El nucli Linux

Josep Jorba Esteve

PID_00239611

Universitat Oberta de Catalunya

Índex

Introducció

El nucli (en anglès *kernel*) del sistema operatiu GNU/Linux (que habitualment anomenem simplement Linux) [Vasb] és la part central del sistema: s'encarrega de posar-lo en funcionament i, una vegada aquest ja és utilitzable per les aplicacions i els usuaris, s'encarrega de gestionar els recursos de la màquina, controlant la gestió de la memòria, els sistemes de fitxers, les operacions d'entrada i sortida, els processos i la seva intercomunicació.

Origen de Linux

El nucli Linux es remunta a l'any 1991, quan Linus Torvalds el va oferir per a l'ús 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 un *newsgroup* de l'època 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 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 de codi font, podem recompilar el nucli per a adaptar millor al nostre sistema i, com veurem més endavant configurar-lo 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 GNU/Linux: com, partint dels arxius font, podem obtenir una nova versió del nucli adaptada al nostre sistema. Veurem com es desenvolupen les fases de configuració, es compila posteriorment i es fan proves amb el nou nucli obtingut.

A més, examinarem com 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 se'ns ofereixen amb suport des del mateix nucli.

Objectius

En aquest mòdul es mostren els continguts i les eines procedimentals per a aconseguir els objectius següents:

1. Conèixer el funcionament del kernel i dels processos de configuració associats.
2. Poder configurar i crear 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, per a sintonitzar els seus sistemes.

1. El nucli del sistema GNU/Linux

El nucli o *kernel* és la part bàsica de qualsevol sistema operatiu [Tan87, Tan06], sobre la qual descansa el codi dels serveis fonamentals per a controlar el sistema complet. Bàsicament, la seva estructura pot separar-se típicament en una sèrie de components o de mòduls de gestió orientats al següent:

- **Gestió de processos**: quines tasques s'executaran, 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 possibles amb els usuaris?

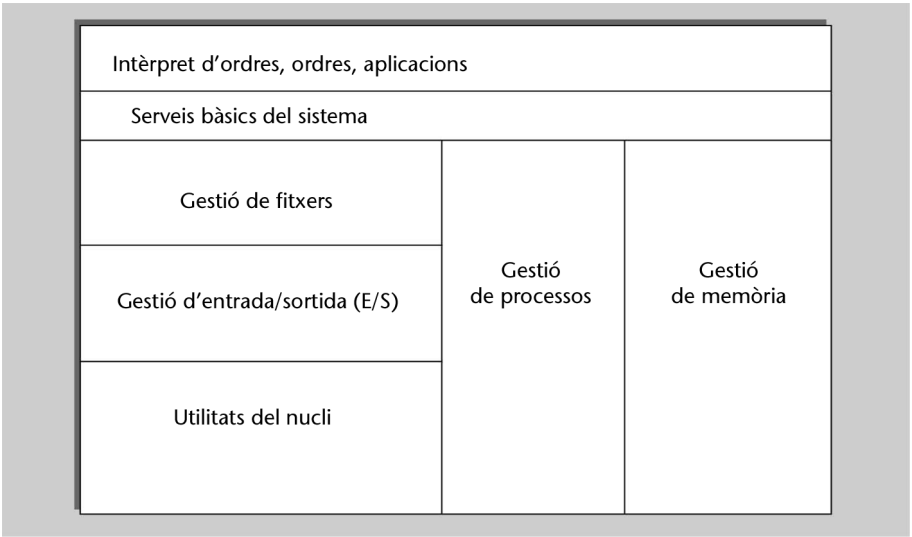
- **Intercomunicació de processos i sincronització**: com es comuniquen tasques entre elles, amb quins diferents mecanismes i com poden sincronitzar-se grups de tasques?

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

- **Gestió de memòria**: optimització de l'ús 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* o *updates*).

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 amb necessitats crítiques controlar-ne millor les aplicacions i el mateix sistema, i poder elaborar sistemes amb el sistema operatiu "a la carta", personalitzat al gust de cada usuari final. També permet disposar, al seu torn, d'un sistema operatiu amb codi obert, desenvolupat per una comunitat de programadors coordinats mitjançant Internet i accessible, 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 (per a dispositius mòbils o altra electrònica de consum).

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][Tan06] (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

MINIX

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

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 gener de 1999, es van incrementar les prestacions de suport SMP de manera significativa, i es van afegir controladors per a una 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.

El juliol de 2011, Linus Torvalds va anunciar la versió Linux 3.0, no pels seus grans canvis tecnològics, sinó per iniciar una sèrie de canvis posteriors i per commemorar el 20è aniversari de Linux. Algunes fites més en la branca 3.x:

- 3.3 (març de 2012): es fa una mescla de les fonts de Linux i s'hi incorpora Android, que bàsicament és un kernel Linux amb certes modificacions; després de diversos intents en aquesta versió, es fa la unió de tots dos. S'hi introdueix també la capacitat de poder arrencar directament mitjançant les noves EFI, substitució de les BIOS.

- 3.6 (setembre de 2012): millores en el suport del sistema de fitxers Btrfs (quotes, *snapshots*), pensat per a substituir en el futur a ext4.

- 3.7 (desembre de 2012): s'hi incorpora el suport per a diverses arquitectures ARM.

- 3.8 (febrer de 2013): com a curiositat, s'elimina el suport a processadors 286, i es fan importants millores en diversos sistemes de fitxers.

- 3.13 (gener de 2014): suport d'NFtables, la següent generació de regles de tallafoc (substituint iptables).

- 3.15 (juny de 2014): es millora el suport d'EFI, l'escriptura en sistemes de fitxers FUSE, i el suport per a repertoris vectorials de noves CPU Intel.

L'abril de 2015 es va anunciar la versió 4.0, inaugurant una nova branca on es van realitzar algunes millores de la infraestructura d'aplicació de pedaços en el kernel. La idea era aplicar alguns canvis que permetessin millorar el desenvolupament de futures versions del kernel. Les següents són algunes de les noves prestacions més destacades:

- **4.0 (abril 2015)**. Integració preliminar de tecnologies basades en kGraft (de SUSE) i kpatch (procedent de RedHat) que permet aplicar pedaços en

el kernel sense reiniciar. S'afegeix suport per a noves CPUs d'Intel (les Skylake).

- **4.1 (juny 2015)**. Suport d'encriptació en ext4.

- **4.2 (agost 2015)**. Drivers gràfics bàsics d'AMD inclosos en el kernel, i driver virtual per GPU per a la millora, suport, i prestacions en virtualització per la GPU.

- **4.5 (març 2016)**. Diverses millores en sistemes de fitxers Btrfs. I estabilització de la jerarquia cgroups (els grups de control són utilitzats per assignar, o limitar, recursos a grups de processos).

- **4.7 (juliol 2016)**. S'afegeix suport per a noves GPUs AMD i suport per a dispositius virtuals USB compartits en xarxa. S'habilita una memòria cau paral·lela de directoris (per a cerques més eficients sense/minimitzant l'accés a disc). I mòdul de seguretat (LoadPin) per restringir l'origen dels mòduls del kernel (evitar intrusions de mòduls no vàlids).

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), es va veure obligat a mantenir les diferents versions estables del nucli gràcies a diferents col·laboradors, mentre que Linus continua (en la mesura del 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 (branca de desenvolupament).
- 2.6 Greg Kroah-Hartman (versions estables, entre altres) / Linus Torvalds, Andrew Morton (*releases* de desenvolupament).
- 3.x Greg Kroah-Hartman, Sasha Levin i Linus Torvalds (*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 de la 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	1.800
2.4	01-2001	3.378
2.6	12-2003	5.930
3.10	06-2013	15.803
4.1	06-2015	19.500

Com podem comprovar, s'ha evolucionat d'unes deu mil línies a sis milions en les primeres versions de la branca 2.6; les últimes versions de la branca 4.x tenen ja més de dinou milions de línies.

En aquests moments, el desenvolupament continua en la branca 4.x del nucli, l'última versió estable, que inclou la majoria de distribucions com a versió principal (algunes encara inclouen 2.6.x, però 3.x o 4.x acostuma a ser l'opció per defecte en la instal·lació).

Tot i que ara la branca principal és la 4.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 van mantenir 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, Red Hat, Suse, Transmeta, etc.). En aquell moment 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 dos o tres 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 catorze mil programadors en les últimes versions del nucli) 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, amb diferents prestacions, per a un mateix component.

Respecte a la numeració de les versions del nucli de Linux [Ker, Ces06, Lov10], 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 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.

En els últims esquemes s'arriba a introduir quarts nombres, per a especificar Z canvis menors, o diferents possibilitats de la revisió (amb diversos pedaços

afegits que corregeixen errades). La versió així definida amb quatre nombres és la que es considera estable (*stable*). També es fan servir altres esquemes per a les diverses versions de prova (normalment no recomanables per a entorns de producció), com sufixos `-rc` (*release candidate*) i `-mm`, que són nuclis experimentals amb gran introducció de pedaços que suposen noves prestacions addicionals com proves de diverses tècniques noves; o els `-git`, que són una mena de "foto" diària del desenvolupament del nucli. Aquests esquemes de numeració estan en constant canvi per a adaptar-se a la forma de treballar de la comunitat del nucli i a les seves necessitats per a accelerar el desenvolupament.

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

3) En la branca 3.x, quan després del 2.6.39 Linus va decidir renombrar 3.x, el segon nombre s'ha fet servir com a nombre de revisió per seqüència temporal a mesura que sortien els nous kernels, i s'hi pot afegir un tercer nombre per a designar correccions de fallades respecte a *bugs* o seguretat. S'espera que, de manera similar, en un determinat moment es progressi de la mateixa manera en les branques 4.x, 5.x, etc. Pel que fa a les versions diferents del kernel, ara s'acostuma a denominar *mainline* la versió de desenvolupament que normalment manté Linus, i que apareix de manera periòdica cada dos o tres mesos. Quan apareix alliberada la versió passa a l'estat de *stable*, i s'hi fan diverses iteracions (indicades pel tercer nombre) per *bugs*, normalment de dues a tres per mes, tot i que només hi ha unes poques revisions de l'estable, perquè mentrestant el *mainline* següent esdevé estable. Com a cas excepcional, hi ha alguns kernels denominats *longterm*, per a propòsits de manteniment de llarga durada, en els quals s'inclou el suport de pedaços que es facin en altres versions, per a portar-los a aquestes. Poden ser, per exemple, kernels que hagin estat escollits per distribucions de tipus LTS (*long term support*) o interessants perquè contenen alguna prestació especial. S'ha de diferenciar que els kernels de distribució acostumen a ser mantinguts per les mateixes distribucions, normalment com a paquets als seus repositoris, i poden incloure nivells de pedaç addicionals propis de la distribució. Aquests kernels especials de les distribucions es poden detectar amb l'ordre `uname -r`, que ens dóna la versió del kernel actual, pels números addicionals als predefinits en la numeració del kernel, inclosos en la sortida de l'ordre.

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

Enllaç d'interès

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

Algunes de les característiques tècniques [Ces06, Kan, Lov10] 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 a 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 elles. 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 i prosseguir-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 les branques 2.6 i 3. Del rendiment SMP i *multicore* 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), ZFS (Oracle), Btrfs, 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 és bo 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 personalitzable: 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 i en processadors o arquitectures obsoletes).

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 format antic de 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 a 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 32 o 64 bits o bé virtualitzar màquines completes, es pot disposar d'imatges virtuals de distribucions GNU/Linux ja preparades per a actuar de sistemes convidats en gestors de virtualització.

8) Linux disposa d'un completíssim suport: no hi ha cap altre sistema que tingui la rapidesa i la quantitat de pedaços i actualitzacions que té 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. Una deficiència important per a l'adopció de Linux en certs àmbits 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, AMD ATI) i impressores (Epson, HP) ja comencen a proporcionar els controladors per als seus dispositius, 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 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 ofereixen una barreja de controladors de dispositius [Cor05], 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 3.x/4.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 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 contràriament, una vella) 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-ne característiques.
- 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. 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 manualment 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.

Vegeu també

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

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.

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, per al sistema operatiu, l'element central.

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.

No obstant això, davant sistemes en producció, sempre és important prendre les mesures de precaució i fer les còpies de seguretat necessàries. O bé treballar amb un sistema equivalent, en estat de test o preproducció, en el qual puguem prèviament provar els efectes d'un nou kernel.

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

1) El procés de compilació per a les branques 2.6.x i 3.x/4.x

2) Detalls específics de les versions 3.x./4.x

3) 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.

Respecte a les versions 2.6.x, en aquest mòdul mantenim l'explicació per raons històriques, ja que les distribucions actuals pràcticament ja 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 per incompatibilitats o per l'existència de maquinari antic no suportat per les distribucions actuals.

Els conceptes generals del procés de compilació i configuració s'explicaran en el primer subapartat (2.6.x), ja que la majoria són genèrics, i observarem posteriorment les diferències respecte a les noves versions.

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, dels següents fets:

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

- Per a sintonitzar el nucli, antigament calia compilar-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 compilació, 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 a una nova *release* d'aquesta.

