complexitat significatius.

Per a veure una mica la complexitat del nucli de Linux, vegem una taula amb la seva història resumida en les diferents versions i en la mida respectiva del codi font. A la taula només s'indiquen les versions de producció; la mida està especificada en milers de línies del codi dels font del nucli:

Versió	Data de publicació	Línies de codi (en milers)
0.01	09-1991	10
1.0	03-1994	176
1.2	03-1995	311
2.0	06-1996	649
2.2	01-1999	1800
2.4	01-2001	3378
2.6	12-2003	5930

Com podem comprovar, hem passat d'unes deu mil línies a sis milions en les primeres versions de la branca 2.6; les últimes versions d'aquesta branca es mouen entre els deu i els quinze milions de línies.

En aquests moments el desenvolupament continua en la branca 2.6.x del nucli, l'última versió estable, que inclou la majoria de distribucions com a versió

principal (algunes encara inclouen algunes 2.4.x, però 2.6.x sol ser l'opció per defecte a la instal·lació).

Tot i que ara la branca principal és la 2.6.x, un cert coneixement de les versions anteriors és imprescindible, ja que fàcilment podem trobar màquines amb distribucions antigues que no s'hagin actualitzat i que és possible que hàgim de mantenir o en les quals hàgim de fer un procés de migració a versions més actuals.

En la branca del 2.6, durant el seu desenvolupament es van accelerar de manera significativa els treballs del nucli, ja que tant Linus Torvalds com Andrew Morton (que mantenen algunes de les branques de Linux 2.6 en desenvolupament) es van incorporar (durant el 2003) a l'Open Source Developement Laboratory (OSDL), un consorci d'empreses la finalitat del qual és promocionar l'ús d'Open Source i GNU/Linux en l'empresa (en el consorci es troben, entre moltes altres empreses amb interessos en GNU/Linux: HP, IBM, Sun, Intel, Fujitsu, Hitachi, Toshiba, Xarxa Hat, Suse, Transmeta, etc.). En aquells moments es va donar una situació interessant, ja que el consorci OSDL va fer de patrocinador dels treballs, tant per al mantenidor de la versió estable del nucli (Andrew) com per al de la de desenvolupament (Linus), treballant a temps complet en les versions i en els temes relacionats. Linus es manté independent, treballant en el nucli, mentre que Andrew va anar a treballar a Google, on continua a temps complet els seus desenvolupaments, fent pedaços amb diferents i noves aportacions al nucli, en el que es coneix com a branca de desenvolupament `-mm`. Després d'un cert temps, OSDL es va reconvertir en la fundació The Linux Foundation.

Cal tenir en compte que amb les versions actuals del nucli s'ha assolit ja un alt grau de desenvolupament i maduresa, cosa que farà que cada vegada s'ampliï més el temps entre la publicació de les versions estables, però no de les revisions parcials o de desenvolupament, aspecte en el qual els mantenidors esperen una nova versió cada 2 o 3 mesos.

Enllaç d'interès

Linux Foundation: http://www.linuxfoundation.org

A més, un altre factor a considerar és la mida i el nombre de persones que estan treballant en el desenvolupament actual. Al començament hi havia unes quantes persones que tenien un coneixement global de tot el nucli, mentre que avui en dia tenim un gran nombre de persones que el desenvolupen (es creu que prop de diversos milers) amb diferents contribucions, tot i que el grup dur s'estima en unes quantes dotzenes de desenvolupadors.

També es pot tenir en compte que la majoria de desenvolupadors (dels milers) només tenen uns coneixements parcials del nucli i, ni tots treballen simultàniament, ni la seva aportació és igual de rellevant (algunes aportacions només corregeixen errors senzills). En l'altre extrem, són unes quantes persones (com els mantenidors) les que disposen d'un coneixement total del nucli. Això implica que es puguin allargar els desenvolupaments i que s'hagin de depurar les aportacions, per a comprovar que no entrin en conflicte entre elles, o que s'hagi d'escollir entre possibles alternatives de prestacions.

Respecte a la numeració de les versions del nucli de Linux [Ker, Ces06], cal tenir en compte els aspectes següents:

1) Fins a la branca del nucli 2.6.x, les versions del nucli Linux es regien per una divisió en dues sèries: una era l'anomenada "experimental" (amb numeració senar en la segona xifra, com 1.3.xx, 2.1.x o 2.5.x) i l'altra era la de producció (sèrie parella, com 1.2.xx, 2.0.xx, 2.2.x, 2.4.x i més). La sèrie experimental eren versions que es movien ràpidament i s'utilitzava per a provar noves prestacions, algorismes, controladors de dispositiu, etc. Per la pròpia naturalesa dels nuclis experimentals, podien tenir comportaments impredictibles, com ara pèrdues de dades, bloquejos aleatoris de la màquina, etc. Per tant, no estaven destinades a utilitzar-se en màquines per a la producció, tret que es volgués provar una característica determinada (amb els consegüents perills).

Els nuclis de producció (sèrie parella) o estables eren els nuclis amb un conjunt de prestacions ben definit, amb un nombre baix d'errors coneguts i controladors de dispositius provats. Es publicaven amb menys freqüència que els experimentals i n'hi havia diverses versions, unes de més o menys qualitat que d'altres. Les distribucions GNU/Linux se solen basar en una determinada versió del nucli estable, no necessàriament l'últim nucli de producció publicat.

2) En la numeració actual del nucli Linux (utilitzada en la branca 2.6.x), es continuen conservant alguns aspectes bàsics: la versió s'indica per uns números X.Y.Z, en què normalment X és la versió principal, que representa els canvis importants del nucli; Y és la versió secundària, i habitualment implica millores en les prestacions del nucli: Y és parell en els nuclis estables i senar en els desenvolupaments o proves; Z és la versió de construcció, que indica el número de la revisió de X.Y, quant a pedaços o correccions fetes.

Els distribuïdors no solen incloure l'última versió del nucli, sinó la que ells hagin provat amb més freqüència i que puguin verificar que és estable per al programari i els components que ells inclouen. Partint d'aquest esquema de numeració clàssic (que es va seguir durant les branques 2.4.x fins als inicis de la 2.6), hi va haver algunes modificacions per a adaptar-se al fet que el nucli (branca 2.6.x) es torna més estable (fixant X.Y a 2.6) i cada vegada les revisions són menors (perquè signifiquen un salt de versió dels primers números), però el desenvolupament continu i frenètic segueix.

En els últims esquemes s'arriba a introduir quarts números, per a especificar Z canvis menors, o diferents possibilitats de la revisió (amb diferents pedaços afegits que corregeixen errors). La versió així definida amb quatre números és la que es considera estable (*stable*). També s'usen altres esquemes per a les diverses versions de prova (normalment no recomanables per a entorns de producció), com sufixos `-rc` (*release candidate*), `-mm`, que són nuclis experimentals amb gran introducció de pedaços i que representen noves prestacions addicionals, com ara proves de diferents tècniques noves, o els `-git`, que són una espècie de "foto" diària del desenvolupament del nucli. Aquests esquemes de

numeració estan en constant canvi per a adaptar-se a la manera de treballar de la comunitat del nucli i a les seves necessitats per a accelerar-ne el desenvolupament.

3) Per a obtenir l'últim nucli publicat (que normalment s'anomena *vanilla* o *pristine*), cal anar a l'arxiu de nuclis Linux (disponible a http://www.kernel.org) o a la rèplica (*mirror*) local (a Espanya http://www.es.kernel.org). També podran trobar-s'hi alguns pedaços del nucli original, que corregeixen errors detectats *a posteriori* de la publicació del nucli.

Enllaç d'interès

Dipòsit de nuclis Linux: http://www.kernel.org

Algunes de les característiques tècniques [Ces06, Kan] del nucli Linux que podríem destacar són:

- Nucli de tipus monolític: bàsicament és un gran programa creat com una unitat, però conceptualment dividit en diversos components lògics.

- Té suport per a càrrega i descàrrega de porcions del nucli sota demanda; aquestes porcions s'anomenen *mòduls* i solen ser característiques del nucli o controladors de dispositiu.

- Fils de nucli: per al funcionament intern s'utilitzen diversos fils (*threads* en anglès) d'execució interns al nucli, que poden estar associats a un programa d'usuari o bé a una funcionalitat interna del nucli. En Linux no es feia un ús intensiu d'aquest concepte originalment, però ha passat a ser un concepte fonamental per al rendiment, en especial a causa de l'aparició de les CPU *multicore*. En les diferents revisions de la branca 2.6.x es va oferir un suport millor, i gran part del nucli s'executa usant diversos fils d'execució.

- Suport d'aplicacions multifil: suport d'aplicacions d'usuari de tipus multifil (*multithread*), ja que molts paradigmes de computació de tipus client/servidor necessiten servidors capaços d'atendre múltiples peticions simultànies dedicant un fil d'execució a cada petició o grup de peticions. Linux té una biblioteca pròpia de fils que pot usar-se per a les aplicacions multifil, amb les millores que es van introduir en el nucli, que també han permès un ús millor per a implementar biblioteques de fils per al desenvolupament d'aplicacions.

- El nucli és de tipus no preemptiu (*nonpreemptive*): això implica que dins del nucli no poden passar-se crides a sistema (en mode supervisor) mentre s'està resolent la tasca de sistema, i quan aquesta acaba, es prossegueix l'execució de la tasca anterior. Per tant, el nucli dins d'una crida no pot ser interromput per a atendre una altra tasca. Normalment, els nuclis preemptius estan associats a sistemes que treballen en temps real, en què s'ha de permetre la situació anterior per a tractar esdeveniments crítics. Hi ha algunes versions especials del nucli de Linux per a temps real (branques `-rt`, de *realtime*), que ho permeten per mitjà de la introducció d'uns punts fixos en què les tasques del nucli poden interrompre's entre si. També s'ha

millorat especialment aquest concepte en la branca 2.6.x del nucli, que en alguns casos permet interrompre algunes tasques del nucli, reassumibles, per a tractar-ne d'altres, prosseguint-ne posteriorment l'execució. Aquest concepte de nucli preemptiu també pot ser útil per a millorar tasques interactives, ja que si es produeixen crides costoses al sistema, poden provocar retards en les aplicacions interactives.

- Suport per a multiprocessador, tant el que s'anomena *multiprocessament simètric* (SMP) com el *multicore*. Aquest concepte sol englobar màquines que van des del cas simple de 2 fins 64 CPU col·locades en diferents sòcols físics de la màquina. Aquest tema s'ha posat d'especial actualitat amb les arquitectures de tipus *multicore*, que permeten de 2 a 8 o més nuclis de CPU en un mateix sòcol físic, en màquines accessibles als usuaris domèstics. Linux pot usar múltiples processadors, i cada processador pot manejar una o més tasques. Però originalment hi havia algunes parts del nucli que disminuïen el rendiment, ja que estan pensades per a una única CPU i obliguen a parar el sistema sencer en determinats bloquejos. SMP és una de les tècniques més estudiades en la comunitat del nucli de Linux, i s'han obtingut millores importants en la branca 2.6. Del rendiment SMP depèn en gran mesura l'adopció de Linux en els sistemes empresarials, en l'aspecte de sistema operatiu per a servidors.

- Sistemes de fitxers: el nucli té una bona arquitectura dels sistemes de fitxers, ja que el treball intern es basa en una abstracció d'un sistema virtual (VFS, *virtual file system*), que pot ser adaptada fàcilment a qualsevol sistema real. Com a resultat, Linux és potser el sistema operatiu que més sistemes de fitxers suporta, des del seu propi ext2 inicial, fins a msdos, vfat, ntfs, sistemes amb *journal* com ext3, ext4, ReiserFS, JFS(IBM), XFS(Silicon), NTFS, iso9660 (CD), udf, etc. i s'hi van afegint més en les diferents revisions del nucli.

Altres característiques menys tècniques (una mica de màrqueting) que podríem destacar són:

1) Linux és gratuït: juntament amb el programari GNU, i l'inclòs en qualsevol distribució, podem tenir un sistema tipus UNIX complet pràcticament pel cost del maquinari; i per la part dels costos de la distribució GNU/Linux, podem obtenir-la pràcticament gratis. Però no està de més pagar per una distribució completa, amb els manuals i el suport tècnic, a un cost més baix comparat amb el que es paga per alguns sistemes privatius, o contribuir amb la seva compra al desenvolupament de les distribucions que més ens agradin o ens resultin pràctiques.

2) Linux és personalizable: la llicència GPL ens permet llegir i modificar el codi font del nucli (sempre que tinguem els coneixements adequats).

3) Linux s'executa en maquinari antic bastant limitat; és possible, per exemple, crear un servidor de xarxa amb un 386 amb 4 MB de RAM (hi ha distribucions especialitzades en recursos baixos).

4) Linux és un sistema d'altes prestacions: l'objectiu principal en Linux és l'eficiència i s'intenta aprofitar al màxim el maquinari disponible.

5) Alta qualitat: els sistemes GNU/Linux són molt estables, amb una baixa proporció d'errors, i redueixen el temps dedicat a mantenir els sistemes.

6) El nucli és bastant reduït i compacte: és possible col·locar-lo, juntament amb alguns programes fonamentals, en un sol disc d'1,44 MB (hi ha diverses distribucions d'un sol disquet amb programes bàsics).

7) Linux és compatible amb una gran part dels sistemes operatius, pot llegir fitxers de pràcticament qualsevol sistema de fitxers i pot comunicar-se per xarxa per oferir i rebre serveis de qualsevol d'aquests sistemes. A més, també amb certes biblioteques pot executar programes d'altres sistemes (com MS-DOS, Windows, BSD, Xenix, etc.) en l'arquitectura x86 o bé virtualitzar màquines completes.

8) Linux disposa d'un amplíssim suport: no hi ha cap altre sistema que tingui la rapidesa i la quantitat de pedaços i actualitzacions que Linux, ni tan sols en els sistemes privatius. Per a un problema determinat, hi ha infinitat de llistes de correu i fòrums que en poques hores poden permetre solucionar qualsevol problema. L'únic problema és en els controladors de maquinari recent, que molts fabricants encara es resisteixen a proporcionar, si no és per a sistemes privatius. Però això està canviant a poc a poc, i alguns dels fabricants més importants de sectors com els de les targetes de vídeo (NVIDIA, ATI) i impressores (Epson, HP) ja comencen a proporcionar els controladors per als seus dispositius, o bé de codi obert, o bé binaris utilitzables pel nucli.

2. Personalització o actualització del nucli

Com a usuaris o administradors de sistemes GNU/Linux, hem de tenir en compte les possibilitats que ens ofereix el nucli per a adaptar-lo a les nostres necessitats i als nostres equips.

Normalment, construïm els nostres sistemes GNU/Linux a partir de la instal·lació en els nostres equips d'alguna de les distribucions de GNU/Linux, tant si són comercials, com Red Hat, Mandriva o Suse, com "comunitàries", com Debian i Fedora.

Aquestes distribucions aporten, en el moment de la instal·lació, una sèrie de nuclis Linux binaris ja preconfigurats i compilats, i normalment hem d'escollir quin nucli del conjunt dels disponibles s'adapta més bé al nostre maquinari. Hi ha nuclis genèrics per a una arquitectura, per a un model de processador o bé orientats a disposar d'una sèrie de recursos de memòria, d'altres que ofereixen una barreja de controladors de dispositius [Ar05], possibilitats de virtualització, etc.

Una altra opció d'instal·lació acostuma a ser la versió del nucli. Normalment les distribucions usen una versió per a la instal·lació que consideren prou estable i provada perquè no causi problemes als usuaris. Per exemple, avui dia moltes distribucions vénen amb una versió de la branca 2.6.x del nucli per defecte, que es considerava la versió més estable en el moment en què va sortir la distribució. En alguns casos, en el moment de la instal·lació pot oferir-se la possibilitat d'usar com a alternativa versions més modernes, amb un suport millor per a dispositius més moderns (d'última generació), però potser no tan provades.

A més, els distribuïdors acostumen a modificar el nucli per a millorar el comportament de la seva distribució o corregir errors que han detectat en el nucli en el moment de les proves. Una altra tècnica bastant comuna en les distribucions comercials és deshabilitar prestacions problemàtiques, que poden causar errors o que necessiten una configuració específica de la màquina. També pot passar que es consideri que una determinada prestació no es prou estable per a incloure-la activada.

Personalització del nucli

La possibilitat d'actualitzar i personalitzar el nucli a mida ofereix una bona adaptació a qualsevol sistema, fet que permet l'optimització i la sintonització del nucli al sistema destí.

Això ens duu a considerar que, per molt bé que un distribuïdor faci la feina d'adaptar el nucli a la seva distribució, sempre ens podem trobar amb una sèrie de problemes o objectius que no podem dur a terme amb la situació actual:

- El nucli no està actualitzat en l'última versió estable disponible; no es disposa de suport per a alguns dispositius moderns.

- El nucli estàndard no disposa de suport per als dispositius que tenim, perquè no han estat habilitats.
- Els controladors que ens ofereix un fabricant necessiten una nova versió del nucli o modificacions.
- A la inversa, el nucli és massa modern i tenim maquinari antic que ja no té suport en els últims nuclis.
- El nucli, tal com està, no obté les màximes prestacions dels nostres dispositius.
- Algunes aplicacions que volem fer servir requereixen suport d'un nucli nou o d'algunes de les seves prestacions.
- Volem estar a l'última i ens arrisquem instal·lant últimes versions del nucli Linux.
- Ens agrada investigar o provar els nous avenços del nucli o bé volem tocar-lo o modificar-lo.
- Volem programar un controlador per a un dispositiu no suportat.
- Etc.

