Estadístiques de crides al sistema en plataforma GNU/Linux

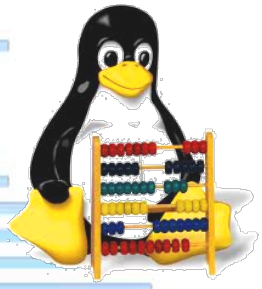
Enginyeria Tècnica en Informàtica de Sistemes (ETIS) - UOC

Autor: **German Guirado Pascual**

Consultor: **Jordi Massaguer Pla**

Memòria del Projecte

Data: **10/01/2012**



DEDICATÒRIES I AGRAÏMENTS

A Jordi Massaguer, el meu consultor, per la seva assistència i el seu suport, així com pel seguiment, els consells i l'atenció que m'ha proporcionat durant tot el projecte.

A la Silvia, per la paciència que ha tingut amb mi durant tot el desenvolupament del projecte, i per haver aguantat tants dies i nits al meu costat.

Al Ferran i al Carles, pels seus consells i recomanacions; així com pels ànims que m'han trasmés quan a mi me'n faltaven.

A Richard Stallmant (GNU) i Linus Torvalds (Linux), per la seva dedicació al món del programari lliure i per mantenir viu l'esperit de col·laboració entre els usuaris de de la informàtica.

A tots ells, la meva dedicatòria i el meu agraïment.
German Guirado

RESUM

Aquest projecte es basa en la modificació del *kernel* (nucli) del sistema operatiu GNU/Linux, per dotar-lo de la capacitat d'extreure estadístiques de les crides al sistema (*syscalls*). A partir de la compilació i instal·lació d'un nou nucli, es registra la informació del nombre de vegades i la freqüència amb la que es fan aquestes crides al sistema, i posteriorment es representa en un informe d'estadístiques explicatives.

Aprofitant els avantatges del paradigma del programari lliure, i especialment, la possibilitat que ofereix per utilitzar, modificar i redistribuir el seu codi font de forma lliure (sota els termes de la GPL [General Public License: Llicència Pública General de GNU]), es procedeix a la modificació del nucli del sistema operatiu Linux, per extreure'n estadístiques detallades de les *syscalls*.

En concret, sobre una distribució Ubuntu 11.04, s'instal·la el codi font del *kernel*, així com eines per compilar-lo. A continuació, un cop compilat, s'instal·la el nou nucli, que ja està en disposició de ser editat. Per fer-hi les modificacions, es fa servir el llenguatge de programació C, editant un fitxer que fa la funció de *dispatching*, i enregistrant les crides al sistema.

Amb la informació un cop emmagatzemada, s'extreuen les dades del fitxer generat, per estructurar-les i presentar-les en un full de càlcul, proveint-lo de gràfiques aclaridores. L'anàlisi d'aquesta informació permet la realització d'unes estadístiques i conclusions sobre les crides al sistema a la plataforma GNU/Linux.

Paraules clau: *Plataforma GNU/Linux, crides al sistema, syscalls, modificació del kernel, modificació del nucli, estadística de crides al sistema Linux, estadístiques de syscalls Linux.*

Àrea: Plataforma GNU/Linux.

RESUMEN

Este proyecto se basa en la modificación del *kernel* (núcleo) del sistema operativo GNU/Linux, para dotarlo de la capacidad de extraer estadísticas de las llamadas al sistema (*syscalls*). A partir de la compilación e instalación de un nuevo núcleo, se registra la información del número de veces y la frecuencia con la que se hacen estas llamadas al sistema, y posteriormente se representa en un informe de estadísticas explicativas.

Aprovechando las ventajas del paradigma del software libre, y especialmente, la posibilidad que ofrece para utilizar, modificar y redistribuir su código fuente de forma libre (bajo los términos de la GPL [General Public License: Licencia Pública General de GNU]), se procede a la modificación del núcleo del sistema operativo Linux, para extraer estadísticas detalladas de las *syscalls*.

En concreto, sobre una distribución Ubuntu 11.04, se instala el código fuente del *kernel*, así como herramientas para compilarlo. A continuación, una vez compilado, se instala el nuevo núcleo, que ya está en disposición de ser editado. Para hacerle las modificaciones, se utiliza el lenguaje de programación C, editando un fichero que hace la función de *dispatching*, y registrando las llamadas al sistema.

Con la información una vez almacenada, se extraen los datos del fichero generado, para estructurarlos y presentarlos en una hoja de cálculo, proveyéndolo de gráficos aclaradores. El análisis de esta información permite la realización de unas estadísticas y conclusiones sobre las llamadas al sistema en la plataforma GNU/Linux.

Palabras clave: *Plataforma GNU/Linux, llamadas al sistema, syscalls, modificación del kernel, modificación del núcleo, estadística de llamadas al sistema Linux, estadísticas de syscalls Linux.*

Área: Plataforma GNU/Linux.

SUMMARY

This project is based on the modification of the GNU/Linux operating system kernel (core), in order to provide it with the capability of extracting statistics about the system calls (*syscalls*). Starting with the compilation and installation of the new kernel, the information about the times and frequency which the system calls are done, is registered. Subsequently, this information is presented in a explanatory statistics report.

Taking advantage of the open software paradigm, and especially, the possibility that it offers to use, modify and redistribute its source code freely (under the terms of the GPL [General Public License]), the modification of the Linux operating system kernel is carried out, for extracting detailed statistics about the *syscalls*.

Specifically, on a Ubuntu 11.04 distribution, the kernel source code is installed, likewise the tools to compile it. Next, once compiled, the new kernel –now prepared to be edited- is installed. To perform the modifications, the C programming language is used, editing a file that makes the *dispatching* function, and registering the system calls.

Once the information is stored, the data is extracted from the generated file, for being structured and presented in a spreadsheet, providing it with clarifying graphs. The analysis of this information permits the carrying out of statistics and conclusions about the system calls in the GNU/Linux platform.

Key words: *GNU/Linux platform, system calls, syscalls, kernel modification, core modification, Linux system calls statistics, Linux syscalls statistics.*

Area: GNU/Linux platform.

ÍNDEX DE CONTINGUTS

ÍNDEX DE FIGURES

IMATGES

TAULES

1.- PROJECTE "Estadístiques de crides al sistema en plataforma GNU/Linux"

1.1.- INTRODUCCIÓ

1.1.1.- Context i justificació del TFC

En el món de la informàtica, especialment en l'àmbit dels sistemes operatius i, en concret, dins el paradigma de programari lliure, destaca la plataforma GNU/Linux pels avantatges que ofereix.

Tot i la seva potència, qualitat i fiabilitat (cosa que està fent que s'estengui de forma important per instal·lacions de servidors), el sistema encara no té gaire presència entre el usuaris finals de computadores.

En molts casos, el desconeixement de la seva existència; i en molts altres, el desconeixement del seu funcionament, són les causes de que Linux no arribi al 1% de les instal·lacions com a sistema operatiu d'escriptori.

Per a un usuari novell en aquest sistema operatiu, com sóc jo, l'aprenentatge del seu funcionament i l'adquisició de coneixements sobre les seves particularitats, són bàsics per guanyar la experiència necessària com a professional de sistemes.

I per a aconseguir aquest objectiu, aquest projecte el focalitzo en aprofundir sobre dos temes cabdals en Linux: el kernel i les crides al sistema. La modificació del nucli per la posterior extracció d'estadístiques de *syscalls* servirà, per una banda, per aprendre a compilar el kernel; i per una altra, per guanyar coneixements de la pròpia informació sobre crides al sistema.

1.1.2.- Objectius

Els objectius bàsics d'aquest projecte es poden dividir principalment es dues vessants:

· Per una banda, em fixo uns objectius "personals":

- Aprendre a modificar i compilar el nucli (kernel) del sistema Linux.
- Adquirir coneixements sobre les crides al sistema més utilitzades.

· I, per una altra banda, plantejo uns objectius "públics":

- Documentar, de forma explicativa, el procés portat a terme.
- Oferir informació, en forma d'estadístiques, sobre les *syscalls*.

1.1.3.- Enfocament i mètode seguit

Per a abordar el desenvolupament del projecte, es segueix un procediment basat en la execució ordenada d'una sèrie d'accions, estructurades en fases.

Primerament, es procedirà a preparar l'entorn; seguidament, es modificarà el kernel per enregistrar les crides al sistema i, finalment, es presentarà la informació obtinguda.

Així doncs, sobre una instal·lació de Linux en una computadora (en concret, sobre una distribució Ubuntu 11.04), es planifiques les següents fases:

FASE 1: Preparació de l'entorn i compilació del kernel

- **Instal·lació del sistema operatiu**
  · Establir les particions, el sistema d'arxius i formatar el disc dur.
  · Instal·lar Linux, distribució 'Ubuntu 11.04'.

- **Instal·lació del codi font del kernel i eines de compilació**
  · Instal·lar el codi font del nucli de Linux.
  · Instal·lar les eines de compilació ('gcc' i 'make').

- **Compilació i instal·lació del nou nucli**
  · Configurar el nucli.
  · Compilar el nucli.
  · Instal·lar el nou nucli.

FASE 2: Modificació del kernel

- **Programació per enregistrar les syscalls**
  · Editar fitxer .C amb funció de *dispatching*.

FASE 3: Informació i resultats

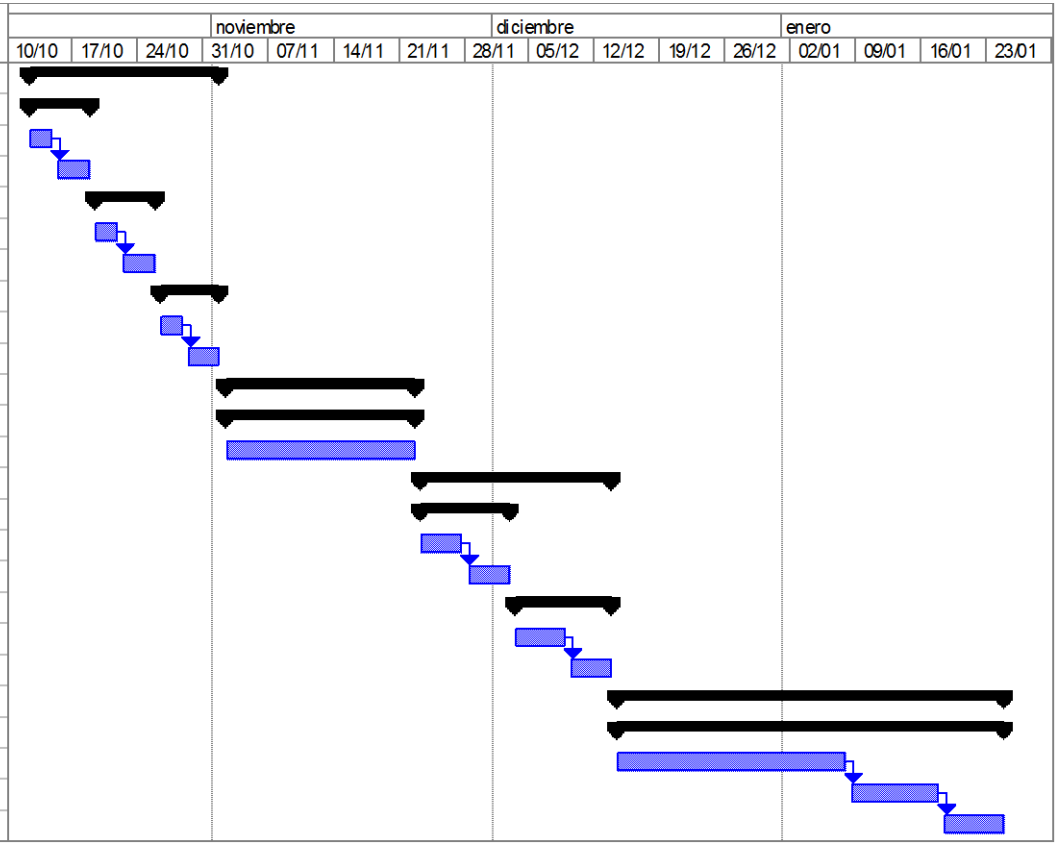
- **Representació de la informació**
  · Extreure la informació del fitxer generat.
  · Estructurar les dades en un full de càlcul.

- **Conclusions**
  ·Analitzar i valorar la informació.
  ·Valoracions i conclusions finals.

1.1.4.- Planificació del projecte

Temporalització de les fases i fites del projecte:

Id	Nombre de tarea	Días	Comienzo	Fin
1	**FASE 1: Preparació de l'entorn i compilació del kernel**	**21 días**	**mié 12/10/11**	**mar 01/11/11**
2	**Instal·lació del sistema operatiu**	**7 días**	**mié 12/10/11**	**mar 18/10/11**
3	Particionar, Stma.Arxius i formatejar disc dur	3 días	mié 12/10/11	vie 14/10/11
4	Instal·lar Linux Ubuntu 11.04	4 días	sáb 15/10/11	mar 18/10/11
5	**Codi font del kernel i eines de compilació**	**7 días**	**mié 19/10/11**	**mar 25/10/11**
6	Instal·lar el codi font del kernel	3 días	mié 19/10/11	vie 21/10/11
7	Instal·lar eines de compilació 'gcc' i 'make'	4 días	sáb 22/10/11	mar 25/10/11
8	**Compilació i instal·lació del nucli**	**7 días**	**mié 26/10/11**	**mar 01/11/11**
9	Executar eina de compilació 'make'	3 días	mié 26/10/11	vie 28/10/11
10	Instal·lar el nou kernel	4 días	sáb 29/10/11	mar 01/11/11
11	**FASE 2: Modificació del kernel**	**21 días**	**mié 02/11/11**	**mar 22/11/11**
12	**Programació per enregistrar syscalls**	**21 días**	**mié 02/11/11**	**mar 22/11/11**
13	Editar fitxer .c amb funció de dispatching	21 días	mié 02/11/11	mar 22/11/11
14	**FASE 3: Informació i resultats**	**21 días**	**mié 23/11/11**	**mar 13/12/11**
15	**Representació de la informació**	**10 días**	**mié 23/11/11**	**vie 02/12/11**
16	Extreure informació del fitxer generat	5 días	mié 23/11/11	dom 27/11/11
17	Estructurar dades en full de càlcul	5 días	lun 28/11/11	vie 02/12/11
18	**Conclusions**	**11 días**	**sáb 03/12/11**	**mar 13/12/11**
19	Analitzar i valorar informació	6 días	sáb 03/12/11	jue 08/12/11
20	Valoracions i conclusions finals	5 días	vie 09/12/11	mar 13/12/11
21	**FASE FINAL: Lliuraments**	**42 días**	**mié 14/12/11**	**mar 24/01/12**
22	**Lliuraments finals**	**42 días**	**mié 14/12/11**	**mar 24/01/12**
23	Memòria	25 días	mié 14/12/11	sáb 07/01/12
24	Producte	10 días	dom 08/01/12	mar 17/01/12
25	Presentació	7 días	mié 18/01/12	mar 24/01/12



Imatge 01: Planificació del projecte

1.1.5.- Productes obtinguts

Com a resultat d'aquest projecte, al final del mateix es disposarà dels següents productes:

- Fitxer de full de càlcul, que inclou les estadístiques obtingudes: [SysCallStatistics.ods].
- Fitxer patch, que conté les diferències entre el codi font original i el codi modificat per a l'obtenció de les estadístiques de crides al sistema: [SysCallStatistics.patch]
- Memòria del projecte, que inclou la documentació de tot el procés, així com l'anàlisi i les conclusions finals extretes de la informació obtinguda: [gguirado_memoria.PDF].

1.1.6.- Estructura de la memòria

Aquest document es presenta i estructura en forma de procediment lineal, on la solució es basa en la consecució ordenada de les fases que la formen. D'aquesta manera, cada fase (que consta d'una sèrie d'accions que es porten a terme de forma ordenada), segueix a una altra fase amb la mateixa estructura, i es presenta en un capítol diferent.

Tant a l'índex com a l'apartat "*1.1.3.- Enfocament i mètode seguit*", es pot veure aquesta estructura de fases, i les accions que les componen. Cada capítol de l'apartat "1.2.- DESENVOLUPAMENT" documenta una d'aquestes fases. Més endavant, es presenta un capítol de valoració econòmica de la solució. I finalment, s'inclou un capítol de conclusions.

Durant la memòria, a mode de documentació, es presenten les instruccions introduïdes al terminal de Linux, i les sortides per pantalla a aquestes instruccions (es fa servir una font diferent per a les `entrades (inputs)`, i una altra per a les `sortides (outputs)`. Quan la sortida és massa llarga i s'obvien algunes parts, es fa servir el símbol: `[...]`). Tanmateix, també s'adjunta el codi de la programació de la funció de *dispatching*.

Al final d'aquesta memòria es poden trobar els capítols dedicats al **glossari** de termes utilitzats, la **bibliografia** (on es poden trobar les referències a la documentació emprada) i els **annexos**.

1.2.- DESENVOLUPAMENT

1.2.1.- Disseny de la solució

El resultat final d'aquest projecte és la documentació de les estadístiques obtingudes sobre les crides al sistema, gracies a la modificació del kernel de Linux. Per a obtenir aquest resultat, s'ha realitzat el següent disseny de la solució, que es pot trobar en els propers tres capítols, pertanyents a les fases del projecte:

1.2.2.- Fase de preparació de l'entorn i compilació del kernel

- Instal·lar Linux, distribució Ubuntu 11.04 en una computadora.