- 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 exemple sis mesos) 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 i 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 compilació de les diferents branques de desenvolupament del nucli Linux.

3.1. Compilació de les branques 2.6.x i 3.x/4.x del kernel Linux

Les instruccions són específiques per a l'arquitectura x86/x86_64(o amd64) d'Intel, mitjançant usuari *no-root* en la major part del procés (es pot fer com a usuari normal i, de fet, és altament aconsellable per seguretat), i només al procés final d'instal·lació és necessari *root* per a integrar el kernel final en el sistema. La compilació del kernel és un procés costós en temps i disc per als kernels actuals, i es recomana disposar d'espai de disc abundant (100 GB, o més, per a kernels actuals), i portar a terme el procés amb el màxim de CPU i memòria disponible per a accelerar-lo. Les diferents fases que hi estan involucrades són:

1) Obtenir el nucli. Podem acudir a http://www.kernel.org (o al seu servidor ftp) i descarregar la versió *stable* o *longterm* que volem provar. D'altra banda, en la majoria de distribucions de GNU/Linux, com ara 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 fer servir aquestes (mitjançant els paquets `kernel-source`, `kernel-version`, `linux-source`, o similars). Si volem els últims nuclis, potser no estaran disponibles en la distribució i haurem d'acudir a *kernel.org*.

2) Desempaquetar el nucli. Els paquets font del nucli acostumaven a col·locar-se i desempaquetar-se sobre el directori `/usr/src`, tot i que es recomana fer servir algun directori a part 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-2.6.32.63.tar.bz2 | tar xvf -
```

Si els paquets font venien en un fitxer `gz` (o `tar.gz`), reemplacem bzip2 per gzip, o en els recents (fets servir per kernel.org) `.tar.xz` o `.xz` reemplacem per `unxz` (paquet xz-utils):

```
xz -cd linux-4.X.tar.xz | tar xvf -
```

En descomprimir els paquets font, s'haurà generat un directori denominat `linux-version_kernel`, on entrarem per establir la configuració del nucli. És recomanable disposar de prou espai per a la compilació del nucli. Depenent de la versió poden ser necessaris de 20 a 100 GB d'espai, depenent de les opcions escollides. Així mateix es necessita disposar d'espai suficient per a la instal·lació posterior del nucli creat a la partició arrel /. En aquest casos són recomanables de 10 a 20 GB lliures addicionals.

En aquest punt, podria ser interessant apedaçar el nucli, la qual cosa podríem fer ja que 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 es podria donar el cas que un fabricant de maquinari oferís algun suport o correcció de fallades per a un controlador de dispositiu, com un pedaç per a una versió de nucli concreta.

Vegeu també

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

Una vegada disposem dels paquets necessaris, procedim al procés de compilació en el directori fet servir pels paquets font. Cal recordar que tot el procés es pot portar a terme com a usuari normal; només parts molt concretes, com la instal·lació final del nucli o de mòduls dinàmics, és necessari fer-les fent servir l'usuari *root*.

També poden sorgir problemes de seguretat per a fer la compilació en mode *root* de fonts de nucli desconegudes o no fiables. Respecte a això, tant els paquets font de *kernel.org* com els proporcionats per les distribucions acostumen a contenir firmes (o repositoris firmts) que es poden fer servir per a verificar la integritat dels fitxers de fonts. S'ha d'evitar, costi el que costi, fer servir fonts de nucli o de mòduls proporcionats per emissors no fiables.

Eines de configuració i compilació

Abans de començar els passos previs a la compilació, hem d'assegurar-nos de disposar de les eines correctes, especialment del compilador gcc, make i altres utilitats GNU complementàries en el procés. Un exemple són les *module-init-tools*, que ofereixen les diferents utilitats per a l'ús i 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 utilitzada (per exemple, les ncurses per a la interfície menuconfig).

Per exemple, en una distribució Debian estable (en altres distribucions o versions, haureu de consultar els paquets similars), són necessaris, entre d'altres, els paquets *build-essential libncurses5-dev libqt4-dev libqt4-qt3support qt4-qmake libgtk2.0-dev libglib2.0-dev libglade2-dev modules-init-tools gcc g++*. En el cas de Debian, necessitarem una instal·lació prèvia d'aquests paquets, per exemple, feta amb la llista anterior, i un *apt-get install* dels paquets esmentats.

Es recomana, en general, consultar la documentació del nucli (per mitjà del paquet o en el directori arrel de les fonts del nucli) per a saber quins prerequisits, i versions d'aquests, són necessaris per al procés. Es recomana examinar els fitxers `README` en el directori "arrel", i el `Documentation/Changes`, que esmenta els requisits mínims de versions dels programes, prerequisits de la compilació, o l'índex de documentació del nucli en `Documentation/00-INDEX`.

Si hem fet anteriors compilacions en el mateix directori, hem 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 s'acostumen a emmagatzemar

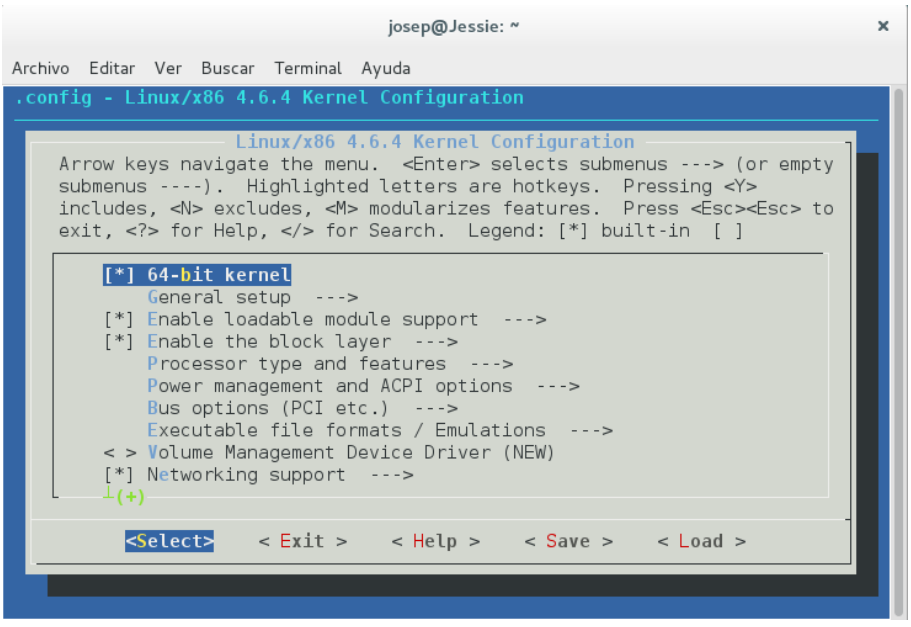
en un fitxer de configuració, normalment `.config` en el directori "arrel" de les fonts). Les diferents alternatives són:

- `make config`: des de la línia d'ordres, se'ns pregunta per cada opció i se'ns demana confirmació (*y/n*), si desitgem o no l'opció, o se'ns demanen els valors necessaris, o seleccionar una de les possibilitats de l'opció. És la configuració llarga, en la qual se'ns demanen moltes respostes; podem haver de respondre uns quants centenars de preguntes (o més, depenent de la versió).

- `make oldconfig`: serveix per si volem reutilitzar una configuració ja emprada (normalment emmagatzemada en un fitxer `.config`, en el directori "arrel" de les fonts prèvies que hàgim completat amb anterioritat). 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 seves opcions. No obstant això, en les últimes versions del nucli, aquesta opció detecta la configuració anterior i només ens pregunta per les opcions noves disponibles en la nova versió, estalviant així un temps considerable en les eleccions. D'altra banda, en general la configuració d'un kernel disponible en la distribució està també guardada en el fitxer `/boot/config-version` corresponent (recordeu que amb `uname -r` tenim la versió del kernel actual), el qual podem copiar directament com a fitxer `.config` en l'arrel de les fonts del kernel, i amb `make oldconfig` configurar només les noves opcions.

- `make menuconfig`: configuració basada en menús textuals, molt còmoda (figura 2); podem habilitar o inhabilitar el que vulguem, i és més ràpida i versàtil per a la navegació entre opcions que el `make config`.

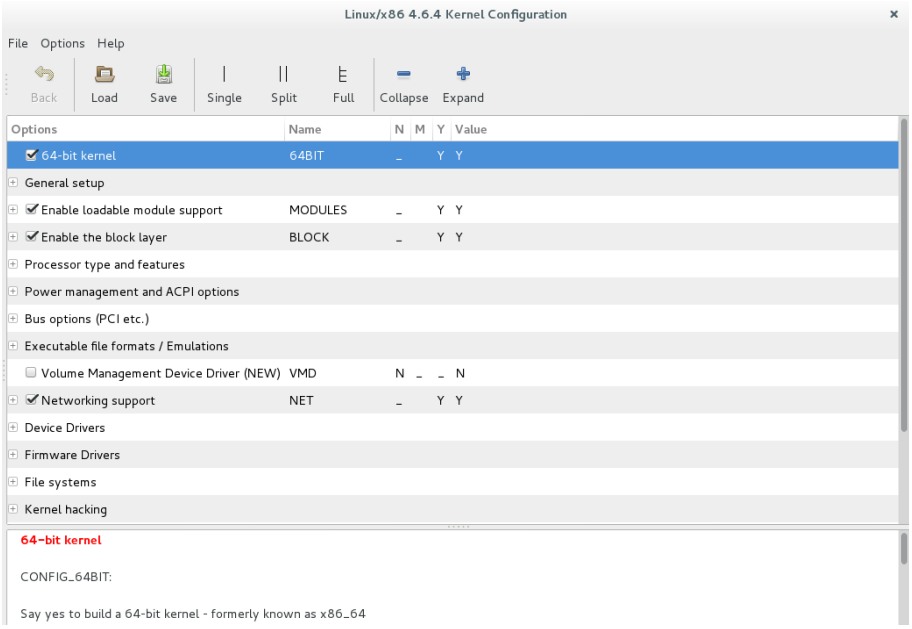
Figura 2. Configuració del nucli 4.x des de la interfície textual de menuconfig



- `make xconfig`: la més còmoda, basada en diàlegs gràfics en X Window. És necessari tenir instal·lades les biblioteques de desenvolupament de qt3 o qt4, ja que aquesta configuració té aquests requeriments en estar basada en la biblioteca Qt3/Qt4 (KDE). La configuració es basa en quadres de diàleg, botons i caselles d'activació. És prou ràpida i disposa d'ajuda amb comentaris de moltes de les opcions.

- `make gconfig`: similar al cas de xconfig, però amb la interfície basada en gtk (Gnome) (figura 3). Com a prerequisit, necessita els paquets de desenvolupament de les biblioteques Gnome (entre d'altres, *libgtk2.0-dev*, *libglib2.0-dev* i *libglade2-dev*; els noms són dependents de la distribució i versió).

Figura 3.Configuració basada en GNOME d'un nucli 4.x



Un cop s'hagi fet el procés de configuració, cal guardar el fitxer `.config`, ja que la configuració consumeix un temps important. A més, pot ser d'utilitat disposar de la configuració efectuada (`.config`) si s'està planejant fer-la posteriorment en diferents màquines similars 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 la mida i el rendiment del nucli (i, per tant, del sistema sencer) en alguns casos pot dependre de la nostra elecció.

Vegeu també

La gestió de mòduls es veu en l'apartat 5.

El nucli de Linux, com a imatge binària un cop compilat, ha començat a ser molt gran, tant per complexitat com pels controladors de dispositiu [Cor05] que inclou. Si hi integréssim totes les seves possibilitats, es podria crear un fitxer del nucli prou gran 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 petites porcions, que s'encarregaran de ma-

nera dinàmica a demanda quan, per càrrega explícita o per l'ús de les seves característiques, siguin necessàries.

L'elecció més normal és integrar dins del nucli allò que es consideri bàsic per al funcionament, o crític en rendiment, i deixar com a mòduls aquelles parts o controladors dels quals es faci un ús esporàdic o que es necessitin 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 sapiguem amb seguretat quin maquinari tindrà (per exemple, quina targeta de xarxa), però sí 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, podem seleccionar-ne alguns (o tots) i posar-los com a mòduls. D'aquesta manera, 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 només hi hagués un controlador integrat en el nucli i canviem la targeta, això obligaria a reconfigurar i recompilar el nucli amb el controlador de la nova targeta.

- Un altre cas que acostuma a aparèixer (tot i que no és gaire comú) té lloc quan necessitem dos dispositius que són incompatibles entre ells, o està funcionant l'un o l'altre (això passava antigament, per exemple, amb impressores amb cable paral·lel i alguns dispositius de maquinari que es connectaven al port paral·lel). Per tant, en aquest cas hem de col·locar com a mòduls els controladors i carregar o descarregar allò que sigui necessari en cada moment. Tot i que actualment també s'han solucionat aquests problemes, amb suports de *daemos* que controlen casos de *hotplug* (connexió en calent) de dispositius, com per exemple *udev*, que aporta solucions en aquest sentit ja que gestiona dinàmicament les entrades del directori `/dev` a mesura que els dispositius es connecten o desconnecten en el sistema.

- Un altre exemple el podrien formar els sistemes de fitxers (*filesystems*). Normalment, esperarem que el nostre sistema tingui accés a alguns d'aquests, per exemple ext2, ext3 o ext4 (propis de Linux), vfat, i els donarem d'alta en la configuració del nucli. Si en un altre moment haguéssim de llegir un altre tipus no esperat, per exemple dades guardades en un disc o partició de sistema NTFS de Windows, 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'hagi d'accedir a aquests sistemes, podem deixar els altres sistemes de fitxers com a mòduls (també en aquests casos és possible utilitzar altres alternatives, com FUSE, per donar suport a nivell usuari, en lloc d'en el nucli).

vfat

vfat (FAT32) és el format propi d'antigues versions de Windows i bastant utilitzat en dispositius d'emmagatzematge USB.

4) Compilació del nucli. Un cop disponible una configuració de la versió del nucli, mitjançant `make` començarem el procés de compilació:

```
$ make
```

Aquest procés es pot accelerar si disposem d'una màquina multiprocessador o *multicore*; `make` disposa de l'opció `-jn`, amb la qual podem especificar, amb `n`, el nombre de processadors o *cores* que farem servir per a la compilació.

Quan aquest procés finalitzi, tindrem la part integral del nucli i ens faltaran les parts que es va demanar col·locar com a mòduls (en les últimes branques de 3.x/4.x aquesta opció no és necessària, perquè es generen ja amb `make` directament, encara que d'altra banda sí que caldrà instal·lar els mòduls):

```
$ make modules
```

Fins a aquest moment, hem fet la configuració i compilació del nucli. Aquesta part es podia fer des d'un usuari normal (altament recomanable) o bé el *root*, però ara necessitarem forçosament usuari *root*, perquè passarem a la part de la instal·lació del nou kernel compilat en el sistema.

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

```
# make modules_install
```

Això ens instal·larà els mòduls construïts en el sistema de fitxer perquè el nou kernel els pugui trobar al lloc adequat. Normalment, els mòduls acostumen a estar disponibles en el directori `/lib/modules/version_kernel`, on `version_kernel` serà la numeració del kernel que acabem de construir.

I per a realitzar la instal·lació del nou nucli (des del directori de compilació de les fonts, essent `version` la versió emprada), simplement amb:

```
# make install
```

Per a aquest últim pas, la majoria de distribucions inclouen un suport extern amb un *script* denominat *installkernel* (normalment proporcionat pel paquet *mkinitrd*), que el fa servir el sistema de compilació del kernel per a col·locar-lo al lloc adequat, i modificar el *bootloader* (LILO o Grub), de manera que no s'hagin de fer passos extra. Cal indicar que algunes distribucions de tipus *from scratch*, com *Gentoo*, no inclouen l'script, i per tant cal consultar la documentació específica de kernel per a aquestes distribucions.

En el procés d'aquesta última fase *install*, es desenvolupen les fases següents:

- Es verifica que el kernel hagi estat ben construït.

Cas Debian

Algunes distribucions permeten modes més simplificats; per exemple, Debian permet mitjançant *make deb-pkg* portar a terme tot el procés de compilació i preparar la instal·lació mitjançant la creació dels paquets `.deb` per a la instal·lació del kernel. Finalment, només quedarà instal·lar amb *dpkg -i* els paquets resultants.

- S'instal·la la porció estàtica del kernel en el directori `/boot`, i posa nom al fitxer amb la versió del kernel que s'ha construït (típicament haurà de ser `/boot/vmlinuz-version`).

- Les imatges de ramdisk que puguin ser necessàries són creades fent servir aquells mòduls que s'hagin creat de manera prèvia i siguin necessaris per a aquestes imatges quan arrenqui la màquina. Típicament, aquesta imatge apareixerà com a `/boot/initrd.img-version`. Respecte a la creació de *initrd*, cal disposar de les eines adequades; en Debian, per exemple, en el procés de compilació és necessari com a prerequisit el paquet *initramfs-tools* (per a kernels superiors a 2.6.12, antigament complien aquesta funció les *mkinitrd-tools*, ja obsoletes). També durant aquestes primeres fases s'acostumen a generar el fitxer `/boot/System.map-version` i el fitxer `/boot/config-version`, que contenen, de manera respectiva, informació sobre els símbols disponibles en el kernel i la configuració feta servir en el kernel en la seva compilació.

- El *bootloader* (LILO/Grub) corresponent del sistema rep la notificació de l'existència del nou kernel, i es crea l'entrada necessària del menú, de manera que l'usuari pugui seleccionar-lo en la següent arrencada de la màquina.

Finalment, després d'aquestes fases, el kernel està instal·lat correctament i podem procedir al reinici, i provar la nostra imatge nova del kernel. Si el procés s'ha fet correctament, disposarem d'una entrada de *bootloader* corresponent i, en cas que sorgissin problemes amb la nova imatge, disposaríem de les entrades dels kernels antics per a tornar a la situació estable anterior.

Com a processos alternatius, ja sigui per complementar la instal·lació, o bé per realitzar-la si apareixen problemes des del `make install`, cal tenir en compte el següent. En la instal·lació, depenent del kernel i del seu ús posterior, també pot ser necessària la instal·lació dels *headers* de desenvolupament del kernel, i dels *firmwares* associats (els últims kernels han inclòs alguns *firmwares* de dispositius directament en el kernel). Aquests últims processos els desenvoluparem amb:

```
# make headers_install
# make firmware_install
```

I, d'aquesta manera, finalitzarem el procés d'instal·lació del kernel complet. En aquest últim cas els *firmwares* són instal·lats en */lib/firmware*, i els *headers* necessaris en el directori mateix de les fonts, al subdirectori *usr/include/linux*.

Aquest últim procés, d'instal·lació automàtica mitjançant *make install* (i afegits), també es pot fer de manera manual, si no es disposa de l'*script* abans comentat (*installkernel*) amb el següent procés (alguns detalls depenen de l'arquitectura):

```
# make modules_install
# make kernelversion
```

(L'última ordre obté un número KERNEL_VERSION de la versió construïda, en el que segueix s'ha de col·locar el número obtingut on aparegui KERNEL_VERSION.)

En algunes últimes branques 3.x/4.x cal substituir el directori i386 de la següent ordre per x86, on trobarem la versió binària comprimida del nucli compilat.

```
# cp arch/i386/boot/bzImage /boot/bzImage-KERNEL_VERSION
# cp System.map /boot/System.map-KERNEL_VERSION
```

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-versión-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 fallades o un mal funcionament del nou i així poder recuperar el vell. El fitxer `System.map`, fitxer que conté els símbols disponibles en el nucli, necessari per al procés d'arrencada del nucli, també es col·loca en el mateix directori `/boot`.

També pot ser necessària la creació de la imatge de disc ramdisk `initrd` amb les utilitats necessàries segons la distribució (o la seva versió).

Respecte a la creació del `initrd`, en Fedora / Red Hat aquest es crearà de manera automàtica amb l'opció `make install`. En Debian haurem de fer servir les tècniques conegudes com a *Debian Way* o bé crear-lo de manera explícita amb `mkinitrd` (versions de nucli $\leq$ 2.6.12) o, posteriorment, amb kernels actuals utilitzar, bé `mkinitramfs`, o una eina denominada `update-initramfs`, indicant la versió del nucli (s'assumeix que aquest es diu `vmlinuz-version` en el directori `/boot`):

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

Tot i que cal indicar que en les distribucions actuals, i versions del kernel, és habitual que el initramfs es creï sol en el procés de compilació i posterior instal·lació (en el *make install* igualment). Tot i així, es recomana la remodelació d'aquest (via prèvia configuració en */etc/initramfs-tools/initramfs.conf*), ja que acostuma a presentar una mida respectable, que pot ser ineficient en algunes màquines perquè inclou més mòduls dels imprescindibles. Per a això, es pot jugar (en el mencionat initramfs.conf) amb la configuració *MODULES=most* o *dep* (que segurament minimitzarà la mida de manera significativa). Després, podem recrear el initrd mitjançant l'ordre anterior o, per exemple, substituint els paràmetres per:

```
#update-initramfs -t -u -k version
```

que en aquest cas ens actualitzarà el initrd si el procés d'instal·lació ja en va crear un de previ.

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

***Aquest pas només s'ha de fer si la instal·lació final és manual; d'usar-se `make install` les últimes branques del nucli ja permeten la inclusió del nou nucli en el *bootloader*.**

- Si s'arrenca amb LILO** [Skoa], ja sigui en el Master Boot Record (MBR) o des de partició pròpia, cal afegir-hi el fitxer de configuració (localitzat en `/etc/lilo.conf`), per exemple, les línies:

  ****Actualment és gairebé omnipresent Grub2 com a sistema d'arrencada. Aquí esmentem LiLo per raons històriques.**

  ```
  image = /boot/bzImage-KERNEL_VERSION
      label = KERNEL_VERSION
  ```

  on `image` és el nucli que s'arrencarà i `label` serà el nom amb què apareixerà l'opció en l'arrencada. Podem afegir-hi aquestes línies o modificar les que hi hagués del nucli antic. Es recomana afegir-les i deixar el nucli antic per a recuperar-lo si hi ha 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"` on s'indica el label per defecte que s'arrencarà. També podem afegir a les línies anteriors un `"root=/dev/..."` per a indicar la partició de disc on estarà el sistema d'arxius principal (el /). Recordeu que els discos tenen dispositius com ara `/dev/sda` (primer disc) `/dev/sdb` (segon), i la participació s'indicaria com `"root=/dev/sda2"` si el / del nostre Linux estigués en la segona partició del primer disc. A més 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 la següent arrencada:

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

- Arrencada amb Grub: en aquest cas, la gestió és molt simple; cal introduir una nova configuració formada pel nucli nou i afegir-la com una opció més en el fitxer de configuració del Grub, i tornar a arrencar procedint de manera semblant a la del LILO, però recordant que en Grub n'hi ha prou d'editar el fitxer i tornar a arrencar. També és millor deixar l'antiga configuració per a recuperar-se de possibles errors o problemes amb el nucli acabat de compilar. Cal assenyalar que la configuració de Grub (normalment disponible en `/boot/grub/menu.lst o grub.cfg`) depèn molt de la versió, tant si és Grub-Legacy (Grub 1.x) com Grub 2. Es recomana examinar la configuració de kernels ja presents, per a adaptar l'acabat de crear, i consultar la documentació GNU i el manual disponible via *info grub*.

Lectura recomanada

Sobre Grub, podeu consultar *Grub Manual* accessible des del web del projecte GNU:
https://www.gnu.org/software/grub/

https://www.gnu.org/software/grub/manual/

Un cop disposem dels fitxers correctes en `/boot` i del *bootloader* actualitzat, ja podem procedir a tornar a arrencar 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 ara `/var/log/messages` (o *logs* equivalents, depenent de la distribució) i l'ordre `dmesg`, per a examinar el registre de sortida de missatges produïts pel nou nucli en l'arrencada i detectar si ha sorgit algun problema de funcionalitat o amb algun dispositiu concret.

Si tinguéssim problemes, podem recuperar l'antic nucli, escollir l'opció del vell nucli i després retocar el `bootloader` per a tornar a l'antiga configuració o estudiar el problema i reconfigurar i compilar el nucli una altra vegada.

3.2. Compilació del nucli en Debian (*Debian Way*)

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

Enllaç d'interés

Per a Fedora, recomanem consultar el següent 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`, `bzip2` i `unxz`.

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 en: http://kernel-handbook.alioth.debian.org/

Pel que fa als paquets de fonts del nucli, Debian proporciona els paquets font utilitzats en la seva distribució, que poden arribar a ser molt diferents dels de la versió *vanilla* o *pristine* obtinguda com a estable de *kernel.org*. Això es deu al fet que en Debian es produeixen múltiples revisions amb diversos pedaços que s'hi van afegint, molts d'aquests 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.xx o 3.xx, de manera que el nucli acostuma a quedar-se (generalment) en aquesta numeració per a tota la vida de la versió concreta de Debian estable i, així, quan s'actualitza amb revisions menors la distribució, només s'actualitzen pedaços del nucli (sense canviar el número principal). Quan Debian produeix la següent versió estable, se salta a una nova versió del nucli. Mentre dura 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

repositori Debian una versió `linux-source-version` que conté la versió de les fonts del nucli amb els últims pedaços aplicats (vegeu l'apartat 4). Aquesta versió del paquet de les fonts del nucli és la que farem servir per a crear un nucli personalitzat, en l'esmentada *Debian Way*. Aquest paquet font és usat per a crear els paquets binaris de la distribució associats al nucli i també és l'indicat per a fer servir en cas de voler aplicar pedaços al nucli actual de la distribució, o per si volem provar modificacions del nucli en un nivell de codi.

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

Per a la reconstrucció del kernel actual Debian, podem procedir de dues maneres: o bé obtenim el kernel directament de les fonts (amb tots els pedaços inclosos), o bé reconstruïm els paquets Debian oficials del kernel.

En el primer cas, obtenim les fonts directes i per a la compilació del nucli actual procedim usant el paquet disponible com *linux-source-version*; en aquest cas d'exemple version=3.2 (versió principal del kernel per a una Debian estable concreta) com a versió del kernel Debian de la distribució, però depenent en cada moment del kernel actual, haurem d'escollir les fonts que es corresponguin amb la versió principal del nostre kernel actual (vegeu *uname -r*):