Per aquests i altres motius podem no estar contents amb el nucli que tenim. Se'ns plantegen llavors dues possibilitats: actualitzar el nucli binari de la distribució o bé personalitzar-lo a partir dels paquets font.

Veurem a continuació algunes qüestions relacionades amb les diferents opcions i què impliquen:

1) Actualització del nucli de la distribució. El distribuïdor normalment publica també les actualitzacions del nucli que van apareixent. Quan la comunitat Linux crea una nova versió del nucli, cada distribuïdor la uneix a la seva distribució i en fa les proves pertinents. Després del període de prova, s'identifiquen possibles errors, els corregeix i produeix l'actualització del nucli pertinent respecte al que oferia en els CD de la distribució. Els usuaris poden descarregar la nova revisió de la distribució del lloc web o bé actualitzar-la mitjançant algun sistema automàtic de paquets via dipòsit de paquets. Normalment, es verifica quina versió té el sistema, es descarrega el nucli nou i es fan els canvis necessaris perquè la propera vegada el sistema funcioni amb el nucli nou, i es manté la versió antiga per si hi ha problemes.

Aquesta mena d'actualització ens simplifica molt el procés, però no necessàriament ha de solucionar els nostres problemes, ja que pot ser que el nostre maquinari no estigui encara suportat o que la característica que volem provar del nucli no estigui encara en la versió que tenim de la distribució; cal recordar que no ha de fer servir per força l'última versió disponible (per exemple a kernel.org), sinó aquella que el distribuïdor consideri estable per a la seva distribució.

Si el nostre maquinari tampoc no ve habilitat per defecte en la nova versió, ens trobem en la mateixa situació. O senzillament, si volem la darrera versió, aquest procés no ens serveix.

2) Personalització del nucli. En aquest cas, anirem als paquets font del nucli i adaptarem "a mà" el maquinari o les característiques que volem. Passarem per un procés de configuració i compilació dels paquets font del nucli per a, finalment, crear un nucli binari que instal·larem en el sistema i, així, el tindrem disponible en la següent arrencada del sistema.

La personalització del nucli és un procés que es descriu amb detall en els apartats següents.

També aquí podem trobar-nos amb dues opcions més: o bé per defecte obtenim la versió "oficial" del nucli (kernel.org) o bé podem acudir als paquets font proporcionats per la mateixa distribució. Cal tenir en compte que distribucions com Debian i Fedora fan una tasca important d'adequació del nucli i de correcció d'errors que afecten la seva distribució, i gràcies a això podem disposar, en alguns casos, de correccions addicionals al codi original del nucli. Una vegada més, els paquets font oferts per la distribució no necessàriament han de correspondre a l'última versió estable publicada.

Aquest sistema ens permet la màxima fiabilitat i control, però a un cost d'administració alt, ja que hem de disposar de coneixements amplis dels dispositius i de les característiques que estem escollint (què signifiquen i quines implicacions poden tenir), i també de les conseqüències que puguin tenir les decisions que prenguem.

3. Procés de configuració i compilació

La personalització del nucli [Vasb] és un procés costós, necessita amplis coneixements del procés que s'ha de dur a terme i, a més, és una de les tasques crítiques, de la qual depèn l'estabilitat del sistema, per la mateixa naturalesa del nucli, ja que n'és l'element central.

Obtenció d'un nucli personalitzat

El procés d'obtenció d'un nou nucli personalitzat passa per obtenir els paquets font, adaptar la configuració, compilar i instal·lar el nucli obtingut en el sistema.

Qualsevol error de procediment pot comportar la inestabilitat o la pèrdua del sistema. Per tant, no és sobrer fer tasques de còpia de seguretat de les dades d'usuaris, dades de configuracions que hàgim personalitzat o, si disposem de dispositius adequats, fer una còpia de seguretat completa del sistema. També és recomanable disposar d'algun disquet d'arrencada (o distribució *live CD* amb eines de rescat) que ens serveixi d'ajuda si sorgeixen problemes, o bé un disquet/CD/arxiu de rescat (*rescue disk*) que la majoria de distribucions permeten crear des dels CD de la distribució (o en proporcionen directament algun com a CD de rescat per a la distribució). Actualment molts dels CD autònoms (*live CD*) de les distribucions ja proporcionen prou eines de rescat per a aquestes tasques, tot i que també hi ha algunes distribucions que hi estan especialitzades.

Sense ànims d'exagerar, gairebé mai no apareixen problemes si se segueixen els passos adequadament, es té consciència dels passos fets i es prenen algunes precaucions. Evidentment, davant de sistemes en producció, sempre és important prendre mesures de precaució i fer les còpies de seguretat necessàries.

A continuació veurem el procés necessari per a instal·lar i configurar un nucli Linux. En els subapartats següents, examinarem:

1) El cas de les versions antigues 2.4.x.

2) Algunes consideracions sobre la migració a les 2.6.x a partir de 2.4.x.

3) Detalls específics de les versions 2.6.x.

4) Un cas particular per a la distribució Debian, que disposa d'un sistema propi (*Debian way*) de compilació més flexible.

Enllaç d'interès

Per a Fedora recomanem consultar l'enllaç següent: http://fedoraproject.org/wiki/Building_a_custom_kernel.

Pel que fa a les versions 2.4.x, mantenim en aquest mòdul l'explicació per raons històriques, ja que les distribucions actuals ja gairebé no les ofereixen, però hem de considerar que en més d'una ocasió ens veurem obligats a migrar un determinat sistema a noves versions, o bé a mantenir-lo en les antigues, a causa d'incompatibilitats o existència de maquinari antic no suportat.

Els conceptes generals del procés de compilació i configuració s'explicaran en el primer subapartat (2.4.x), ja que la majoria són genèrics, i observarem posteriorment les diferències respecte a les noves versions. Tot i així, cada subapartat pot examinar-se de manera autosuficient.

També cal afegir que, amb les últimes distribucions, cada vegada és més casual la necessitat de reconstruir o recompilar el nucli mateix, a causa, entre altres consideracions, del fet que:

- Antigament la majoria dels controladors estaven integrats en el nucli i calia recompilar-lo per complet si volíem incloure o excloure un controlador determinat. Avui en dia, com veurem en l'apartat 5, poden recompilar-se els controladors o mòduls concrets, no el nucli en si mateix.

- Per a sintonitzar el nucli, antigament calia recompilar-lo. En molts casos (no tots) es poden sintonitzar alguns elements del nucli mitjançant l'accés als sistemes de fitxers `/proc` o `/sys`.

- En alguns casos de distribucions comercials (versions empresarials per a les quals es paga suport), els nuclis i el sistema complet estan suportats per l'equip de la distribució, i de vegades poden perdre's les llicències de suport o les garanties per a fer canvis d'aquest estil.

- D'altra banda, les distribucions tenen una velocitat bastant ràpida quant a integrar pedaços i nous nuclis a mesura que es generen.

En canvi, una personalització del nucli, mitjançant una recompilació, ens pot permetre:

- Escollir quines parts incloure o excloure del nucli, donant un suport concret a una plataforma o a un maquinari molt concret. Això és imprescindible si ens trobem, per exemple, en situacions de maquinari encastat (*embedded*).

- Sintonitzar, d'aquesta última manera, el consum de memòria o altres recursos per a adaptar-se millor a recursos limitats (CPU, memòria, disc, etc.).

- Versions concretes de les distribucions impliquen usar certes versions específiques del nucli. En molts casos no podem obtenir noves actualitzacions del nucli si no actualitzem la distribució concreta completament.

- Provar versions de nucli encara no disponibles en les distribucions. Malgrat que hi ha certa rapidesa d'integració dels nous nuclis, es pot trigar setmanes o mesos a disposar dels nous nuclis via distribució. D'altra banda, algunes distribucions són molt conservadores quant a la versió de nucli utilitzada, i cal esperar diverses versions completes de la distribució (un període llarg de temps) per a arribar a una versió concreta de nucli.

- Provar versions beta, o pedaços de nucli, amb l'objectiu d'integrar-lo ràpidament en algun sistema amb problemes concrets, o bé senzillament per qüestions d'avaluació de les noves possibilitats o d'un millor rendiment.

- Participar directament en el desenvolupament del nucli, estudiant possibles millores del codi actual, proposant i provant propostes concretes. Això és típic d'alguns components, i també estudiar diferents estratègies de planificació de CPU, de gestió de memòria, millorar paràmetres del nucli o col·locar alguna prestació nova en algun sistema de fitxers.

En els subapartats següents veurem les diferents possibilitats quant a la configuració i la recompilació de les diferents branques de desenvolupament del nucli Linux.

3.1. Compilació antiga de la branca 2.4.x del nucli

Les instruccions són específiques per a l'arquitectura x86 d'Intel, mitjançant usuari arrel (*root*) (tot i que la major part del procés pot fer-se com a usuari normal i, de fet, és aconsellable per raons de seguretat):

1) Obtenir el nucli. Per exemple, podem acudir a http://www.kernel.org (o al seu servidor ftp) i descarregar la versió *vanilla* que vulguem provar. Hi ha rèpliques per a diferents països; així podem, per exemple, acudir al web: http://www.es.kernel.org. D'altra banda, en la majoria de les distribucions de GNU/Linux, com Fedora/Red Hat o Debian, també s'ofereix com a paquet el codi font del nucli (normalment amb algunes modificacions incloses); si es tracta de la versió del nucli que necessitem, potser és preferible usar aquestes (mitjançant els paquets `kernel-source`, `linux-source` o similars). Si volem els últims nuclis, potser no estan disponibles en la distribució i haurem d'acudir a kernel.org.

2) Desempaquetar el nucli. Els paquets font del nucli solien col·locar-se i desempaquetar-se sobre el directori `/usr/src`, tot i que es recomana utilitzar algun directori a banda per a no barrejar-los amb fitxers font que pugui portar la distribució. Per exemple, si els paquets font venien en un fitxer comprimit de tipus bzip2:

```
bzip2 -dc linux-2.4.0.tar.bz2 | tar xvf -
```

Si els paquets font vénen en un fitxer gz, aleshores reemplacem bzip2 per gzip. En descomprimir els paquets font s'haurà generat un directori anomenat `linux-version-kernel`, en què entrarem per a establir la configuració del nucli.

Eines de configuració i compilació

Abans de començar els passos previs a la compilació, ens hem d'assegurar de disposar de les eines correctes, en especial del compilador gcc, make i altres utilitats gnu complementàries durant el procés. Un exemple són les modutils, que disposen les diferents

utilitats per a l'ús i la gestió dels mòduls de nucli dinàmics. Així mateix, per a les diferents opcions de configuració cal tenir en compte una sèrie de prerequisits en forma de biblioteques associades a la interfície de configuració usada (per exemple les ncurses per a la interfície menuconfig).

Es recomana, en general, consultar la documentació del nucli (via paquet o en el directori arrel de les font del nucli) per a conèixer quins prerequisits i versions dels paquets són necessaris per al procés. Es recomana examinar els fitxers `README` en el directori "arrel", i el `Documentation/Changes`, o l'índex de documentació del nucli en `Documentation/00-INDEX`.

Si hem fet compilacions anteriors en el mateix directori, haurem de garantir que el directori utilitzat estigui net de compilacions anteriors; podem netejar-lo amb `make mrproper` (fet des del directori "arrel" de les fonts).

3) Configuració del nucli. Per al procés de configuració del nucli [Vasb], tenim diversos mètodes alternatius, que ens presenten interfícies diferents per a ajustar els múltiples paràmetres del nucli (que solen emmagatzemar-se en un fitxer de configuració, normalment `.config` en el directori "arrel" de les fonts). Les diferents alternatives són:

- `make config`: des de l'indicador d'ordres se'ns pregunta per cada opció i se'ns demana confirmació (y/n), si volem o no l'opció, o se'ns demanen els valors necessaris. És la configuració llarga, en la qual se'ns demanen moltes respostes, podem haver de respondre a gairebé un centenar de preguntes (o més, depenent de la versió).

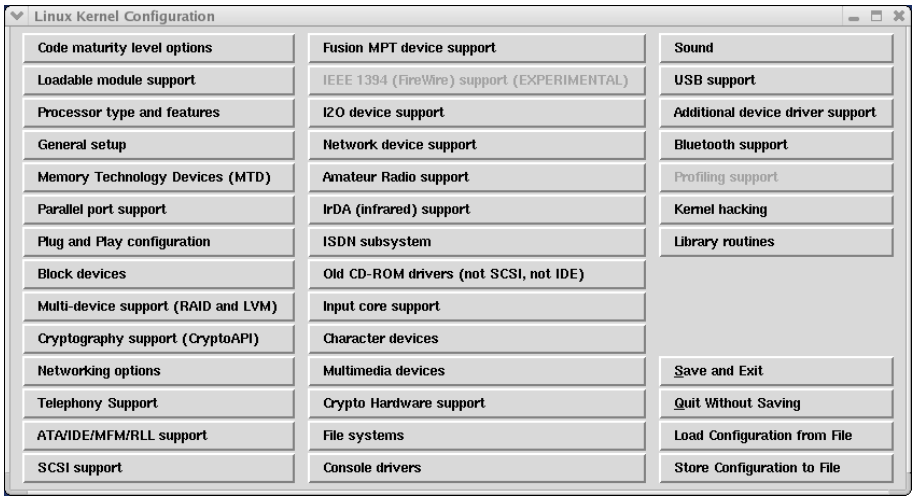
- `make oldconfig`: serveix per si volem reutilitzar una configuració ja usada (normalment emmagatzemada en un fitxer `.config`, en el directori "arrel" de les fonts); cal tenir en compte que només és vàlida si estem compilant la mateixa versió del nucli, ja que diferents versions del nucli poden variar en les opcions.

- `make menu config`: configuració basada en menús textuals, bastant còmoda; podem habilitar o inhabilitar el que vulguem i és més ràpida que el `make config`.

- `make xconfig`: la més còmoda, basada en diàlegs gràfics en X Window (figura 2). Cal tenir instal·lades les biblioteques de tcl/tk, ja que aquesta configuració està programada en aquest llenguatge. La configuració es basa en quadres de diàleg, botons i caselles d'activació. És bastant ràpida i disposa d'ajuda amb comentaris de moltes de les opcions. Però hi ha un defecte: pot ser que algunes de les opcions no apareguin (depèn del fet que el programa de configuració estigui actualitzat, i de vegades no ho està). En aquest darrer cas, el `make config` (o el `menuconfig`) és l'únic que garanteix disposar de totes les opcions elegibles; en els altres tipus de configuració depèn del fet que s'hagin pogut adaptar a temps els programes a les noves opcions quan s'allibera el nucli. Tot i així, en general s'intenten mantenir de manera equivalent.

Una vegada s'ha fet el procés de configuració, cal guardar el fitxer `.config`, ja que la configuració consumeix un temps important. A més, pot ser d'utilitat

Figura 2. Configuració del nucli 2.4.x des de la interfície gràfica en X Window



disposar de la configuració feta (`.config`) si està planejat fer-la en diverses màquines semblants o idèntiques.

Un altre tema important en les opcions de configuració és que en molts casos se'ns preguntarà si una determinada característica la volem integrada en el nucli o com a mòdul. Aquesta és una decisió més o menys important, ja que el rendiment del nucli (i, per tant, del sistema sencer) en alguns casos pot dependre de la nostra elecció.

Els mòduls es tracten en l'apartat 5.

El nucli de Linux ha començat a tenir una grandària considerable, tant per complexitat com pels controladors de dispositiu [Ar05] que inclou. Si l'integréssim tot, es podria crear un fitxer del nucli bastant gros i ocupar molta memòria, fet que alentiria alguns aspectes de funcionament. Els mòduls del nucli [Hen] són un mètode que permet separar part del nucli en petits trossos, que es carregaran dinàmicament sota demanda quan, per càrrega explícita o per ús de la característica, siguin necessaris.

L'elecció més normal és integrar dins del nucli el que es consideri bàsic per al funcionament o crític en rendiment, i deixar com a mòduls les parts o els controladors dels quals es farà un ús esporàdic o que calgui conservar per si es produeixen futures ampliacions de l'equip.

- Un cas clar són els controladors de dispositiu: si estem actualitzant la màquina, pot ser que a l'hora de crear el nucli no coneguem amb seguretat quin maquinari tindrà; per exemple, quina targeta de xarxa, però sí que sabem que estarà connectada a xarxa. En aquest cas el suport de xarxa estarà integrat en el nucli, però pel que fa als controladors de les targetes, podrem seleccionar uns quants (o tots) i posar-los com a mòduls. Així, quan tinguem la targeta podrem carregar el mòdul necessari, o si després hem de canviar una targeta per una altra, només haurem de canviar el mòdul que es carregarà. Si hi hagués només un controlador integrat en el nucli i canviem la targeta, això obligaria a reconfigurar i recompilar el nucli amb el controlador de la targeta nova.

- Un altre cas que sol aparèixer (encara que no és gaire comú) és quan necessitem dos dispositius que són incompatibles entre ells, o està funcionant l'un o l'altre (això podria passar, per exemple, amb impressores amb cable paral·lel i maquinari que es connecta al port paral·lel). Per tant, en aquest cas hem de col·locar com a mòduls els controladors i carregar o descarregar el que sigui necessari.