- Instal·lar el codi font del kernel:
  ```
  $ apt-get install kernel-source
  ```

- Instal·lar les eines de compilació 'gcc' i 'make':
  ```
  $ sudo apt-get install gcc
  $ sudo apt-get install make
  ```

- Compilar i instal·lar el nou nucli.

1.2.3.- Fase de modificació del kernel

- Programar el fitxer .c amb la funció de dispatching, perquè enregistri les crides al sistema. Per a aquest pas, es compta amb la documentació facilitada pel consultor:
  http://www.ibm.com/developerworks/linux/library/l-linux-kernel/
  http://www.win.tue.nl/~aeb/linux/lk/lk-4.html
  http://www.linuxjournal.com/article/3326
  http://linux.die.net/man/2/syscalls
  http://stackoverflow.com/questions/2103315/linux-kernel-system-call-hooking-example

1.2.4.- Fase d'informació i resultats

- Extreure la informació del fitxer generat.

- Representar les dades en forma d'estadístiques explicatives, sobre un full de càlcul que inclogui les pròpies dades, així com gràfiques aclaridores.

Finalment, en el capítol "1.4.- CONCLUSIONS", es procedeix a reflexionar sobre tot el procés i analitzar la informació, tot extraient-ne valoracions i conclusions.

1.2.2.- Fase de preparació de l'entorn i compilació del kernel

a) Instal·lació del Sistema Operatiu "Linux":

Els detalls de la instal·lació de Linux (particionat del disc dur, tria de sistema d'arxius i format de cada partició), no formen part d'aquest projecte i, per tant, no es documenten en la memòria. Existeixen infinitat de manuals al respecte.

En aquest sentit, l'únic detall remarcable és la versió instal·lada: Linux Ubuntu 11.04.

b) Instal·lació del codi font del kernel i eines de compilació:

b.1) Instal·lar el codi font del nucli de Linux:

Al intentar descarregar el codi font del kernel… :

```
$ apt-get install kernel-source
```

… es rep un missatge d'error indicant la denegació de permís per fer-ho si no es tracta de l'usuari administrador "*root*", donat que suposa una acció potencialment perillosa:

```
E: Could not open lock file /var/lib/dpkg/lock – open (13: Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
```

El sistema ens adverteix que per solucionar-ho, s'ha de fer servir "sudo":

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Per tant, s'ha de modificar la sentència (aquest procediment és vàlid només per usuaris del grup d'administradors):

```
$ sudo apt-get install kernel-source
```

Tal com indica la documentació que passaria ("man sudo_root"), se'ns demana la contrasenya:

```
[sudo] password for username: _
```

Però al fer-ho, es rep un altre error, donat que no existeix cap paquet amb aquest nom:

```
Reading package list… Done
Building dependency tree
Reading state information… Done
Package kernel-source is not available, but it is refered to by another package.
This may mean that the package is missing, has benn obsoleted, or is only
available from another source.
E: Package 'kernel-source' has no installation candidate
```

Buscant informació i documentació a Internet[1], es pot arribar a la conclusió que '*kernel*' és el terme genèric per referir-se al nucli del sistema operatiu, i es fa servir en distribucions Debian. Per a Ubuntu, s'ha d'indicar el nucli que es vol instal·lar ('linux'):

```
$ sudo apt-get install linux-source
```

1= "The reason that your apt-get command didn't work is simply that you have to specify which kernel you want the source or headers for." [http://forums.debian.net/viewtopic.php?f=10&t=38885]

En aquesta ocasió, el procediment de descàrrega sí que troba un paquet amb aquest nom i es disposa a baixar-lo, avisant de que alguns dels subpaquets ja estan instal·lats, indicant els paquets 'extra' que s'instal·laran perquè no ho estan, el volum d'arxius i espai en disc que necessitarà, i demanant confirmació per continuar:

```
Reading package list… Done
Building dependency tree
Reading state information… Done
The following packages were automatically installed and are no longer required:
       linux-headers-2.6.38-8 linux-headers-2.6.38-8-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
       linux-source-2.6.38
Suggested packages:
       libncurses-dev ncurses-dev kernel-package libqt3-dev
The following NEW packages will be installed:
       linux-source linux-source-2.6.38
0 upgraded, 2 newly installed, 0 to remove and 64 not upgraded.
Need to get 77.9 MB of archives.
After this operation, 85.8 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get:1  http://es.archive.ubuntu.com/ubuntu/  natty-updates/main  linux-source-
2.6.38 all 2.6.38-12.51 [77.9 MB]
Get:2 http:// es.archive.ubuntu.com/ubuntu/ natty-updates/main linux-source all
2.6.38.12.27 [2,500 B]
Fetched 77.9 MB in 41min 14s (306 kB/s)
Selecting previously deselected package linux-source-2.6.38.
(Reading database … 168580 files and directories currently installed.)
Unpacking linux-source-2.6.38 (from …/linux-source-2.6.38_2.6.38-12.51_all.deb)…
Selecting previously deselected package linux-source.
Unpacking linux-source (from …/linux-source_2.6.38.12.27_all.deb) …
Setting up linux-source-2.6.38 (2.6.38-12.51) …
Setting up linux-source (2.6.38.12.27) …
```

En aquest punt, ja està descarregat el codi font del kernel (nucli del sistema operatiu). S'ha generat el directori "linux-source-2.6.38" i el link "linux-source-2.6.38.tar.bz2" al directori "/usr/src/".

A continuació, s'ha de descomprimir el contingut de l'arxiu descarregat: "linux-source-2.6.38.tar.bz2". Per a fer-ho, es fa servir l'ordre "tar":

```
/usr/src$ sudo tar jxvf linux-source-2.6.38.tar.bz2
```

Després d'introduir la contrasenya d'usuari, el procediment comença la descompressió del contingut de l'arxiu en el directori "/usr/src/linux-source-2.6.38":

```
[...]
linux-source-2.6.38/ubuntu/fsam7400/README
linux-source-2.6.38/ubuntu/Kconfig
linux-source-2.6.38/ubuntu/aufs-update
linux-source-2.6.38/ubuntu/Makefile
linux-source-2.6.38/dropped.txt
```

En aquest punt, ja està disponible el codi font del kernel.

b.2) Instal·lar les eines de compilació ('gcc' i 'make'):

Les eines de compilació que es necessiten són: '*gcc*' i '*make*'. Amb la experiència de l'apartat anterior, es pot passar directament a l'ordre definitiva, utilitzant "sudo" com a prefix de la sentència per a la instal·lació d'aquestes eines. (En els dos casos, es demana la contrasenya d'usuari per poder continuar):

gcc (GNU Compiler Collection):

```
$ sudo apt-get install gcc
```

En el cas concret d'aquesta distribució (Ubuntu 11.04), i en aquesta data (22/10/11), la instal·lació no es realitza perquè no és necessària, donat que el sistema ja té la última versió disponible, i així ho mostra en el missatge informatiu:

```
Reading package list… Done
Building dependency tree
Reading state information… Done
gcc is already the newest version.
The following packages were automatically installed and are no longer required:
       linux-headers-2.6.38-8 linux-headers-2.6.38-8-generic
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 64 not upgraded.
```

En aquest punt, ja està instal·lada l'eina de compilació 'gcc'.

make:

```
$ sudo apt-get install make
```

Com en el cas anterior, aquesta distribució en aquesta data ja té instal·lada la versió més nova:

```
Reading package list… Done
Building dependency tree
Reading state information… Done
make is already the newest version.
The following packages were automatically installed and are no longer required:
       linux-headers-2.6.38-8 linux-headers-2.6.38-8-generic
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 64 not upgraded.
```

En aquest punt, ja està instal·lada la eina de compilació 'make'.

c) Compilació i instal·lació del nou nucli

c.1) Configurar el nucli:

Abans de compilar el nucli, primer s'ha d'establir una configuració. En el meu cas, he optat per fer servir una configuració coneguda: la que ve per defecte amb Ubuntu. El seu arxiu de configuració "config-2.6.38-8-generic" es pot trobar al directori "boot". S'ha de copiar aquest arxiu al directori "usr/src/linux-source-2.6.38", amb el nom ".config":

```
/usr/src/linux-source-2.6.38$ sudo cp /boot/config-2.6.38-8-generic .config
```

Existeixen diverses maneres de modificar la configuració del kernel. Triaré 'xconfig' perquè proporciona una interfície gràfica molt còmoda.
Ara ja es pot intentar la configuració del nucli amb l'eina 'make':

```
/usr/src/linux-source-2.6.38$ sudo make oldconfig xconfig
```

El procés de configuració llegeix el valor per defecte de cada opció que hem proveït al pas anterior, i només demana confirmació pels paràmetres nous, als que també deixo els valors per defecte:

```
  HOSTCC scripts/basic/fixdep
  HOSTCC scripts/basic/docproc
  HOSTCC scripts/kconfig/conf.o
  HOSTCC scripts/kconfig/kxgettext.o
  HOSTCC scripts/kconfig/zconf.tab.c
  HOSTCC scripts/kconfig/lex.zconf.c
  HOSTCC scripts/kconfig/zconf.hash.c
  HOSTCC scripts/kconfig/zconf.tab.o
  HOSTCC scripts/kconfig/conf
scripts/kconfig/conf –oldconfig Kconfig
*
* Restart config...
*
*
* Kernel kacking
*
Show timing information on printks (PRINTK_TIME) [Y/n/?] y
[...]
*
* Tracers
*
Tracers (FTRACE) [Y/n/?] y
[...]
*
* Sample kernel code
*
Sample kernel code (SAMPLES) [N/y/?] n
*
* KGDB: kernel debugger
*
KGDB: kernel debugger (KGDB) [Y/n/?] y
[...]
Test kstrto*() family of functions at runtime (TEST_KSTRTOX) [N/m/y] N
[...]
#
# configuration written to .config
#
```

En aquest punt, acaba la recopilació de paràmetres, que es guarda a l'arxiu ".config".

Però quan fa la comprovació de l'eina 'qt' (eina de configuració), no la troba:

```
[...]
  CHECK  qt
* Unable to find QT4 tool qmake. Trying to use QT3
*
* Unable to find any QT installation. Please make sure that
* the QT4 or QT3 development package is correctly installed and
* either qmake can be found or install pkg-config or set
* the QTDIR enviroment variable to the correct location.
*
sed < scripts/kconfig/lkc_proto.h > scripts/kconfig/lkc_defs.h
       's/P(\([^,]*\),.*/#define \1 (\*\1_p)/'
  HOSTCC scripts/kconfig/kconfig_load.o
make[l]: *** No rule to make target 'scripts/kconfig/.tmp_qtcheck', needed by
       'scripts/kconfig/qconf.o'. Stop.
make: *** [xconfig] Error 2
```

Per tant, és necessari instal·lar el paquet libqt4-dev amb totes les seves dependències:

```
/usr/src$ sudo apt-get install libqt4-dev
```

Aquesta ordre també demana contrasenya d'usuari. Després, avisa dels paquets que ja estan instal·lats, quins paquets extra i nous s'instal·laran, i quins s'actualitzaran. Al final, mostra un resum i demana confirmació per a continuar:

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.38-8 linux-headers-2.6.38-8-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
[...]
The following NEW packages will be installed:
[...]
The following packages will be upgraded:
[...]
2 upgraded, 38 newly installed, 0 to remove and 88 not upgraded.
Need to get 27.9 MB of archives.
After this operation, 118 MB of additional disk space will be used.
Do you want to continue [Y/n] Y
Get:1  http://es.archive.ubuntu.com/ubuntu  natty/mail  libkms1  i386  2.4.23-
1ubuntu6 [8,936 B]
[...]
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
```

En aquest punt, ja està instal·lat el paquet qt4 necessari.

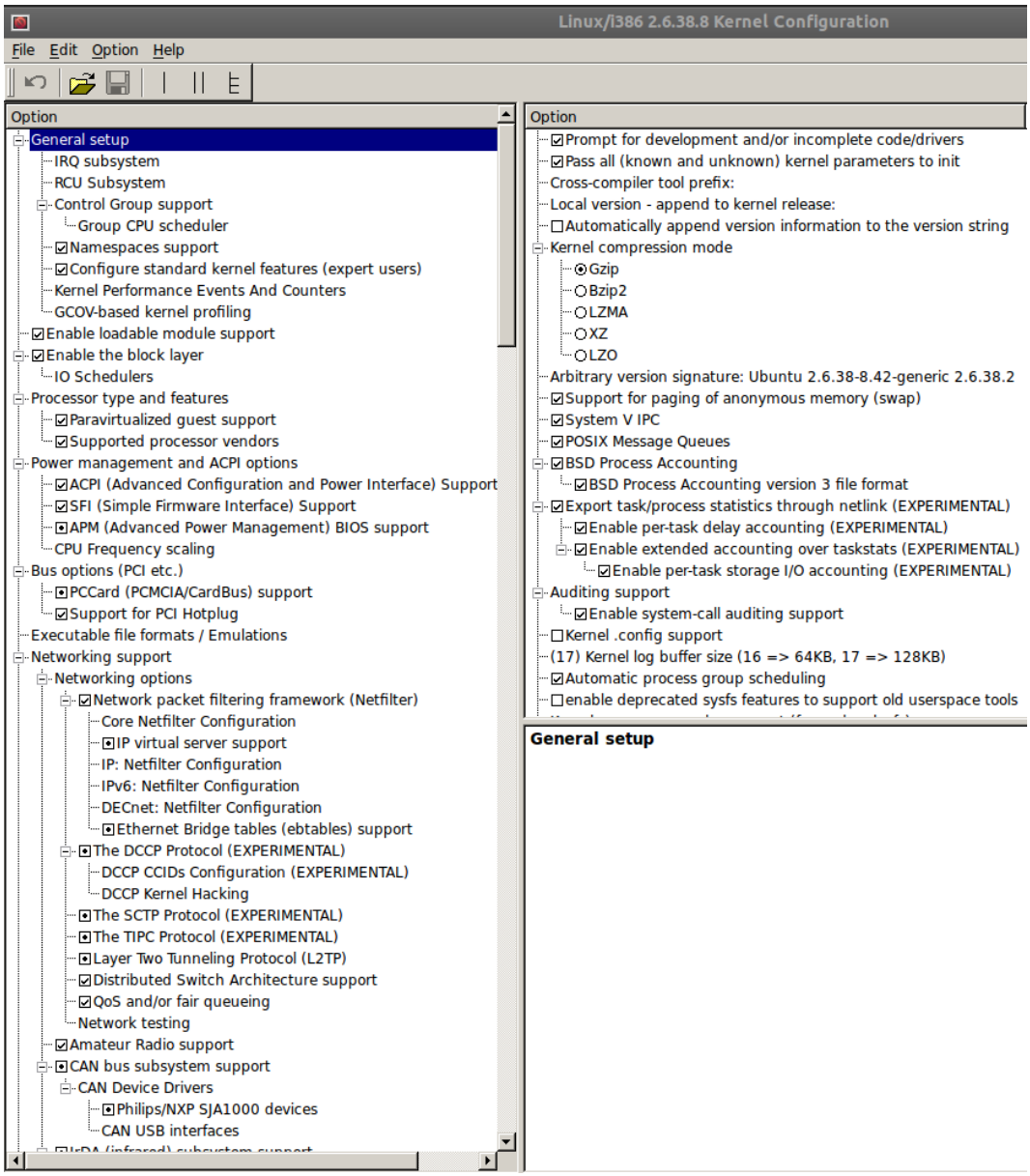
Ara ja es pot procedir definitivament a la configuració del nucli amb l'eina 'make':

```
/usr/src/linux-source-2.6.38$ sudo make oldconfig xconfig
```

Després d'iniciar-se i comprovar que ara sí està disponible 'qt', el procés continua... :

```
scripts/kconfig/conf –oldconfig Kconfig
#
# configuration written to .config
#
  CHECK  qt
/usr/bin/moc –i scripts/kconfig/qconf.h –o scripts/kconfig/qconf.moc
  HOSTCXX scripts/kconfig/qconf.o
  HOSTLD  scripts/kconfig/qconf
scripts/kconfig/qconf Kconfig
```

... i obre l'eina gràfica de configuració (qconf):



Imatge 02: pconf (eina gràfica de configuració del kernel)

Per a configurar el nucli, he fet servir una configuració que ve per defecte amb Ubuntu: la de l'arxiu "config-2.6.38-8-generic" del directori "boot", que he copiat a "usr/src/linux-source-2.6.38"). El paràmetres són correctes; així que no modifico cap opció.

c.2) Compilar el nucli:

Un cop configurades les opcions del kernel, es pot procedir a compilar-lo.
Després de la configuració i abans de la compilació, i amb l'objectiu de que algunes opcions que s'han d'utilitzar (com '-append_to_version') tinguin efecte, s'han d'eliminar alguns arxius creats durant la configuració (per exemple: 'include/linux/version.h', arxiu que el procés de compilació no crearà si ja existeix). Per fer-ho de manera còmoda i segura, s'aconsella executar l'ordre de "neteja" 'make-kpkg clean'. Si no es té el programa 'make-kpkg', s'ha d'instal·lar amb:

```
$ sudo apt-get install kernel-package
```

Després d'avisar dels paquets que ja estan instal·lats, quins paquets extra i nous s'instal·laran, i quins s'actualitzaran, mostra un resum i demana confirmació per a continuar:

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.38-8 linux-headers-2.6.38-8-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
[...]
Suggested packages:
[...]
The following NEW packages will be installed:
[...]
0 upgraded, 4 newly installed, 0 to remove and 88 not upgraded.
Need to get 653 kB of archives.
After this operation, 3,138 kB of additional disk space will be used.
Do you want to continue [Y/n] Y
Get:1 http://es.archive.ubuntu.com/ubuntu natty/main po-debconf all 1.0.16+nmul
[212 B]
[...]
Setting up libsys-hostname-long-perl (1.4-2) ...
Setting up libmail-perl (0.79.16-1) ...
```