```
# apt-get install linux-source-3.16
$ tar -Jxf /usr/src/linux-source-3.16.tar.xz
```

Nota

Per a la segona comanda hem suposat un format `xz`. En les últimes branques s'utilitza `xz`; en anteriors podia ser `bzip2`, amb la qual cosa l'opció del `tar` passaria de `J` a `j` per al format bzip2.

En aquest cas les descarregarà normalment en `/usr/src` en forma comprimida, encara que podem copiar-les a un altre espai o disc de treball, ja que el procés necessita un espai important (prop de 10 GB o més), si no disposem d'aquest a `/usr/src`. Una vegada tinguem la font, amb tar, descomprimirà els paquets font i els deixarà en un arbre de directoris a partir del directori `linux-version`, on la versió ara serà la revisió del kernel disponible (3.16.xx). Per a la configuració i compilació posterior a partir d'aquest directori, cal fer servir el mètode clàssic (vist en l'anterior subapartat) o la *Debian Way* que examinem en aquest apartat.

Per al segon cas (la reconstrucció dels paquets Debian oficials), l'objectiu és obtenir uns paquets *deb* finals equivalents als que ofereix la distribució, però amb la personalització que vulguem.

Per a aquest procés, començarem per obtenir algunes eines que necessitarem:

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

Aquestes línies bàsicament ens instal·len l'entorn de compilació per al nucli (de fet, els paquets del compilador gcc necessaris i eines pròpies de la *Debian Way*) i finalment es comproven les dependències de les fonts per si són necessaris paquets font extra de desenvolupament. Després d'aquesta configuració d'eines inicials, podem passar a la descàrrega de les fonts (es pot fer des de qualsevol usuari, no és necessari *root* per al procés):

```
$ apt-get source linux
```

que ens descarregarà les fonts i les descomprimirà en el directori actual, a més d'altres paquets que inclouen pedaços respecte a la versió original del kernel. El directori necessari on començar el procés el trobarem com `linux-version`. En aquest moment, podrem personalitzar la configuració del kernel pels mètodes examinats en la secció prèvia, ja que la configuració és la que porta el kernel Debian per defecte, de manera que podrem canviar aquells components o opcions del kernel que ens interessin abans de tornar a reconstruir els paquets.

Per a portar a terme la construcció dels paquets kernel per a l'arquitectura (es recomanen com a mínim 20 GB disponibles de disc), segons la configuració preestablerta del paquet (semblant a la inclosa en els paquets oficials del nucli `linux-image` en Debian):

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

Amb això disposarem dels paquets binaris (i uns altres de suport independents de l'arquitectura) del nucli (fitxers `*.deb`) disponibles per a la instal·lació (via `dpkg -i`).

Hi ha alguns procediments extra per a la creació del nucli sobre la base de diferents nivells de pedaç (*patch*) proporcionats per la distribució, i possibilitats de generar diferents configuracions finals (es pot veure la referència de la nota a propòsit de *Debian Way*, a l'inici d'aquest subapartat, per a complementar aquests aspectes).

Passem ara a l'altra opció que havíem comentat al principi, als aspectes de personalització, quan volem canviar certes opcions del nucli i crear-ne una versió personal (a la qual donarem una numeració de revisió pròpia).

En aquest cas, més habitual, quan desitgem un nucli personalitzat, haurem de seguir un procés semblant mitjançant un pas de personalització típic (per exemple, mitjançant `make menuconfig` o `oldconfig`). Els passos són, en primer lloc, l'obtenció i preparació del directori (aquí obtenim els paquets de la distribució, però és equivalent obtenint els paquets font des de *kernel.org*).

En aquest cas sigui 3.x la versió de kernel disponible de fonts en el repositori, o també començant en el segon pas, el kernel estable obtingut de *kernel.org* (aquest últim potser estigui en format *xz* en lloc de *bzip2*):

```
# apt-get install linux-source-3.16
$ tar -xvJf /usr/src/linux-source-3.16.tar.xz
$ cd linux-source-3.16
```

d'on obtenim els paquets font i els descomprimim (la instal·lació del paquet deixa l'arxiu de fonts en `/usr/src`; en cas que procedeixi de kernel.org, ja depèn d'on haguem fet la descàrrega inicial). És recomanable que abans de procedir al pas de descompressió amb tar, el portem a terme en un espai de disc amb capacitat suficient (recomanats 10-20 GB).

A continuació, fem la configuració de paràmetres. Com sempre, podem basar-nos en fitxers `.config` que hem utilitzat anteriorment per a partir d'una configuració coneguda (per a la personalització, amb *make oldconfig*, també es pot fer servir qualsevol dels altres mètodes, `xconfig`, `gconfig`, etc.):

```
$ make menuconfig
```

El kernel preparat d'aquesta manera, tret que indiquem el contrari, té la depuració activada, la qual cosa resulta molt útil per a la depuració en cas d'errors (utilitzant eines com *kdump, crash* o *SystemTap*), encara que, d'altra banda, ens augmenta l'espai necessari del kernel per a la seva construcció; quan això no sigui necessari, abans de passar a la construcció podem deshabilitar l'opció d'incloure informació de depuració:

```
$ scripts/config --disable DEBUG_INFO
```

A continuació, la construcció final del nucli:

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

on vam crear 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 dels paquets associats a la imatge del nucli, que podrem finalment instal·lar (examinar dels passos anteriors el número del paquet obtingut, que serà el que inclourem aquí):

```
# dpkg -i ../linux-image-3.xx.yy_custom.1.0_i386.deb
```

Això ens descomprimirà i instal·larà el nucli i generarà una imatge `initrd` addicional si és necessari. 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).

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

Afegim també una altra peculiaritat que cal tenir en compte en Debian, que pot ser d'utilitat amb algun maquinari: l'existència d'utilitats per a afegir mòduls dinàmics de nucli proporcionats per tercers; en particular, la utilitat `module-assistant`, que permet automatitzar tot aquest procés a partir dels paquets font del mòdul.

Enllaç d'interès

Habitualment no serà necessari, però si es necessita alguna reconfiguració del `initrd` generat, es recomana llegir el següent enllaç, on es comenten algunes eines Debian disponibles: http://kernel-handbook.alioth.debian.org/ch-initramfs.html

Necessitem disposar dels *headers* del nucli instal·lat (disponible en el paquet `linux-headers-version`) o bé dels paquets font que utilitzem en la seva compilació (també es poden obtenir durant la instal·lació del nucli mitjançant un `make headers_install`, o mitjançant el paquet deb obtingut amb *Debian Way*). A partir d'aquí, `module-assistant` es pot utilitzar 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.

En la utilització des de la línia d'ordres, també podem simplement especificar (`m-a`, equivalent a `module-assistant`):

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

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

4. Aplicació de pedaços (*patch*) al nucli

En alguns casos, també pot ser útil l'aplicació de pedaços (*patches*) al 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, depenent de la distribució, en `/usr/src/linux`).

Els pedaços s'acostumen a necessitar quan un maquinari especial necessita alguna modificació en el nucli, o s'han detectat alguns errors (*bugs*) de funcionalitat posteriors a alguna distribució concreta d'una versió del nucli, o bé 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 hi ha uns quants pedaços, s'uneixen amb diverses millores del nucli anterior per a formar-ne una nova versió. En tot cas, si tenim maquinari problemàtic o l'error afecta la funcionalitat o l'estabilitat del sistema i no podem esperar a la següent versió del nucli, serà necessari aplicar els pedaços.

El pedaç s'acostuma a distribuir en un fitxer comprimit tipus bz2 (bunzip2, encara que també pot trobar-se en gzip amb extensió .gz, o xz amb extensió .xz), com per exemple podria ser:

```
patchxxxx-version-pversion.xz
```

on xxxx sol ser algun missatge sobre el tipus o finalitat del pedaç, `version` 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 més d'una. Cal tenir en compte que estem parlant d'aplicar pedaços als paquets font del nucli (normalment instal·lats, com vam veure, en `/usr/src/linux` o directori similar de l'usuari utilitzat en la compilació del nucli).

Una vegada disposem del pedaç, haurem d'aplicar-lo. Veurem el procés que cal 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-version, o directori que fem servir).
unxz patch-xxxxx-version-pversion.xz
patch -p1 < patch-xxxxx-version-pversion
```

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

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

Els pedaços es poden obtenir de diferents llocs. El més normal és trobar-los en el lloc de magatzem dels nuclis *vanilla* (http://www.kernel.org), que en té un arxiu complet. Determinades comunitats Linux (o usuaris individuals) també solen oferir algunes correccions, però és millor buscar en els llocs estàndard per a assegurar un mínim de confiança en aquests pedaços i evitar problemes de seguretat amb possibles pedaços "pirata". Una altra via és el fabricant de maquinari que pot oferir certes 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 propietaris per a les seves targetes gràfiques).

Finalment, assenyalarem que moltes distribucions de GNU/Linux (Fedora / Red Hat, Debian) ja ofereixen nuclis pedaç 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 l'últim nucli publicat, sinó el que creï 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 test o de necessitat de les últimes prestacions, sempre es pot acudir a http://www.kernel.org i obtenir el nucli més modern publicat, quan la compatibilitat de la distribució ho permeti.

Finalment, cal comentar la incorporació d'una tecnologia comercial a l'ús de pedaços en Linux, Ksplice (mantinguda per Oracle), que permet a un sistema Linux afegir pedaços al nucli sense necessitat d'aturar i tornar a arrencar 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 en memòria com afecten la imatge del nucli en memòria que s'està executant. S'intenta llavors aturar l'execució en el moment en què no hi hagi dependències de tasques que necessitin les parts del nucli que cal apedaçar. Llavors es procedeix a canviar, en el codi objecte del nucli, les funcions afectades, apuntant les no-

Ksplice

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

ves funcions amb el pedaç aplicat i modificant dades i estructures de memòria que hagin de reflectir els canvis. Actualment és un producte comercial (d'Oracle per a la seva distribució Oracle Linux); altres productes semblants estan apareixent en diverses distribucions empresarials comercials. En els casos de producció a empreses, amb servidors en els quals és important no disminuir el temps de servei, pot ser una tecnologia crítica per a fer servir, altament recomanable tant per a disminuir el temps de pèrdua de servei com per a minimitzar incidents de seguretat que afectin el nucli.

Bàsicament, ofereixen un servei denominat *Ksplice Uptrack* que és una espècie 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 es puguin incorporar 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.

SUSE també va anunciar la seva idea d'integrar una tecnologia semblant, denominada *kGraft*, com a mecanisme per a aplicar pedaços al kernel sense reiniciar el sistema, de manera semblant a *Ksplice*. Tanmateix, en aquest cas la idea dels desenvolupadors és intentar incloure-la en la línia principal del kernel, de manera que estaria disponible com a part del codi del kernel. En aquesta tecnologia, de la qual ja hi ha algunes versions que s'estan integrant en les versions de desenvolupament del kernel, bàsicament el pedaç s'integra com un mòdul del kernel (.ko), el qual és carregat amb *insmod*, i reemplaça les funcions kernel afectades pel pedaç, fins i tot durant el temps d'execució d'aquestes funcions. La idea de kGraft és permetre arreglar *bugs* crítics de kernel sense afectar els sistemes que necessiten gran estabilitat i disponibilitat.

kGraft

Tecnologia de SUSE per a pedaços en execució del kernel: https://www.suse.com/promo/kgraft.html

També existeix la tecnologia kpatch de Red Hat, que juntament amb kGraft de SUSE, s'ha començat a introduir en les branques 4.x del nucli, amb la qual cosa s'espera que en futures iteracions de la branca 4.x es disposi d'aquesta funcionalitat.

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) a 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, a 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`, encara que en les recents distribucions s'ha migrat a una estructura de subdirectoris on s'inclou un fitxer d'opcions per mòdul, que es pot veure en el subdirectori `/etc/modprobe.d`.

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 saber 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`):

- `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.

  Exemple

  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 és 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` (que, mitjançant un disc RAM en memòria principal, implica una precàrrega en temps d'arrencada del sistema) o mitjançant `udev` (en situacions dinàmiques d'ús de dispositius removibles, o de connexió en calent).

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` (o una posterior ordre *update-initramfs*) permet crear-lo a partir de les seves opcions genèriques, que es poden configurar en l'arxiu

```
/etc/initramfs-tools/initramfs.conf
```

i, de manera específica, els mòduls que es carregaran en inici automàticament, i que podem trobar-los en */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 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. En aquest cas s'utilitzen tècniques de llistes negres de mòduls (típicament la llista negra es desa a `/etc/modprobe.d/blacklist.conf`), tot i que es pot crear en un fitxer arbitrari en aquest directori, simplement afegint al fitxer creat una línia *blacklist nommodul*.

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, o si estava configurat en arrencada, un cop produïda aquesta, el nou mòdul s'utilitzarà.

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); normalment n'hi ha prou amb els *headers*, però depenent del mòdul poden ser necessaris més paquets de fonts.