- Un altre exemple podrien formar-lo els sistemes de fitxers (*filesystems*). Normalment esperarem que el nostre sistema tingui accés a alguns, per exemple ext2, ext3 o ext4 (propis de Linux), vfat (dels Windows 95/98/ME) i els donarem d'alta en la configuració del nucli. Si en un altre moment hagués-sim de llegir un altre tipus no esperat, per exemple dades guardades en un disc o partició de sistema NTFS de Windows NT/XP, no podríem: el nucli no sabria o no tindria suport per a fer-ho. Si tenim previst que en algun moment (però no habitualment) s'ha d'accedir a aquests sistemes, podem deixar els altres sistemes de fitxers com a mòduls.

4) Compilació del nucli. Mitjançant `make` començarem la compilació. Primer cal generar les possibles dependències entre el codi i després, el tipus d'imatge de nucli que es vol (en aquest cas una imatge comprimida, que sol ser la normal):

```
make dep
make bzImage
```

Quan aquest procés acabi, tindrem la part integrada del nucli i ens faltaran les parts que hàgim posat com a mòduls:

```
make modules
```

Fins aquest moment hem fet la configuració i la compilació del nucli. Aquesta part podia fer-se des d'un usuari normal o bé des del root, però ara necessitarem forçosament usuari root, perquè passarem a la part de la instal·lació.

5) Instal·lació. Comencem instal·lant els mòduls:

```
make modules_install
```

i la instal·lació del nucli nou (des de `/usr/src/linux-version`, on `xx` és la versió utilitzada):

```
cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.xx
cp System.map /boot/System.map-2.4.xx
```

L'arxiu `bzImage` és el nucli acabat de compilar, que es col·loca en el directori `/boot`. Normalment el nucli antic es trobarà en el mateix directori `/boot` amb el nom `vmlinuz` o bé `vmlinuz-versió-anterior` i `vmlinuz` com un enllaç simbòlic al nucli antic. Una vegada tinguem el nostre nucli, és millor conservar l'antic, per si es produeixen errors o mal funcionament del nou i

així poder recuperar el vell. El fitxer `System.map` és un fitxer que conté els símbols disponibles en el nucli i és necessari per al procés d'arrencada del nucli; també es col·loca en el mateix directori.

6) Configuració de l'arrencada. El pas següent és dir al sistema amb quin nucli ha d'arrencar. Aquest pas depèn del sistema d'arrencada de Linux, i del *bootloader* que fem servir:

- Des de l'arrencada amb LiLo [Skoa], ja sigui en el Master Boot Record (MBR) o ja sigui des d'una partició pròpia, cal afegir al fitxer de configuració (a `/etc/lilo.conf`), per exemple, les línies:

  ```
  image = /boot/vmlinuz-2.4.0
      label = 2.4.0
  ```

  on `image` és el nucli que arrencarà i `label` serà el nom amb què apareixerà l'opció en l'arrencada. Podem afegir aquestes línies o modificar les que hi hagués del nucli antic. Es recomana afegir-les i deixar el nucli antic, per a poder recuperar el nucli antic si apareixen problemes. En el fitxer `/etc/lilo.conf` hi pot haver una o més configuracions d'arrencada, tant de Linux com d'altres sistemes (com Windows); cada arrencada s'identifica per la seva línia `image` i el `label` que apareix en el menú d'arrencada. Hi ha una línia `"default=label"` en què s'indica el *label* per defecte que s'arrencarà. També podem afegir a les línies anteriors un `"root=/dev/..."` per indicar la partició de disc on és el sistema d'arxius principal (el /). Recordeu que els discos tenen dispositius com `/dev/hda` (primer disc IDE) `/dev/hdb` (segon IDE), i la partició s'indicaria com a `"root=/dev/hda2"` si el / del nostre Linux estigués en la segona partició del primer disc ide. A més, amb `append=` podem afegir paràmetres a l'arrencada del nucli. Després de canviar la configuració del LiLo cal escriure-la físicament en el disc (es modifica el sector d'arrencada) perquè estigui disponible en l'arrencada següent:

  ```
  /sbin/lilo -v
  ```

  A continuació reiniciem i arrenquem amb el nou nucli.

  Si tinguéssim problemes, podem recuperar el nucli antic, escollint l'opció del vell, i després retocar el `lilo.conf` per tornar a l'antiga configuració o estudiar el problema i reconfigurar i recompilar el nucli de nou.

- Arrencada amb Grub: La gestió en aquest cas és bastant simple, es pot afegir una nova configuració formada pel nucli nou i afegir-la com una opció més al fitxer de configuració del Grub, i rearrencar procedint de manera semblant a la del LiLo, però recordant que a Grub n'hi ha prou d'editar el fitxer i rearrencar. També és millor deixar l'antiga configuració per a poder recuperar-se de possibles errors o problemes amb el nucli acabat de compilat.

Lectura recomanada

Sobre Grub podeu consultar *Grub bootloader* i *Grub Manual* accessibles des del web del projecte GNU: http://www.gnu.org.

3.2. Migració de 2.4 a la branca 2.6.x del nucli

En cas d'haver d'actualitzar versions de distribucions antigues, o bé fer el canvi de generació del nucli mitjançant els paquets font, caldrà tenir en compte una sèrie de consideracions, a causa de les novetats introduïdes en la branca 2.6.x del nucli.

Presentem a continuació alguns punts concrets que cal considerar:

- Alguns dels mòduls del nucli han canviat de nom, i alguns han pogut desaparèixer; cal comprovar la situació dels mòduls dinàmics que es carreguen (per exemple, examinar `/etc/modules` i/o `/etc/modules.conf`), i editar-los per a reflectir els canvis.

- S'han afegit noves opcions per a la configuració inicial del nucli: com `make gconfig`, una interfície basada en gtk (Gnome). Cal considerar, com a prerequisit en aquest cas, les biblioteques de Gnome. L'opció de `make xconfig` s'ha implementat ara amb les biblioteques de qt (KDE).

- S'incrementen les versions mínimes necessàries de diverses utilitats necessàries per al procés de compilació (consulteu `Documentation/Changes` en els paquets font del nucli). En especial, la versió del compilador gcc mínima. Ha canviat el paquet per defecte per a les utilitats de mòduls, que passa a ser `module-init-tools` (en lloc de `modutils` que es feia servir en les 2.4.x). Aquest paquet és un prerequisit per a la compilació de nuclis 2.6.x, ja que el carregador de mòduls dinàmics està basat en aquesta nova versió.

- El sistema `devfs` queda obsolet en favor de `udev`, el sistema que controla l'arrencada (connexió) "en calent" (*hotplug*) de dispositius (i el seu reconeixement inicial, de fet simulant una arrencada en calent en iniciar el sistema) i crea dinàmicament les entrades en el directori `/dev`, només per als dispositius que estiguin presents actualment.

- En Debian, a partir de certes versions del nucli 2.6.x, per a les imatges binàries de nucli, *headers* de desenvolupament i el codi font del nucli, els noms de paquets canvien de l'antic `kernel-images/source/headers` a `linux-image/source/headers`.

- En alguns casos, els dispositius de tecnologies noves (com SATA) poden haver passat de `/dev/hdX` a `/dev/sdX`. En aquests casos caldrà editar les configuracions de `/etc/fstab` i el *bootloader* (LiLo o Grub) per a reflectir els canvis.

- Hi pot haver alguns problemes amb dispositius d'entrada/sortida concrets. El canvi de noms de mòduls de nucli ha afectat, entre d'altres, els dispositius de ratolí, fet que pot afectar també l'execució de X-Window, la veri-

ficació de quins models són necessaris i carregar els mòduls correctes (per exemple el `psmouse`). D'altra banda, en el nucli s'integren els controladors de so Alsa. Si disposem dels antics OSS, caldrà eliminar-los de la càrrega de mòduls, ja que Alsa ja s'encarrega de l'emulació d'aquests darrers.

- Respecte a les arquitectures que suporta el nucli, cal tenir en compte que amb els 2.6.x, en les diferents revisions, s'han anat incrementant les arquitectures suportades, la qual cosa ens permetrà disposar d'imatges binàries del nucli en les distribucions (o les opcions de compilació del nucli) més adequades per al suport dels nostres processadors. En concret, podem trobar-nos amb arquitectures com i386 (per a Intel i AMD), que suporta la compatibilitat d'Intel en 32 bits per a tota la família de processadors (en algunes distribucions s'usa 486 com a arquitectura general); en algunes distribucions s'integren versions diferenciades per a i686 (Intel a partir de Pentium Pro i posteriors), per a k7 (AMD Athlon i posteriors) i les específiques de 64 bits, per a AMD de 64 bits (x86_64) i Intel amb extensions em64t de 64 bits com en els Xeon i algunes famílies de multicores. D'altra banda, també hi ha l'arquitectura IA64 per als models de 64 bits Intel Itanium. En la majoria dels casos, les arquitectures disposen de les capacitats SMP activades en la imatge del nucli (tret que la distribució suporti versions amb SMP o sense, creades independentment; en aquest cas, sol afegir-se el sufix *-smp* a la imatge binària del nucli que el suporta).

- En Debian, per a la generació d'imatges `initrd`, a partir de determinades versions del nucli ($\geq$ 2.6.12) es consideren obsoletes les `mkinitrd-tools`, que se substitueixen per utilitats noves com `initramfs-tools` o com `yaird`. Totes dues permeten construir la imatge `initrd`, la primera de les quals és la recomanada per Debian.

3.3. Compilació de la branca 2.6.x del nucli

En les versions 2.6.x, tenint en compte les consideracions comentades abans, la compilació es desenvolupa de manera anàloga a com s'ha exposat anteriorment.

Una vegada descarregats de kernel.org o proporcionats per la distribució, els paquets font del nucli 2.6.xx (en què xx és el número de revisió del nucli estable), es procedeix a extreure'ls en el directori que s'usarà per a la compilació. Si el paquet obtingut és de tipus gzip (`tar.gz`):

```
gzip -cd linux-2.6.xx.tar.gz | tar xvf -
```

o si és de tipus bzip2 (`tar.bz2`):

```
bzip2 -dc linux-2.6.xx.tar.bz2 | tar xvf -
```

Una vegada extrets els arxius font en el directori de compilació, es comproven els paquets de l'entorn de compilació necessari. Normalment cal instal·lar alguns paquets, com per exemple `build-essentials`, `libncurses-dev`, `libqt3-dev`, `libgtk2-dev` (els noms depenen de la distribució), que incorporen l'entorn bàsic de compilació (gcc) i les necessitats per a la construcció dels menús posteriors de compilació dels arxius font (`make menuconfig` o `xconfig`).

És recomanable llegir el fitxer `README` que està situat en el directori arrel dels paquets font.

En aquest punt, podria ser interessant apedaçar el nucli, cosa que podríem fer perquè tenim un codi font addicional (en forma de pedaç) que millora algun problema conegut de la versió o bé perquè volem proposar o provar un canvi de codi en el nucli. També podria donar-se el cas que un fabricant de maquinari oferís algun suport o correcció d'errors per a un controlador de dispositiu, com un pedaç per a una versió de nucli concreta.

El procés d'apedaçar el nucli es tracta en l'apartat 4.

Una vegada disposem dels paquets necessaris, procedim al procés de compilació en el directori utilitzat per als paquets font. Cal recordar que tot el procés pot fer-se com a usuari normal; només cal fer amb l'usuari arrel parts molt concretes, com la instal·lació final del nucli o de mòduls dinàmics.

També poden sorgir problemes de seguretat per a fer la compilació en mode root, de fonts de nucli desconegudes o no fiables. Sobre això, tant els paquets font de kernel.org com els proporcionats per les distribucions solen contenir signatures (o dipòsits signats) que poden fer-se servir per a verificar la integritat dels fitxers de fonts. Cal evitar, de totes totes, usar fonts de nucli o de mòduls proporcionats per emissors no fiables.

Iniciem el procés en el directori de fonts, començant amb la neteja de compilacions anteriors:

```
# make mrproper
```

El pas següent és la configuració de paràmetres. Podem usar el fitxer `.config` o bé a partir de compilacions prèvies que hàgim fet o bé partir de la configuració del nucli actual existent (recordeu que amb `uname -r` tenim la versió de nucli). En funció de la distribució, podem obtenir aquesta configuració de `/boot/config-version-kernel` i podem copiar aquest fitxer com a `.config` en el directori arrel dels paquets font del nucli. Així, partim d'una sèrie d'opcions de nucli ja preparades, en què només haurem de fer els canvis que vulguem o mirar les noves opcions no presents en el nucli antic. Si som en el directori dels paquets font, llavors un:

Recordeu que si disposem d'un `.config` previ, ens permetrà no començar de zero la configuració del nucli.

```
cp /boot/config-‘uname -r‘ ./.config
```

ens deixarà aquestes opcions del nucli actual preparades. També podríem fer servir un fitxer de configuració de compilacions de nucli prèvies que hàgim fet. Si no proposem cap fitxer de configuració previ, llavors partim d'unes

opcions per defecte en les quals haurem de mirar i comprovar totes les opcions del nucli.

Fem la configuració per mitjà de l'opció escollida de `make`, passant per les opcions següents, entre d'altres:

- `make config`: interfície plana de text.
- `make menuconfig`: interfície basada en menús textuals.
- `make xconfig`: interfície gràfica basada en `toolkit Qt` de KDE.
- `make gconfig`: interfície gràfica basada en `toolkit Gtk` de Gnome.
- `make oldconfig`: es basa en el fitxer `.config` previ i pregunta textualment per les opcions noves que no estaven prèviament en la configuració antiga (de `.config`).

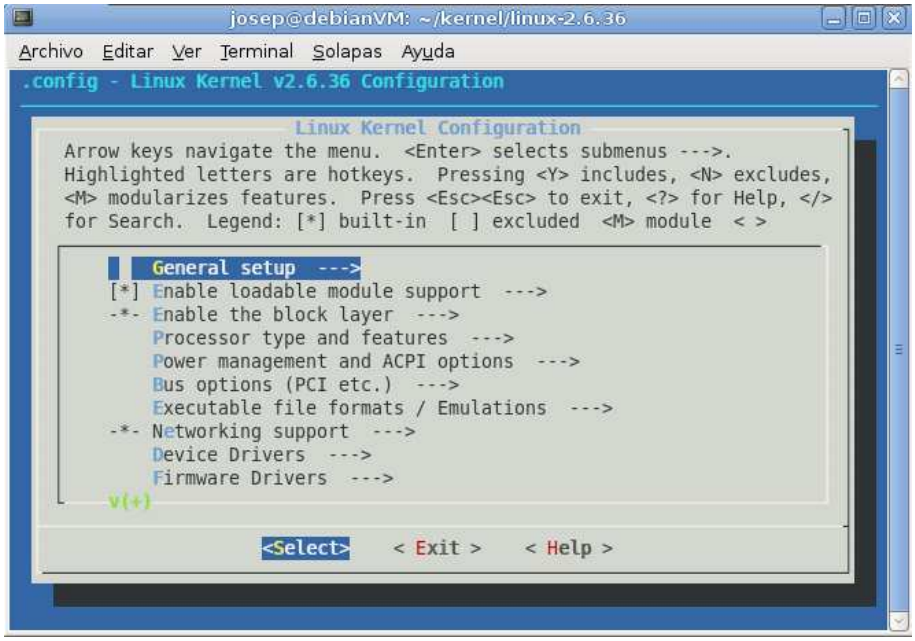
Per exemple, escollim la interfície textual (figura 3):

```
# make menuconfig
```

A continuación construïm la imatge binària del nucli, simplemente amb:

```
# make
```

Figura 3. Configuració del nucli (`make menuconfig`) des de la interfície textual



Cal esmentar que per a aquests nuclis genèrics (normalment anomenats *vanilla* o *pristine*), també hi ha algunes dreceres més ràpides per a la construcció final del nucli (que salten la majoria dels passos següents); per exemple, en Debian és possible substituir l'actual `make` per `make deb-pkg`, que ens obtindrà paquets binaris DEB instal·lables en el sistema i actualitzarà *bootloaders* i els fitxers necessaris. En qualsevol cas, sempre tenim l'alternativa de fer els passos següents:

1) Construcció dels mòduls (els especificats com a tals):

```
# make modules
```

2) instal·lació dels mòduls creats (en `/lib/modules/version-kernel`):

```
# make modules_install
```

3) còpia de la imatge a la seva posició final, suposant i386 com a arquitectura (cal vigilar aquest punt, perquè hi ha detalls en els noms i els directoris que canvien en funció de la distribució GNU/Linux):

```
# cp arch/i386/boot/bzimage /boot/vmlinuz-2.6.xx.img
```

4) i finalment, si cal, creació de la imatge de disc RAM `initrd`, amb les utilitats necessàries segons la versió de la distribució*, i ajustament de les entrades en el *bootloader* LiLo o Grub segons quin fem servir en la nostra distribució.

***Vegeu els comentaris posteriors en aquest apartat i en l'apartat 5.**

Els últims passos (`vmlinuz`, `system.map` i `initrd`) de moviment d'arxius a `/boot` també es poden fer normalment amb el procés (en root):

```
# make install
```

Però aquesta opció fa tot el procés i actualitzarà els *bootloaders* traient antigues configuracions o modificant-les. Poden alterar-se també els enllaços per defecte en el directori `/boot`. Abans d'utilitzar aquesta opció automàtica, seria recomanable un procés de còpia de seguretat de la configuració del *bootloader* (LiLo en `/etc/lilo.conf` o Grub en `/etc/grub/menu.lst`) i possibles enllaços de `/boot` a l'hora de pensar en les configuracions passades que volem guardar.