A continuació, ja es pot executar l'ordre de neteja:

```
/usr/src/linux-source-2.6.38$ sudo make-kpkg clean
```

```
exec make kpkg_version=12.036+nmu1 -f /usr/share/kernel-package/ruleset/minimal
.mk clean
====== making target minimal_clean [new prereqs: ] =====
This is kernel package version 12.036+nmu1.
test ! -f .config || cp -pf .config config.precious
test ! -e stamp-building || rm -f stamp-building
test ! -f Makefile || \
       make  ARCH=i386 distclean
make[1]: Entering directory '/usr/src/linux-source-2.6.38'
  CLEAN scripts/basic
  CLEAN scripts/kconfig
  CLEAN include/config include/generated
  CELAN .config .config.old
make[1]: Leaving directory '/usr/src/linux-source-2.6.38'
test ! -f config.precious || mv -f config.precious .config
rm -f modules/modversions.h modules/ksyms.ver scripts/cramfs/cramfsck scripts/
cramfs/mkcramfs
```

L'últim pas de la compilació és la pròpia construcció del paquet que es farà servir per a la instal·lació. Si hem inclòs al pas de configuració de manera estàtica (no amb mòduls) els controladors pel bus, disc i sistema d'arxius del nostre directori arrel), la opció '--initrd ' es fa prescindible, però igualment recomanable, doncs garanteix la correcta iniciació del sistema. Aquest paràmetre crea una imatge *initrd* al directori '/boot'.
Amb l'objectiu de diferenciar aquesta instal·lació de la resta de versions que puguin coexistir al sistema, es pot fer servir el paràmetre "--append-to-version=*foo*", on "*foo*" poden ser caràcters alfanumèrics en minúscula i els símbols ~ - . +.
Jo, personalment, faré servir la data de compilació per a diferenciar-la: '*.281011*':

```
/usr/src/linux-source-2.6.38$ sudo make-kpkg --append-to-version=.281011 --initrd kernel_image
```

L'execució d'aquesta ordre produeix la següent sortida, que mostra un error abans d'acabar la compilació:

```
exec make kpkg_version=12.036+nmul –f /usr/share/kernel-package/ruleset/minimal
.mk debian APPEND_TO_VERSION=.281011  INITRD=YES
===== making target debian/stamp/conf/minimal_debian [new prereqs: ] =====
This is kernel package version 12.036+nmu1.
test –d debian|| mkdir debian
test ! –e stamp-building || rm –f stamp-building
install –p –m 755 /usr/share/kernel-package/rules debian/rules
for file in ChangeLog  Control  Control.bin86 config templates.in rules; do
              cp –f  /usr/share/kernel-package/$file ./debian/;
              \
       done
for dir  in Config docs examples ruleset scripts pkg po;  do
              \
       cp –af /usr/share/kernel-package/$dir  ./debian/;
         \
       done
[...]
ld: cannot find /ubuntu/omnibook/sections.lds: No such file or directory
make[3]: *** [ubuntu/omnibook/omnibook.o] Error 1
make[2]: *** [ubuntu/omnibook] Error 2
make[1]: *** [ubuntu] Error 2
make[1]: Leaving directory 'usr/src/linux-source-2.6.38'
make: *** [debian/stamp/build/kernel] Error 2
```

Buscant informació i documentació a Internet[2], es pot trobar la explicació i solució d'aquest error, que es basa en la modificació de l'arxiu "../*ubuntu/omnibook/Makefile*":

```
/usr/src/linux-source-2.6.38$ sudo vi ubuntu/omnibook/Makefile
```

```
       160: #EXTRA_LDFLAGS +=  $(src)/sections.lds
       161: EXTRA_LDFLAGS += $(PWD)/ubuntu/omnibook/sections.lds
```

Es tracta de descomentar la línia 160 i comentar la 161:

```
       160: EXTRA_LDFLAGS +=  $(src)/sections.lds
       161: #EXTRA_LDFLAGS += $(PWD)/ubuntu/omnibook/sections.lds
```

2="http://thangamaniarun.wordpress.com/2010/07/08/how-to-quickly-build-custom-kernel-on-ubuntu-10-04/" i "https://bugs.launchpad.net/ubuntu/+source/linux/+bug/505420".

Torno a executar el procés de compilació, amb la garantia de que funcionarà correctament:

```
/usr/src/linux-source-2.6.38$ sudo make-kpkg --append-to-version=.281011 --initrd kernel_image
```

Això és part del resultat (sortida) quan el procés finalitza[3] correctament:

```
[...]
dpkg –build   /usr/src/linux-source-2.6.38/debian/linux-image-2.6.38.281011 ..
dpkg-deb: building package ‘linux-image-2.6.38.8.281011’ in ‘../linux-image-2.6.38.8.281011_2.6.38.8.281011-10.00.Custom_i386.deb’.
make[2]: Leaving directory ‘/usr/src/linux-source-2.6.38’
make[1]: Leaving directory ‘/usr/src/linux-source-2.6.38’
```

En aquest punt, ja es troba compilat el nucli. El resultat és un arxiu a “usr/src/”, anomenat “kernel-image-2.6.38.**281011**_10.00.Custom_i386.deb”.

3= El temps necessari per a la execució d'aquest procés, depèn bàsicament de les característiques de la computadora i de la quantitat de mòduls triats, però amb diferència, aquest és el procés més llarg de tota la fase. Per a un usuari principiant (que prefereix no desactivar cap opció per precaució per si falla alguna cosa en el procés), deixar tota la configuració per defecte és la opció més còmoda i segura, però també la que més temps requereix. He realitzat el procés amb dues màquines diferents, per comprovar aquest diferència, i aquests són els resultats:
*PC Sobretaula: Clònic [i gens modern] (Processador **Intel Celeron @2,66GHz, 768Mb RAM**). Temps: **5h 30min**.*
*PC Portàtil: HP Compaq 8510p (Processador **Intel Core 2 Duo T7500@2,20GHz, 2Gb RAM**). Temps: **1h 45min**.*

c.3) Instal·lar el nou nucli:

Finalment, la instal·lació del nou nucli es basa en l'arxiu d'imatge que el procés anterior ha creat al directori "/usr/src". Aquest arxiu conté el nou nucli compilat i llest per instal·lar, i té el nom: "kernel-image-2.6.38.**281011**_10.00.Custom_i386.deb".
Ara només es necessita executar la següent ordre per a la seva instal·lació:

```
/usr/src$ sudo dpkg –i linux-image-2.6.38.8.281011_2.6.38.8.281011-10.00.Custom_i386.deb
```

El procés genera la següent sortida:

```
Selecting previously deselected package linux-image-2.6.38.8.281011.
(Reading database ... 176008 files and directories currently installed.)
Unpacking linux-image-2.6.38.8.281011 (from linux-image-2.6.38.8.281011_2.6.38.8.281011-10.00.Custom_i386.deb) ...
Done.
Setting up linux-image-2.6.38.8.281011 (2.6.38.8.281011-10.00.Custom) ...
Running depmod.
Examining /etc/kernel/postinst.d.
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 2.6.38.8.281011 /boot/vmlinuz-2.6.38.8.281011
update-initramfs: Generating /boot/initrd.img-2.6.38.8.281011
run-parts: executing /etc/kernel/postinst.d/pm-utils 2.6.38.8.281011 /boot/vmlinuz-2.6.38.8.281011
run-parts: executing /etc/kernel/postinst.d/update-notifier 2.6.38.8.281011 /boot/vmlinuz-2.6.38.8.281011
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 2.6.38.8.281011 /boot/vmlinuz-2.6.38.8.281011
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-2.6.38.8.281011
Found initrd image: /boot/initrd.img-2.6.38.8.281011
Found linux image: /boot/vmlinuz-2.6.38-11-generic
Found initrd image: /boot/initrd.img-2.6.38-11-generic
Found linux image: /boot/vmlinuz-2.6.38-8-generic
Found initrd image: /boot/initrd.img-2.6.38-8-generic
Found memtest86+ image: /boot/memtest86+.bin
Found Windows 7 (loader) on /dev/sda1
done
```

Amb això, queda instal·lat el nou nucli . Al reiniciar, apareix una nova opció en la llista de sistemes operatius, que porta el sufix indicat al paràmetre "--append-to-version=":

Aquesta és la opció que triaré per arrancar d'ara en endavant, per treballar amb el nou kernel. Per tant, aquí finalitza la fase de preparació de l'entorn i compilació del kernel. [4]

4= He basat bona part d'aquesta fase en la documentació existent a Internet; en especial en un document publicat a: "http://www.ubuntu-es.org/node/431", que a la seva vegada és una adaptació d'un document a la wiki de Ubuntu. Ja no està disponible a la referència facilitada en aquest document: "http://www.ubuntulinux.org/wiki/KernelHowto", però sí un de molt complet, i en anglès a l'adreça: "http://www.faqs.org/docs/Linux-HOWTO/Kernel-HOWTO.html".

RESUM DE LA FASE DE PREPARACIÓ DE L'ENTORN I COMPILACIÓ DEL KERNEL:

A continuació, es presenta un recull de les 12 ordres que s'han de fer servir per portar a terme correctament aquesta fase, ordenades i corregides per evitar els errors:

1.- Obtenció del codi font del nucli:

```
$ sudo apt-get install linux-source
```

2.- Descompressió de l'arxiu que conté el codi font del nucli:

```
/usr/src$ sudo tar jxvf linux-source-2.6.38.tar.bz2
```

3.- Obtenció de l'eina 'gcc' [si no estava instal·lada prèviament]:

```
$ sudo apt-get install gcc
```

4.- Obtenció de l'eina 'make' [si no estava instal·lada prèviament]:

```
$ sudo apt-get install make
```

5.- Copiar la configuració actual per al nou nucli:

```
/usr/src/linux-source-2.6.38$ sudo cp /boot/config-2.6.38-8-generic .config
```

6.- Obtenció de l'eina de configuració 'qt4':

```
/usr/src$ sudo apt-get install libqt4-dev
```

7.- Configurar el nou nucli:

```
/usr/src/linux-source-2.6.38$ sudo make oldconfig xconfig
```

8.- Obtenció del paquet d'eines del kernel:

```
$ sudo apt-get install kernel-package
```

9.- Netejar els arxius creats durant la configuració :

```
/usr/src/linux-source-2.6.38$ sudo make-kpkg clean
```

10.- Modificar els paràmetres de Makefile:

```
/usr/src/linux-source-2.6.38$ sudo vi ubuntu/omnibook/Makefile
      160: #EXTRA_LDFLAGS +=  $(src)/sections.lds
      161: EXTRA_LDFLAGS += $(PWD)/ubuntu/omnibook/sections.lds
      >
      160: EXTRA_LDFLAGS +=  $(src)/sections.lds
      161: #EXTRA_LDFLAGS += $(PWD)/ubuntu/omnibook/sections.lds
```

11.- Compilar el kernel:

```
/usr/src/linux-source-2.6.38$ sudo make-kpkg --append-to-version=.281011 --initrd kernel_image
```

12.- Instal·lar el nou kernel, per poder arrancar-hi:

```
/usr/src$ sudo dpkg –i linux-image-2.6.38.8.281011_2.6.38.8.281011-10.00.Custom_i386.deb
```

1.2.3.- Fase de modificació del kernel

Aquesta fase es basa en la modificació del nucli de Linux, concretament la part que conté la funció de "*dispatching*" de les syscalls.

A partir del kernel de Linux 2.6 (i, a diferència de les seves predecessores [2.4, etc.]), la gestió de les crides al sistema es fa de forma diferent. A continuació s'indiquen els passos donats per intentar assolir l'objectiu esperat: Enregistrar les crides al sistema. En aquest cas, la funció i la crida al sistema es diu "statistics".

Solució 1:

1.- /usr/src/linux-2.6.38/arch/x86/kernel/syscall_table.S

Aquest arxiu conté els noms de les crides al sistema. Incorporarem una nova syscall, a la que anomenarem "statistics". Per fer-ho, s'ha d'afegir una línia al final d'aquest fitxer:

```
[...]
.long sys_statistics
```

2.- /usr/src/linux-2.6.38/include/asm-generic/unistd.h

Aquest arxiu conté els nombres de les crides al sistema que es passen al registre "%eax" del kernel quan aquestes són invocades. S'ha d'afegir la definició de la nova syscall, amb el nombre següent a l'últim utilitzat; i augmentar el nombre total de syscalls definides. P.ex.:

Si la última definició de crida del sistema és: #define __NR_fanotify_mark 263

i la definició del nombre total de syscalls és: #define __NR_syscalls 265

S'ha de canviar:

```
[...]
#define __NR_fanotify_mark 263
__SYSCALL(__NR_fanotify_mark, sys_fanotify_mark)
#undef __NR_syscalls
#define __NR_syscalls 264
[...]
```

Per:

```
[...]
#define __NR_fanotify_mark 263
__SYSCALL(__NR_fanotify_mark, sys_fanotify_mark)
#define __NR_statistics 264
__SYSCALL(__NR_statistics, sys_statistics)
#undef __NR_syscalls
#define __NR_syscalls 265
[...]
```

3.- /usr/src/linux-2.6.38/include/linux/syscalls.h

Aquest arxiu conté les definicions de les crides al sistema. S'hi ha d'afegir la següent línia al final de l'arxiu:

```
[...]
asmlinkage long sys_statistics(int i);
```

4.- /usr/src/linux-2.6.38/Makefile

S'ha d'afegir el nom del directory que contindrà la nostra funció, al paràmetre "core-y":

```
[...]
core-y       += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ statistics/
[...]
```

5.- /usr/src/linux-2.6.38/statistics

S'ha de crear el directori "**/usr/src/linux-2.6.38/statistics**", on es guardaran diversos arxius.

6.- /usr/src/linux-2.6.38/statistics/statistics.c

Crear el nou arxiu: "**/usr/src/linux-2.6.38/statistics/statistics.c**", que contindrà el codi[5] de la nova crida al sistema:

```
#include<linux/linkage.h>
asmlinkage long sys_statistics(int i)
{
      printk(KERN_DEBUG "SysCallNumber:%u \n",i);
}
```

7.- /usr/src/linux-2.6.38/statistics/Makefile

Crea el nou arxiu: "**/usr/src/linux-2.6.38/statistics/Makefile**", que només contindrà la següent línia:

```
obj-y := statistics.o
```

*5= A la funció "**sys_statistics**" se li passa com a paràmetre un nombre enter, que serà el nombre de la syscall cridada. "**printk**", a diferència de "**printf**", s'executa a nivell de kernel. "**KERN_DEBUG**" és l'únic nivell de log que no mostra cap missatge per consola/pantalla, i es limita a enregistrar-ho al fitxer de log ("/var/log/syslog").*

8.- **/usr/src/linux-2.6.38/arch/x86/kernel/entry_32.S**

Aquest arxiu és el punt d'entrada al kernel; el que fa la funció de dispatching de les crides al sistema. S'ha de modificar perquè, abans de cridar a una syscall demanada, primer cridi a la nova syscall creada. Per a això, s'ha d'afegir:

```
[...]
/*
 * syscall stub including irq exit should be protected against kprobes
 */
       .pushsection .kprobes.text, "ax"
       # system call handler stub
ENTRY(system_call)
       RING0_INT_FRAME     # can't unwind into user space anyway
       pushl_cfi %eax      # save orig_eax
       SAVE_ALL
       GET_THREAD_INFO(%ebp)
                           # system call tracing in operation / emulation
       testl $_TIF_WORK_SYSCALL_ENTRY,TI_flags(%ebp)
       jnz syscall_trace_entry
       cmpl $(nr_syscalls), %eax
       jae syscall_badsys

syscall_call:
       call *sys_call_table(,264,4)
       call *sys_call_table(,%eax,4)
       movl %eax,PT_EAX(%esp)          # store the return value
syscall_exit:
[...]
```

9.- **Tornar a compilar el kernel i instal·lar-lo.**

Per a fer-ho, s'han de seguir els dos últims passos de la fase anterior:

Compilar el kernel:

```
/usr/src/linux-source-2.6.38$ sudo make-kpkg --append-to-version=.021111 --initrd kernel_image
```

Instal·lar el nou kernel, per poder arrancar-hi:

```
/usr/src$ sudo dpkg –i linux-image-2.6.38.8.021111_2.6.38.8.021111-10.00.Custom_i386.deb
```

10.- **Reiniciar l'equip amb aquest nucli.**

Amb això, queda modificat el kernel per intentar fer l'enregistrament de les syscalls.[6]

El procediment explicat anteriorment, però, no ha funcionat correctament.

6= He basat bona part d'aquesta fase en la documentació existent a Internet; en especial en un document publicat a: "http://tldp.org/HOWTO/html_single/Implement-Sys-Call-Linux-2.6-i386/"

Al arrancar, el sistema es queda bloquejat. La solució es basava en executar la meva crida al sistema personalitzada (sys_statistics) just abans de fer la syscall demanada, fent una crida a la taula de crides al sistema, però passant-li el número de la meva crida en comptes del valor d'%eax:

```
call *sys_call_table(,264,4)
```

El valor "264" és el número de la meva crida. El valor "4" és un multiplicador que fa servir el sistema per ubicar la posició en memòria de cada crida en la *array*, en la que cada crida ocupa 4 bytes. Però, a més, faltava passar-li a la meva funció un paràmetre indicant la crida al sistema demanada (valor que resideix al registre '%eax').