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-lo, 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/nom-modul` 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` i `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 es poden haver de modificar 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 i la seva integració amb entorns *cloud* de tipus públic/privat o híbrid, com veurem més endavant. 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 d'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, per a observar diferències entre aquestes o personalitzar les nostres tasques o *scripts* per a cada distribució.

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

- KVM
- Xen

- LXC
- OpenVZ
- VirtualBox
- VMware

VMware és un dels líders comercials en solucions de virtualització, i disposa de productes de virtualització per a escriptori (VMware Workstation / Fusion), mentre que per a servidor disposa d'un producte VMware vSphere Hypervisor (ESXi) 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. Altres línies de productes VMware implementen necessitats més grans per a centres de dades (*data centers*) i entorns de *cloud computing*, 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

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).

LXC (Linux Containers) és un sistema de virtualització a escala operativa que permet córrer múltiples sistemes Linux de manera aïllada. En lloc de crear màquines virtuals completes, es basa en l'ús d'una funcionalitat del kernel de Linux, denominada cgroups, juntament amb *chroot*, que permet encapsular un espai d'execució de processos i gestió de xarxa de manera aïllada. La idea és similar a OpenVZ, però en aquest cas no necessita actuacions de mòduls o pedaços de kernel, ja que funciona sota el kernel *vanilla*, sense modificacions. Alguns sistemes per desplegament de contenidors, com *Docker*, l'utilitzaren com a base per a implementar els contenidors (encara que ara depenen d'altres solucions com *libcontainer*).

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, o svm en processadors AMD, 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

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

Enllaç d'interès

Enllaç web de KVM a:
http://www.linux-kvm.org

existent, que es complementa amb altres solucions. És l'opció per defecte per a la virtualització en distribucions com Debian i Fedora.

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

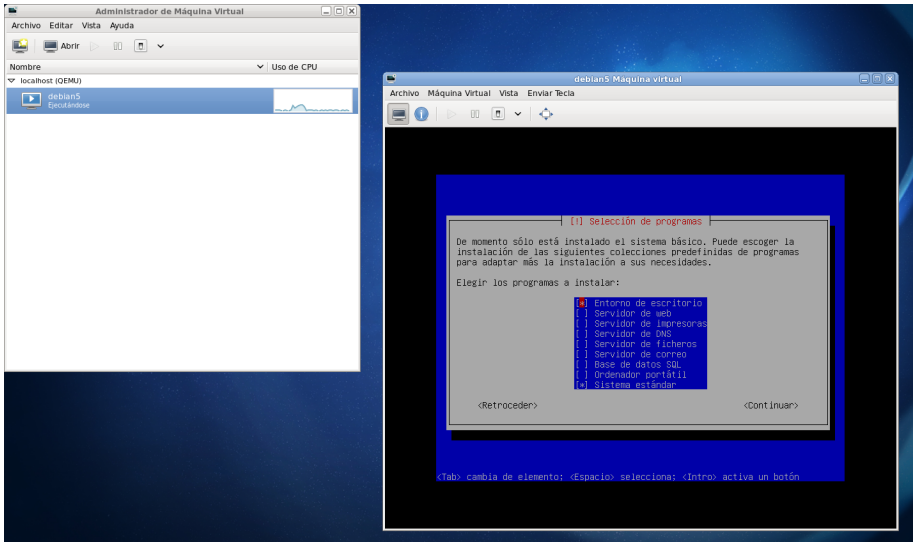
- El mòdul de nucli `kvm.ko`, que proporciona la infraestructura base de virtualització, i, a més, un mòdul addicional segons el model de processador, `kvm-intel.ko` o `kvm-amd.ko`. Aquests mòduls s'hauran de carregar manualment (`modprobe`) o habilitar-los durant l'arrencada per a disposar de suport 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` (en algunes distribucions `libvirt-bin`). Hem d'assegurar-nos de posar en marxa el servei (`/etc/init.d/libvirtd start`, `/etc/init.d/libvirt-bin start` o equivalents amb Systemd) 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-bin`, `virt-viewer` entre altres. 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, ja que la biblioteca libvirt ens permet abstreure diversos sistemes diferents de virtualització.

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):

```
# adduser <usuari> libvirt
```

En aquest moment, podem executar l'ordre de *shell* per a la gestió de màquines KVM, `virsh`), que ens permetria veure les disponibles:

```
$ virsh --connect qemu:///system list
```

això permet connectar-se al gestor local, i preguntar-li per la llista de màquines disponibles.

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:

```
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
```

Enllaç d'interès

Per altra banda, un tema important és la gestió de xarxa, que és un dels temes complexos en virtualització; en el cas de KVM es recomana examinar: http://www.linux-kvm.org/page/Networking.

Aquesta configuració permet que es creï un dispositiu `br0` per a reemplaçar `eth0`. Així, les targetes de xarxa virtuals redirigiran el trànsit assignat 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-n'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 o d'arxius d'imatges ISO de la distribució Debian).

Creem l'espai de disc (en el directori actual) 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 la 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 CD/DVD amb

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

Cal vigilar que el dispositiu, `/dev/cdrom` en aquest cas, sigui el correcte, ja que segons la distribució pot canviar.

que ens genera la imatge del CD/DVD o per contra directament descarregar la imatge ISO de la distribució.

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` (en algunes distribucions, la comanda kvm ja no es fa servir o només existeix per compatibilitat):

```
$ qemu-system-x86_64 --enable-kvm -m 512 -cdrom debian-install.iso \
-boot d -hda debianVM.img
```

En aquest cas, crearia una màquina virtual (arquitectura x86_64) 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 és 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 debian -r 512 \
--vcpus=2 -f debianVM.img -s 8 -c debianinstall.iso --vnc \
--noautoconsole --os-type linux --os-variant debianwheezy
```

que entre altres paràmetres, col·loca el mètode de connexió a la màquina virtual, el nom de la VM `debian`, defineix 512 MB de memòria, fa disponibles 2 nuclis de la màquina física, utilitza el disc virtual `debianVM`, de 8 GB (`-s` permet que si no existeix, el creï de manera prèvia 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, especialment 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 seva 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 pot trobar-se en: http://www.linux-kvm.org/page/guest_support_status

En la comanda anterior de `virt-install` s'ha utilitzat la configuració de xarxa virtual per defecte, l'anomenada *default*. Si apareix algun error per no estar correctament inicialitzada, podem realitzar-lo amb:

```
virsh net-start default
```

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. Amb `virt-manager` podem afegir la connexió a la màquina de la qual hem creat la imatge, per exemple en *localhost*, fet que ens permetrà després tenir-la accessible mitjançant connexió a *qemu:///system*, i també arrencar-la de seguida.

Després podem visualitzar la consola (de text o gràfica) de la màquina acabada de crear mitjançant

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

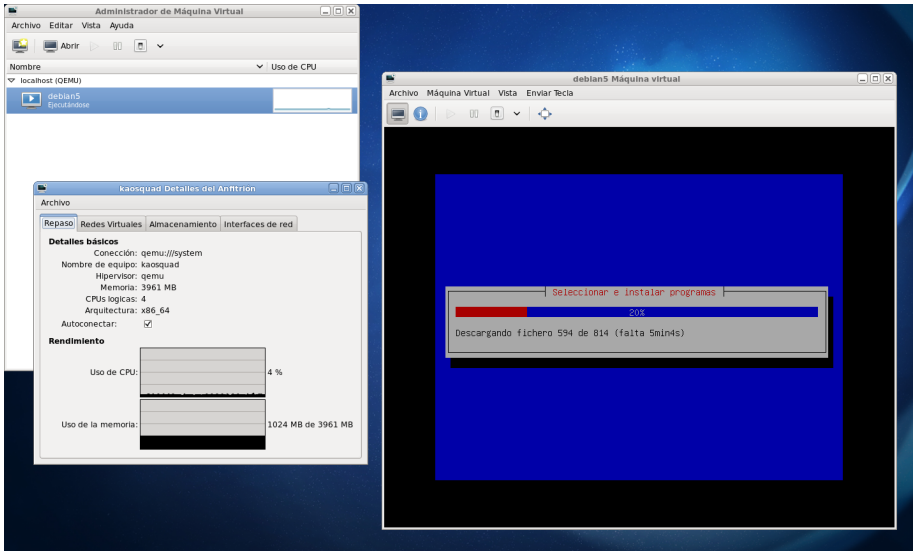
si és al mateix sistema *localhost* o, si som en una màquina remota, amb

```
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 `qemu-system-x86_64`). 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

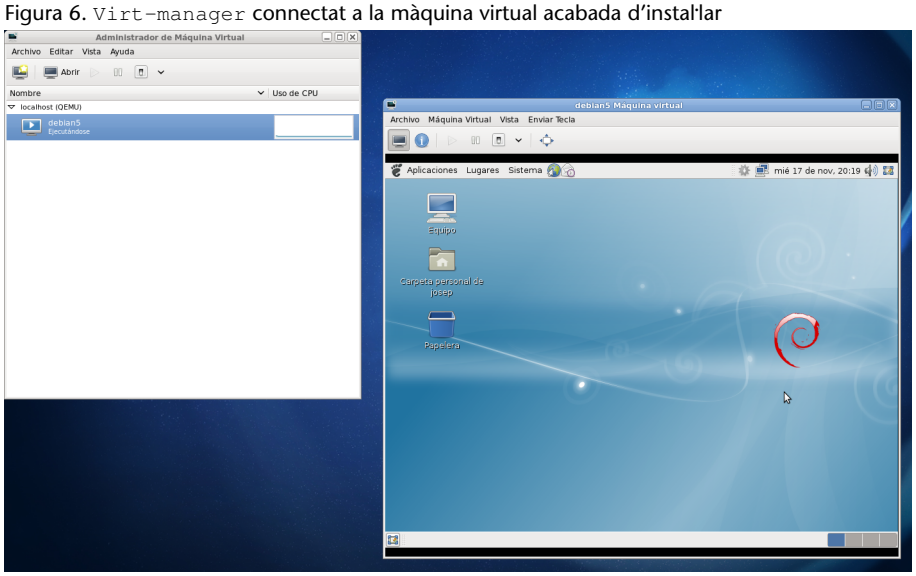


Per exemple, amb `virsh`:

```
virsh --connect qemu:///system
```

connectem amb el *shell*, on 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



6.2. VirtualBox

VirtualBox és una solució de virtualització d'escriptori que està obtenint resultats importants, un projecte en aquests moments mantingut per Oracle.

En el cas de Debian, que utilitzarem, està disponible en el repositori oficial. Però per a altres distribucions, poden trobar-se en https://www.virtualbox.org /wiki/linux_downloads paquets preparats per a un gran nombre de distribucions, i repositoris propis que cal afegir a les distribucions.

Aquesta solució de virtualització inclou tant eines de tipus gràfic com utilitats en el nivell de línia d'ordres per a virtualitzar màquines de 32 bits i 64 bits d'arquitectura Intel (x86 i x86_64). VirtualBox s'ofereix majoritàriament amb llicència GPL; tot i així hi ha una part addicional propietària, denominada *Extension Pack*, que ofereix lliure de cost Oracle per a ús personal. Aquest paquet addicional ofereix algunes facilitats extra relacionades amb emulació de maquinari extra (USB2, per exemple, per defecte només s'ofereix USB1 sense pack), i ofereix accés gràfic a les màquines hoste per mitjà d'RDP (*remote desktop protocol*).

Per a la instal·lació en Debian, s'ha d'instal·lar el paquet *VirtualBox* i és recomanat disposar dels *headers* corresponents al kernel actual (paquet *linux-headers-version* i la versió és la corresponent al kernel actual). Els *headers* s'utilitzaran per a la compilació d'una sèrie de mòduls que VirtualBox instal·la dinàmicament en el kernel. Normalment, la instal·lació proporciona també una configuració de *dkms* per a aquests mòduls que permet que es recompilin després de canvis de kernel en la distribució. En algunes distribucions, de vegades passa un temps fins que estan disponibles els mòduls actualitzats per a una versió concreta de kernel; per tant, abans d'actualitzar el kernel en un sistema amb virtualització VirtualBox, és recomanable comprovar que la versió de kernel ja té el suport adequat.

```
# apt-get install linux-headers-version virtualbox
```

VirtualBox es pot arrencar ja mitjançant l'ordre `virtualbox`. Els mòduls, si teníem els *headers* adequats, s'hauran compilat i carregat en el sistema, i es carregaran en cada arrencada. Si no desitgem que s'arrenquin automàticament, llavors podem canviar en `/etc/default/virtualbox` el paràmetre LOAD_VBOXDRV_MODULE i posar-lo a 0.

Un cop arrencat l'entorn gràfic amb `VirtualBox`, podem crear una nova VM mitjançant l'opció de la icona "Nova", que ens arrencarà l'assistent de creació de màquines virtuals. El procés passa per determinar:

- L'operatiu de la màquina hoste i el seu nom.

- La memòria del nostre sistema, que proporcionarem a la màquina hoste; generalment, es recomana que el total de màquines VM no consumeixin més del 50% de la memòria. Depenent de l'ús, és freqüent de 512 MB a 2 GB per màquina virtual.