Respecte a la creació del `initrd`, en Fedora/Red Hat aquest es crearà automàticament amb l'opció `make install`. En Debian haurem de fer servir les tècniques de l'apartat següent o bé crear-lo explícitament amb `mkinitrd` (versions de nucli *leq* 2.6.12) o, posteriorment, amb `mkinitramfs` o una utilitat anomenada `update-initramfs`, especificant la versió del nucli (s'assumeix que aquest s'anomena `vmlinuz-version` dins del directori `/boot`):

```
# update-initramfs -c -k 'version'
```

Una vegada disposem dels fitxers correctes a `/boot` i del *bootloader* actualitzat, ja podem procedir a la rearrencada amb `shutdown -r now`, escollir el nostre nucli i, si tot ha anat bé, podrem comprovar amb `uname -r` que disposem de la nova versió del nucli. També és particularment interessant examinar alguns registres, com `/var/log/messages` i l'ordre `dmesg`, per a examinar el registre de sortida de missatges produïts pel nou nucli en l'arrencada i detectar si ha aparegut algun problema de funcionalitat o amb algun dispositiu concret.

3.4. Compilació del nuci en Debian (*Debian Way*)

En Debian, a més dels mètodes comentats en els subapartats previs, cal afegir la configuració pel mètode denominat *Debian Way*. És un mètode que ens permet construir el nucli d'una manera flexible i ràpida, adaptada a la distribució.

Enllaç d'interès

Per a Fedora recomanem consultar l'enllaç: http://fedoraproject.org/wiki/Building_a_custom_kernel.

Per al procés necessitarem una sèrie d'utilitats (cal instal·lar els paquets o similars): `kernel-package`, `ncurses-dev`, `fakeroot`, `wget` i `bzip2`.

Podem observar el mètode [Debk] des de dues perspectives: reconstruir un nucli equivalent al proporcionat per la distribució com a nucli base (canviant opcions) o bé crear un nucli amb una numeració de versió-revisió personalitzada.

Enllaç d'interès

Es pot veure el procés *Debian Way* de manera detallada a: http://kernel-handbook.alioth.debian.org/.

Pel que fa als paquets de fonts del nucli, Debian proporciona els paquets font usats en la seva distribució, que poden arribar a ser bastant diferents dels de la versió *vanilla* o *pristine* obtinguda de kernel.org. Això és a causa que en Debian produeixen múltiples revisions amb diversos pedaços que van afegint, molts a partir de fallades que es detecten *a posteriori* en les següents versions *vanilla* del nucli.

En les versions estables de Debian, la distribució sol escollir una revisió xx de la branca 2.6, de manera que el nucli 2.6.xx sol quedar-se (generalment) en aquesta numeració per a la versió de Debian estable i, així, quan la distribució s'actualitza amb revisions menors, només s'actualitzen pedaços en revisions menors del nucli (sense canviar el número principal). Quan Debian produeix la versió estable següent se salta a una nova versió del nucli. Durant la durada d'una versió estable de la distribució, Debian sol produir diferents modificacions (*patchlevels*) o revisions del nucli que es va escollir.

Debian ha canviat diverses vegades la gestió dels seus paquets associats als paquets font del nucli. A partir de la versió 2.6.12, és habitual trobar en el dipòsit Debian una versió `linux-source-version` que conté la versió dels fonts del nucli amb els últims pedaços aplicats (vegeu l'apartat 4). Aquesta versió del paquet dels paquets font del nucli és la que farem servir per a crear un nucli personalitzat.

A partir de la versió esmentada (2.6.12), es va incloure en els dipòsits *source* de Debian un nou paquet denominat simplement *linux-2.6*, que inclou els paquets font i utilitats preparades per a generar el nucli en l'esmentada *Debian Way*. Aquest paquet font s'utilitza per a crear els paquets binaris de la distribució associats al nucli i també és l'indicat per a usar en cas de voler aplicar pedaços al nucli actual de la distribució, o per si volem provar modificacions del nucli a nivell de codi.

Examinem primer aquesta opció i després, al final del subapartat, comentarem la personalització.

Per a la compilació del nucli actual procedim utilitzant el segon paquet:

```
# apt-get source linux-2.6
```

En aquest cas ens descarregarà i descomprimirà els paquets font deixant-los en un arbre de directoris a partir del directori `linux-2.6-version`.

Obtenim després algunes eines que necessitarem:

```
# apt-get install build-essential fakeroot
# apt-get build-dep linux-2.6
```

Aquestes línies bàsicament ens instal·len l'entorn de compilació per al nucli (de fet, els paquets del compilador gcc necessaris i les eines pròpies de la *Debian Way*) i finalment es comproven les dependències dels fonts per si calen paquets font addicionals de desenvolupament.

Per a la construcció del binari per a l'arquitectura, segons la configuració preestablerta del paquet (semblant a la inclosa en els paquets oficials del nucli `linux-image` en Debian):

```
$ cd linux-2.6-version
$ fakeroot debian/rules binary
```

Amb això tindrem els paquets binaris de nucli (fitxers `*.deb`) disponibles per a la instal·lació (via `dpkg`).

Existeixen alguns procediments addicionals per a la creació del nucli partint de diferents nivells de pedaç (*patch*) proporcionats per la distribució, i possibilitats de generar diferents configuracions finals (pot veure's la referència de la nota per a complementar aquests aspectes).

Passem ara a l'altra opció que havíem comentat a l'inici, als aspectes de personalització, quan volem canviar certes opcions del nucli i crear-ne una versió personal.

En aquest cas, més habitual, quan volem un nucli personalitzat, haurem de dur a terme un procés semblant mitjançant un pas de personalització típic (per exemple, per mitjà de `make menuconfig`). Els passos són, en primer lloc, obtenir i preparar el directori (aquí obtenim els paquets de la distribució, però és equivalent obtenint els paquets font des de kernel.org):

```
# apt-get install linux-source-2.6.xx
```

```
$ tar -xvjf /usr/src/linux-source-2.6.xx.tar.bz2
$ cd linux-source-2.6.xx
```

on obtenim els paquets font i els descomprimim (la instal·lació del paquet deixa l'arxiu de fonts a `/usr/src`).

A continuació configurem els paràmetres. Com sempre, podem basar-nos en fitxers `.config` que hàgim utilitzat anteriorment, per a partir d'una configuració coneguda (per a la personalització també pot usar-se qualsevol dels altres mètodes, `xconfig`, `gconfig`, etc.):

```
$ make menuconfig
```

A continuació, la construcció final del nucli:

```
$ make clean
$ make KDEB_PKGVERSION=custom.1.0 deb-pkg
```

on creem un identificador per al nucli construït (`custom.1.0`) que s'afegirà al nom del paquet binari del nucli, posteriorment visible en l'arrencada amb l'ordre `uname`.

Així, el procés finalitzarà amb l'obtenció del paquet associat a la imatge del nucli, que podrem finalment instal·lar:

```
# dpkg -i ../linux-image-2.6.xx_custom.1.0_i386.deb
```

Això ens descomprimirà i instal·larà el nucli i generarà una imatge `initrd` addicional si calgués. A més, ens configura el *bootloader* amb el nou nucli per defecte (cal vigilar amb aquest pas, val la pena haver obtingut abans una còpia de seguretat del *bootloader*, per a no perdre cap configuració estable).

Enllaç d'interès

Normalment no caldrà, però si es necessita alguna reconfiguració del `initrd` generat, es recomana llegir l'enllaç següent, en què es comenten diverses eines Debian disponibles: http://kernel-handbook.alioth.debian.org/ch-initramfs.html.

Ara directament amb un `shutdown -r now` podem provar l'arrencada amb el nucli nou.

Afegim, finalmentl, una altra peculiaritat que hem de tenir en compte en Debian, que és l'existència d'utilitats per a afegir mòduls dinàmics de nucli proporcionats per tercers. En particular la utilitat `module-assistant` permet automatitzar tot aquest procés a partir dels paquets font del mòdul.

Necessitem disposar dels *headers* del nucli instal·lat (disponibles en el paquet `linux-headers-version`) o bé dels paquets font que utilitzem en la compilació. A partir d'aquí `module-assistant` pot utilitzar-se interactivament i

seleccionar entre una àmplia llista de mòduls registrats prèviament en l'aplicació, i pot encarregar-se de descarregar el mòdul, compilar-lo i instal·lar-lo en el nucli existent.

També, en l'ús des de l'indicador d'ordres, simplement podem especificar (`m-a` és equivalent a `module-assistant`):

```
# m-a prepare
# m-a auto-install nom_modul
```

que prepara el sistema per a possibles dependències, descarrega fonts del mòdul, compila i, si no hi ha problemes, instal·la per al nucli present. De la llista interactiva de `module-assistant` podem observar el nom del mòdul.

4. Aplicació de pedaços del nucli

En alguns casos també pot ser habitual l'aplicació de pedaços (*patches*) en el nucli [Lkm].

Un fitxer de pedaç (*patch file*) respecte al nucli de Linux és un fitxer de text ASCII que conté les diferències entre el codi font original i el nou codi, amb informació addicional de noms de fitxer i línies de codi. El programa patch (vegeu `man patch`) serveix per a aplicar-lo a l'arbre del codi font del nucli (normalment, en funció de la distribució, a `/usr/src/linux`).

Els pedaços solen ser necessaris quan un maquinari especial necessita alguna modificació en el nucli, s'han detectat alguns errors (*bugs*) posteriors a alguna distribució concreta d'una versió del nucli o es vol afegir una nova prestació sense generar una versió de nucli nova. Per a corregir el problema (o afegir la nova prestació), se sol distribuir un pedaç en lloc d'un nou nucli complet. Quan ja existeixen alguns d'aquests pedaços, s'uneixen amb diverses millores del nucli anterior per a formar-ne una nova versió. En qualsevol cas, si tenim maquinari problemàtic o l'error afecta la funcionalitat o l'estabilitat del sistema i no podem esperar la versió del nucli següent, caldrà aplicar el pedaç o els pedaços.

El pedaç se sol distribuir en un fitxer comprimit tipus bz2 (bunzip2, tot i que també pot trobar-se en gzip amb extensió .gz), com per exemple podria ser:

```
patchxxxx-2.6.xx-pversion.bz2
```

en què xxxx acostuma a ser algun missatge sobre el tipus o la finalitat del pedaç. 2.6.xx seria la versió del nucli a la qual s'aplicarà el pedaç, i `pversion` faria referència a la versió del pedaç, del qual també en poden existir algunes. Cal tenir en compte que estem parlant d'aplicar pedaços als paquets font del nucli (normalment instal·lats, com hem vist, a `/usr/src/linux` o un directori similar de l'usuari usat en la compilació del nucli).

Una vegada disposem del pedaç, haurem d'aplicar-lo. Veurem el procés que s'ha de seguir en algun fitxer `Readme` que acompanya el pedaç, però generalment el procés segueix els passos (una vegada comprovats els requisits previs)

de descomprimir el pedaç en el directori dels fitxers font i aplicar-lo sobre les fonts del nucli, com per exemple:

```
cd /usr/src/linux (o /usr/src/linux-2.6.xx o la versió que sigui).
bunzip2 patch-xxxxx-2.6.xx-version.bz2
patch -p1 < patch-xxxxx-2.6.xx-version
```

També pot aplicar-se prèviament amb l'opció `patch -p1 -dry-run` que només fa un primer test, per a assegurar-nos que no hi hagi cap condició d'error quan se substitueixi el codi. Si no hi ha error, tornem a aplicar sense l'opció de test.

Posteriorment, un cop aplicat el pedaç, haurem de recompilar el nucli per tornar-lo a generar.

Els pedaços poden obtenir-se de diferents llocs. El més normal és trobar-los al lloc d'emmagatzematge dels nuclis *vanilla* (http://www.kernel.org), que en té un arxiu complet. Determinades comunitats Linux (o usuaris individuals) també en solen oferir algunes correccions, però és millor buscar als llocs estàndard per a garantir un mínim de confiança en aquests pedaços i evitar problemes de seguretat amb possibles pedaços "pirates". Una altra via és el fabricant de maquinari, que pot oferir determinades modificacions del nucli (o de controladors en forma de mòduls dinàmics de nucli) perquè funcionin millor els seus dispositius (un exemple conegut és NVIDIA i els seus controladors Linux de propietat per a les seves targetes gràfiques).

Finalment, assenyalarem que moltes distribucions de GNU/Linux (Fedora/Red Hat, Mandriva) ja ofereixen nuclis apedaçats per ells mateixos i sistemes per a actualitzar-los (alguns fins i tot de manera automàtica, com en el cas de Fedora/Red Hat i Debian). Normalment, en sistemes de producció és més recomanable seguir les actualitzacions del fabricant, encara que aquest no oferirà necessàriament el darrer nucli publicat, sinó el que consideri més estable per a la seva distribució, amb l'inconvenient de perdre prestacions d'última generació o alguna novetat en les tècniques incloses en el nucli.

Nucli actualitzat

En sistemes que es vulguin tenir actualitzats, per raons de prova o de necessitat de les últimes prestacions, sempre es pot acudir a http://www.kernel.org i obtenir el nucli més modern publicat.

Un exemple, en el cas de distribució de pedaços, podria ser la distribució Debian per als paquets font de nucli proporcionats per la distribució. Debian produeix revisions del nucli aplicant diferents sèries de pedaços sobre els fonts originals; així, el seu nucli ofert en dipòsits no és l'original, sinó el resultat d'aplicar una sèrie de pedaços. Un exemple de procés per a obtenir un nucli amb diferents pedaços aplicats podria ser el següent (en què xx és la revisió de la branca 2.6 usada per Debian):

```
# apt-get install linux-source-2.6.xx
```

Per a desempaquetar el nucli (el trobem a `/usr/src/`):

```
# apt-get install linux-patch-debian-2.6.xx
# cd linux-source-2.6.xx
# /usr/src/kernel-patches/all/2.6.xx/apply/debian 2.6.xx-1
```

Aquesta última part ens instal·la el conjunt de pedaços que Debian ha generat per a la versió font del nucli utilitzat i, amb la darrera ordre, ens permet tornar a una revisió Debian concreta del nucli (cosa que s'acostuma a denominar un *patchlevel* dels arxius font del nucli). Així, per exemple, si la revisió del nucli és 2.6.xx-2, l'ordre anterior retornaria (procés anomenat *rollback*) els paquets font del nucli a la versió anterior *patchlevel* 2.6.xx-1. Així, podem escollir una versió del nucli segons el subconjunt de pedaços aplicats.

Per acabar, es pot comentar la incorporació d'una tecnologia recent a l'ús de pedaços en Linux, Ksplice, que permet a un sistema Linux afegir pedaços al nucli sense necessitat de parar i rearrencar el sistema. Bàsicament, Ksplice determina, a partir dels paquets font, quins són els canvis introduïts per un pedaç o una sèrie de pedaços, i comprova com afecten la imatge del nucli en memòria que es troba executant-se. S'intenta llavors parar l'execució en el moment en què no existeixin dependències de tasques que necessitin les parts del nucli a apedaçar. A continuació es procedeix a canviar, en el codi objecte del nucli, les funcions afectades, apuntant a les noves funcions amb el pedaç aplicat i modificant dades i estructures de memòria que hagin de reflectir els canvis. Actualment, és un producte comercial, però algunes distribucions de comunitat ja l'estan incloent gràcies al suport gratuït que s'ofereix per a algunes. En els casos de producció en empreses, amb servidors en els quals és important no disminuir el temps de servei, pot ser una tecnologia crítica, altament recomanable tant per a disminuir el temps de pèrdua de servei com per a minimitzar incidents de seguretat que afectin el nucli.

Ksplice

Ksplice és una tecnologia molt útil per a servidors empresarials en producció. Podeu consultar el seu web: http://www.ksplice.com

Bàsicament, ofereixen un servei anomenat *Ksplice Uptrack*, que és una mena d'actualitzador de pedaços per al nucli en execució. La gent de Ksplice segueix el desenvolupament dels pedaços font del nucli, els prepara en forma de paquets que puguin incorporar-se a un nucli en execució i els fa disponibles en aquest servei *uptrack*. Una eina gràfica gestiona aquests paquets i els fa disponibles per a l'actualització durant l'execució.

5. Mòduls del nucli

Flexibilitat del sistema

Els mòduls aporten una flexibilitat important al sistema, i permeten que s'adapti a situacions dinàmiques.

El nucli és capaç de carregar dinàmicament porcions de codi (mòduls) sota demanda [Hen] per a complementar la seva funcionalitat (es disposa d'aquesta possibilitat des de la versió 1.2 del nucli). Per exemple, els mòduls poden afegir suport per a un sistema de fitxers o per a dispositius de maquinari específics. Quan la funcionalitat proporcionada pel mòdul no és necessària, el mòdul pot ser descarregat i així alliberar memòria.