Però com ja he comentat abans, aquesta solució no funciona. Així que he buscat altres possibles solucions:

Solució 2:

He intentat cridar directament a la meva syscall (sys_statistics) des de l'arxiu 'entry_32.S', pel que he hagut de tornar a un kernel estable, i modificar la següent línia:

```
[...]
syscall_call:
        call *sys_statistics(,%eax)
        call *sys_call_table(,%eax,4)
        movl %eax,PT_EAX(%esp)          # store the return value
syscall_exit:
[...]
```

La idea, en aquest cas, era passar com a paràmetre a la meva funció 'sys_statistics' el valor contingut al registre '%eax', de manera que el meu codi enregistrés aquest valor fent un 'printk'. Després de tornar a compilar i instal·lar el nou nucli, he pogut comprovar que aquesta solució tampoc funciona, i el sistema es queda bloquejat en intentar arrancar.

Solució 3:

En aquesta ocasió, la idea és cridar directament a la funció 'printk', sense fer ús de la meva crida al sistema que havia desenvolupat. Per a fer-ho, he fet una crida directa a 'printk', modificant de nou l'arxiu 'entry_32.S', des d'un nucli estable:

```
[...]
syscall_call:
        call *printk(KERN_DEBUG, %eax)
        call *sys_call_table(,%eax,4)
        movl %eax,PT_EAX(%esp)        # store the return value
syscall_exit:
[...]
```

La idea, en aquest cas, era passar com a paràmetre a la funció 'printk' el valor 'KERN_DEBUG' (que indica que la sortida ha d'anar a parar a un arxiu de log, i no pas a consola), i el valor que s'imprimiria: el número de syscall demanada, present al registre '%eax'. De nou, aquesta solució no ha funcionat. El compilador no reconeix 'KERN_DEBUG'. Així que aquesta prova no ha passat d'aquest punt.

Solució 4:

Tenint en compte el resultat anterior, la nova proposta es basa en obviar el paràmetre 'KERN_DEBUG', ja que he pogut comprovar que aquest és el nivell per defecte en Linux 2.6 (tal com s'indica a 'DEFAULT_MESSAGE_LOGLEVEL' amb un valor enter a l'arxiu 'kernel/printk.c'). Encara que aquest no fos el nivell de log per defecte, es podria canviar amb el paràmetre indicat. Per tant, he pogut evitar l'ús d'aquest paràmetre i he modificat l'arxiu 'entry_32.S', amb la següent ordre:

```
[...]
syscall_call:
      call *printk(%eax)
      call *sys_call_table(,%eax,4)
      movl %eax,PT_EAX(%esp)        # store the return value
syscall_exit:
[...]
```

Un altre cop, el compilador no permet aquesta ordre perquè estic fent servir una ordre invàlida en llenguatge assemblador, al intentar passar-li un paràmetre directament. Per tant, també he de descartar aquesta solució.

Solució 5:

He pensat en fer servir la pila i els registres per passar el paràmetre a 'printk': La idea és posar el valor de '%eax' (el número de syscall cridada) a la pila, perquè 'printk' el tingui disponible. En acabar la meva crida, hauré de "netejar" la pila perquè el procés continuï amb normalitat. Per tant, aquesta solució es basa en guardar el valor que conté '%eax' a la pila, cridar a 'printk' perquè faci servir com a paràmetre el primer valor disponible, i després buidar el cim de la pila utilitzat. Per a això, he fet els següents canvis a l'arxiu 'entry_32.S':

```
[...]
syscall_call:
      push %eax
      call *printk
      pop %eax
      call *sys_call_table(,%eax,4)
      movl %eax,PT_EAX(%esp)        # store the return value
syscall_exit:
[...]
```

En aquesta ocasió, el compilador ha superat aquest punt, i ha construït un paquet amb el que arrancar. Al instal·lar-lo i intentar arrancar-hi, el sistema torna a quedar-se bloquejat. Aquesta solució tampoc és vàlida, i he d'arrancar de nou des d'un nucli estable.

Solució 6, Solució 7, Solució 8 i Solució 9:

He fet petites variacions de les solucions anteriors mitjançant diferents tècniques basades en crides a la funció 'sys_statistics', crides a la funció 'printk', diferents maneres de passar el registre '%eax' com a paràmetre, etc. Però cap d'aquestes solucions ha servit, ja fos perquè el compilador no admetia alguna ordre, o bé perquè compilava però el sistema es bloquejava al reiniciar amb el nou nucli, o bé perquè no produïa resultat algun.

Solució 10:

Veient la dificultat d'enregistrar les crides al sistema mitjançant un únic punt d'entrada, he optat per una solució que resulta segura: modificar cada una de les crides al sistema, de manera que totes facin una crida a 'printk' en ser executades. Aquesta solució, a més, evita que la crida 'printk' s'enregistri pel fet d'estar enregistrant la resta; cosa que desvirtuaria el resultat.

A partir d'una taula aconseguida a Internet[7], he pogut comprovar que són 72 els arxius que contenen funcions de crides al sistema. Per tant, s'han de modificar al voltant de 300 crides en uns 72 arxius. Tot i que és una tasca laboriosa, garanteix l'acompliment dels objectius d'aquest projecte. Aquesta és la llista de fitxers i funcions que s'han de canviar:

num	name	Source	
101	ioperm	arch/i386/kernel/ioport.c	1
110	iopl	arch/i386/kernel/ioport.c	
123	modify_ldt	arch/i386/kernel/ldt.c	2
2	fork	arch/i386/kernel/process.c	3
11	execve	arch/i386/kernel/process.c	
112	idle	arch/i386/kernel/process.c	
120	clone	arch/i386/kernel/process.c	
190	vfork	arch/i386/kernel/process.c	
26	ptrace	arch/i386/kernel/ptrace.c	4
67	sigaction	arch/i386/kernel/signal.c	5
72	sigsuspend	arch/i386/kernel/signal.c	
119	sigreturn	arch/i386/kernel/signal.c	
173	rt_sigreturn	arch/i386/kernel/signal.c	
179	rt_sigsuspend	arch/i386/kernel/signal.c	
186	sigaltstack	arch/i386/kernel/signal.c	
29	pause	arch/i386/kernel/sys_i386.c	6
42	pipe	arch/i386/kernel/sys_i386.c	
59	olduname	arch/i386/kernel/sys_i386.c	
82	old_select	arch/i386/kernel/sys_i386.c	
90	old_mmap	arch/i386/kernel/sys_i386.c	
109	olduname	arch/i386/kernel/sys_i386.c	
117	ipc	arch/i386/kernel/sys_i386.c	
113	vm86old	arch/i386/kernel/vm86.c	7
166	vm86	arch/i386/kernel/vm86.c	
243	set_thread_area	arch/mips/kernel/syscall.c:222	8
102	socketcall	cket.c	9
245	io_setup	fs/aio.c:1245	10
246	io_destroy	fs/aio.c:1283	
248	io_submit	fs/aio.c:1711	
249	io_cancel	fs/aio.c:1746	
247	io_getevents	fs/aio.c:1808	
36	sync	fs/buffer.c	11
118	fsync	fs/buffer.c	
134	bdflush	fs/buffer.c	
148	fdatasync	fs/buffer.c	
183	getcwd	fs/dcache.c	12
131	quotactl	fs/dquot.c	13
328	eventfd2	fs/eventfd.c:409	14
323	eventfd	fs/eventfd.c:434	
329	epoll_create1	fs/eventpoll.c:1187	15
254	epoll_create	fs/eventpoll.c:1215	
255	epoll_ctl	fs/eventpoll.c:1228	
256	epoll_wait	fs/eventpoll.c:1320	
319	epoll_pwait	fs/eventpoll.c:1373	
86	uselib	fs/exec.c	16
41	dup	fs/fcntl.c	17

num	name	Source	
16	lchown	fs/open.c	
30	utime	fs/open.c	
33	access	fs/open.c	
61	chroot	fs/open.c	
92	truncate	fs/open.c	
93	ftruncate	fs/open.c	
94	fchmod	fs/open.c	
95	fchown	fs/open.c	
99	statfs	fs/open.c	
100	fstatfs	fs/open.c	
111	vhangup	fs/open.c	
133	fchdir	fs/open.c	
182	chown	fs/open.c	
307	faccessat	fs/open.c:286	
306	fchmodat	fs/open.c:474	
212	chown	fs/open.c:539	
298	fchownat	fs/open.c:558	
198	lchown	fs/open.c:583	
207	fchown	fs/open.c:602	
295	openat	fs/open.c:913	
331	pipe2	fs/pipe.c:1101	26
3	read	fs/read_write.c	27
4	write	fs/read_write.c	
19	lseek	fs/read_write.c	
140	sys_llseek	fs/read_write.c	
145	readv	fs/read_write.c	
146	writev	fs/read_write.c	
180	pread	fs/read_write.c	
181	sys_pwrite	fs/read_write.c	
333	preadv	fs/read_write.c:759	
334	pwritev	fs/read_write.c:784	
239	sendfile64	fs/read_write.c:916	
89	old_readdir	fs/readdir.c	28
141	getdents	fs/readdir.c	
220	getdents64	fs/readdir.c:273	
142	select	fs/select.c	29
168	poll	fs/select.c	
308	pselect6	fs/select.c:675	
309	ppoll	fs/select.c:950	
327	signalfd4	fs/signalfd.c:211	30
321	signalfd	fs/signalfd.c:265	
316	vmsplice	fs/splice.c:1692	31
313	splice	fs/splice.c:1718	
315	tee	fs/splice.c:2061	
18	stat	fs/stat.c	32
28	fstat	fs/stat.c	

55	fcntl	fs/fcntl.c		84	lstat	fs/stat.c	
63	dup2	fs/fcntl.c		85	readlink	fs/stat.c	
221	fcntl64	fs/fcntl.c:452		106	sys_newstat	fs/stat.c	
330	dup3	fs/fcntl.c:53		107	sys_newlstat	fs/stat.c	
169	nfsservctl	fs/filesystems.c	18	108	sys_newfstat	fs/stat.c	
54	ioctl	fs/ioctl.c	19	305	readlinkat	fs/stat.c:284	
290	ioprio_get	fs/ioprio.c:192	20	195	stat64	fs/stat.c:358	
289	ioprio_set	fs/ioprio.c:76		196	lstat64	fs/stat.c:369	
143	flock	fs/locks.c	21	197	fstat64	fs/stat.c:380	
9	link	fs/namei.c	22	300	fstatat64	fs/stat.c:391	
10	unlink	fs/namei.c		268	statfs64	fs/statfs.c:118	33
14	mknod	fs/namei.c		269	fstatfs64	fs/statfs.c:154	
38	rename	fs/namei.c		21	mount	fs/super.c	34
39	mkdir	fs/namei.c		22	umount	fs/super.c	
40	rmdir	fs/namei.c		52	umount2	fs/super.c	
83	symlink	fs/namei.c		62	ustat	fs/super.c	
297	mknodat	fs/namei.c:2012		135	sysfs	fs/super.c	
296	mkdirat	fs/namei.c:2093		322	timerfd_create	fs/timerfd.c:164	35
301	unlinkat	fs/namei.c:2341		325	timerfd_settime	fs/timerfd.c:194	
304	symlinkat	fs/namei.c:2377		326	timerfd_gettime	fs/timerfd.c:252	
303	linkat	fs/namei.c:2470		320	utimensat	fs/utimes.c:173	36
302	renameat	fs/namei.c:2671		299	futimesat	fs/utimes.c:191	
217	pivot_root	fs/namespace.c:2184	23	271	utimes	fs/utimes.c:219	
332	inotify_init1	fs/notify/inotify/inotify_user.c:640	24	226	setxattr	fs/xattr.c:279	37
291	inotify_init	fs/notify/inotify/inotify_user.c:680		227	lsetxattr	fs/xattr.c:298	
292	inotify_add_watch	fs/notify/inotify/inotify_user.c:685		228	fsetxattr	fs/xattr.c:317	
293	inotify_rm_watch	fs/notify/inotify/inotify_user.c:726		229	getxattr	fs/xattr.c:376	
5	open	fs/open.c	25	230	lgetxattr	fs/xattr.c:390	
6	close	fs/open.c		231	fgetxattr	fs/xattr.c:404	
8	creat	fs/open.c		232	listxattr	fs/xattr.c:449	
12	chdir	fs/open.c		233	llistxattr	fs/xattr.c:463	
15	chmod	fs/open.c		234	flistxattr	fs/xattr.c:477	

Continuació de la taula de fitxers i funcions de crides al sistema:

num	name	Source		num	name	Source	
235	removexattr	fs/xattr.c:509		23	setuid	kernel/sys.c	55
236	lremovexattr	fs/xattr.c:527		43	times	kernel/sys.c	
237	fremovexattr	fs/xattr.c:545		46	setgid	kernel/sys.c	
281	mq_notify	ipc/mqueue.c:1023	38	57	setpgid	kernel/sys.c	
282	mq_getsetattr	ipc/mqueue.c:1154		60	umask	kernel/sys.c	
277	mq_open	ipc/mqueue.c:673		65	getpgrp	kernel/sys.c	
278	mq_unlink	ipc/mqueue.c:746		66	setsid	kernel/sys.c	
279	mq_timedsend	ipc/mqueue.c:840		70	setreuid	kernel/sys.c	
280	mq_timedreceive	ipc/mqueue.c:934		71	setregid	kernel/sys.c	
51	acct	kernel/acct.c	39	74	sethostname	kernel/sys.c	
184	capget	kernel/capability.c	40	75	setrlimit	kernel/sys.c	
185	capset	kernel/capability.c		76	getrlimit	kernel/sys.c	
136	personality	kernel/exec_domain.c	41	77	getrusage	kernel/sys.c	
1	exit	kernel/exit.c	42	80	getgroups	kernel/sys.c	
7	waitpid	kernel/exit.c		81	setgroups	kernel/sys.c	
114	wait4	kernel/exit.c		88	reboot	kernel/sys.c	
252	exit_group	kernel/exit.c:1087		96	getpriority	kernel/sys.c	
284	waitid	kernel/exit.c:1655		97	setpriority	kernel/sys.c	
310	unshare	kernel/fork.c:1624	43	121	setdomainname	kernel/sys.c	
258	set_tid_address	kernel/fork.c:920		122	uname	kernel/sys.c	
311	set_robust_list	kernel/futex.c:2351	44	132	getpgid	kernel/sys.c	
312	get_robust_list	kernel/futex.c:2373		138	setfsuid	kernel/sys.c	
240	futex	kernel/futex.c:2605		139	setfsgid	kernel/sys.c	
205	getgroups	kernel/groups.c:203	45	147	sys_getsid	kernel/sys.c	

206	setgroups	kernel/groups.c:232		164	setresuid	kernel/sys.c	
116	sysinfo	kernel/info.c	46	165	getresuid	kernel/sys.c	
104	setitimer	kernel/itimer.c	47	170	setresgid	kernel/sys.c	
105	getitimer	kernel/itimer.c		171	getresgid	kernel/sys.c	
283	kexec_load	kernel/kexec.c:939	48	172	prctl	kernel/sys.c	
127	create_module	kernel/module.c	49	191	getrlimit	kernel/sys.c:1237	
128	init_module	kernel/module.c		318	getcpu	kernel/sys.c:1621	
129	delete_module	kernel/module.c		204	setregid	kernel/sys.c:484	
130	get_kernel_syms	kernel/module.c		214	setgid	kernel/sys.c:531	
167	query_module	kernel/module.c		203	setreuid	kernel/sys.c:594	
336	perf_event_open	kernel/perf_event.c:5065	50	213	setuid	kernel/sys.c:655	
267	clock_nanosleep	kernel/posix-timers.c:1001	51	208	setresuid	kernel/sys.c:696	
259	timer_create	kernel/posix-timers.c:522		209	getresuid	kernel/sys.c:746	
261	timer_gettime	kernel/posix-timers.c:702		210	setresgid	kernel/sys.c:761	
262	timer_getoverrun	kernel/posix-timers.c:732		211	getresgid	kernel/sys.c:800	
260	timer_settime	kernel/posix-timers.c:800		215	setfsuid	kernel/sys.c:819	
263	timer_delete	kernel/posix-timers.c:855		216	setfsgid	kernel/sys.c:852	
264	clock_settime	kernel/posix-timers.c:941		149	sysctl	kernel/sysctl.c	56
265	clock_gettime	kernel/posix-timers.c:954		13	time	kernel/time.c	57
266	clock_getres	kernel/posix-timers.c:971		25	stime	kernel/time.c	
103	syslog	kernel/printk.c	52	78	gettimeofday	kernel/time.c	
20	getpid	kernel/sched.c	53	79	settimeofday	kernel/time.c	
24	getuid	kernel/sched.c		124	adjtimex	kernel/time.c	
27	alarm	kernel/sched.c		199	getuid	kernel/timer.c:1359	58
34	nice	kernel/sched.c		201	geteuid	kernel/timer.c:1365	
47	getgid	kernel/sched.c		200	getgid	kernel/timer.c:1371	
49	geteuid	kernel/sched.c		202	getegid	kernel/timer.c:1377	
50	getegid	kernel/sched.c		224	gettid	kernel/timer.c:1493	
64	getppid	kernel/sched.c		144	msync	mm/filemap.c	59
154	sched_setparam	kernel/sched.c		187	sendfile	mm/filemap.c	
155	sched_getparam	kernel/sched.c		257	remap_file_pages	mm/fremap.c:123	60
156	sched_setscheduler	kernel/sched.c		219	madvise	mm/madvise.c:335	61
157	sched_getscheduler	kernel/sched.c		274	mbind	mm/mempolicy.c:1232	62
158	sched_yield	kernel/sched.c		276	set_mempolicy	mm/mempolicy.c:1254	
159	sched_get_priority_	kernel/sched.c		294	migrate_pages	mm/mempolicy.c:1273	
160	sched_get_priority_	kernel/sched.c		275	get_mempolicy	mm/mempolicy.c:1348	
161	sched_rr_get_inter	kernel/sched.c		317	move_pages	mm/migrate.c:1075	63
162	nanosleep	kernel/sched.c		218	mincore	mm/mincore.c:256	64
241	sched_setaffinity	kernel/sched.c:4765		150	mlock	mm/mlock.c	65
242	sched_getaffinity	kernel/sched.c:4817		151	munlock	mm/mlock.c	
37	kill	kernel/signal.c	54	152	mlockall	mm/mlock.c	
48	sys_signal	kernel/signal.c		153	munlockall	mm/mlock.c	
68	sgetmask	kernel/signal.c		45	brk	mm/mmap.c	66
69	ssetmask	kernel/signal.c		91	munmap	mm/mmap.c	
73	sigpending	kernel/signal.c		192	mmap_pgoff	mm/mmap.c:1091	
126	sigprocmask	kernel/signal.c		125	mprotect	mm/mprotect.c	67
174	rt_sigaction	kernel/signal.c		163	mremap	mm/mremap.c	68
175	rt_sigprocmask	kernel/signal.c		87	swapon	mm/swapfile.c	69
176	rt_sigpending	kernel/signal.c		115	swapoff	mm/swapfile.c	
177	rt_sigtimedwait	kernel/signal.c		337	recvmmsg	net/socket.c:2168	70
178	rt_sigqueueinfo	kernel/signal.c		98	profil	profile	71
270	tgkill	kernel/signal.c:2383		288	keyctl	security/keys/keyctl.c	72