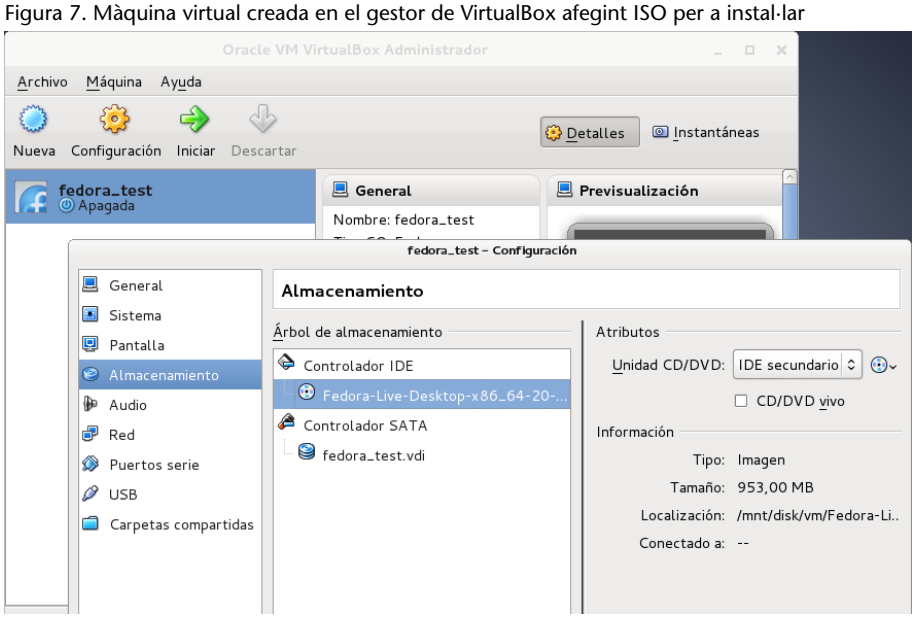
- Creació del disc dur virtual de la màquina. Podem crear un disc nou o fer servir una imatge de disc prèviament creada per a la nova màquina.

- Format de la imatge del disc. En VirtualBox, per defecte, és VDI. Tanmateix, podem crear-ne d'altres, que potser puguem compartir amb altres sistemes de virtualització.

- Assignem la grandària del disc virtual de manera dinàmica o reservem el total de l'espai. Això ens permetrà reservar espai de disc a mesura que es vagi fent servir o, per contra, tenir un disc més ràpid, la qual cosa ens permetrà millors prestacions del disc virtual.

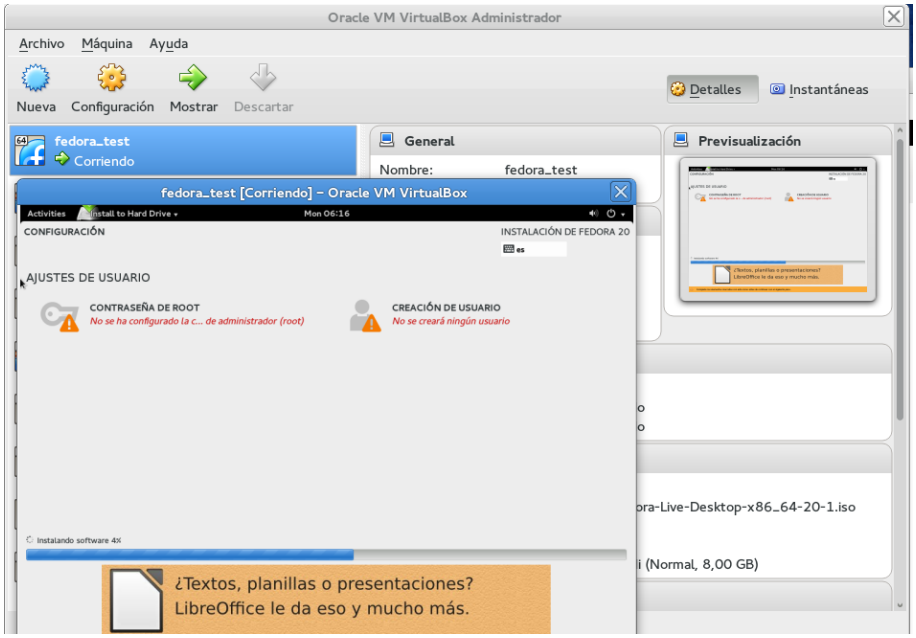
Amb aquest procés, ja disposem d'una màquina creada i podem retocar la seva configuració seleccionant-la; especialment, podem connectar en emmagatzematge una imatge ISO d'una distribució al seu CD-ROM virtual. Això ens permetrà arrencar la màquina des de CD, arrencant així la imatge d'instal·lació. En el procés inicial, també podem canviar la configuració de xarxa; hi ha diverses maneres disponibles: des del NAT, per defecte, a maneres que permeten xarxes privades per a les màquines VM, o incloure una IP pública per a la nostra màquina VM (vegeu documentació VirtualBox per a l'administració de xarxa http://www.virtualbox.org/manual/usermanual.html). Vegeu en la figura 7 una màquina creada, amb la seva secció de configuració d'emmagatzematge a la qual s'acaba de connectar una imatge ISO (d'una distribució Fedora), si procedim a l'arrencada de la màquina amb *Iniciar*. En la figura 8 podem veure la màquina hoste durant el procés d'instal·lació.

Figura 7. Màquina virtual creada en el gestor de VirtualBox afegint ISO per a instal·lar



Una vegada disposem de la VM amb sistema operatiu instal·lat, ja la tindrem funcional. Per a millorar el rendiment de les màquines hoste, VirtualBox també proporciona una sèrie de controladors per al sistema hoste, coneguts com a *VirtualBox Guest Additions*, que permeten millores de la gestió de la xarxa

Figura 8. Màquina virtual en el procés d'instal·lació d'una distribució GNU/Linux



virtual, de l'acceleració dels gràfics i la possibilitat de compartir carpetes amb la màquina *host* (amfitrió). Una vegada arrencada i instal·lada la VM, des del menú Dispositius -> Inserir imatge CD Guest Additions, es poden instal·lar.

Cal assenyalar que per realitzar aquest procés, que necessita instal·lar en el sistema alguns controladors addicionals, cal disposar del paquet dels *headers* (`linux-headers` o `kernel-devel` segons la distribució). A més a més, és recomanable tenir instal·lat el paquet `dkms`, que ens permetrà a posteriori que aquests controladors es recompilin sota canvis de nucli del sistema.

Cal assenyalar que, a més de tot aquest procés, determinats proveïdors proporcionen imatges VDI de màquines VirtualBox ja preparades per al seu funcionament, perquè no hàgim de passar per tot el procés de creació. Normalment són imatges denominades *cloud*, i haurem de buscar imatges VDI ja preparades o altres formats compatibles amb VirtualBox. No obstant això, es pot incórrer en certs riscos de seguretat. Convé evitar l'ús de VM ja personalitzades si el proveïdor no és de confiança o no és directament la mateixa distribuïdora; a més, algunes màquines VM arbitràries podrien contenir problemes de seguretat o introduir *malware* o eines destinades a l'atac dels nostres sistemes. Per això, es recomana fer servir només imatges de proveïdors adequats.

Per acabar, cal assenyalar el tema del control mitjançant interfície d'usuari. VirtualBox proporciona les ordres `vboxmanage` i `VBoxHeadless` per permetre'ns la major part de gestió de les màquines des de línia d'ordres, i fins i tot si ja disposem d'imatges VDI, podrem fer tota la gestió sense cap ús d'interfície gràfica.

Un resum d'algunes ordres interessants:

- Arrencar la màquina en mode servidor sense interfície VirtualBox; en arrencar, ens donarà el port de connexió al servidor gràfic (VRDE). També amb vboxmanage, amb interfície GUI i sense interfície:

```
$ VBoxHeadless -startvm "nom_maquina"
$ vboxmanage startvm "nom_maquina"
$ vboxmanage startvm "nom_maquina" --type headless
```

- Informació de la màquina virtual, els detalls de la seva configuració, discos, xarxa, memòria, CPU i altres:

```
$ vboxmanage showvminfo "nom_maquina"
```

- Apagar la màquina:

```
$ vboxmanage controlvm "nom_maquina" poweroff
```

- Modificar el servei de pantalla:

```
$ vboxmanage modifyvm "nom_maquina" --vrde on --vrdeport
port_VRDE --vrdeaddress ip_maquina
```

- Connexió gràfica a la màquina executant-se. Si la màquina està en NAT sense IP assignada, la ip=0.0.0.0 per a la connexió, si no la IP corresponent. Per a ple funcionament es necessiten instal·lades les Guest Additions, les quals tenen com a requisit el paquet de *headers* del kernel corresponent a la màquina hoste:

```
$ rdesktop -a 16 -N ip_maquina:port_VRDE
```

- En particular, amb el següent veurem múltiples opcions de control de les VM que tenim disponibles:

```
$ vboxmanage controlvm
```

6.3. Xen

En el cas de Xen, tractem amb un entorn d'hypervisor en un nivell de màquina física que permet córrer múltiples instàncies del sistema operatiu o de diferents operatius en paral·lel en una sola màquina.

Algunes de les característiques de Xen que cal destacar són les següents:

Enllaç d'interès

Es recomana llegir la introducció http://wiki.xen.org/wiki/Xen_Overview, per a comprendre alguns conceptes de virtualització associats a Xen.

- L'hypervisor, o component monitor de màquines virtuals, és relativament petit i consumeix pocs recursos de la màquina.

- Xen és agnòstic del sistema operatiu. La majoria d'instal·lacions usen Linux com a sistema base; Xen aprofita els serveis bàsics, el denominat *domini 0* de Xen. Dom0 és una VM especialitzada amb privilegis per a l'accés directe al maquinari. Com a base per a Xen, es poden fer servir altres operatius, com ara alguns de la família BSD.

- Xen és capaç d'aïllar el funcionament d'un *driver* de sistema en una màquina virtual. Si aquest falla o és compromès, la VM que conté el *driver* es pot tornar a arrencar sense afectar la resta del sistema.

- És possible distingir dos modes de funcionament. Un és el denominat *Xen Full Virtualization* (HVM), que fent servir extensions maquinari de virtualització (les HVM), utilitza emulació de PC (Xen es basa en Qemu per a això); en aquest cas no es requereix suport de kernel, però per contra acostuma a ser una solució de menor rendiment a causa de l'emulació necessària. Per altra banda, el mode de funcionament *Xen Paravirtualization* (PV) és una virtualització eficient i lleugera en ús de recursos que no requereix extensions HVM, però sí disposar de kernels i *drivers* amb suport de PV, de manera que l'hypervisor es pot executar de manera eficient sense emulació de màquina o emulació de components virtuals de maquinari.

- El suport de Paravirtualization. Els hostes d'aquest tipus són optimitzats per a poder executar-se com a màquina virtual (alguns estudis han proporcionat només sobrecàrregues entre el 2% i el 8%, mentre que en emulació les penalitzacions s'acostumen a apropar a un 20%), i donen millor rendiment que altres sistemes que es basen en components addicionals. A més, Xen fins i tot es pot executar en maquinari que no suporta extensions de virtualització.

  Linux disposa del suport de paravirtualització des de la versió 2.6.23, anomenat *paravirt_ops* (o simplement *pvops*). Es tracta d'un kernel Linux de Domini 0 (disponible per a arquitectures x86, x86_64 i ia64). Aquest dom0 és el sistema que tenen els controladors de dispositiu per a comunicar-se amb el maquinari subjacent, executa les eines d'administració de Xen i proporciona els discos virtuals i el subsistema de xarxa virtual a la resta de sistemes hoste (denominats *domU*).

  També cal esmentar que en les últimes generacions de processadors ha millorat sensiblement el suport i les possibilitats de les extensions HVM, la qual cosa ha fet viable noves aproximacions híbrides per a passar de virtualització PV a PVHVM; bàsicament consisteix en hoste HVM, però amb *drivers* especials per als discos i xarxa.

Enllaç d'interès

Una referència per a explicar el concepte de PVHVM: http://www.slideshare.net/fullscreen/xen_com_mgr/linux-pv-on-hvm/

- Com que necessitem de base l'esmentat Domini 0, Xen necessitarà portar els sistemes operatius per a adaptar-se a l'API de Xen, a diferència de les VM tradicionals, que proporcionen entorns basats en programari per a simular el maquinari. En aquests moments hi ha ports (dels kernels pvops) per a NetBSD, FreeBSD i Linux, entre d'altres.

- Gràcies al codi aportat per Intel i AMD a Xen, s'han adaptat diferents extensions de virtualització de maquinari que permeten que els sistemes operatius sense modificacions s'executin en màquines virtuals Xen, la qual cosa ha aportat millores de rendiment i ha ofert la possibilitat d'executar sistemes GNU/Linux i Windows sense cap modificació.

- Hi ha suport per a la migració de màquines VM en calent entre equips físics. En aquest sentit, una màquina virtual en execució pot ser copiada i traslladada d'equip físic sense aturar l'execució, només per a petites sincronitzacions.

A continuació, veurem alguns usos simples de Xen sobre una distribució Debian, que disposa de la particularitat que és fàcil construir VM hostes basades en Debian de manera bastant flexible a partir dels repositoris de Debian.