Normalment, sota demanda, el nucli identifica una característica no present en el nucli en aquell moment, contacta amb un fil (*thread*) del nucli anomenat `kmod` (en les versions del nucli 2.0.x el dimoni es deia `kerneld`) i aquest executa una ordre `modprobe` per intentar carregar el mòdul associat a partir d'una cadena amb el nom de mòdul o bé d'un identificador genèric. Aquesta informació en forma d'àlies entre el nom i l'identificador es consulta en el fitxer `/etc/modules.conf`.

A continuació, es busca a `/lib/modules/version-kernel/modules.dep` per a saber si hi ha dependències amb altres mòduls. Finalment, amb l'ordre `insmod` es carrega el mòdul des de `/lib/modules/version_kernel/` (el directori estàndard per als mòduls), la `version-kernel` és la versió del nucli actual i s'utilitza l'ordre `uname -r` per a determinar-la. Per tant, els mòduls en forma binària estan relacionats amb una versió concreta del nucli, i solen collocar-se a `/lib/modules/version-kernel`. Els mòduls es reconeixen com a arxius dins de l'estructura d'aquest directori, amb `.ko` com a extensió d'arxiu.

En general, l'administrador ha de conèixer com es carreguen els mòduls en el sistema. La majoria de vegades, per mitjà del procés anterior, els mòduls de gran part del maquinari i necessitats concretes són detectats automàticament en l'arrencada o per demanda d'ús i carregats en el moment corresponent. En molts casos no haurem de dur a terme cap procés com a administradors. Però en alguns casos, caldrà preparar alguna sintonització del procés o dels paràmetres dels mòduls o, en alguns altres, afegir nous mòduls ja en forma binària o per compilació a partir dels fitxers font.

Si cal compilar alguns mòduls a partir de les seves fonts, s'ha de disposar dels paquets font i/o *headers* de la versió del nucli al qual està destinat.

Hi ha unes quantes utilitats que ens permeten treballar amb mòduls (solien aparèixer en un paquet de programari anomenat `modutils`, que es va reemplaçar per `module-init-tools` per a la gestió de mòduls de la branca 2.6.x):

- `lsmod`: podem veure els mòduls carregats en el nucli (la informació s'obté del pseudofitxer `/proc/modules`). Es crea la llista dels noms, de les dependències amb d'altres (entre claudàtors [ ]), de la mida del mòdul en bytes i del comptador d'ús del mòdul; això permet descarregar-lo si el recompte és zero.

Ejemplo

Alguns mòduls en un Debian:

```
Module                  Size  Used by    Tainted: P
agpgart                37344   3   (autoclean)
apm                    10024   1   (autoclean)
parport_pc             23304   1   (autoclean)
lp                      6816   0   (autoclean)
parport                25992   1   (autoclean) [parport_pc lp]
snd                    30884   0
af_packet              13448   1   (autoclean)
nvidia               1539872  10
es1371                 27116   1
soundcore               3972   4   [snd es1371]
ac97_codec             10964   0   [es1371]
gameport                1676   0   [es1371]
3c59x                  26960   1
```

- `modprobe`: intenta la càrrega manual d'un mòdul i de les seves dependències.
- `insmod`: carrega un mòdul determinat.
- `depmod`: analitza dependències entre mòduls i crea un fitxer de dependències.
- `rmmod`: treu un mòdul del nucli.
- `depmod`: es fa servir per a generar el fitxer de dependències dels mòduls, que es troba a `/lib/modules/version-kernel/modules.dep` i que inclou les dependències de tots els mòduls del sistema. Si s'instal·len nous mòduls de nucli, és interessant executar manualment aquesta ordre per a actualitzar les dependències. També se solen generar automàticament en engegar el sistema.
- Es poden utilitzar altres ordres per a depurar o analitzar els mòduls, com per exemple `modinfo`, que crea una llista d'algunes informacions associades al mòdul (com ara llicències, descripció, ús i dependències), o de fitxers `/proc/kallsyms`, que ens permeten examinar els símbols exportats pels mòduls.

Normalment per a la càrrega, o bé el mateix nucli o bé l'usuari, a mà, especificarà amb `insmod` el nom del mòdul i, opcionalment, determinats paràmetres; per exemple, en el cas de dispositius sol ser habitual especificar les adreces dels ports d'E/S o bé els recursos d'IRQ o DMA. Per exemple:

```
insmod soundx io=0x320 irq=5
```

La càrrega general de mòduls, en funció del moment i la manera, pot fer-se manualment, com hem comentat, mitjançant `initrd/initramfs` o per mitjà de `udev`.

En el cas de `initrd/initramfs`, quan el sistema arrenca, es necessiten immediatament alguns mòduls per a accedir al dispositiu i al sistema de fitxers arrel del sistema (per exemple, controladors específics de disc o tipus de sistemes de fitxers). Aquests mòduls necessaris es carreguen a la RAM mitjançant un sistema de fitxers especial anomenat `initrd/initramfs`. En funció de la distribució GNU/Linux, s'utilitzen aquests termes de manera indiferent, tot i que en alguns casos s'han produït canvis al llarg de la vida de la distribució. Per convenció, aquest element s'acostuma a anomenar *filesystem* RAM inicial, i se'l referencia com a `initramfs`.

El sistema inicial de fitxers en RAM, `initramfs`, és carregat pel *bootloader* en l'especificació de l'entrada pertinent a la càrrega del nucli corresponent (per exemple en la línia/opció `initrd` de l'entrada corresponent de Grub).

En la majoria de distribucions, amb el nucli distribuït originalment o bé amb la nostra configuració del nucli, acostuma a crear-se un `initramfs` inicial. En qualsevol cas, si depenent de la distribució no es produeix (pot no ser necessari), podem crear-lo i sintonitzar-lo manualment. L'ordre `mkinitramfs` permet crear-lo a partir de les seves opcions genèriques, que no s'acostumen a canviar i que es poden configurar a `/etc/initramfs-tools/initramfs.conf` i, específicament, els mòduls que es carregaran a l'inici automàticament, que podem trobar a */etc/initramfs-tools/modules*.

Un `mkinitramfs -o new_initrd_file` ens permetrà crear-lo, i normalment podem procedir a copiar-lo en el directori `/boot` per a fer-lo accessible al *bootloader* utilitzat. Per exemple, mitjançant un canvi en el fitxer de configuració `/boot/grub/menu.lst` de Grub, amb la modificació de la línia de `initrd` oportuna. En qualsevol cas, sempre és interessant establir al *bootloader* una configuració alternativa de text durant aquestes proves, per a poder reiniciar i provar la nova configuració però, alhora, mantenir la configuració estable antiga.

Durant el procés d'arrencada, a més del `initramfs` necessari per a l'arrencada inicial, es carregarà la resta de mòduls per detecció automàtica. Si no es carrega algun mòdul, sempre pot forçar-se'n la càrrega en incloure'n el nom implícitament en el fitxer de configuració `/etc/modules`.

També es pot donar o voler el cas contrari: evitar la càrrega d'un mòdul que es pot detectar erròniament o per al qual existeix més d'una alternativa possible. En aquest cas s'utilitzen tècniques de llistes negres de mòduls (típicament la llista negra es desa a `/etc/modprobe.d/blacklist.conf`).

5.1. DKMS: mòduls recompilats dinàmicament

Pel que fa als mòduls dinàmics, un problema clàssic ha estat la recompilació amb noves versions del nucli. Els mòduls dinàmics de tercers, no inclosos *a*

priori en el nucli, necessiten codi font per a compilar-se, proporcionat per la comunitat o pel fabricant del maquinari del dispositiu.

Durant la compilació, normalment cal disposar dels paquets de desenvolupament del nucli, dels paquets del codi font del nucli i dels seus *headers* de desenvolupament, amb la mateixa numeració que el nucli utilitzat per al qual es vol compilar el mòdul. Aquest fet obliga a una recompilació constant amb el canvi de versions del nucli del sistema, en especial ara que les distribucions distribueixen revisions del nucli en un temps més breu, o bé per a resoldre problemes potencials de seguretat o bé per a corregir errors detectats.

Algunes distribucions, per a minimitzar aquesta problemàtica, distribueixen un entorn anomenat DKMS, que permet facilitar la recompilació automàtica d'un mòdul amb els canvis de versió del nucli. Això normalment es produeix en l'arrencada en detectar un número de nucli: tots els mòduls de tercers registrats pel sistema DKMS es recompilen a partir dels arxius font dels mòduls i dels arxius font o *headers* del nucli nou. D'aquesta manera, el procés total és transparent a l'usuari. Una vegada fet aquest procés, el sistema o l'usuari, mitjançant ordres de manipulació de mòduls (com `modprobe`), poden utilitzar directament el nou mòdul.

Algunes distribucions ofereixen només el paquet base (dkms) del sistema, en algunes es proporcionen paquets dkms preparats per a mòduls concrets o, fins i tot, el fabricant pot oferir el seu mòdul amb suport dkms.

El procés habitualment passa per les etapes següents:

1) Obtenir els arxius font del mòdul (un paquet o un arxiu TGZ acostuma a ser el més normal).

2) Instal·lar els diversos paquets necessaris per al procés: `kernel-source`, `kernel-headers`, `dkms` (els noms depenen de la distribució i, en el cas dels paquets font del nucli, cal tenir en compte que siguin les versions associades a la versió del nucli actual per al qual es vol instal·lar el mòdul).

3) Activar el servei DKMS en arrencada. Habitualment, el servei s'anomena `dkms_autoinstaller`.

4) Crear un directori a `/usr/src` per als paquets font del mòdul i col·locar-los-hi.

5) Crear un fitxer `dkms.conf` en el directori anterior, que especifica com construir (compilar) i instal·lar el mòdul.

6) Afegir el servei a la base de dades DKMS, compilar-lo i instal·lar, normalment amb unes ordres:

```
dkms add -m nom-modul -v numero-versio-modul
dkms build   -m nom-modul -v numero-versio-modul
dkms install -m nom-modul -v numero-versio-modul
```

Respecte al fitxer `dkms.conf` esmentat, podria ser com segueix (en aquest cas `nom-modul` és el nom del mòdul i `versio`, el codi numèric de la versió):

```
#
# /usr/src/nom-modul/dkms.conf
#

PACKAGE_NAME="nom-modul"
PACKAGE_VERSION="versio"
CLEAN="make clean"
MAKE[0]="make module"
BUILD_MODULE_NAME[0]="nom-modul"
DEST_MODULE_LOCATION[0]="/kernel/drivers/video"
AUTOINSTALL="yes"

# End Of File
```

En aquest cas, tenim localitzats els paquets font del mòdul en un directori `/usr/src/nombre-modulo` en què posem aquest fitxer `dkms.conf`. L'opció `MAKE` dóna els passos per a compilar i construir el mòdul, prèviament netejat amb `CLEAN`. `BUILD_MODULE_NAME` estableix el nom del mòdul construït (cal anar amb compte en aquest punt perquè depèn del sistema de compilació i pot coincidir amb el nom del mòdul general o no; alguns paquets font permeten construir diversos controladors/mòduls diferents, amb diferent denominació).

`DEST_MODULE_LOCATION` defineix on s'instal·larà el mòdul final en l'arbre associat al nucli; en aquest cas suposem que és un controlador de vídeo (recordeu que l'arrel és a `/lib/modules/version-kernel`, el que es col·loca aquí és a partir d'aquesta arrel). `AUTOINSTALL` permet que es reconstrueixi automàticament el mòdul durant canvis del nucli actual.

En els casos de

`CLEAN`, `MAKE`, `BUILD_MODULE_NAME` y `DEST_MODULE_LOCATION`

es recomana consultar el fitxer d'explicació (normalment amb nom `README` o `INSTALL`) que acompanya els paquets font dels mòduls, ja que poden caldre ordres addicionals, o haver de modificar-los perquè es compili i s'instal·li correctament el mòdul.

6. Virtualització en el nucli

Una de les àrees en expansió, en l'administració d'IT, és la virtualització de sistemes. Amb el temps, GNU/Linux ha anat incorporant diferents possibilitats provinents tant de solucions comercials, com de diferents projectes de codi obert.

La virtualització de sistemes és un recurs bàsic actual en les empreses i organitzacions per a millorar-ne l'administració de sistemes, disminuir costos i aprofitar els recursos de maquinari de manera més eficient.

En general, quan en el passat necessitàvem diverses instàncies d'un (o més) sistemes operatius, havíem d'adquirir un servidor per a cada instància que volguéssim implantar. El corrent actual és comprar servidors molt més potents i utilitzar la virtualització en aquests servidors per a implantar els diferents sistemes en desenvolupament o producció.

Normalment, en virtualització disposem d'un sistema operatiu instal·lat (que habitualment s'anomena sistema amfitrió o *host*) i una sèrie de màquines virtuals sobre aquest sistema (anomenades sistemes hoste o *guest*). Tot i així, també hi ha solucions que substitueixen el sistema operatiu amfitrió per una capa anomenada *hypervisor*.

La virtualització com a solució ens permet optimitzar l'ús dels nostres servidors o, per exemple en el cas d'escriptori, disposar de màquines de prova d'altres sistemes operatius convivint alhora en la mateixa màquina. En el cas de GNU/Linux disposem de múltiples solucions que permeten tant un cas com l'altre. També podem disposar de GNU/Linux com a sistema amfitrió que allotja màquines virtuals o bé utilitzar-lo com a màquina virtual sobre un altre sistema diferent o bé sobre un altre sistema amfitrió també GNU/Linux. Un esquema, aquest darrer, particularment útil en el cas d'administració, perquè ens permetrà, per exemple, examinar i executar diferents distribucions GNU/Linux sobre un mateix sistema amfitrió base.

Hi ha moltes solucions de virtualització, però per esmentar-ne algunes de les més populars en sistemes GNU/Linux, disposem de (les ordenem de més a menys en relació directa amb el nucli):

- KVM
- Xen
- OpenVZ
- VirtualBox
- VMware

VMware és un dels líders comercials en solucions de virtualització, i disposa de productes de virtualització per a escriptori (VMware Workstation), mentre que per a servidor disposa d'un producte VMware Server que està disponible per a Linux i es pot descarregar gratuïtament. El cas de servidor permet gestionar diverses màquines virtuals amb una interfície de gestió simple. Una altra línia de producte, VMware ESX, implementa necessitats més grans per a centres de dades (*data centers*), amb gestió elaborada de màquines, tolerància a errors, migracions i altres necessitats explícites per a centres de dades.

Enllaç d'interès

Podeu visitar el web de VMware a: http://www.vmware.com

Sun/Oracle VirtualBox ofereix virtualització orientada a escriptori, que ens permet una opció bastant senzilla per a provar màquines amb diferents sistemes operatius. Disposa de versió de codi lliure utilitzable en gran quantitat de distribucions GNU/Linux.

OpenVZ és una solució de virtualització que utilitza el concepte de contenidor de màquina virtual. Així, l'amfitrió arrenca amb un nucli comú a les màquines virtuals (hi ha paquets d'imatges de nucli amb suport OpenVZ integrat en les distribucions, com per exemple en Debian), que permet arrencar màquines amb el nucli en comú però cada una dins d'un entorn aïllat de la resta. OpenVZ només permet màquines virtuals hoste Linux (a causa del nucli compartit).

Xen usa el concepte d'*hypervisor*, utilitzant un tipus de virtualització denominada *paravirtualització*, en què s'elimina el concepte d'amfitrió hoste (*host-guest*) i es delega a la capa d'*hypervisor* la gestió dels recursos físics de la màquina, de manera que permeti a les màquines virtuals el màxim accés als recursos de maquinari. En aquests casos es necessiten nuclis sintonitzats que puguin beneficiar-se de les possibilitats de la paravirtualització. En el cas de GNU/Linux, en la majoria de distribucions s'ofereixen nuclis optimitzats per a Xen (vegeu el cas de Debian, per al qual existeixen imatges binàries dels nuclis per a Xen). En general, la paravirtualització i la capa d'*hypervisor* per a accedir al maquinari aprofiten les facilitats de les CPU actuals, amb recursos de maquinari dedicats a facilitar la virtualització. Si es disposa de les característiques, es poden usar sistemes com Xen, si no, es pot utilitzar un sistema més clàssic de d'amfitrió hoste, com per exemple VirtualBox.

En general, per a veure si la CPU té suport de virtualització, cal examinar-ne dades a `/proc/cpuinfo`, en concret el *flag* `VMX`, per a processadors Intel, que es pot veure en la secció `flags`:

```
$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 23
model name      : Intel(R) Xeon(R) CPU E5405  @ 2.00GHz
stepping        : 10
```

```
cpu MHz         : 1994.999
cache size      : 6144 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 4
apicid          : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep
mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2
ss ht tm syscall   nx lm constant_tsc pni monitor ds_cpl
vmx tm2 ssse3 cx16 xtpr sse4_1 lahf_lm
bogomips        : 3989.99
clflush size    : 64
cache_alignment : 64
address sizes   : 38 bits physical, 48 bits virtual
power management:
```

6.1. KVM

Finalment, la solució en la qual ens centrarem en aquest subapartat, **KVM**, és present des del nucli 2.6.20, com a solució inclosa per a la virtualització de sistemes. És una solució semblant a Xen, però amb diferències quant a la implementació. Xen és un producte complex, amb diferents capes i, en especial, el seu disseny de capa hipervisora. En canvi, KVM s'implementa com un mòdul del nucli existent, que es complementa amb altres solucions. És l'opció per defecte per a la virtualització en distribucions com Debian i Fedora.