Taula 01: SysCalls (taula de crides al sistema, amb el seu número, nom i localització en 72 fitxers .C)

7= Llistat de crides al sistema en un nucli de Linux 2.6.35.4: "http://syscalls.kernelgrok.com/", facilitat per un enllaç disponible a un article de crides al sistema a Wikipedia: "http://en.wikipedia.org/wiki/System_call"

La modificació d'aquests arxius es basa en una tasca laboriosa però repetitiva, en la que he insertat la següent ordre després del bloc de declaracions (a mode d'exemples):

fs/open.c:

```
[...]
SYSCALL_DEFINE3(open, const char __user *, filename, int, flags, int, mode)
{
long ret;
//GGuirado SysCall Statistics
printk(KERN_DEBUG "|open");
if (force_o_largefile())
       flags |= O_LARGEFILE;
ret = do_sys_open(AT_FDCWD, filename, flags, mode);
/* avoid REGPARM breakage on x86: */
asmlinkage_protect(3, ret, filename, flags, mode);
return ret;
}
[...]
SYSCALL_DEFINE1(close, unsigned int, fd)
{
struct file * filp;
struct files_struct *files = current->files;
struct fdtable *fdt;
int retval;
//GGuirado SysCall Statistics
printk(KERN_DEBUG "|close");
spin_lock(&files->file_lock);
[...]
```

kernels/sys.c:

```
[...]
SYSCALL_DEFINE1(setuid, uid_t, uid)
{
const struct cred *old;
struct cred *new;
int retval;
//GGuirado SysCall Statistics
printk(KERN_DEBUG "|setuid");
new = prepare_creds();
[...]
SYSCALL_DEFINE1(uname, struct old_utsname __user *, name)
{
int error = 0;
//GGuirado SysCall Statistics
printk(KERN_DEBUG "|uname");
if (!name)
[...]
```

Després de compilar i instal·lar el nou nucli amb aquestes modificacions, el sistema ha arrancat correctament, i ha començat a enregistrar els missatges generats. Els resultats es mostren en la següent fase.

1.2.4.- Fase d'informació i resultats

var/log/syslog:

Quan el 'MESSAGE_LOGLEVEL' està establert a 'KERN_DEBUG', la crida 'printk', enregistra els seus missatges a un arxiu de log. Aquest arxiu es diu 'syslog', i es troba al directory '/var/log'. Quan va creixent aquest arxiu, es creen arxius històrics 'syslog.1', 'syslog.2', i després es comprimeixen per estalviar espai. En aquest cas, només amb els resultats enregistrats en el fitxer 'syslog' ja es disposa de més de dos milions de missatges.

A continuació es presenta un fragment d'aquest arxiu, extret d'una part on es veuen els missatges enregistrats per la meva crida '`printk(KERN_DEBUG "|NomDeLaSysCall")`', en un inici normal del sistema:

```
[...]
Dec 16 18:12:24 HPCompaq kernel: [   19.699731] |close
Dec 16 18:12:24 HPCompaq kernel: [   19.699730] |read
Dec 16 18:12:24 HPCompaq kernel: [   19.699734] |close
Dec 16 18:12:24 HPCompaq kernel: [   19.699733] |read
Dec 16 18:12:24 HPCompaq kernel: [   19.699736] |close
Dec 16 18:12:24 HPCompaq kernel: [   19.699736] |waitid
Dec 16 18:12:24 HPCompaq kernel: [   19.699739] |close
Dec 16 18:12:24 HPCompaq kernel: [   19.699739] |clock_gettime
Dec 16 18:12:24 HPCompaq kernel: [   19.699742] |close
Dec 16 18:12:24 HPCompaq kernel: [   19.699743] |close
Dec 16 18:12:24 HPCompaq kernel: [   19.699745] |close
Dec 16 18:12:24 HPCompaq kernel: [   19.699746] |close
Dec 16 18:12:24 HPCompaq kernel: [   19.699748] |close
Dec 16 18:12:24 HPCompaq kernel: [   19.699747] |select
Dec 16 18:12:24 HPCompaq kernel: [   19.699751] |close
[...]
```

D'aquest fitxer n'he extret una mostra representativa -un milió de resultats- (1.000.000) per fer una estadística completa i fiable. Mitjançant fórmules a un full de càlcul (i aprofitant el símbol '|' que havia col·locat expressament), he aïllat els noms de les crides al sistema, i n'he fet un recompte.

Un cop ordenades pel número d'ocurrències de cada SysCall al fitxer, es pot veure clarament que 'time', 'gettimeofday' i 'write' són, amb diferència, les crides més habituals. Això fa veure que aquestes SysCalls són conseqüència del mateix fet d'haver escrit resultats al fitxer de log. És a dir: Cada cop que s'inserta un nou missatge a 'syslog', s'escriu amb la marca de temps actual (per exemple: `Dec 16 18:12:24 HPCompaq kernel: [   19.699747] |select`).

Aquesta situació s'hauria produït amb qualsevol de les tècniques utilitzada per enregistrar crides del sistema. Per tant, per a tenir un resultat correcte, obviant els resultats produïts per la pròpia observació, s'han de descartar aquests resultats grans més extrems.

A continuació es mostra la taula inicial, amb totes les crides al sistema que tenen alguna aparició al fitxer de resultats:

num	SysCall	Description	Source	1.000.000
13	time	get time in seconds	kernel/time.c	414.820
78	gettimeofday	get the date and time	kernel/time.c	354.551
4	write	write to a file descriptor	fs/read_write.c	167.434
195	stat64		fs/stat.c:358	8.936
3	read	read from a file descriptor	fs/read_write.c	8.867
265	clock_gettime		kernel/posix-timers.c:954	8.039
158	sched_yield	yield the processor	kernel/sched.c	5.195
6	close	close a file descriptor	fs/open.c	3.541
168	poll	wait for some event on a file descriptor	fs/select.c	3.268
5	open	open a file or device	fs/open.c	2.895
240	futex		kernel/futex.c:2605	2.478
125	mprotect	set protection of memory mapping	mm/mprotect.c	1.929
192	mmap_pgoff		mm/mmap.c:1091	1.923
146	writev	write data into multiple buffers	fs/read_write.c	1.517
33	access	check user's permissions for a file	fs/open.c	1.428
307	faccessat		fs/open.c:286	1.419
197	fstat64		fs/stat.c:380	1.345
221	fcntl64		fs/fcntl.c:452	1.341
54	ioctl	control device	fs/ioctl.c	1.288
104	setitimer	set value of interval timer	kernel/itimer.c	1.273
256	epoll_wait		fs/eventpoll.c:1320	995
142	select	sync. I/O multiplexing	fs/select.c	979
91	munmap	unmap pages of memory	mm/mmap.c	563
85	readlink	read the contents of a symbolic link	fs/stat.c	348
305	readlinkat		fs/stat.c:284	347
220	getdents64		fs/readdir.c:273	336
174	rt_sigaction		kernel/signal.c	324
196	lstat64		fs/stat.c:369	324
45	brk	change the amount of space allocated for the calling…	mm/mmap.c	213
140	sys_llseek	move extended read/write file pointer	fs/read_write.c	208
24	getuid	get real user ID	kernel/sched.c	146
199	getuid		kernel/timer.c:1359	146
331	pipe2		fs/pipe.c:1101	137
309	ppoll		fs/select.c:950	112
42	pipe	create an interprocess channel	arch/i386/kernel/sys_i386.c	102
284	waitid		kernel/exit.c:1655	95
114	wait4	wait for process termination, BSD style	kernel/exit.c	76
63	dup2	duplicate a file descriptor	fs/fcntl.c	68
330	dup3		fs/fcntl.c:53	66
119	sigreturn	return from signal handler and cleanup stack frame	arch/i386/kernel/signal.c	59
49	geteuid	get effective user ID	kernel/sched.c	56
201	geteuid		kernel/timer.c:1365	56
175	rt_sigprocmask		kernel/signal.c	47
64	getppid	get parent process ID	kernel/sched.c	46
7	waitpid	wait for process termination	kernel/exit.c	44
252	exit_group		kernel/exit.c:1087	44
141	getdents	read directory entries	fs/readdir.c	43

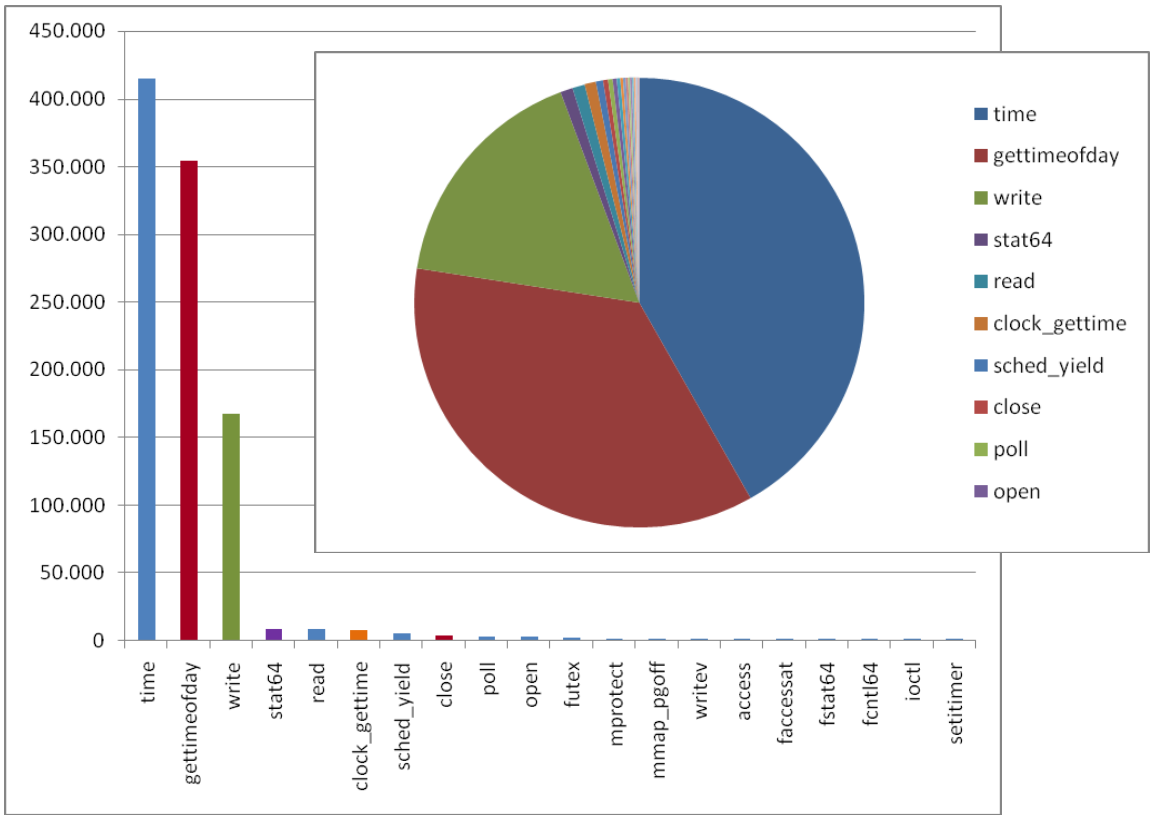
183	getcwd	Get current working directory	fs/dcache.c	25
311	set_robust_list		kernel/futex.c:2351	23
117	ipc	System V IPC system calls	arch/i386/kernel/sys_i386.c	21
219	madvise		mm/madvise.c:335	21
50	getegid	get effective group ID	kernel/sched.c	20
162	nanosleep	pause execution for a specified time (nano seconds)	kernel/sched.c	20
202	getegid		kernel/timer.c:1377	20
12	chdir	change working directory	fs/open.c	16
296	mkdirat		fs/namei.c:2093	15
39	mkdir	create a directory	fs/namei.c	14
255	epoll_ctl		fs/eventpoll.c:1228	14
76	getrlimit	get maximum system resource con sumption	kernel/sys.c	13
191	getrlimit		kernel/sys.c:1237	13
266	clock_getres		kernel/posix-timers.c:971	13
292	inotify_add_watch		fs/notify/inotify/inotify_user.c:685	13
10	unlink	delete a name and possibly the file it refers to	fs/namei.c	12
38	rename	change the name or location of a file	fs/namei.c	12
60	umask	set file creation mask	kernel/sys.c	12
302	renameat		fs/namei.c:2671	12
258	set_tid_address		kernel/fork.c:920	11
268	statfs64		fs/statfs.c:118	9
80	getgroups	get list of supplementary group IDs	kernel/sys.c	8
205	getgroups		kernel/groups.c:203	8
20	getpid	get process identification	kernel/sched.c	7
26	ptrace	allows a parent process to control the execution of a child… process	arch/i386/kernel/ptrace.c	7
47	getgid	get real group ID	kernel/sched.c	7
155	sched_getparam	get scheduling parameters	kernel/sched.c	7
157	sched_getscheduler	get scheduling algorithm parameters	kernel/sched.c	7
200	getgid		kernel/timer.c:1371	7
293	inotify_rm_watch		fs/notify/inotify/inotify_user.c:726	7
15	chmod	change permissions of a file	fs/open.c	5
43	times	get process times	kernel/sys.c	5
77	getrusage	get maximum system resource con sumption	kernel/sys.c	5
156	sched_setscheduler	set scheduling algorithm parameters	kernel/sched.c	5
270	tgkill		kernel/signal.c:2383	5
306	fchmodat		fs/open.c:474	5
1	exit	terminate the current process	kernel/exit.c	4
41	dup	duplicate an open file descriptor	fs/fcntl.c	4
66	setsid	creates a session and sets the process group ID	kernel/sys.c	4
138	setfsuid	set user identity used for file system checks	kernel/sys.c	4
139	setfsgid	set group identity used for file system checks	kernel/sys.c	4
215	setfsuid		kernel/sys.c:819	4
216	setfsgid		kernel/sys.c:852	4
332	inotify_init1		fs/notify/inotify/inotify_user.c:640	4
27	alarm	set an alarm clock for delivery of a signal	kernel/sched.c	3
30	utime	set file access and modification times	fs/open.c	3
37	kill	send signal to a process	kernel/signal.c	3
95	fchown	change owner and group of a file	fs/open.c	3

97	setpriority	set program scheduling priority	kernel/sys.c	3
165	getresuid	get real, effective and saved user or group ID	kernel/sys.c	3
171	getresgid	get real, effective and saved user or group ID	kernel/sys.c	3
172	prctl	operations on a process	kernel/sys.c	3
207	fchown		fs/open.c:602	3
209	getresuid		kernel/sys.c:746	3
211	getresgid		kernel/sys.c:800	3
81	setgroups	set list of supplementary group IDs	kernel/sys.c	2
94	fchmod	change access permission mode of file	fs/open.c	2
133	fchdir	change working directory	fs/open.c	2
143	flock	apply or remove an advisory lock on an open file	fs/locks.c	2
182	chown	change ownership of a file	fs/open.c	2
184	capget	get process capabilities	kernel/capability.c	2
185	capset	set process capabilities	kernel/capability.c	2
206	setgroups		kernel/groups.c:232	2
212	chown		fs/open.c:539	2
8	creat	create a file or device ("man 2 open" for information)	fs/open.c	1
9	link	make a new name for a file	fs/namei.c	1
23	setuid	set real user ID	kernel/sys.c	1
46	setgid	set real group ID	kernel/sys.c	1
61	chroot	change root directory	fs/open.c	1
75	setrlimit	set maximum system resource con sumption	kernel/sys.c	1
148	fdatasync	synchronize a file's in-core data with that on disk	fs/buffer.c	1
150	mlock	lock pages in memory	mm/mlock.c	1
164	setresuid	set real, effective and saved user or group ID	kernel/sys.c	1
170	setresgid	set real, effective and saved user or group ID	kernel/sys.c	1
208	setresuid		kernel/sys.c:696	1
210	setresgid		kernel/sys.c:761	1
213	setuid		kernel/sys.c:655	1
214	setgid		kernel/sys.c:531	1
224	gettid		kernel/timer.c:1493	1
233	llistxattr		fs/xattr.c:463	1
254	epoll_create		fs/eventpoll.c:1215	1
269	fstatfs64		fs/statfs.c:154	1
271	utimes		fs/utimes.c:219	1
291	inotify_init		fs/notify/inotify/inotify_user.c:680	1
299	futimesat		fs/utimes.c:191	1
300	fstatat64		fs/stat.c:391	1
301	unlinkat		fs/namei.c:2341	1
303	linkat		fs/namei.c:2470	1
328	eventfd2		fs/eventfd.c:409	1
329	epoll_create1		fs/eventpoll.c:1187	1

Taula 02: Recompte complet de SysCalls (en un inici normal del sistema [Linux 2.6.38.8-021111: Ubuntu 11.04]).