Començarem per la instal·lació del domini 0 (`dom0`). Bàsicament, es fa amb una instal·lació Debian típica, en la qual únicament es tindran en compte les particions que cal emprar per a reservar espai posterior als hostes. En les versions actuals de Xen se sol reservar 4 GB de disc pel `dom0` (el seu /), i 1 GB de memòria per a la RAM que necessitarà. L'espai extra pot dedicar-se a emmagatzemar les VM hostes, o en el cas de *storage* podem, per exemple, fer servir volums LVM que ens permetran fer créixer els *filesystems* a mesura que els necessitem per part del `dom0` o les domU (hostes).

Per exemple, podríem fer 3 particions (sda1,2,3), i col·locar `/ (root)` i `swap` en les dues primeres i un LVM en la tercera, amb un volum físic i un grup (*vg0*, per exemple). Instal·lem el sistema Debian.

Passem a instal·lar l'hypervisor mitjançant un metapaquet disponible en Debian que ens fa tot el treball de dependències i utilitats necessàries:

```
# apt-get install xen-linux-system
```

Podem comprovar també si la nostra màquina suporta extensions HVM, ja siguin d'Intel o AMD (en alguns casos poden estar desactivades en la BIOS de la màquina, i llavors haurem d'habilitar-les prèviament), amb:

```
egrep '(vmx|svm)' /proc/cpuinfo
```

Si estan disponibles, Xen podrà utilitzar més eficientment la paravirtualització, basant-se en el suport de virtualització dels processadors moderns. En diversos estudis (de *benchmarking*) s'ha comprovat que PV+HVM (sobre PV pura) en diferents *bencharmaks* obté beneficis que van del 15% fins al 200% o 300%.

Benchmarking PVHVM

Algunes referències: https://developer.rackspace.com/blog/welcome-to-performance-cloud-servers-have-some-benchmarks/, https://xen-orchestra.com/debian-pvhvm-vs-pv/ i http://wiki.xen.org/wiki/PV_on_HVM

Un cop feta la instal·lació prèvia del sistema Xen, observarem que la distribució Debian, en la seva arrencada `Grub2` (en aquest cas), té una nova entrada denominada `Xen4`. És la que ens permetrà arrencar la màquina, sota l'hypervisor Xen gestionant el `dom0`. L'entrada Xen no està posada per defecte, amb la qual cosa se seguirà carregant per defecte el kernel del sistema. Amb aquestes instruccions, la posarem per defecte:

```
dpkg-divert --divert /etc/grub.d/08_linux_xen --rename /etc/grub.d/20_linux_xen
update-grub
```

El següent pas serà configurar la xarxa per al domini 0. El més habitual és fer servir un *bridge* per programari (necessitem disposar del paquet *bridge-utils*); un exemple senzill per a `/etc/network/interfícies`:

```
#The loopback network interface
auto lo
iface lo inet loopback

iface eth0 inet manual

auto xenbr0
iface xenbr0 inet dhcp
   bridge_ports eth0

#other possibly useful options in a virtualized environment
  #bridge_stp off       # disable Spanning Tree Protocol
  #bridge_waitport 0    # no delay before a port becomes available
  #bridge_fd 0          # no forwarding delay

## configure a (separate) bridge for the DomUs without
## giving Dom0 an IP on it
#auto xenbr1
#iface xenbr1 inet manual
#   bridge_ports eth1
```

Altres tècniques en la configuració són opcionals, com per exemple: ajustar l'ús de memòria per `dom0`, afegir a `/etc/default/grub` la següent línia per a reservar memòria per al `dom0` (1 GB en aquest cas), i posteriorment un *update-grub*:

```
GRUB_CMDLINE_XEN="dom0_mem=1024M"
```

En Xen es fa servir una tècnica denominada *ballooned* que el que fa és assignar inicialment la majoria de memòria al `dom0`, i a mesura que van apareixent domU la va reduint. Amb això, la fixem de manera estàtica. Cal comprovar que sigui suficient; si patim algun *crash* del kernel Xen cal augmentar-la. També podem obviar aquesta personalització de memòria i deixar la tècnica de *balloning* per defecte. També cal fer el procés equivalent en el fitxer `/etc/xen/xend-config.sxp` amb:

```
(dom0-min-mem 1024)
(enable-dom0-ballooning no)
```

Amb aquests canvis, ja podrem arrencar la màquina de nou amb el `dom0` de Xen disponible amb opcions bàsiques. La configuració de Xen en producció escapa als termes d'espai d'aquesta secció, i recomanem consultar els manuals Xen per a configurar, especialment: a) les CPU disponibles a les màquines *guest* i `dom0`; b) comportament de *guests* en rearrencada; c) també és possible, per a depuració, activar la consola per port sèrie per a missatges de depuració.

Amb la configuració actual, passarem a detallar una instal·lació bàsica de domU (hoste) de tipus Debian, ja que se'ns ofereixen certes facilitats per al procés automàtic:

```
apt-get install xen-tools
```

Així, instal·lem *scripts* d'utilitats que ens proporciona Xen i després modificarem `/etc/xen-tools/xen-tools.conf` per a definir `dir`=/mnt/disk i `passwd`=1, amb l'adreça del pedaç on posarem les nostres màquines creades (en /mnt/disk, en el nostre cas un volum especial per a guardar les màquines hoste –*guests*–).

Amb l'ordre següent, i la configuració prèvia, podrem construir una imatge de Xen, una VM ja preparada amb la imatge de Debian corresponent (en aquest cas, una *wheezy*):

```
# xen-create-image --hostname nom_maquina --ip ip_maquina
--vcpus 1 --pygrub --dist wheezy --genpass=0
```

Col·loquem en l'ordre anterior un nom de VM i una IP disponible (ja sigui pública o privada), i començarà la construcció de la imatge. En aquest cas, s'utilitzen els repositoris de Debian per a descarregar els paquets adequats per a construir la imatge.

Nota

Per defecte, sense l'opció `--genpass = 0`, en l'ordre `xen-create-image`, es genera un *password* de root aleatori, que es mostraria durant la sortida següent, encara que pot obtenir-se *a posteriori* en el *log* de la màquina, `/var/log/xen-tools/deb.log` en aquest cas.

Observarem que ens dóna les característiques per defecte de la màquina, i que ens comença a generar les imatges de les particions de *root* i *swap* en el directori (a partir del nostre cas /mnt/disk), *domains/nom_maquina* on residiran les imatges de la VM *guest* que s'està creant. Finalment ens demanarà el *password* de *root* de la màquina creada i finalitzarà:

```
General Information
--------------------
Hostname       :  deb
Distribution   :  wheezy
Mirror         :  http://mirror.switch.ch/ftp/mirror/debian/
Partitions     :  swap            128Mb (swap)
                  /               4Gb   (ext3)
Image type     :  sparse
Memory size    :  128Mb
Kernel path    :  /boot/vmlinuz-3.2.0-4-amd64
Initrd path    :  /boot/initrd.img-3.2.0-4-amd64

Networking Information
----------------------
IP Address 1   : 10.0.0.2 [MAC: 00:16:3E:2D:D7:A2]

Creating partition image: /mnt/disk/domains/deb/swap.img
Creating swap on /mnt/disk/domains/deb/swap.img
Creating partition image: /mnt/disk/domains/deb/disk.img
Creating ext3 filesystem on /mnt/disk/domains/deb/disk.img
Installation method: debootstrap
Running hooks
Creating Xen configuration file

Setting up root password
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
All done

Logfile produced at:
 /var/log/xen-tools/deb.log

Installation Summary
---------------------
Hostname        :  deb
Distribution    :  wheezy
IP-Address(es)  :  10.0.0.2
RSA Fingerprint :  4e:f3:0a:47:c1:15:14:71:15:28:da:e2:9f:df:42:2b
Root Password   :  N/A
```

En els següents exemples, les ordres utilitzades depenen del *toolstack* o conjunt d'utilitats xen que s'estigui utilitzant. En molts dels casos és possible canviar les ordres `xm` existents per `xl`, depenent de la versió de xen i/o distribució. `xl` és un nou *toolstack* per xen (que manté compatibilitat amb `xm`).

Per a executar la màquina creada (*nom_maquina=deb* en aquest exemple):

```
$ xm create /etc/xen/deb.cfg
```

I per a esborrar una imatge de VM:

```
$ xen-delete-image VMs_name
```

Per a fer una llista dels dominis actius:

```
$xm list
Name              ID   Mem VCPUs      State   Time(s)
Domain-0           0  1023     1     r-----    247.9
deb                2   128     1     -b----      5.4
```

I per a connectar una consola a la màquina disponible (al domini amb ID=2 en aquest cas):

```
$xm console 2
```

Fet que ens permet portar a terme la connexió a la màquina virtual i fer el procés de *login*.

Enlaces de interés

Sobre Toolstacks Xen podeu consultar la següent pàgina web: http://wiki.xen.org/wiki/Choice_of_Toolstacks. Per comparar `xm` i `xl` podeu veure la següent pàgina web: http://wiki.xen.org/wiki/XL_vs_Xend_Feature_Comparison

7. Present 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 i les branques posteriorment derivades 3.x/4.x, cada vegada passa més temps entre les versions principals 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 organitzada d'accedir a dades del nucli i del maquinari.
- 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 l'antiga branca del nucli 2.6.x [Ces06, Pra03, Lov10], 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 l'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 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 en el seu codi font, 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 (en l'adreça http://kernelnewbies.org/LinuxVersions).

En les versions 3.x/4.x del kernel, bàsicament s'han millorat diferents aspectes de la branca prèvia 2.6.x, aportant noves prestacions, que també són el futur immediat de futures versions en el Kernel, les quals se centraran en:

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

- Millores en sistemes de fitxers estàndard Linux per a adaptar-los a noves necessitats, com el cas de Btrfs i diverses millores introduïdes en ext4, i també les millores introduïdes i millor suport en el kernel per XFS i ZFS.

- Per contra, la millora en nuclis més optimitzats per a dispositius mòbils (*smartphones*, *tablets*, etc.). Per exemple, s'ha incorporat el nucli d'Android,

com a plataforma, al codi font del kernel. I hi ha diverses iniciatives com Ubuntu Phone o Tizen, per a proporcionar noves plataformes basant-se en kernel Linux per a entorns mòbils. Especialment, el suport del kernel per a arquitectures ARM (de les més utilitzades en entorns mòbils) ha portat la possibilitat d'usar Linux en multitud de nous dispositius.

- Millora en el compliment dels estàndards POSIX.

Enllaç d'interès

Per a saber més sobre POSIX, podeu visitar el següent web: http://www.unix.org/

- Millora de la planificació de la CPU. Encara que es van fer molts avenços en aquest aspecte, encara hi ha un baix rendiment 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.

- Suport per a les noves EFI/UEFI com a substitutes de les antigues BIOS en entorns de PC x86/x86_64.

- Suport per a un nou sistema de filtratge IP NFtables, que substituirà progressivament els tallafocs mitjançant iptables.

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 a saber 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 respecte 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 a 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 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 arrencar el nostre sistema en cas de fallades: el conjunt de CD/DVD de la instal·lació, el disquet (o CD) de rescat (actualment s'acostuma a fer servir el primer CD/DVD de la distribució) o alguna distribució en LiveCD que ens permeti accedir al sistema de fitxers de la màquina, per a refer configuracions que hagin causat problemes. 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 en 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 en Debian

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

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

1) Actualitzar la llista de paquets:

```
# apt-get update
```

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

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

3) Triar una versió adequada a la nostra arquitectura (genèrica, x86 o i386 per a Intel o AMD o, en particular per a 64 bits, versions amd64 per a Intel i AMD). El codi de la versió indica la versió del nucli, la revisió de Debian del nucli i l'arquitectura. Per exemple, 3.16.0-4-amd64 és un nucli per a x86_64 AMD/Intel, revisió Debian 4 del nucli 3.16.

4) Comprovar, per a la versió escollida, que hi ha els mòduls accessoris extres. Amb `apt-cache` busquem si hi ha altres mòduls dinàmics que puguin ser interessants per al nostre maquinari, segons la versió del nucli que cal instal·lar. Recordeu que, com vam veure en la *Debian Way*, també trobem la utilitat `module-assistant`, que ens permet automatitzar aquest procés si els mòduls estan suportats. En el cas en què els mòduls necessaris no estiguessin 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 també del codi font del nucli, els `linux-source-version` (en aquest cas ha de ser 3.14, és a dir, el número principal) i els `linux-headers` corresponents, per si més tard volem fer un nucli personalitzat (en aquest cas, el nucli genèric corresponent pedaç per a Debian).

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

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

(si fossin necessaris alguns mòduls) i

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

7) Instal·lar el nou nucli, per exemple en el *bootloader* LILO o Grub (en les últimes versions trobarem aquest per defecte). Normalment això es fa automàticament, però no estaria de més fer alguna còpia de seguretat prèvia de la configuració del *bootloader* (ja sigui `/etc/lilo.conf` o `/boot/grub/menu.lst` o `grub.cfg`).