Enllaç d'interès

Podeu accedir al web de KVM a: http://www.linux-kvm.org

Normalment, una solució de virtualització basada en KVM es compon d'una barreja del següent:

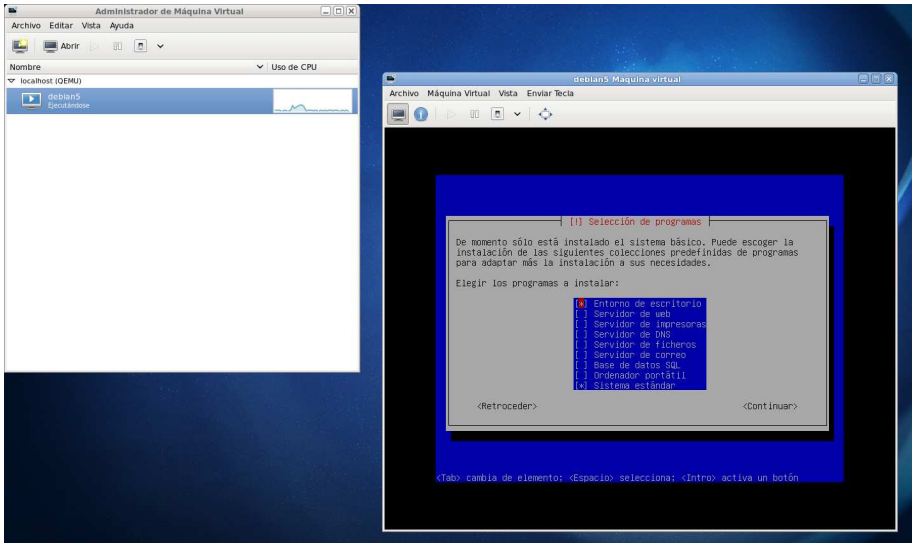
- El mòdul de nucli `kvm.ko`, que proporciona la infraestructura base de virtualització, i, a més, un mòdul addicional segons el model de processador, `kvm-intel.ko` o `kvm-amd.ko`. Aquests mòduls s'hauran de carregar manualment (`modprobe`) o habilitar-los durant l'arrencada per a disposar de suport de virtualització KVM.

- Qemu, un simulador creuat d'arquitectures de CPU que ens permet simular una CPU virtual sobre una altra d'arquitectura real igual o diferent. KVM utilitza una versió modificada de Qemu per a la gestió i la creació de les màquines hoste (aquest paquet sol aparèixer com a `qemu-kvm` en les distribucions).

- La biblioteca `libvirt`, que és una API que permet una gestió de la virtualització independent del sistema de virtualització utilitzat (Xen, KVM o d'altres).

- Utilitats basades en `libvirt` per a la creació de les VM hoste i el seu manteniment, com `virt-manager`, una interfície gràfica de gestió de màquines virtuals (figura 4); `virt-install`, una utilitat de línia per a la gestió de màquines virtuals, o `virtsh`, un intèrpret d'ordres (*shell*) basat en ordres de gestió de les màquines virtuals. Aquestes utilitats solen necessitar un servei de sistema, anomenat `libvirtd`. Hem d'assegurar-nos de posar en marxa el servei (`/etc/init.d/libvirtd start`) o bé de carregar-lo a l'inici.

Figura 4. `Virt-manager` durant la instal·lació d'una màquina virtual



A continuació, veurem els processos bàsics associats a la utilització de KVM i descriurem algunes de les fases de la instal·lació de KVM i la posada en marxa d'algunes màquines virtuals.

Per començar, hem d'examinar si disposem de suport de maquinari a la CPU: com hem esmentat, busquem el *flag* `vmx` a `/proc/cpuinfo` (per a processadors Intel) o el *flag* `svm` (per a processadors AMD). Per exemple, amb (substituïu `vmx` per `svm` en AMD):

Suport de virtualització

Cal anar amb compte amb el fet que moltes de les màquines actuals permeten desactivar/activar a la BIOS el suport de virtualització; comproveu primer que no estigui desactivat.

```
grep vmx /proc/cpuinfo
```

obtindrem com a resultat la línia de *flags* (si existeix el *flag* `vmx`, si no, cap resultat). També podem obtenir diverses línies en CPU *multicore* i/o Intel amb *hyperthreading*, on obtenim una línia de *flags* per cada element de còmput (CPU amb HT o múltiples nuclis amb HT o sense).

Una vegada determinat el suport correcte de virtualització, instal·lem els paquets associats a KVM; aquests ja dependran de la distribució, però en general,

el mateix "kvm" ja obtindrà la majoria de paquets requerits com a dependències. En general, els paquets recomanats per a instal·lar (via apt o yum) són `kvm, qemu-kvm, libvirt`. Depenent de la distribució, poden canviar alguns noms de paquets i, en especial, amb el nom `virt` hi ha diversos paquets d'utilitats de generació i monitoratge de màquines virtuals, tant de KVM com de Xen.

Si es permet als usuaris del sistema l'ús de màquines virtuals, cal afegir els noms d'usuari a grups específics (via ordres `adduser` o `usermod`), normalment als grups `kvm` o `libvirt` (depenent de la distribució, Fedora o Debian, i de la versió de KVM).

El pas següent (opcional) és facilitar l'ús de xarxa a les màquines virtuals. Per defecte, KVM utilitza NAT, i dóna adreces IP privades de tipus 10.0.2.x, i hi accedeix mitjançant la xarxa de la màquina amfitrió. En un altre cas, si volem una configuració diferent (per exemple que permeti accés extern a les màquines virtuals) haurem de permetre que la xarxa actual faci de *bridge*; en aquest cas és necessari instal·lar el paquet `bridge-utils` i configurar un dispositiu especial de xarxa denominat `br0`: en Debian en la configuració de xarxa que es troba a `/etc/network/interface` i en Fedora pot crear-se un fitxer associat al dispositiu com ara `/etc/sysconfig/network-scripts/ifcfg-br0`. Per exemple, en Debian podria col·locar-se un exemple de configuració com el següent:

Enllaç d'interès

La gestió de xarxa és un dels temes complexos en virtualització; en el cas de KVM es recomana examinar: http://www.linux-kvm.org/page/Networking.

```
auto lo
iface lo inet loopback

auto br0
iface br0 inet static
       address 192.168.0.100
       network 192.168.0.0
       netmask 255.255.255.0
       broadcast 192.168.0.255
       gateway 192.168.0.1
       bridge_ports eth0
       bridge_fd 9
       bridge_hello 2
       bridge_maxage 12
       bridge_stp off
```

Aquesta configuració permet que es creï un dispositiu `br0` per a reemplaçar `eth0`. Així, les targetes de xarxa virtuals redirigiran el trànsit assignades a aquest dispositiu. `bridge_ports` especifica quin serà el dispositiu físic real que s'utilitzarà.

Tal com hem comentat, aquesta part de configuració de xarxa és opcional, i només té sentit si volem accedir des de l'exterior a les nostres màquines

virtuals. Ben al contrari, en un entorn de virtualització d'escriptori pot haver-hi prou amb el mode NAT per defecte, ja que les màquines disposaran de sortida de xarxa a través de la xarxa de l'amfitrió.

A partir d'aquestes configuracions, ja estarem capacitats per a crear les imatges de les màquines virtuals. Hi ha diferents conjunts d'ordres per a fer-ho, fent servir kvm directament (ordre `kvm`), utilitats associades a qemu per a la creació d'aquestes imatges (`qemu-img`) o utilitats associades a libvirt (`virt-install`). El procés passa per crear la imatge de disc (o espai de disc) associat a la màquina hoste com un fitxer i, després, fer la instal·lació del sistema operatiu en aquest fitxer (imatge de disc), des del CD/DVD d'instal·lació del sistema operatiu o també des d'una imatge `*.iso` del CD/DVD.

Per exemple, suposem una instal·lació d'un hoste determinat (per exemple, disposem d'uns CD/DVD de la distribució Debian).

Creem l'espai de disc per a la màquina virtual (en aquest cas 8 GB):

```
# dd if=/dev/zero of=~/debianVM.img bs=1M count=8192
```

Mitjançant l'ordre `dd`, creem un fitxer de sortida de 8.192 blocs d'1 MB, és a dir, 8 GB, que ens formarà la unitat de disc per a la nostra màquina virtual (també existeix una ordre alternativa per a crear imatges, `qemu-img`, vegeu pàgina man).

Després, cal obtenir el mitjà d'instal·lació del sistema operatiu a instal·lar, proporcionat com a CD/DVD i, per tant, accessible en el dispositiu `/dev/cdrom` (o equivalent si tenim més unitats de CD/DVD) o, en canvi, a partir d'una imatge `iso` del suport. En qualsevol cas, aquest darrer sempre el podem obtenir a partir dels discos amb:

```
dd if=/dev/cdrom of=debian-install.iso
```

que ens genera la imatge del CD/DVD.

Ara que disposem de la imatge binària i el mitjà d'instal·lació del sistema operatiu, podem instal·lar-lo, amb diferents utilitats. En general, acostuma a utilitzar-se `virt-install` com a ordre per a crear la VM, però també hi ha la possibilitat d'usar el `qemu-kvm` esmentat, directament com a opció més simple, que sol aparèixer (depenent de la distribució) com a ordre `qemu-kvm`, `qemu-system-x86_64` o simplement com a `kvm`:

```
kvm -m 512 -cdrom debian-install.iso -boot d -hda debianVM.img
```

En aquest cas, crearia una màquina virtual bàsica de 512 MB de memòria principal, fent servir el nostre disc de 8 GB creat prèviament i arrencant la

màquina a partir de la nostra imatge `iso` del mitjà d'instal·lació del sistema operatiu. Es pot substituir el fitxer `iso` per `/dev/cdrom` si utilitzem els discos d'instal·lació.

L'altra possible alternativa de creació es fa mitjançant la utilitat de l'indicador d'ordres `virt-install`, molt més completa, però amb la dificultat d'haver d'especificar molts més paràmetres. Per exemple, podríem indicar la creació de la màquina anterior mitjançant:

```
virt-install --connect qemu:///system -n debian5 -r 512 \
--vcpus=2 -f debianVM.img -s 8 -c debianinstall.iso --vnc \
--noautoconsole --os-type linux --os-variant debianLenny
```

que, entre altres paràmetres, col·loca el mètode de connexió a la màquina virtual, el nom de la VM `debian5`, defineix 512 MB de memòria, fa disponibles dos nuclis de la màquina física, utilitza el disc virtual `debianVM`, de 8 GB (`-s` permet que si no existeix, el creï prèviament amb aquesta grandària de GB), utilitza la `iso` com a mitjà d'instal·lació i permetrà connexions gràfiques amb vnc amb la màquina virtual. A més, definim que el sistema que s'instal·larà és Linux, en especial una versió de Debian. Els paràmetres de tipus i variant del sistema operatiu són altament dependents de la versió de `virt-install`, de manera que val la pena consultar la pàgina man (`man virt-install`) i la llista de sistemes operatius compatibles amb la versió KVM del nucli i el conjunt d'utilitats `qemu` i `libvirt`.

Enllaç d'interès

La llista de compatibilitat de KVM es pot trobar a: http://www.linux-kvm.org/page/Guest_Support_Status

En aquest punt només hem creat la màquina, però no hi estem connectats, i n'hem configurat de manera molt bàsica els paràmetres. Amb altres utilitats, per exemple les basades en `libvirt`, com la interfície gràfica `virt-manager`, podem personalitzar més la VM creada, per exemple afegint o traient maquinari virtual al sistema.

Podem connectar després amb la màquina acabada de crear mitjançant:

```
virt-viewer -c qemu:///system nombreVM
```

si és al mateix sistema o, si som en una màquina remota, amb:

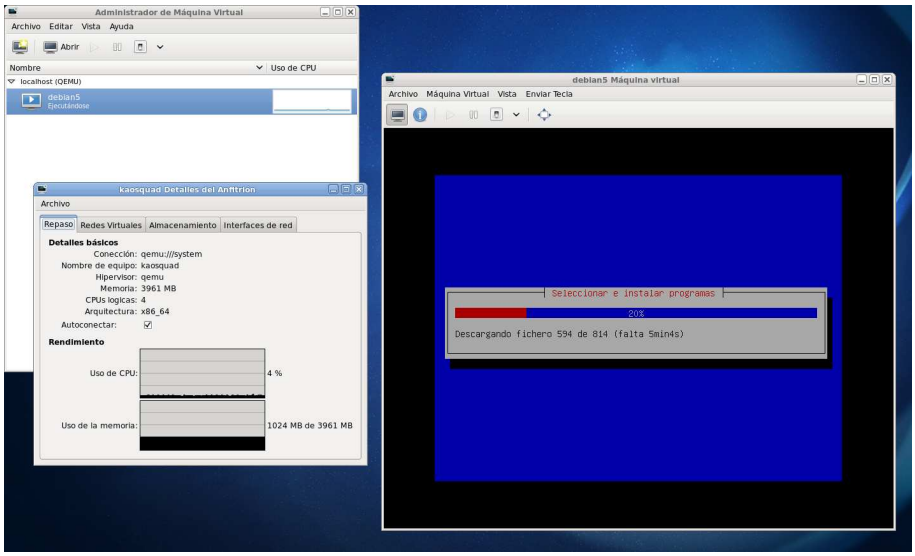
```
virt-viewer -c qemu+ssh://ip/system nombreVM
```

o utilitzant directament la interfície `virt-manager` (figura 5).

Això ens permet connectar gràficament amb la màquina virtual creada i, per exemple, continuar amb la instal·lació del sistema operatiu (s'haurà iniciat

amb l'arrencada anterior amb `virt-install` o `kvm`). Una vegada instal·lat el sistema, ja podem usar qualsevol sistema, tant si és gràfic (`virt-manager`) com d'indicador d'ordres (`virtsh`), per a gestionar les màquines hoste virtuals i poder arrencar-les i parar-les.

Figura 5. `Virt-manager` connectat a la màquina virtual durant el procés d'instal·lació, observant els recursos utilitzats per la màquina virtual

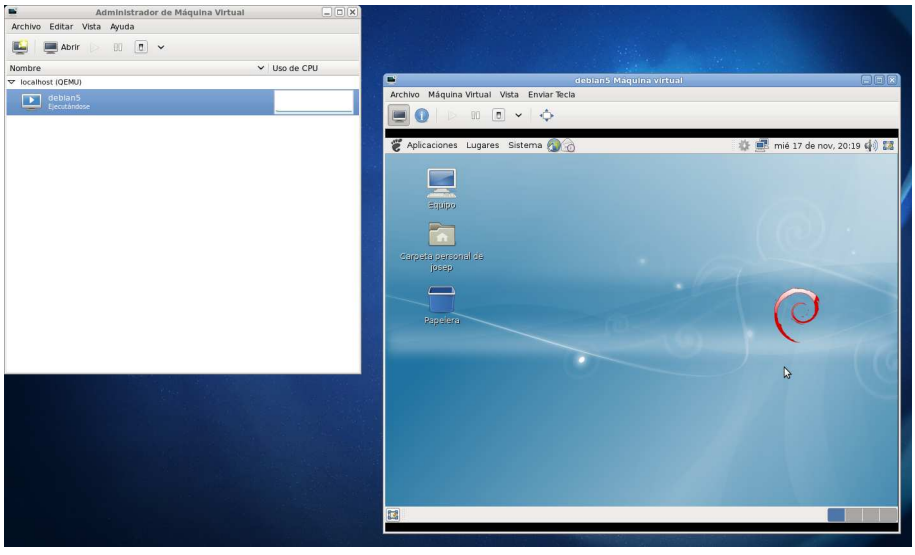


Per exemple, amb `virtsh`:

```
virtsh --connect qemu:///system
```

connectem amb l'intèrpret d'ordres. Ordres com `list` ens donen les màquines actives, `list -all`, totes les màquines disponibles, i altres com `start`, `shutdown`, `destroy`, `suspend` o `resume` ens donen diferents possibilitats de gestió de cada màquina virtual.

Figura 6. `Virt-manager` connectat a la màquina virtual acabada d'instal·lar



7. Futur del nucli i alternatives

Els avenços en el nucli de Linux en determinats moments van ser molt ràpids, però actualment, ja amb una situació bastant estable amb els nuclis de la branca 2.6.x, cada vegada passa més temps entre les versions que van apareixent. En certa manera això és força positiu: permet tenir temps per a corregir errors comesos, veure quines idees no han funcionat bé i provar-ne de noves, que, si tenen èxit, s'inclouen.

El nucli, en evolució

El nucli continua evolucionant, incorporant les últimes novetats en suport de maquinari i millores en les prestacions.

Comentarem en aquest apartat algunes de les idees dels últims nuclis i algunes que estan previstes, per a donar indicacions del que serà el futur pròxim en el desenvolupament del nucli.