El fet de no descartar les tres crides més repetides al fitxer, produiria una interpretació incorrecta de les estadístiques.

Aquests són els gràfics de les 20 primeres crides al sistema amb més de 1.000 aparicions:

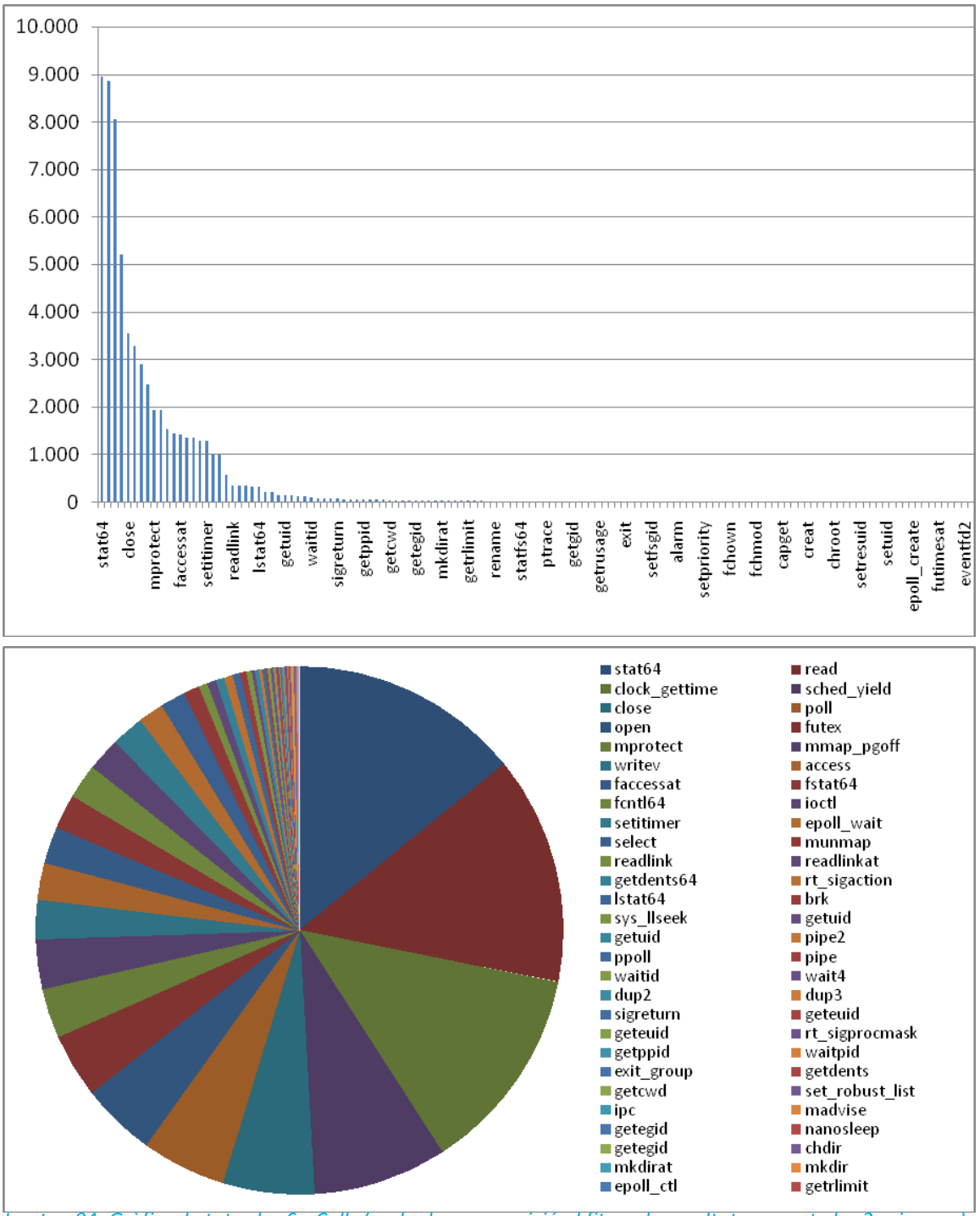


Imatge 03: Gràfics de les primeres 20 SysCalls (amb més de 1.000 aparicions al fitxer de resultats).

Com es pot apreciar, entre les tres primeres crides ('time', 'gettimeofday' i 'write'), sumen la pràctica totalitat dels 20 primers resultats (936.805 aparicions de 993.487: 94,29%), i la gran majoria d'entre el total de 137 resultats (936.805 aparicions d'1.000.000: 93,68%).

Així doncs, per oferir una visió clara i realista dels resultats, s'han d'obviar aquestes crides, resultat del mateix enregistrament.

A continuació, l'estadística es realitza a partir dels resultats descartant les crides 'time', 'gettimeofday' i 'write'. Però, de nou, com es pot observar a continuació, englobar totes les crides al sistema en un mateix gràfic, no ajuda a interpretar la informació (hi ha massa elements [134] a estudiar, i no es pot apreciar el detall perquè els valors són massa diferents):



Imatge 04: Gràfics de totes les SysCalls (amb alguna aparició al fitxer de resultats, excepte les 3 primeres).

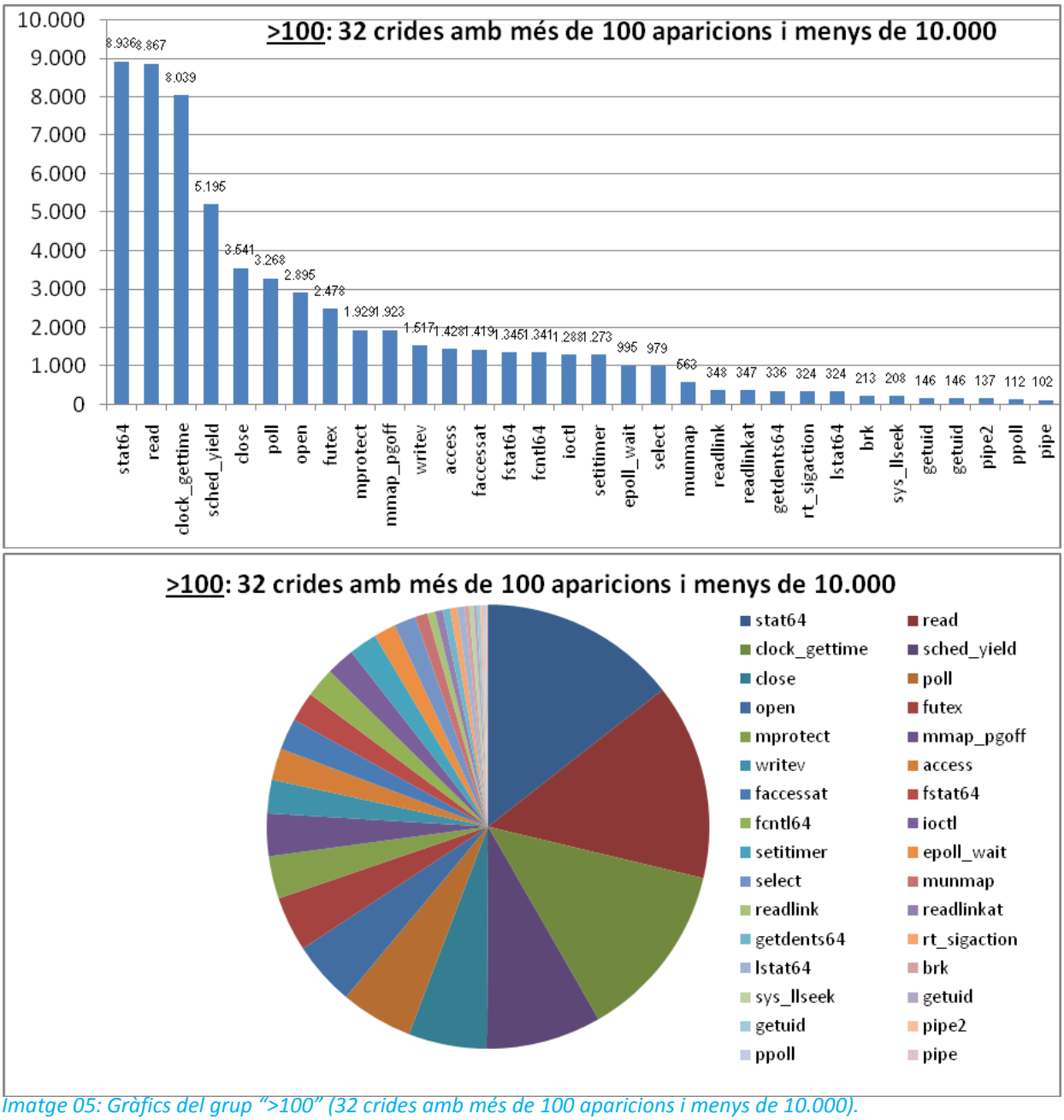
Amb l'objectiu d'oferir una visió clara de les estadístiques, he dividit els resultats en tres categories, en funció d'una agrupació per nombre d'aparicions al fitxer de resultats:

>100: 32 crides amb més de 100 aparicions i menys de 10.000:

stat64, read, clock_gettime, sched_yield, close, poll, open, futex, mprotect, mmap_pgoff, writev, access, faccessat, fstat64, fcntl64, ioctl, setitimer, epoll_wait, select, munmap, readlink, readlinkat, getdents64, rt_sigaction, lstat64, brk, sys_llseek, getuid, getuid, pipe2, ppoll, pipe.

num	SysCall	Description	Source	1.000.000
195	stat64		fs/stat.c:358	8.936
3	read	read from a file descriptor	fs/read_write.c	8.867
265	clock_gettime		kernel/posix-timers.c:954	8.039
158	sched_yield	yield the processor	kernel/sched.c	5.195
6	close	close a file descriptor	fs/open.c	3.541
168	poll	wait for some event on a file descriptor	fs/select.c	3.268
5	open	open a file or device	fs/open.c	2.895
240	futex		kernel/futex.c:2605	2.478
125	mprotect	set protection of memory mapping	mm/mprotect.c	1.929
192	mmap_pgoff		mm/mmap.c:1091	1.923
146	writev	write data into multiple buffers	fs/read_write.c	1.517
33	access	check user's permissions for a file	fs/open.c	1.428
307	faccessat		fs/open.c:286	1.419
197	fstat64		fs/stat.c:380	1.345
221	fcntl64		fs/fcntl.c:452	1.341
54	ioctl	control device	fs/ioctl.c	1.288
104	setitimer	set value of interval timer	kernel/itimer.c	1.273
256	epoll_wait		fs/eventpoll.c:1320	995
142	select	sync. I/O multiplexing	fs/select.c	979
91	munmap	unmap pages of memory	mm/mmap.c	563
85	readlink	read the contents of a symbolic link	fs/stat.c	348
305	readlinkat		fs/stat.c:284	347
220	getdents64		fs/readdir.c:273	336
174	rt_sigaction		kernel/signal.c	324
196	lstat64		fs/stat.c:369	324
45	brk	change the amount of space allocated for the calling…	mm/mmap.c	213
140	sys_llseek	move extended read/write file pointer	fs/read_write.c	208
24	getuid	get real user ID	kernel/sched.c	146
199	getuid		kernel/timer.c:1359	146
331	pipe2		fs/pipe.c:1101	137
309	ppoll		fs/select.c:950	112
42	pipe	create an interprocess channel	arch/i386/kernel/sys_i386.c	102

Taula 03: Recompte del grup ">100" (32 crides amb més de 100 aparicions i menys de 10.000).



Imatge 05: Gràfics del grup ">100" (32 crides amb més de 100 aparicions i menys de 10.000).

SysCalls del grup ">100":

stat64, read, clock_gettime, sched_yield, close poll, open, futex, mprotect, mmap_pgoff, writev, access, faccessat, fstat64, fcntl64, ioctl, setitimer, epoll_wait, select, munmap, readlink, readlinkat, getdents64, rt_sigaction, lstat64, brk, sys_llseek, getuid, getuid, pipe2, ppoll, pipe.

Al capítol "1.4.- CONCLUSIONS", s'analitzen i valoren aquests resultats.

stat64 (8.936 aparicions [14,14%]): Ofereix informació sobre un arxiu (ID del dispositiu que el conté, protecció, ID del usuari propietari, tamany en bytes, últim accés/modificació, etc). *http://linux.die.net/man/2/stat64*

read (8.867 ap. [14,03%]): Llegeix informació d'una memòria intermèdia ('*buffer*'). / Intenta llegir *n* bytes de dades de l'objecte referenciat pel descriptor *d* al '*buffer*' apuntat per *buf*. *http://codewiki.wikidot.com/c:system-calls:read* / *http://comsci.liu.edu/~murali/unix/read.htm*

clock_gettime (8.039 [12,72%]): Permet, al procés que la crida, obtenir el valor d'un rellotge indicat per clock_id (CLOCK_REALTIME, CLOCK_MONOTONIC, CLOCK_UPTIME, ...). *http://fuse4bsd.creo.hu/localcgi/man-cgi.cgi?clock_gettime+2*

sched_yield (5.195 [8,22%]): Un procés pot abandonar i cedir el processador voluntàriament sense bloquejar-se cridant a aquesta syscall, i posant-se a la cua mentre un altre s'executa. *http://es.tldp.org/Paginas-manual/man-pages-es-1.28/man2/sched_yield.2.html*

close (3.541 [5,60%]): Tanca el descriptor d'un arxiu obert. / Elimina un descriptor de la taula de referències a objecte per procés. Si és la última referència al objecte indicat, serà desactivat. *http://codewiki.wikidot.com/c:system-calls:close* / *http://comsci.liu.edu/~murali/unix/close.htm*

poll (3.268 [5,17%]): Espera un event en un descriptor d'arxiu. / Espera a que un d'un grup de descriptors estigui preparat per fer una entrada sortida (I/O). [Similar a '*select*'.] *http://linux.die.net/man/2/poll* / *http://www.tutorialspoint.com/unix_system_calls/poll.htm*

open (2.895 [4,58%]): Obre un nou fitxer i obté el seu descriptor. / Localitza els recursos associats a l'arxiu (el descriptor d'arxiu) i retorna un punter que el procés utilitzarà per referir-s'hi. *http://codewiki.wikidot.com/c:system-calls:open* / *http://en.wikipedia.org/wiki/Open_(system_call)*

futex (2.478 [3,92%]): *Fast Userspace mutEX [Locking]*: Construcció de Linux per implementar bloqueig bàsic (o bloquejos abstractes com semàfors, POSIX *mutex* o variables condició). *http://linux.die.net/man/2/futex* / *http://en.wikipedia.org/wiki/Futex*

mprotect (1.929 [3,05%]): *Memory protection*: Mecanisme per controlar els drets d'accés a memòria en una computador. / Canvia la protecció de les pàgines indicades al nou valor indicat. *http://en.wikipedia.org/wiki/Memory_protection* / *http://www.manpagez.com/man/2/mprotect/*

mmap_pgoff (1.923 [3,04%]): Similar a mmap, però es tracta d'una *callback* (mecanisme de resposta), que es cridada abans d'obtenir l'actual *mutex* de mmap. *http://comments.gmane.org/gmane.linux.nfs/30265*

writev (1.517 [2,40%]): Escriu dades a un fitxer o descriptor de *socket*. Ofereix un mètode perquè les dades que han de ser escrites, puguin ser guardades en varis *buffers* diferents. *http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=%2Fapis%2Fwritev.htm*

access (1.428 [2,26%]): Comprova els permisos d'accés d'un usuari a un fitxer. *http://man.freetechsecrets.com/access.2.html*