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 al següent llistat (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, i l'altra és el nucli antic. Així doncs, des del menú LILO podrem demanar-ne una o una altra o, simplement canviant el `default`, recuperar l'antiga. Sempre que fem modificacions en l'arxiu `/etc/lilo.conf` no hem d'oblidar-nos de reescriure'l en el sector corresponent amb l'ordre `/sbin/lilo` o `/sbin/lilo -v`.

En el cas de Grub, que acostuma a ser l'opció normal en les distribucions (ara Grub2 en particular), s'hauria creat una nova entrada (cal tenir present fer la còpia de seguretat prèvia, perquè podem haver perdut alguna entrada anterior depenent del funcionament o 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 instal·lat 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 en la detecció dels discos pot canviar l'ordre d'aquests en el sistema; així, es prefereix etiquetar les particions. Aquestes etiquetes es poden obtenir mitjançant l'ordre del subsistema udisks `udisksctl info -b /dev/particion`, comanda `blkid`, o també amb l'ordre `dumpe2fs /dev/particion | grep UUID` o, si la partició es troba muntada en el sistema, consultant `/etc/fstab`.

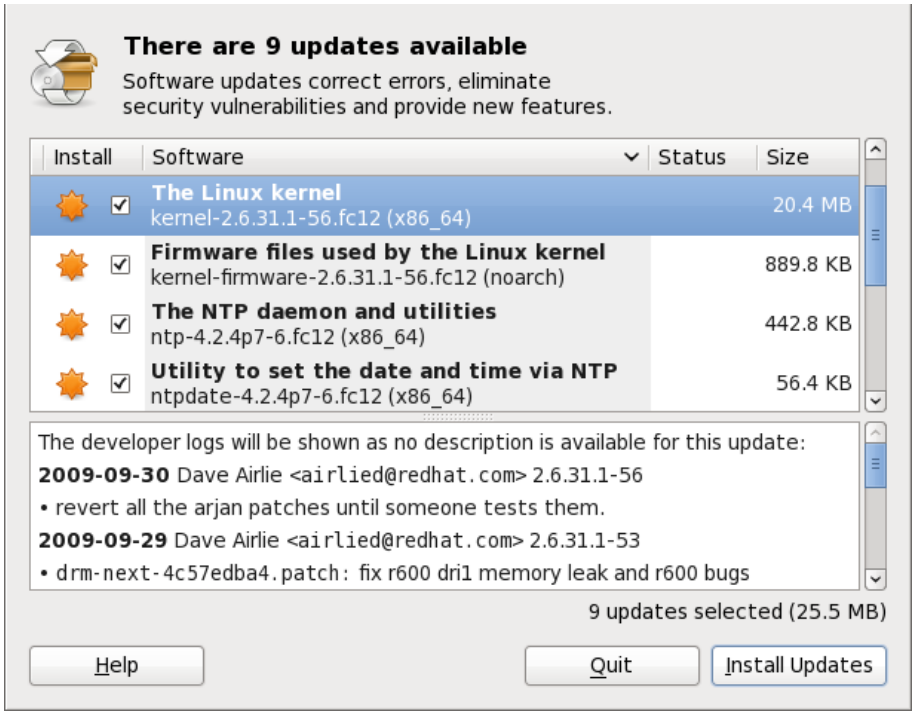
8.2. Configuració del nucli en Fedora/Red Hat

L'actualització del nucli en la distribució Fedora/Red Hat és totalment automàtica per mitjà del seu servei de gestió de paquets (`yum` en línia d'ordres, per exemple), o bé mitjançant els programes gràfics que inclou la distribució, depenent de la seva versió, per a l'actualització (PackageKit en Fedora o pup en Red Hat empresarial o equivalents com CentOS). Normalment, les 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 enfront d'una base de dades de Fedora/Red Hat, i ofereix la possibilitat de descarregar els paquets actualitzats, entre aquests els del nucli. Aquest servei, en el cas de Red Hat empresarial, funciona per un compte de servei i Red Hat l'ofereix per 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 seva versió, perquè són actualitzades amb freqüència.

Per exemple, en la figura 9, 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 9. Eina de Fedora que mostra l'actualització del nucli disponible



En Fedora podem utilitzar les eines gràfiques equivalents o fer servir directament `yum`/`dnf`, si coneixem la disponibilitat de nous nuclis (obtenim nucli i fitxers de desenvolupament per a compilació de mòduls):

```
# dnf install kernel kernel-devel kernel-headers
```

Una vegada descarregada, es procedirà a la seva instal·lació, normalment també de manera automàtica, ja que disposem de Grub o LILO com a gestors d'arrencada. En el cas de Grub (legacy, vegeu més endavant els comentaris per a l'actual Grub 2), sol 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 està en `/boot/grub/grub.cfg` o bé

/boot/grub/menu.lst), tenim dos nuclis diferents, amb els números respectius 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
```

Observeu que la configuració actual és Grub-Legacy (Grub < 2.x); per a Grub2, els camps són semblats però fa falta consultar les *menuentry* corresponents a cada opció.

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 la Fedora/Red Hat, el sistema també l'actualitza (fitxer /etc/lilo.conf), però després caldrà reescriure l'arrencada amb l'ordre /sbin/lilo -v manualment.

***Per defecte en Fedora es fa servir Grub.**

Cal assenyalar, així mateix, 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 es poden configurar i compilar pel procediment habitual, com si fossin un nucli genèric. Cal esmentar que l'empresa Red Hat duu a terme un gran treball de pedaços i correccions per al nucli (usat després en Fedora), i que els seus nuclis són modificacions de l'estàndard genèric amb bastants afegits, per la qual cosa pot ser millor utilitzar les fonts pròpies de Red Hat, tret que vulguem un nucli més nou o experimental que aquell que ens proporcionen.

8.3. Configuració d'un nucli genèric

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

Normalment, tindrem un directori `linux, linux-version` o senzillament el nombre versió; aquest serà l'arbre dels paquets font del nucli. Aquests poden provenir de la mateixa distribució (o pot ser que els hàgim descarregat d'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 últimes i genèriques versions, podem anar a *kernel.org* i descarregar l'última versió disponible (millor l'estable que les experimentals, tret que estiguem interessats en el desenvolupament del nucli). Descarreguem l'arxiu i descomprimim en `/usr/src` (o un altre d'escollit, potser millor) els paquets font del nucli. També podríem buscar si hi ha pedaços per al nucli i aplicar-los (segons hem comentat en l'apartat 4).

Vegeu també

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 depenent de la distribució usada. Es recomana seguir en tant que sigui possible el subapartat 3.1., on es comenta el cas de configuració d'un nucli *vanilla*, o bé els comentaris en el cas Debian, en el subapartat 3.2., o la referència [Fedk] per al cas Fedora.

Recordeu que cal disposar d'espai suficient per a la compilació. És recomanable espai lliure en disc superior a 20 GB.

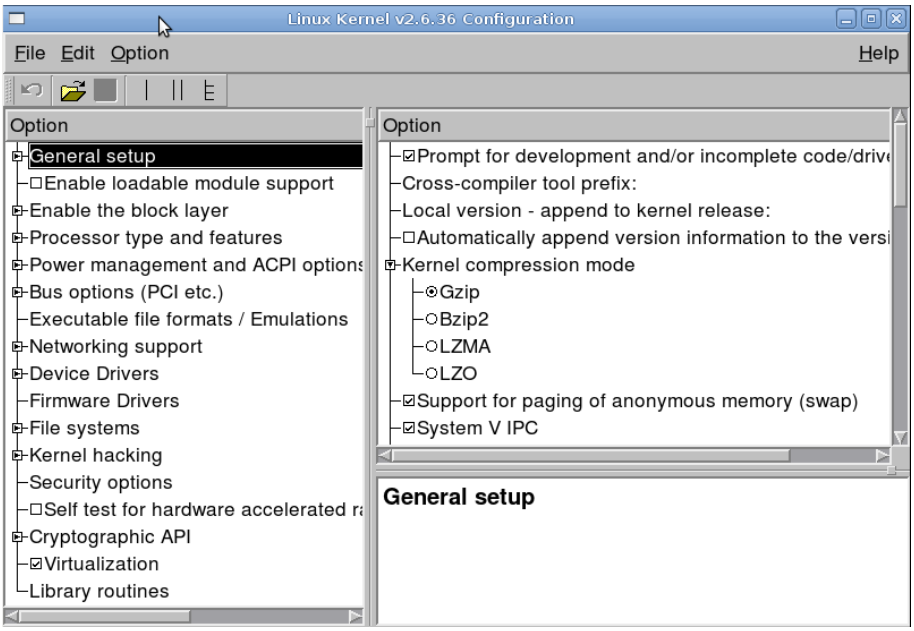
Amb les consideracions esmentades, podem seguir també el procés següent:

1) Netejar el directori de proves anteriors (si és el cas):

```
make mrproper
```

2) Configurar el nucli, per exemple, amb `make menuconfig` (o `xconfig`, figura 10, `gconfig` o `oldconfig`). Ho vam veure en el subapartat 3.1.

Figura 10. Configuració (xconfig) del nucli per menús gràfics



3) Compilar i crear la imatge del nucli: `make`. El procés pot durar desenes de minuts en maquinari modern o diverses hores en maquinari molt antic. Quan

finalitza, la imatge és a `/usr/src/directori-fonts/arch/i386/boot` (en el cas d'arquitectura Intel de 32/64 bits; depenent de la versió el nom del directori i386 canvia per x86).

4) Ara compilem els mòduls amb `make modules` (en les últimes versions això és automàtic). Fins a aquest moment, no hem modificat res en el nostre sistema. Ara haurem de procedir a la instal·lació.

Tanmateix, també s'ha d'anar amb compte si estem compilant una versió que és la mateixa (exacta numeració) 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 versionkernel-antigua/
```

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

```
# make modules_install
```

5) Ara podem passar a la instal·lació del nucli (s'ha de substituir i386 or x86 depenent de la versió), per exemple (de manera manual) amb:

```
# cd /usr/src/directorio-Fuentes/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`.

6) Ja només ens queda posar la configuració necessària en el fitxer de configuració del gestor d'arrencada, ja sigui LILO (`/etc/lilo.conf`), Grub-Legacy o Grub2 (`/boot/grub/menu.lst o grub.cfg`) segons les configuracions que ja vam veure amb Fedora o Debian. I recordeu que en el cas de LILO caldrà tornar a actualitzar la configuració amb `/sbin/lilo` o `/sbin/lilo -v`.

Cal recordar que, com vam veure, tot aquest procés (punts 5 i 6) en els kernels moderns pot fer-se simplement amb un:

```
# make install
```

7) Reiniciar la màquina i observar els resultats (si tot el procés ha estat correcte disposarem d'una nova entrada de nucli en el gestor d'arrencada).

Resum

En aquest mòdul s'han examinat diferents característiques del nucli (*kernel*) de Linux en les seves branques 2.6 i 3.x/4.x. Així mateix, s'han comentat alguns dels processos d'actualització, configuració i sintonització per al sistema, útils tant per a l'optimització de l'ús de memòria com per a maximitzar les prestacions en una arquitectura de CPU concreta, i l'adaptació al maquinari (dispositius) present en el sistema.

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/dnf).

2. Feu una actualització automàtica de la vostra distribució. Comproveu possibles dependències amb altres mòduls utilitzats i amb el *bootloader* (LILO o Grub-Legacy o Grub2) 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 https://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

[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.

[Cor05] **Corbet, J.; Rubini, A.; Kroah-Hartman, G.** (2005). *Linux Device Drivers* (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.

[Grub1] **GNU**. *Grub bootloader*. <http://www.gnu.org/software/grub/grub-legacy.html>

[Grub2] **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*. <https://www.kernel.org/doc/htmldocs/kernel-hacking/index.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>

[Lov10] **Love, R.** (2010). *Linux Kernel Development*. (3a. ed.). Addison-Wesley.

[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>

[Pra11] **Pranevich, J.** (2011). *The Wonderful World of Linux 3.0*. <http://www.kniggit.net/wwol30/>

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

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

[Tan06] **Tanenbaum, A.; Woodhull, A. S.** (2006). *Operating Systems Design and Implementation* (3a. ed.). 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.

Sobre aquestes fonts de referència i informació:

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

[Kera] [lkm] Llocs web que inclouen 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. S'acostuma a actualitzar amb els canvis produïts en la distribució. [Fedk] aporta una referència similar per al cas Fedora.

[Ces06] Llibre sobre el nucli de Linux 2.4, que 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. [Lov10] és un text alternatiu més actualitzat. Per a la branca 3.x aquests llibres continuen essent útils, ja que la majoria de conceptes del nucli es mantenen inalterats.

[Pra03] Article que descriu algunes de les principals novetats de la branca 2.6 del nucli Linux. [Pra11] és l'equivalent per a 3.0+.

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

[Bac86] [Vah96] [Tan87] [Tan86] Alguns textos sobre els conceptes, disseny i implementació dels nuclis de diferents versions UNIX i Linux.

[Skoa][Zan01][Kan][Grub1][Grub2] Recursos per a tenir més informació sobre els carregadors LILO, Grub i Grub2.

[Gru2][Grub1] Llocs oficials de Grub2 i l'anterior original Grub (ara conegut com a Grub Legacy.)