En l'antiga branca 2.4.x del nucli [Ces06] es van fer algunes aportacions:

- Compliment dels estàndards IEEE POSIX, fet que permet que molts dels programes existents d'UNIX puguin recompilar-se i executar-se en Linux.
- Millor suport de dispositius: PnP, USB, port paral·lel, SCSI, etc.
- Suport per a nous sistemes de fitxers, com UDF (CD-ROM reescrivibles com un disc). Altres sistemes amb *journal*, com els Reiser d'IBM o l'ext3, que permeten tenir un registre (*journal*) de les modificacions dels sistemes de fitxers i, així, poder recuperar-se d'errors o tractaments incorrectes dels fitxers.
- Suport de memòria fins a 4 GB. Al seu dia van sorgir alguns problemes (amb nuclis 1.2.x) que no suportaven més de 128 MB de memòria (una quantitat que en aquell temps era molta memòria).
- Millora de la interfície `/proc`. Es tracta d'un pseudosistema de fitxers (el directori `/proc`) que no existeix realment en disc, sinó que és simplement una manera d'accedir a dades del nucli i del maquinari d'una manera organitzada.
- Suport del so en el nucli: es van afegir parcialment els controladors Alsa que abans es configuraven separadament.
- Es va incloure suport preliminar per al RAID programari i el gestor de volums dinàmics LVM1.

En la sèrie actual, la branca del nucli 2.6.x [Ces06, Pra03], es va disposar d'importants avenços respecte a l'anterior (amb les diferents revisions .x de la branca 2.6):

- Millors prestacions en SMP, important per a sistemes multiprocessador molt utilitzats en entorns empresarials i científics.

- Millores en el planificador de CPU (*scheduler*). En particular, s'introdueixen avenços per a millorar l'ús de tasques interactives d'usuari, imprescindibles per a millorar l'ús de Linux en un ambient d'escriptori.

- Millores en el suport *multithread* per a les aplicacions d'usuari. S'incorporen nous models de fils NGPT (IBM) i NPTL (Red Hat) (amb el temps es va consolidar finalment la NPTL).

- Suport per a USB 2.0 i, posteriorment, per a USB 3.0.

- Controladors Alsa de so incorporats en el nucli.

- Noves arquitectures de CPU de 64 bits: se suporten AMD x86_64 (també coneguda com a amd64) i PowerPC 64 i IA64 (arquitectura dels Intel Itanium).

- Sistemes de fitxers amb *journal*: JFS, JFS2 (IBM) i XFS (Silicon Graphics).

- Millores amb *journal* en els sistemes de fitxers propis, `ext3` i `ext4`, amb millores de la mida màxima dels arxius i de rendiment general.

- Millores de prestacions d'entrada/sortida i nous models de controladors unificats.

- Millores en la implementació de TCP/IP i el sistema NFSv4 (compartició de sistema de fitxers per xarxa amb altres sistemes).

- Millores significatives per a nucli preemptiu: permet que internament el nucli gestioni diverses tasques que es poden interrompre entre elles, característica imprescindible per a implementar eficaçment sistemes de temps real i també per a augmentar el rendiment de tasques interactives.

- Suspensió del sistema i restauració després de reiniciar (per nucli).

- UML, User Mode Linux, una mena de màquina virtual de Linux sobre Linux que permet veure un Linux (en mode usuari) executant-se sobre una màquina virtual. Això és ideal per a la depuració, ja que es pot desenvolupar i provar una versió de Linux sobre un altre sistema, i és útil tant per al mateix desenvolupament del nucli com per a una anàlisi de seguretat. En versions posteriors aquest concepte va evolucionar cap al mòdul KVM.

- Tècniques de virtualització incloses en el nucli: en les distribucions s'han anat incorporant diferents tècniques de virtualització, que necessiten extensions en el nucli. Podem destacar, per exemple, nuclis modificats per a Xen, Virtual Server (Vserver), OpenVZ o el mateix mòdul KVM.

- Nova versió del suport de volums LVM2.

- Nou pseudosistema de fitxers `/sys`, destinat a incloure la informació del sistema, i dispositius que s'aniran migrant des del sistema `/proc`, de manera que deixin aquest darrer amb informació relacionada amb els processos i el seu desenvolupament en execució, i també la informació dinàmica del mateix nucli.

- Mòdul FUSE per a implementar sistemes de fitxers en espai d'usuari (es fa servir en especial per al cas d'NTFS).

Per a conèixer els canvis de les versions més recents de Linux, poden examinar-se els fitxers `ChangeLog` que acompanyen cada versió del nucli, o consultar un registre històric que es conserva a *Kernelnewbies.org*, en especial a l'adreça http://kernelnewbies.org/LinuxChanges, que manté els canvis de l'última versió i poden consultar-se els de la resta de versions (mateixa adreça amb `/Linux26Changes`).

En el futur es té pensat millorar els aspectes següents:

- Increment de la tecnologia de virtualització en el nucli, per a suportar diferents configuracions de sistemes operatius i diferents tecnologies de virtualització, i també disposar d'un millor suport del maquinari per a virtualització inclòs en els processadors que sorgeixin en les noves arquitectures. Les arquitectures x86 i x86_64 estan bastant suportades, amb KVM, per exemple, però d'altres no ho estan o només parcialment.

- El suport d'SMP (màquines multiprocessador), de CPU de 64 bits (Xeon, nous multicore d'Intel i Opteron d'AMD), el suport de CPU *multicore* i l'escalabilitat d'aplicacions multifil en aquestes CPU.

- La millora de sistemes de fitxers per a clusterització i grans sistemes distribuïts.

- En canvi, la millora en nuclis més optimitzats per a dispositius mòbils (PDA, *smartphones*, *tablets*, etc.).

- Millora en el compliment dels estàndard POSIX.

Enllaç d'interès

Per saber-ne més sobre POSIX podeu visitar el web: http://www.unix.org/

- Millores en la planificació de la CPU. Tot i que en la sèrie inicial de la branca 2.6.x es van fer molts avenços en aquest aspecte, encara hi ha un rendiment baix en algunes situacions. En particular, en l'ús d'aplicacions

interactives d'escriptori s'estan estudiant diferents alternatives, per a millorar aquest i altres aspectes relacionats amb l'escriptori i l'ús del rendiment gràfic.

També, malgrat que s'aparta dels sistemes Linux, la Free Software Foundation (FSF) i el seu projecte GNU continuen treballant en el projecte d'acabar un sistema operatiu complet. Cal recordar que el projecte GNU tenia com a principal objectiu aconseguir un clon UNIX de programari lliure, i les utilitats GNU només són el programari de sistema necessari. A partir de 1991, quan Linus aconsegueix conjuntar-ne el nucli amb algunes utilitats GNU, es va fer un primer pas que ha acabat en els sistemes GNU/Linux actuals. Però el projecte GNU continua treballant en la idea d'acabar el sistema complet. En aquest moment disposen ja d'un nucli en el qual poden córrer les utilitats GNU. Aquest nucli s'anomena Hurd, i un sistema construït amb aquest nucli es coneix com a GNU/Hurd. Ja existeixen algunes distribucions de prova, en concret, una Debian GNU/Hurd.

Enllaç d'interès

Per saber-ne més sobre el projecte GNU podeu visitar el web:
http://www.gnu.org/gnu/thegnuproject.html

Hurd va ser pensat com el nucli per al sistema GNU cap al 1990, quan en va començar el desenvolupament, ja que llavors la major part del programari GNU ja estava desenvolupat i només en faltava el nucli. Va ser el 1991 quan Linus va combinar GNU amb el nucli Linux i va crear així l'inici dels sistemes GNU/Linux. Però Hurd continua en procés de desenvolupament. Les idees de desenvolupament a Hurd són més complexes, ja que Linux podria considerar-se un disseny "conservador" que parteix d'idees ja conegudes i implantades.

Enllaç d'interès

Podeu llegir les opinions de Richard Stallman sobre GNU i Linux a:
http://www.gnu.org/gnu/linux-and-gnu.html

En concret, Hurd estava pensat com una col·lecció de servidors implementats sobre un micronucli Mach [Vah96], que és un disseny de nucli tipus micronucli (a diferència de Linux, que és de tipus monolític) desenvolupat per la Universitat Carnegie Mellon i posteriorment per la Universitat d'Utah. La idea bàsica era modelitzar les funcionalitats del nucli d'UNIX com a servidors que s'implementarien sobre un nucli bàsic Mach. El desenvolupament de Hurd es va retardar mentre s'estava acabant el disseny de Mach, i aquest es va publicar finalment com a programari lliure, cosa que permetria fer-lo servir per a desenvolupar Hurd. En aquest punt hem de comentar la importància de Mach, ja que molts sistemes operatius s'han basat en idees extretes d'aquest nucli, el més destacat dels quals és el MacOS X d'Apple.

El desenvolupament de Hurd es va retardar més per la complexitat interna, ja que existien diversos servidors amb diferents tasques de tipus *multithread* (d'execució de múltiples fils) i la depuració era extremadament difícil. Però avui en dia es disposa d'algunes versions de prova, i també de versions de prova de distribució GNU/Hurd produïdes per Debian. Tot i així, el projecte en si mateix no és especialment optimista amb vista a obtenir sistemes en producció, a causa tant de la complexitat com de la manca de suport per a dispositius.

Potser en un futur no tan llunyà hi podrà haver avenços i coexistència de sistemes GNU/Linux amb GNU/Hurd, o fins i tot el nucli Linux serà substituït

pel Hurd si es fan avenços importants en el seu desenvolupament. Això seria una solució si en algun moment Linux s'estanca (ja que el seu disseny monolític pot causar problemes si es fa molt més gran). En qualsevol cas, tant uns sistemes com els altres tenen un futur prometedor davant seu. El temps dirà cap on s'inclina la balança.

8. Taller de configuració del nucli a les necessitats de l'usuari

En aquest apartat veurem un petit taller interactiu per al procés d'actualització i de configuració del nucli en el parell de distribucions utilitzades: Debian i Fedora.

Una primera cosa imprescindible, abans de començar, és conèixer la versió actual que tenim del nucli, mitjançant `uname -r`, per a poder determinar quina és la versió següent que volem actualitzar o personalitzar. I una altra és la de disposar de mitjans per a engegar el nostre sistema en cas d'errors: el conjunt de CD de la instal·lació, el disquet (o CD) de rescat (actualment sol utilitzar-se el primer CD de la distribució) o alguna distribució en Cd autònom que ens permeti accedir al sistema de fitxers de la màquina, per a refer configuracions que hagin causat problemes. A més a més, hauríem de fer una còpia de seguretat de les nostres dades o configuracions importants.

Veurem les possibilitats següents:

1) Actualització del nucli de la distribució. Cas automàtic de Debian.

2) Actualització automàtica amb Fedora.

3) Personalització d'un nucli genèric (tant Debian com Fedora). En aquest últim cas els passos són bàsicament els mateixos que els que es presenten en l'apartat de configuració, però farem alguns comentaris addicionals.

8.1. Configuració del nucli amb Debian

En el cas de la distribució Debian, la instal·lació pot fer-se també de manera automàtica, mitjançant el sistema de paquets d'APT. Pot fer-se tant des de l'indicador d'ordres com amb gestors APT gràfics (synaptic, gnome-apt, etc.).

Farem la instal·lació mitjançant l'indicador d'ordres amb `apt-get`, suposant que l'accés als paquets font apt (sobretot als Debian originals) és ben configurat en el fitxer de `/etc/apt/sources.list`. Vegem-ne els passos:

1) Actualitzar la llista de paquets:

```
# apt-get update
```

2) Fer una llista de paquets associats a imatges del nucli:

```
# apt-cache search linux-image
```

3) Escollir una versió adequada a la nostra arquitectura (genèrica, 386/486/686 per a Intel, k6 o k7 per a AMD o, en particular, per a 64 bits, versions amd64 per a Intel i AMD o ia64 per a Intel Itanium). El codi de la versió indica la versió del nucli, la revisió de Debian del nucli i l'arquitectura. Per exemple, 2.6.xx-4-k7 és un nucli per a AMD Athlon, revisió Debian 4 del nucli 2.6.xx.

4) Comprovar, per a la versió triada, que hi hagi els mòduls accessoris addicionals (amb el mateix número de versió). Amb `apt-cache` busquem si hi ha altres mòduls dinàmics que puguin ser interessants per al nostre maquinari, segons la versió del nucli a instal·lar. Recordeu que, tal com vam veure amb la *Debian Way*, també existeix la utilitat `module-assistant`, que ens permet automatitzar aquest procés després de la compilació del nucli. En el cas en què els mòduls necessaris no fossin suportats, això ens podria impedir actualitzar el nucli si considerem que el funcionament del maquinari problemàtic és vital per al sistema.

5) Buscar, si volem disposar a més a mes del codi font del nucli, els paquets `linux-source-version` (només el 2.6.xx, és a dir, els números principals) i els `linux-headers` corresponents, per si més tard volem fer un nucli personalitzat (en aquest cas, el nucli genèric corresponent apedaçat per Debian).

6) Instal·lar allò que hàgim decidit. Si volem compilar des dels paquets font o simplement disposar del codi:

```
# apt-get install linux-image-version
# apt-get install xxxx-modules-version
```

(si calguessin alguns mòduls) i

```
# apt-get install linux-source-version-generica
# apt-get install linux-headers-version
```

7) Instal·lar el nucli nou, per exemple al *bootloader* LiLo (haurem de comprovar el *bootloader* utilitzat, ja que les últimes versions de Debian usen Grub per defecte). Normalment, aquest pas es fa automàticament, però no seria sobrer fer alguna còpia de seguretat prèvia de la configuració del *bootloader* (`/etc/lilo.conf` o `/boot/grub/menu.lst`).

Si se'ns pregunta si tenim el `initrd` activat, caldrà verificar el fitxer de LiLo (`/etc/lilo.conf`) i incloure la nova línia en la configuració LiLo de la imatge nova:

```
initrd = /initrd.img-version (o /boot/initrd.img-version)
```

Una vegada feta aquesta configuració, hauríem de tenir un LiLo semblant a la llista següent (fragment del fitxer), suposant que `initrd.img` i `vmlinuz` siguin enllaços a la posició dels fitxers del nou nucli:

```
default = Linux
image = /vmlinuz
```

```
  label = Linux
  initrd = /initrd.img
# restricted
# alias = 1
image = /vmlinuz.old
  label = LinuxOLD
  initrd = /initrd.img.old
# restricted
# alias = 2
```

Tenim la primera imatge per defecte, l'altra és el nucli antic. Així, des del menú LiLo en podrem demanar una o altra o, simplement canviant el `default`, recuperar l'antiga. Quan fem canvis en l'arxiu `/etc/lilo.conf` no ens hem d'oblidar de reescriure'ls en el sector corresponent amb l'ordre `/sbin/lilo` o `/sbin/lilo -v`.

En el cas de Grub, que sol ser l'opció normal en les distribucions, s'hauria creat una nova entrada (hem de recordar fer la còpia de seguretat prèvia, perquè podem haver perdut alguna entrada anterior en funció del funcionament o de la configuració de Grub; per exemple, pot limitar-se el nombre màxim d'entrades):

```
title           Debian GNU/Linux, kernel 2.6.32-5-amd64
root            (hd0,0)
kernel          /boot/vmlinuz-2.6.32-5-amd64 \
                root=UUID=4df6e0cd-1156-444e-bdfd-9a9392fc3f7e ro
initrd          /boot/initrd.img-2.6.32-5-amd64
```

on apareixerien els fitxers binaris del nucli i `initrd`. Amb l'etiqueta `root` en el nucli apareix l'identificador de la partició arrel del sistema on està installat el nucli, identificador que és comú a totes les entrades de nuclis del mateix sistema. Abans s'utilitzava un esquema amb `root=/dev/hda0` o `/dev/sda0`, però aquest esquema ja no és útil, perquè la detecció dels discos pot canviarne l'ordre en el sistema; així, es prefereix etiquetar les particions. Aquestes etiquetes poden obtenir-se mitjançant l'ordre `e2label /dev/particion` o també amb l'ordre `dumpe2fs /dev/particio | grep UUID` o, si la partició es troba muntada en el sistema, consultant `/etc/fstab`.

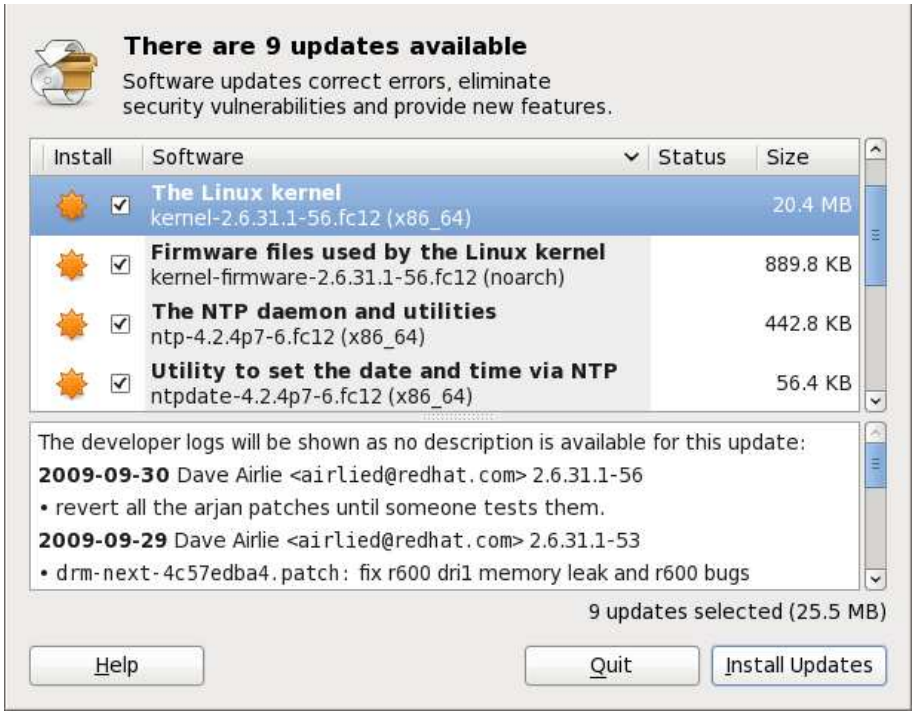
8.2. Configuració del nucli amb Fedora/Red Hat

L'actualització del nucli en la distribució Fedora/Red Hat és totalment automàtica per mitjà del seu servei de gestió de paquets (`yum` en l'indicador d'ordres, per exemple), o bé mitjançant els programes gràfics que inclou la distribució, segons la versió, per a l'actualització (PackageKit a Fedora o pup a Red Hat empresarial o equivalents, com CentOS). Normalment, els trobarem en la barra de tasques o en el menú d'eines de sistema de Fedora/Red Hat.