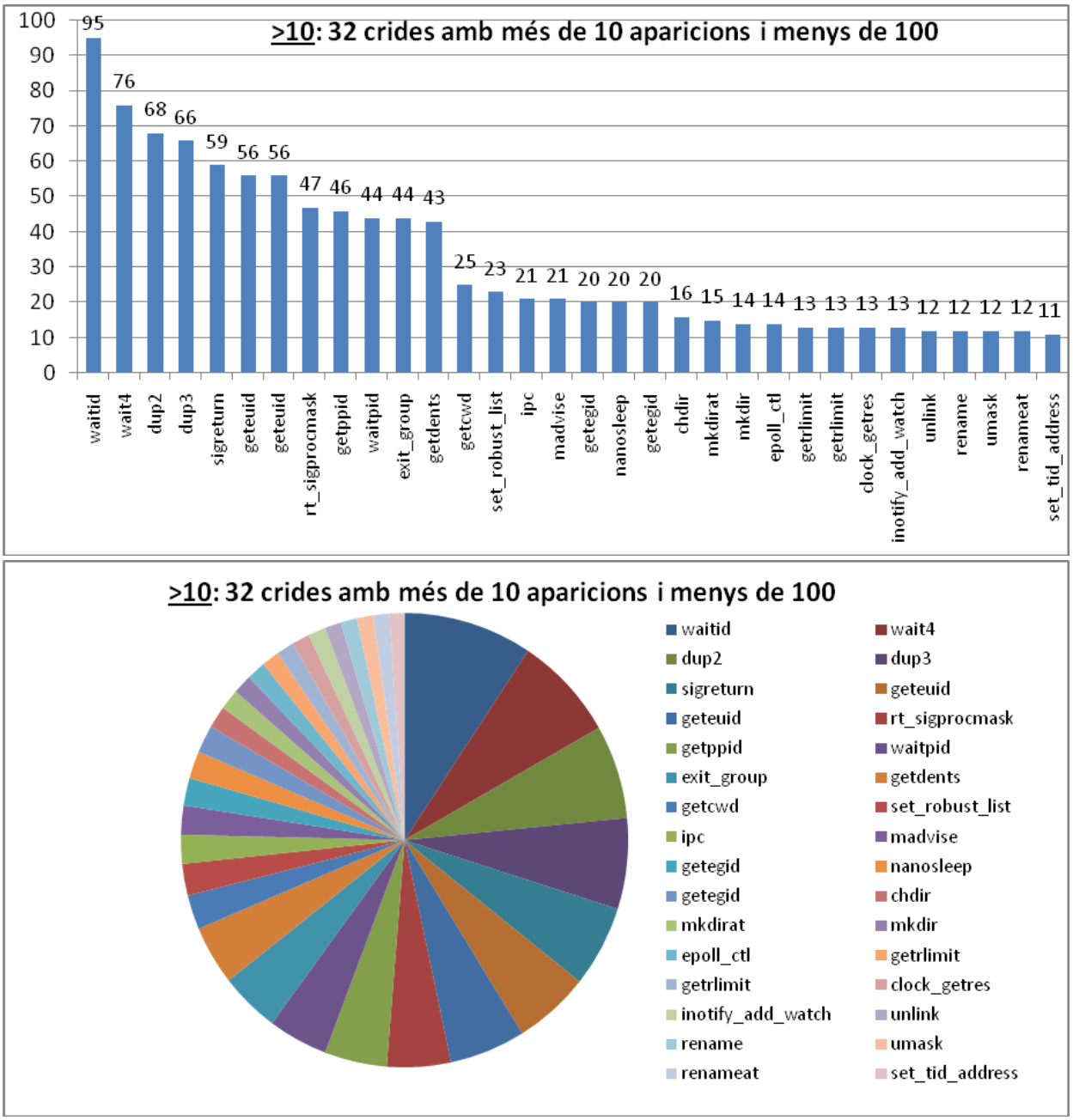
[...]

>10: 32 crides amb més de 10 aparicions i menys de 100:

exit_group, getdents, getcwd, set_robust_list, ipc, madvise, getegid, nanosleep, getegid, chdir, mkdirat, mkdir, epoll_ctl, getrlimit, getrlimit, clock_getres, inotify_add_watch, unlink, rename, umask, renameat, set_tid_address.

num	SysCall	Description	Source	1.000.000
284	waitid		kernel/exit.c:1655	95
114	wait4	wait for process termination, BSD style	kernel/exit.c	76
63	dup2	duplicate a file descriptor	fs/fcntl.c	68
330	dup3		fs/fcntl.c:53	66
119	sigreturn	return from signal handler and cleanup stack frame	arch/i386/kernel/signal.c	59
49	geteuid	get effective user ID	kernel/sched.c	56
201	geteuid		kernel/timer.c:1365	56
175	rt_sigprocmask		kernel/signal.c	47
64	getppid	get parent process ID	kernel/sched.c	46
7	waitpid	wait for process termination	kernel/exit.c	44
252	exit_group		kernel/exit.c:1087	44
141	getdents	read directory entries	fs/readdir.c	43
183	getcwd	Get current working directory	fs/dcache.c	25
311	set_robust_list		kernel/futex.c:2351	23
117	ipc	System V IPC system calls	arch/i386/kernel/sys_i386.c	21
219	madvise		mm/madvise.c:335	21
50	getegid	get effective group ID	kernel/sched.c	20
162	nanosleep	pause execution for a specified time (nano seconds)	kernel/sched.c	20
202	getegid		kernel/timer.c:1377	20
12	chdir	change working directory	fs/open.c	16
296	mkdirat		fs/namei.c:2093	15
39	mkdir	create a directory	fs/namei.c	14
255	epoll_ctl		fs/eventpoll.c:1228	14
76	getrlimit	get maximum system resource con sumption	kernel/sys.c	13
191	getrlimit		kernel/sys.c:1237	13
266	clock_getres		kernel/posix-timers.c:971	13
292	inotify_add_watch		fs/notify/inotify/inotify_user.c:685	13
10	unlink	delete a name and possibly the file it refers to	fs/namei.c	12
38	rename	change the name or location of a file	fs/namei.c	12
60	umask	set file creation mask	kernel/sys.c	12
302	renameat		fs/namei.c:2671	12
258	set_tid_address		kernel/fork.c:920	11

Taula 04: Recompte del grup ">10" (32 crides amb més de 10 aparicions i menys de 100).



Imatge 06: Gràfics del grup ">10" (32 crides amb més de 10 aparicions i menys de 100).

SysCalls del grup ">10":

exit_group, getdents, getcwd, set_robust_list, ipc, madvise, getegid, nanosleep, getegid, chdir, mkdirat, mkdir, epoll_ctl, getrlimit, getrlimit, clock_getres, inotify_add_watch, unlink, rename, umask, renameat, set_tid_address.

Al capítol "1.4.- CONCLUSIONS", s'analitzen i valoren aquests resultats.

waitid (95 aparicions [0,15%]): Espera a que un procés (fill de qui fa la crida) canviï d'estat (procés fill terminat, procés fill detingut per un *signal*, o procés fill reanudat per un *signal*).
http://linux.die.net/man/2/waitid

wait4 (76 ap. [0,12%]): Similar a 'waitid', però addicionalment retorna informació sobre l'ús dels recursos del procés fill. Pot ser utilitzada per especificar un o varis processos fills concrets.
http://www.kernel.org/doc/man-pages/online/pages/man2/wait4.2.html

dup2 (68 [0,11%]): Duplica un '*file descriptor*' obert en un altre '*file descriptor*', convertint-los en 'àlies', i esborrant el descriptor de fitxer antic.
http://www.mkssoftware.com/docs/man3/dup2.3.asp

dup3 (66 [0,10%]): Desenvolupa la mateixa tasca que 'dup2', però afegeix un argument addicional ('*flags*'), una màscara de bits que modifica el comportament de la crida al sistema.
http://my.safaribooksonline.com/book/programming/linux/9781593272203/file-i-o-further-details/file_i_solidus_o_at_a_specified_offset_c

sigreturn (59 [0,09%]): Retorn d'un *signal*. Neteja la pila per tal de que el procés pugui continuar pel punt on havia estat interromput pel signal.
http://www.tutorialspoint.com/unix_system_calls/sigreturn.htm

geteuid (56 [0,09%]): *Get Effective User ID*: Obté i retorna la identificació de l'usuari efectiu que ha cridat al procés, a diferència de '*getuid*', que retorna la identificació de l'usuari real.
http://www.nxmnpg.com/2/geteuid

rt_sigprocmask (47 [0,07%]): Examina i canvia *signals* bloquejats. S'utilitza per recuperar o canviar la màscara de *signal* del *thread* que ha fet la crida.
http://linux.die.net/man/2/rt_sigprocmask

getppid (46 [0,07%]): Obté la identificació d'un procés. Retorna l'ID del procés que ha fet la crida (normalment utilitzat per rutines que generen noms d'arxiu únics temporals).
http://linux.die.net/man/2/getppid

waitpid (44 [0,07%]): Espera la terminació d'un procés. A diferència de 'wait', permet esperar la terminació d'un procés determinat (indicat per l'argument *pid* [Process ID]).
http://sop.upv.es/sso/transparencias/proc/tsld011.htm

exit_group (44 [0,07%]): Surt de tots els fils ('*threads*') d'un procés. Similar a '*exit*', amb la diferència que no termina només el fil que l'ha cridat, sinó tots els fils del grup del procés.
http://linux.die.net/man/2/exit_group

getdents (43 [0,07%]): *Get Directory Entries*: Obté les entrades d'un directori. Llegeix varies estructures *dir_ent* del directori indicat per '*fd*' dins l'àrea de memòria apuntada per '*dirp*'.
http://linux.about.com/library/cmd/blcmdl2_getdents.htm

getcwd (25 [0,04%]): *Get Current Working Directory*: Obté la ruta del directori de treball actual.
http://pubs.opengroup.org/onlinepubs/000095399/functions/getcwd.html

[...]

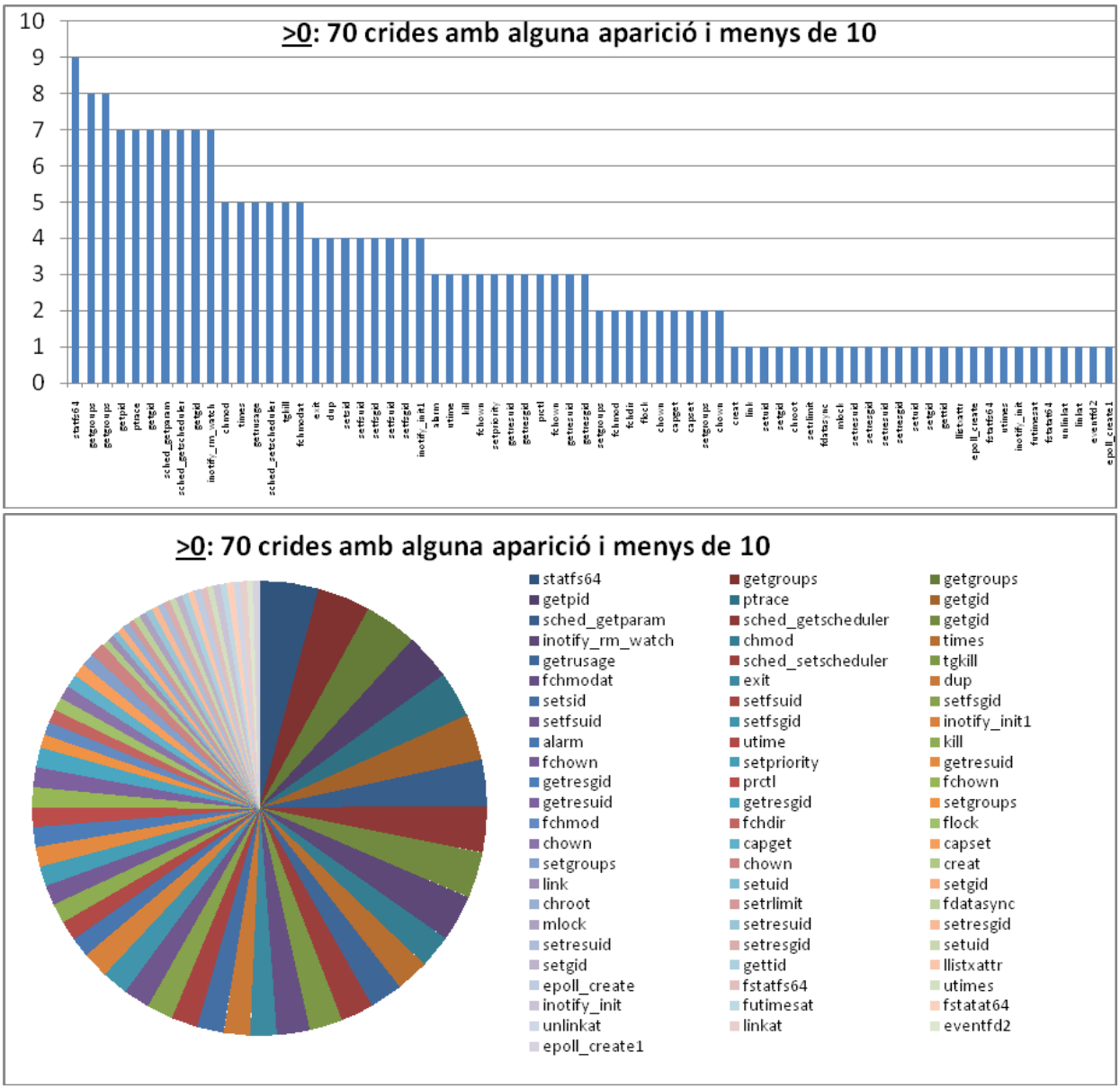
>0: 70 crides amb alguna aparició i menys de 10:

statfs64, getgroups, getgroups, getpid, ptrace, getgid, sched_getparam, sched_getscheduler, getgid, inotify_rm_watch, chmod, times, getrusage, sched_setscheduler, tgkill, fchmodat, exit, dup, setsid, setfsuid, setfsgid, setfsuid, setfsgid, inotify_init1, alarm, utime, kill, fchown, setpriority, getresuid, getresgid, prctl, fchown, getresuid, getresgid, setgroups, fchmod, fchdir, flock, chown, capget, capset, setgroups, chown, creat, link, setuid, setgid, chroot, setrlimit, fdatasync, mlock, setresuid, setresgid, setresuid, setresgid, setuid, setgid, gettid, llistxattr, epoll_create, fstatfs64, utimes, inotify_init, futimesat, fstatat64, unlinkat, linkat, eventfd2, epoll_create1.

num	SysCall	Description	Source	1.000.000
268	statfs64		fs/statfs.c:118	9
80	getgroups	get list of supplementary group IDs	kernel/sys.c	8
205	getgroups		kernel/groups.c:203	8
20	getpid	get process identification	kernel/sched.c	7
26	ptrace	allows a parent process to control the execution of a child…	arch/i386/kernel/ptrace.c	7
47	getgid	get real group ID	kernel/sched.c	7
155	sched_getparam	get scheduling parameters	kernel/sched.c	7
157	sched_getscheduler	get scheduling algorithm parameters	kernel/sched.c	7
200	getgid		kernel/timer.c:1371	7
293	inotify_rm_watch		fs/notify/inotify/inotify_user.c:726	7
15	chmod	change permissions of a file	fs/open.c	5
43	times	get process times	kernel/sys.c	5
77	getrusage	get maximum system resource con sumption	kernel/sys.c	5
156	sched_setscheduler	set scheduling algorithm parameters	kernel/sched.c	5
270	tgkill		kernel/signal.c:2383	5
306	fchmodat		fs/open.c:474	5
1	exit	terminate the current process	kernel/exit.c	4
41	dup	duplicate an open file descriptor	fs/fcntl.c	4
66	setsid	creates a session and sets the process group ID	kernel/sys.c	4
138	setfsuid	set user identity used for file system checks	kernel/sys.c	4
139	setfsgid	set group identity used for file system checks	kernel/sys.c	4
215	setfsuid		kernel/sys.c:819	4
216	setfsgid		kernel/sys.c:852	4
332	inotify_init1		fs/notify/inotify/inotify_user.c:640	4
27	alarm	set an alarm clock for delivery of a signal	kernel/sched.c	3
30	utime	set file access and modification times	fs/open.c	3
37	kill	send signal to a process	kernel/signal.c	3
95	fchown	change owner and group of a file	fs/open.c	3
97	setpriority	set program scheduling priority	kernel/sys.c	3
165	getresuid	get real, effective and saved user or group ID	kernel/sys.c	3
171	getresgid	get real, effective and saved user or group ID	kernel/sys.c	3
172	prctl	operations on a process	kernel/sys.c	3
207	fchown		fs/open.c:602	3
209	getresuid		kernel/sys.c:746	3

211	getresgid		kernel/sys.c:800	3
81	setgroups	set list of supplementary group IDs	kernel/sys.c	2
94	fchmod	change access permission mode of file	fs/open.c	2
133	fchdir	change working directory	fs/open.c	2
143	flock	apply or remove an advisory lock on an open file	fs/locks.c	2
182	chown	change ownership of a file	fs/open.c	2
184	capget	get process capabilities	kernel/capability.c	2
185	capset	set process capabilities	kernel/capability.c	2
206	setgroups		kernel/groups.c:232	2
212	chown		fs/open.c:539	2
8	creat	create a file or device ("man 2 open" for information)	fs/open.c	1
9	link	make a new name for a file	fs/namei.c	1
23	setuid	set real user ID	kernel/sys.c	1
46	setgid	set real group ID	kernel/sys.c	1
61	chroot	change root directory	fs/open.c	1
75	setrlimit	set maximum system resource con sumption	kernel/sys.c	1
148	fdatasync	synchronize a file's in-core data with that on disk	fs/buffer.c	1
150	mlock	lock pages in memory	mm/mlock.c	1
164	setresuid	set real, effective and saved user or group ID	kernel/sys.c	1
170	setresgid	set real, effective and saved user or group ID	kernel/sys.c	1
208	setresuid		kernel/sys.c:696	1
210	setresgid		kernel/sys.c:761	1
213	setuid		kernel/sys.c:655	1
214	setgid		kernel/sys.c:531	1
224	gettid		kernel/timer.c:1493	1
233	llistxattr		fs/xattr.c:463	1
254	epoll_create		fs/eventpoll.c:1215	1
269	fstatfs64		fs/statfs.c:154	1
271	utimes		fs/utimes.c:219	1
291	inotify_init		fs/notify/inotify/inotify_user.c:680	1
299	futimesat		fs/utimes.c:191	1
300	fstatat64		fs/stat.c:391	1
301	unlinkat		fs/namei.c:2341	1
303	linkat		fs/namei.c:2470	1
328	eventfd2		fs/eventfd.c:409	1
329	epoll_create1		fs/eventpoll.c:1187	1

Taula 05: Recompte del grup ">0" (70 crides amb alguna aparició i menys de 10).