Aquest programa d'actualització bàsicament verifica els paquets de la distribució actual respecte a una base de dades de Fedora/Red Hat, i ofereix la possibilitat de descarregar els paquets actualitzats, entre els quals els del nucli. Aquest servei, en el cas de Red Hat empresarial, funciona mitjançant un compte de servei i Red Hat l'ofereix de pagament. Amb aquest tipus d'utilitats, l'actualització del nucli és automàtica. Cal comprovar les utilitats disponibles en els menús Eines/Administració, ja que les eines gràfiques disponibles en una distribució són altament dependents de la versió, ja que sovint són actualitzades.

Per exemple, en la figura 7 observem que una vegada posat en execució, ens ha detectat una nova versió del nucli disponible i podem seleccionar-la perquè ens la descarregui.

Figura 7. Eina de Fedora que mostra l'actualització del nucli disponible



Amb Fedora podem utilitzar les eines gràfiques equivalents o fer servir directament yum, si coneixem la disponibilitat de nous nuclis:

```
# yum install kernel kernel-source
```

Un cop descarregada, es procedirà a instal·lar-la, normalment també de manera automàtica, tant si disposem de Grub o de LiLo com a gestors d'arrencada. En el cas de Grub, això acostuma a ser automàtic i deixa un parell d'entrades en el menú, una per a la versió més nova i una altra per a l'antiga. Per exemple, en aquesta configuració de Grub (el fitxer és a `/boot/grub/grub.conf` o bé `/boot/grub/menu.lst`) tenim dos nuclis diferents, amb els respectius números de versió:

```
#fichero grub.conf
default = 1
```

```
timeout = 10
splashimage = (hd0,1)/boot/grub/splash.xpm.gz

title Linux (2.6.30-2945)
root (hd0,1)
kernel /boot/vmlinuz-2.6.30-2945 ro
       root = UUID=4df6e0cd-1156-444e-bdfd-9a9392fc345f
initrd /boot/initrd-2.6.30-18.9.img

title LinuxOLD (2.6.30-2933)
root (hd0,1)
kernel /boot/vmlinuz-2.4.30-2933 ro
       root = UUID=4df6e0cd-1156-444e-bdfd-9a9392fc345f
initrd /boot/initrd-2.4.30-2933.img
```

Cada configuració inclou un títol, que apareixerà en l'arrencada; el root, o partició del disc des d'on arrencar; el directori on es troba el fitxer corresponent al nucli i el fitxer `initrd` corresponent.

En cas que disposem de LiLo* com a gestor en Fedora/Red Hat, el sistema també l'actualitza (fitxer `/etc/lilo.conf`), però després caldrà reescriure manualment l'arrencada amb l'ordre `/sbin/lilo -v`.

*Per defecte es fa servir Grub.

Cal assenyalar, també, que amb la instal·lació anterior teníem possibilitats de descarregar els paquets font del nucli; aquests, una vegada instal·lats, són a `/usr/src/linux-version`, i poden configurar-se i compilar-se pel procediment habitual, com si fos un nucli genèric. Cal esmentar que l'empresa Red Hat duu a terme una gran tasca de pedaços i correccions per al nucli (utilitzat després a Fedora), i que els seus nuclis són modificacions de l'estàndard genèric amb bastants afegits, de manera que pot ser millor utilitzar els paquets font propis de Red Hat, tret que vulguem un nucli més nou o experimental que el que ens proporcionen.

8.3. Configuració d'un nucli genèric

Veurem ara el cas general d'instal·lació d'un nucli a partir de les seves fonts. Suposem que tenim unes fonts ja instal·lades a `/usr/src` (o el prefix corresponent).

Normalment, tindrem un directori `linux, linux-version` o senzillament el número de versió; aquest serà l'arbre dels paquets font del nucli. Aquests poden provenir de la mateixa distribució (o pot ser que els hàgim baixat en una actualització prèvia) i, en primer lloc, serà interessant comprovar si són els últims disponibles, com ja hem fet abans amb Fedora o Debian. D'altra banda, si volem tenir les versions últimes i genèriques, podem anar a kernel.org i baixar l'última versió disponible (millor l'estable que les experimentals, tret que

estiguem interessats en el desenvolupament del nucli). Descarreguem l'arxiu i descomprimim a `/usr/src` (o un altre escollit, potser millor) els paquets font del nucli. També podríem buscar si hi ha pedaços per al nucli i aplicar-los (tal com hem vist en l'apartat 4).

Seria convenient rellegir l'apartat 3.

A continuació, comentarem els passos que caldrà fer. El procediment que s'indica en aquesta part del taller és genèric, però pot donar algun problema en funció de la distribució que utilitzem. Es recomana seguir, en la mesura del que sigui possible, el subapartat 3.3., en què es comenta el cas de configuració d'un nucli *vanilla* de la branca 2.6.xx o bé els comentaris en el cas de Debian, en el subapartat 3.4., o la referència [Fedk] per al cas de Fedora.

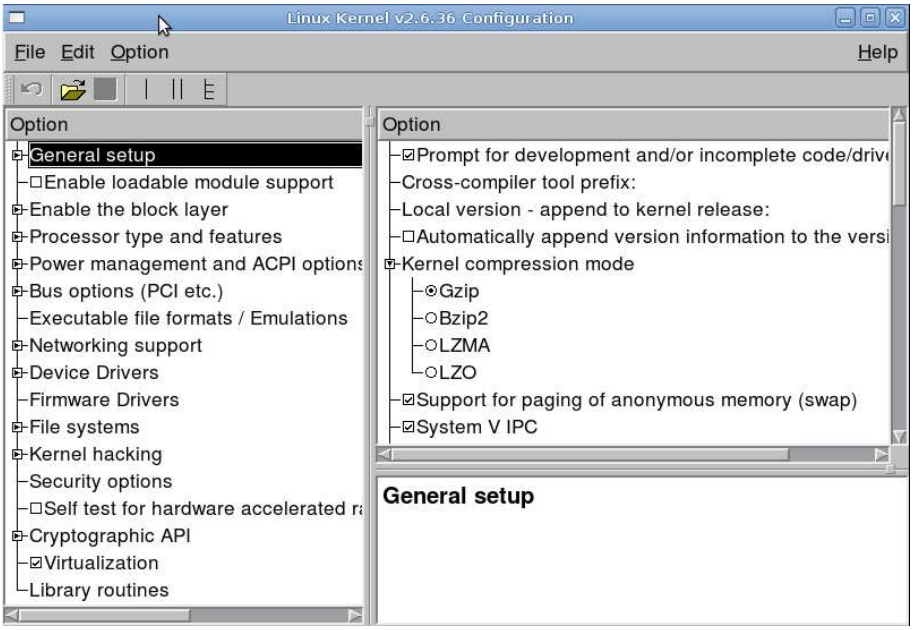
Amb les consideracions comentades, podem seguir també el procés següent:

1) Netejar el directori de proves anteriors (si escau):

```
make mrproper
```

2) Configurar el nucli amb, per exemple, `make menuconfig` (o `xconfig`, figura 8, `gconfig` o `oldconfig`). Ho hem vist en el subapartat 3.3.

Figura 8. Configuració (xconfig) del nucli per menús gràfics



3) Dependències i neteja de compilacions anteriors:

```
make dep
```

4) Compilació i creació de la imatge del nucli: `make bzImage`. També seria possible `zImage` si la imatge fos més petita, però és més normal `bzImage`, que optimitza el procés de càrrega i la compressió per a nuclis més grans. En algun maquinari molt antic pot ser que no funcioni, a causa de la mida final del nucli, i llavors és necessari usar `zImage`. El procés pot durar

des d'algunes desenes de minuts fins a una hora en maquinari modern o diverses hores en maquinari molt antic. Quan finalitza, la imatge es troba a: `/usr/src/directori-fonts/arch/i386/boot` (en el cas d'arquitectura Intel de 32 bits).

5) Ara compilem els mòduls amb `make modules`. Fins aquest moment no hem modificat res en el nostre sistema. Ara haurem de fer la instal·lació.

6) En el cas dels mòduls, si provem alguna versió antiga del nucli (branca 2.2 o primeres de la 2.4), cal anar amb compte, ja que en alguna se sobreescrivien els mòduls antics (en les últimes versions 2.4.x o la branca actual 2.6.x ja no és així).

Però també s'ha d'anar amb compte si estem compilant una versió que és la mateixa (numeració exacta) que la que tenim (els mòduls se sobreescriuran); en aquest cas, és millor fer una còpia de seguretat dels mòduls:

```
cd /lib/modules
tar -cvzf old_modules.tgz versiokernel-antiga/
```

Així, tenim una versió a `.tgz` que podríem recuperar després en cas de problemes. I, finalment, instal·lem els mòduls amb:

```
make modules install
```

7) Ara podem passar a la instal·lació del nucli, per exemple amb:

```
# cd /usr/src/directori-Fonts/arch/i386/boot
# cp bzImage /boot/vmlinuz-versionkernel
# cp System.map /boot/System.map-versionkernel
# ln -s /boot/vmlinuz-versionkernel /boot/vmlinuz
# ln -s /boot/System.map-versionkernel /boot/System.map
```

Així, col·loquem el fitxer de símbols del nucli (`System.map`) i la imatge del nucli. Cal recordar que pot ser necessària també una imatge de `initrd`.

8) Ja només ens queda posar la configuració necessària en el fitxer de configuració del gestor d'arrencada, tant si és LiLo (`/etc/lilo.conf`) com Grub (`/boot/grub/menu.lst`), segons les configuracions que ja hem vist amb Fedora o Debian. I recordeu, en el cas de LiLo, que caldrà tornar a actualitzar la configuració amb `/sbin/lilo` o `/sbin/lilo -v`.

9) Reiniciar la màquina i observar els resultats (si tot ha anat bé).

Resum

En aquest mòdul hem examinat diferents característiques del nucli (*kernel*) de Linux en la seva branca principal 2.6. Així mateix, hem comentat alguns dels processos d'actualització, configuració i sintonització per al sistema, útils tant per a optimitzar-ne l'ús de memòria com per a les prestacions en una arquitectura determinada, i també l'adaptació al maquinari de destinació.

També hem examinat les possibilitats que ofereixen els mòduls dinàmics com a mecanisme d'extensió del nucli i ampliació del maquinari suportat.

La inclusió en el nucli de tècniques de virtualització ens permet, a partir del nucli i certes utilitats, construir un entorn de virtualització potent i flexible per a generar diferents combinacions de màquines virtuals que resideixen en un sistema amfitrió GNU/Linux.

Activitats

1. Determineu la versió actual del nucli Linux incorporada en la vostra distribució. Comproveu les actualitzacions disponibles de manera automàtica, tant a Debian (apt) com a Fedora/Red Hat (mitjançant yum o una eina gràfica d'actualització).

2. Feu una actualització automàtica de la vostra distribució. Comproveu possibles dependències amb altres mòduls utilitzats i amb el *bootloader* (LiLo o Grub) utilitzat. En funció del sistema, pot ser recomanable una còpia de seguretat de les dades importants del sistema (comptes d'usuaris i fitxers de configuració modificats), o bé fer el procés en un altre sistema de què es disposi per a proves.

3. Per a la vostra branca del nucli, determineu l'última versió disponible (consulteu la pàgina web http://www.kernel.org) i feu una instal·lació manual amb els passos examinats en el mòdul. La instal·lació final pot deixar-se com a opcional, o bé posar una entrada dins del *bootloader* per a les proves del nucli nou. Considereu, també en funció del sistema, la possibilitat de fer una còpia de seguretat prèvia, en especial de les configuracions estables dels *bootloaders*.

4. En el cas de la distribució Debian, a més dels passos manuals, hi ha, com hem vist, una manera especial (recomanada) d'instal·lar el nucli a partir de les seves fonts mitjançant el paquet `kernel-package`. Feu els passos necessaris per a crear una versió personalitzada d'un nucli *vanilla*.

5. Elaboreu una màquina virtual basada en KVM que tingui com a hoste una altra distribució GNU/Linux diferent de la del sistema amfitrió, a partir de la seva instal·lació per mitjà de CD-ROM o per mitjà d'una imatge ISO de la distribució.

Bibliografia

[Ar05] **Corbet, J.; Rubini, A.; Kroah-Hartman, G.** (2005). *Linux Device Drivers* (3a. ed.). O'Reilly.

[Arc] **Arcomano, R.**. *Kernel Analysis-HOWTO*. The Linux Documentation Project.

[Bac86] **Bach, M. J.** (1986). *The Design of the UNIX Operating System*. Prentice Hall.

[Ces06] **Cesati, M.; Bovet, D.** (2006). *Understanding the Linux Kernel* (3a ed.). O'Reilly.

[Debk] **Debian Kernel Handbook Project**. *Debian Linux Kernel Handbook*. <http://kernel-handbook.alioth.debian.org>

[Fedk] **Fedora Project**. *Building a custom kernel*. <http://fedoraproject.org/wiki/Building_a_custom_kernel>

[Gor] **Gortmaker, P.** (2003). *The Linux BootPrompt HOWTO*. The Linux Documentation Project.

[Gru] **GNU**. *Grub bootloader*. <http://www.gnu.org/software/grub/>

[Grub] **GNU**. *Grub Manual*. <http://www.gnu.org/software/grub/manual/>

[Hen] **Henderson, B.** *Linux Loadable Kernel Module HOWTO*. The Linux Documentation Project.

[Kan] **Kanis, I.**. *Multiboot with GRUB Mini-HOWTO*. The Linux Documentation Project.

[Ker] **Rusty Russell**. *Unreliable Guide To Hacking The Linux Kernel*. <http://www.kernel.org/doc/htmldocs/kernel-hacking.html>

[Kera] **Kernelnewbies.org**. *Kernel Newbies*. <http://www.kernelnewbies.org>

[Kerb] **Kernel.org**. *Linux Kernel Archives*. <http://www.kernel.org>

[Lkm] **Lkm**. *Linux Kernel Mailing List*. <http://www.tux.org/lkml>

[Mur] **Murphy, G. L.** *Kernel Book Project*.
<http://kernelbook.sourceforge.net>

[OSDa] **OSDL**. *Open Source Developement Laboratories* (ahora *The Linux Foundation*).
<http://www.linuxfoundation.org>

[Pra03] **Pranevich, J.** (2003). *The Wonderful World of Linux 2.6*.
<http://www.kniggit.net/wwol26.html>

[Pro] **GNU Project**. *Grub Manual*.
<http://www.gnu.org/software/grub/manual/>

[Skoa] **Skoric, M.** *LILO mini-HOWTO*. The Linux Documentation Project.

[Tan87] **Tanenbaum, A.** (1987). *Sistemas operativos: Diseño e Implementación*. Prentice Hall.

[Tum] **Tumenbayar, E.** (2002). *Linux SMP HOWTO*. The Linux Documentation Project.

[Vah96] **Vahalia, U.** (1996). *UNIX Internals: The New Frontiers*. Prentice Hall.

[Vasb] **Vasudevan, A.** *The Linux Kernel HOWTO*. The Linux Documentation Project.

[Zan] **Zanelli, R.**. *Win95 + WinNT + Linux multiboot using LILOmini-HOWTO*. The Linux Documentation Project.

Altres fonts de referència i informació

[Kerb] Lloc que proporciona un dipòsit de les diverses versions del nucli Linux i els seus pedaços.

[Kera] [lkm] Llocs web que recullen una part de la comunitat del nucli de Linux. Disposa de diversos recursos de documentació i llistes de correu de l'evolució del nucli, la seva estabilitat i les noves prestacions que es desenvolupen.

[Debk] És un manual imprescindible sobre els processos de compilació del nucli en la distribució Debian. Sol actualitzar-se amb els canvis produïts en la distribució. [Fedk] aporta una referència similar per al cas de Fedora.

[Ces06] Llibre sobre el nucli de Linux 2.4, que en detalla els diferents components i la seva implementació i disseny. Hi ha una primera edició sobre el nucli 2.2 i una nova actualització al nucli 2.6.

[Pra03] Article que descriu algunes de les principals novetats de la branca 2.6 del nucli Linux.

[Ker] [Mur] Projectes de documentació del nucli, incomplets però amb material útil.

[Bac86] [Vah96] [Tan87] Alguns texts sobre els conceptes, el disseny i la implementació dels nuclis de diferents versions UNIX.

[Skoa][Zan01][Kan][Pro] Recursos per a tenir més informació sobre els carregadors LiLo i Grub.

[Gru][Grub] Llocs oficials de Grub2 i l'anterior original Grub (ara conegut com a Grub Legacy).