Imatge 07: Gràfics del grup ">0" (72 crides amb alguna aparició i menys de 10).

SysCalls del grup ">10":

statfs64, getgroups, getgroups, getpid, ptrace, getgid, sched_getparam, sched_getscheduler, getgid, inotify_rm_watch, chmod, times, getrusage, sched_setscheduler, tgkill, fchmodat, exit, dup, setsid, setfsuid, setfsgid, setfsuid, setfsgid, inotify_init1, alarm, utime, kill, fchown, setpriority, getresuid, getresgid, prctl, fchown, getresuid, getresgid, setgroups, fchmod, fchdir, flock, chown, capget, capset, setgroups, chown, creat, link, setuid, setgid, chroot, setrlimit, fdatasync, mlock, setresuid, setresgid, setresuid, setresgid, setuid, setgid, gettid, llistxattr, epoll_create, fstatfs64, utimes, inotify_init, futimesat, fstatat64, unlinkat, linkat, eventfd2, epoll_create1.

Al capítol "1.4.- CONCLUSIONS", s'analitzen i valoren aquests resultats.

statfs64 (9 aparicions [0,014%]): Obté estadístiques del sistema d'arxius. Retorna informació d'un sistema d'arxius muntat (EXT3_SUPER_MAGIC, NTFS_SB_MAGIC, …).
http://linux.die.net/man/2/statfs64

getgroups (8 ap. [0,013%]): Obté la llista d'IDs del grup suplementari. Retorna els identificadors del grup suplementari del procés que ha fet la crida.
http://kernel.org/doc/man-pages/online/pages/man2/setgroups.2.html

getpid (7 [0,011%]): Obté la identificació del procés pare. Retorna l'ID del procés que ha fet la crida (normalment utilitzat per rutines que generen noms d'arxiu temporals únics).
http://linux.die.net/man/2/getpid

ptrace (7 [0,011%]): Rastreja un procés. Proporciona un mitjà pel que un procés pare pot observar i controlar l'execució d'un procés fill, examinar i canviar la seva imatge de memòria i registres.
http://es.tldp.org/Paginas-manual/man-pages-es-1.28/man2/ptrace.2.html

getgid (7 [0,011%]): Obté l'ID del grup real. Retorna l'identificador del grup real del procés que ha fet la crida.
http://pubs.opengroup.org/onlinepubs/009695399/functions/getgid.html

sched_getparam (7 [0,011%]): *Get Scheduling Parameters:* Obté els paràmetres de programació (planificació). Retorna els paràmetres d'*scheduling* d'un procés.
http://linux.die.net/man/2/sched_getparam

sched_getscheduler (7 [0,011%]): Obté els paràmetres i política de programació (planificació). Retorna dues coses: la política d'*scheduling* i els paràmetres associats al procés.
http://linux.die.net/man/2/sched_getscheduler

getid (7 [0,011%]): Obté la identificació d'un procés. Retorna l'ID del procés que ha fet la crida (normalment utilitzat per rutines que generen noms d'arxiu temporals únics).
http://linux.die.net/man/2/getid

inotify_rm_watch (7 [0,011%]): Elimina una vista existent d'una instància '*inotify*'. Esborra la vista associada al descriptor de vista *wd* de la instància inotify associada al descriptor *fd*.
http://linux.die.net/man/2/inotify_rm_watch

chmod (5 [0,008%]): Canvia els bits de mode d'ur arxiu. Modifica el mode d'un arxiu donat, que pot ser una representació simbòlica dels canvis a fer, o un nombre octal amb els nous bits.
http://linux.die.net/man/1/chmod

times (5 [0,008%]): *Obté els temps d'un procés*. Emmagatzema els temps del procés actual en la estructura '*struct tms*' (definida a 'sys/times.h') al que apunta *buf*.
http://linux.die.net/man/2/times

getrusage (5 [0,008%]): Obté l'ús de recursos. Retorna la mesura d'ús de recursos de: RUSAGE_SELF (el propi procés), RUSAGE_CHILDREN (procés fill), RUSAGE_THREAD.
http://linux.die.net/man/2/getrusage

[...]

Altres resultats:

Els resultats que jo he extret, són diferents d'altres que es poden trobar a Internet. En concret, a continuació presento alguns altres resultats d'estadístiques de crides al sistema:

lwn.net/Articles/371448/

```
root@tropicana:~# perf trace record syscall-counts
^C[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 32.165 MB perf.data (~1405309 samples) ]

root@tropicana:~# perf trace report syscall-counts

syscall events:

event                                    count
---------------------------------------  -----------
sys_write                                 269407
sys_getdents                                2782
sys_close                                   2004
sys_swapoff                                 1095
sys_read                                     920
sys_sched_setparam                           511
sys_open                                     331
sys_newfstat                                 326
sys_mmap                                     217
sys_munmap                                   216
sys_select                                   173
sys_getgid                                   128
sys_futex                                     89
sys_poll                                      76
174                                           64
sys_setuid                                    64
sys_setitimer                                 50
15                                            25
sys_rt_sigprocmask                            24
sys_lseek                                      7
sys_inotify_add_watch                          6
sys_writev                                     5
sys_ioctl                                      5
sys_wait4                                      2
sys_perf_event_open                            1
sys_fcntl                                      1
```

Imatge 08: Estadística externa 1 de crides al sistema, trobada a: "http://lwn.net/Articles/371448/"

docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/SystemTap_Beginners_Guide/topsyssect.html

Example 4.15. topsys.stp Sample Output

```
------------------------------------------------------------------
                SYSCALL      COUNT
            gettimeofday       1857
                    read       1821
                   ioctl       1568
                    poll       1033
                   close        638
                    open        503
                  select        455
                   write        391
                  writev        335
                   futex        303
                 recvmsg        251
                  socket        137
           clock_gettime        124
          rt_sigprocmask        121
                  sendto        120
              setitimer        106
                    stat         90
                    time         81
              sigreturn         72
                   fstat         66
------------------------------------------------------------------
```

Imatge 09: Estadística externa 2 de crides al sistema, trobada a:"http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/SystemTap_Beginners_Guide/topsyssect.html"

1.3.- VALORACIÓ ECONÒMICA

La valoració econòmica d'un projecte acostuma a basar-se en els costos i la previsió d'ingressos resultants de l'execució del mateix.

Pel que fa als costos, s'ha de tenir en compte:

Preu dels serveis = hores de dedicació * tarifa programador sènior =
150h * 35€/h = 5.250€

Preu del maquinari = inversió en nou maquinari + tarifa de hosting =
0€ (no es precissa nou maquinari) + 0€ (no cal hosting) = 0€

Preu del programari = preu de les llicències * quantitat de llicències necessàries =
0€ (el programari lliure gratuït no té preu de llicència) * 1 = 0€

TOTAL COSTOS: 5.250€

Pel que fa als ingressos, s'ha de dir que l'objectiu final d'aquest projecte és únicament la generació de coneixement (que, a més, es pugui compartir), i no pas un producte o servei comercialitzable.

D'aquesta manera, en aquest projecte concret es fa difícil aplicar una valoració econòmica, ja que no es pot comparar amb cap altre producte o servei similar al mercat. En tot cas, es pot valorar l'esforç per a desenvolupar-lo, que ha estat de 150h.

1.3.1.- Paradigma de programari lliure

La idea del programari lliure és que qualsevol persona pugui fer servir, copiar, modificar i redistribuir un programa de forma lliure. En aquest sentit, la plataforma GNU/Linux s'engloba dins d'aquest paradigma i permet, per exemple, que els seus usuaris modifiquin el codi font del nucli del sistema.

La distribució de Linux que he fet servir és Ubuntu 11.04, i és totalment lliure i gratuïta. Si el resultat d'aquest projecte pot ser útil o interessant per alguna persona, vull que ho sigui també de forma lliure i gratuïta. D'aquesta manera, acompleixo el compromís de cooperació amb el que s'ha creat aquesta comunitat.

1.4.- CONCLUSIONS

En aquest capítol, presento les conclusions dels projecte. Per una banda, exposo les reflexions de tot el procés; i per una altra, analitzo i valoro els resultats obtinguts. Cadascun d'aquests apartats els mostro separats per fases (les mateixes que han estructurat el projecte), de manera que es veu l'ordre cronològic de les reflexions, anàlisis i valoracions que aporto. S'ha de tenir en compte que, així com els resultats i l'anàlisi són totalment objectius, les reflexions i valoracions tenen un caire més personal, a partir de la meva experiència durant el desenvolupament del projecte.

1.4.1.- Resultats i reflexions

Fase 1: "Disseny de la solució":

És digne de remarcar que el consultor ha participat molt activament i de manera decisiva a establir un guió per a dissenyar la solució d'aquest projecte. A excepció de la dificultat trobada en el pas "Editar fitxer .c que fa la funció de dispatching" (que necessàriament he hagut de modificar degut a la diferència de versions), la totalitat del procediment està ben pensat i dissenyat. D'aquesta manera, he pogut construir una estructura i planificació adients per al projecte, que l'han dut a bon terme.

Fase 2: "Preparació de l'entorn i compilació del kernel":

Instal·lació del sistema operatiu Linux:

El fet d'haver triat "Ubuntu 11.04" per a aquest projecte, és irrellevant. Però és cert que es tracta d'una distribució i versió amb molt suport a Internet, cosa que ha simplificat la recerca d'informació, manuals, exemples, etc. D'igual manera, la instal·lació va ser agraïdament senzilla i ràpida. De tota manera, el fet de comptar amb un nucli de Linux versió 2.6, ha fet que l'enregistrament de crides al sistema hagi estat més complex del que es preveia a l'inici.

Instal·lació del codi font i eines de compilació:

Durant aquest procés, he pogut aprendre i entendre el funcionament d'algunes ordres com "apt-get install", "sudo" o "tar". Els petits entrebancs que hagi pogut tenir amb l'execució d'aquestes ordres, els he solucionat ràpidament (ja sigui gràcies a la pròpia documentació que aporten i els missatges que mostren a l'usuari, o mitjançant la informació abundant existent a Internet). En el desenvolupament del projecte, he documentat aquests entrebancs, i les solucions a cadascun d'ells, de manera que es pot apreciar els problemes que es pot trobar un usuari novell de Linux, i com superar-los.

Compilació i instal·lació del nou nucli:

Configurar, compilar i instal·lar el nou nucli ha estat una tasca senzilla però molt llarga. En el meu cas, vaig començar aquest procés amb una màquina relativament antiga; cosa que va fer que el temps de compilació fos encara molt més llarg. Per la resta, no vaig trobar-ne massa complicacions (llevat d'algunes modificacions que s'han de tenir en compte, i que novament he pogut resoldre fàcilment amb la extensa documentació existent a Internet).

Fase 3: "Modificació del kernel":

Idea inicial:

La primera proposta que s'havia plantejat per tal d'aconseguir que s'enregistressin les crides al sistema, no va ser vàlida. El motiu és que, a partir de la versió 2.6 del nucli de Linux, la taula de crides al sistema (syscall table) ja no és exportada, com a la versió 2.4. Això implica que no és fàcilment accessible, i només es pot aconseguir en temps de compilació.

Solució amb nova *syscall*:

La primera solució que vaig intentar per superar aquest obstacle, va ser la creació d'una nova crida al sistema, que s'executaria just abans de la crida original a qualsevol altra syscall, i faria l'enregistrament pertinent. Però precisament, per motius de seguretat que evitin la intercepció de syscalls, els desenvolupadors del kernel de Linux han fet més difícil afegir noves crides al sistema.

Altres intents de solució:

Així que, després d'intentar-ho de diferents maneres, amb mètodes i solucions diverses, tots els intents van fracassar ja fos perquè el compilador no acceptava el codi que havia insertat, perquè el sistema es bloquejava, o bé perquè les modificacions no generaven cap resultat d'enregistrament de crides al sistema.

Solució definitiva i funcional:

La solució que finalment ha funcionat, la he basat en la modificació dels arxius ".c" que contenen funcions de crides al sistema. Mitjançant l'ordre "printk" (que s'executa a nivell de nucli i no genera una nova crida al sistema –cosa que faria al sistema entrar en un bucle infinit-), he pogut crear un '*log*' de les syscalls. Aquest era l'objectiu d'aquesta fase.

Fase 4: "Informació i resultats":

Obtenció de resultats:

Fent ús de l'ordre '*printk(KERN_DEBUG "|NomDeLaSyscall")*', he pogut enregistrar a l'arxiu '*/var/log/syslog*' les crides al sistema que s'han executat en un inici normal del meu sistema Linux, distribució Ubuntu 11.04. De forma representativa, per a poder fer unes estadístiques amb una quantitat elevada però manejable de resultats, n'he sel·leccionat un milió. Cada línia del fitxer representa una syscall, i té la forma: *Date time host kernel: [seq] |NomDeLaSyscall*

Estructurar la informació:

Per poder estudiar correctament aquesta informació, l'he transferit a un full de càlcul. Agrupant les dades per nom de syscall, he obtingut un total de 137 crides al sistema (descartant les que no tenen cap aparició entre els resultats). El següent pas ha estat ordenar-les per número d'apariccions, amb el que he situat en les primeres posicions les syscalls executades amb més freqüència.

Producte resultant:

Tal com s'indicava al capítol "1.1.5.- Productes obtinguts", el producte resultant està format per tres arxius:

1.- Fitxer de full de càlcul, que inclou les estadístiques obtingudes:
[SysCallStatistics.ods]

2.- Fitxer patch, que conté les diferències entre el codi font original i el codi modificat per a l'obtenció de les estadístiques de crides al sistema:
[SysCallStatistics.patch]

3.- Memòria del projecte, que inclou la documentació de tot el procés, així com l'anàlisi i les conclusions finals extretes de la informació obtinguda:
[gguirado_memoria.PDF]

Per a la creació del fitxer tipus patch, he seguit els següents passos:

- Copiar el codi font original (copia de seguretat prèvia a les modificacions) en un directori amb el nom "linux-source-2.6.38.orig":
  `cp linux-source-2.6.38.backup linux-source-2.6.38.orig`

- Copiar el codi font modificat (amb els canvis per enregistrar les syscalls) en un directori amb el nom "linux-source-2.6.38.modif":
  `cp linux-source-2.6.38 linux-source-2.6.38.modif`

- Comparar el contingut dels dos directoris, mitjançant la comanda "*diff*", i enviant el resultat a l'arxiu "SysCallStatistics.patch":
  `diff –Naur linux-source-2.6.38.orig linux-source-2.6.38.modif > SysCallStatistics.patch`

Això genera un arxiu de tipus patch, amb el següent contingut:

```
diff  -Naur  linux-source-2.6.38.orig/arch/mips/kernel/syscall.c  linux-source-
2.6.38.modif/arch/mips/kernel/syscall.c
--- linux-source-2.6.38.orig/arch/mips/kernel/syscall.c  2011-03-15
02:20:32.000000000 +0100
+++ linux-source-2.6.38.modif/arch/mips/kernel/syscall.c 2011-12-16
12:41:45.000000000 +0100
@@ -272,6 +272,9 @@
 {
        struct thread_info *ti = task_thread_info(current);

+       //GGuirado SysCall Statistics
+       printk(KERN_DEBUG "|set_thread_area");
+
        ti->tp_value = addr;
        if (cpu_has_userlocal)
                write_c0_userlocal(addr);
diff   -Naur   linux-source-2.6.38.orig/arch/x86/kernel/signal.c   linux-source-
2.6.38.modif/arch/x86/kernel/signal.c
[...]
```

D'aquesta manera, qualsevol pot aplicar aquestes modificacions per obtenir resultats similars. Adjunto aquest arxiu [SysCallStatistics.patch] amb aquest projecte.

1.4.2.- Anàlisi i valoracions

El problema de les dades extremes:

El primer que salta a la vista de l'estructuració de la informació (en ordenar les dades per número d'ocurrències), és que hi ha tres crides al sistema ('*time*', '*gettimeofday*' i '*write*') que apareixen amb una freqüència desproporcionada respecte a la resta (més de 100.000 aparicions):

num	SysCall	Description	Source	1.000.000
13	time	get time in seconds	kernel/time.c	414.820
78	gettimeofday	get the date and time	kernel/time.c	354.551
4	write	write to a file descriptor	fs/read_write.c	167.434
195	stat64		fs/stat.c:358	8.936
3	read	read from a file descriptor	fs/read_write.c	8.867
265	clock_gettime		kernel/posix-timers.c:954	8.039
158	sched_yield	yield the processor	kernel/sched.c	5.195
6	close	close a file descriptor	fs/open.c	3.541
168	poll	wait for some event on a file descriptor	fs/select.c	3.268

[...]

Taula 06: Desproporció de les tres crides amb més aparicions, respecte a la resta.

Pensant detingudament en el significat de la informació (i no limitant-me a exposar les dades sense més aportacions que la de recopilació), he fe una anàlisi sobre les dades i el seu origen, arribant a la conclusió de que la informació obtinguda està desvirtuada. Aquesta és la explicació:

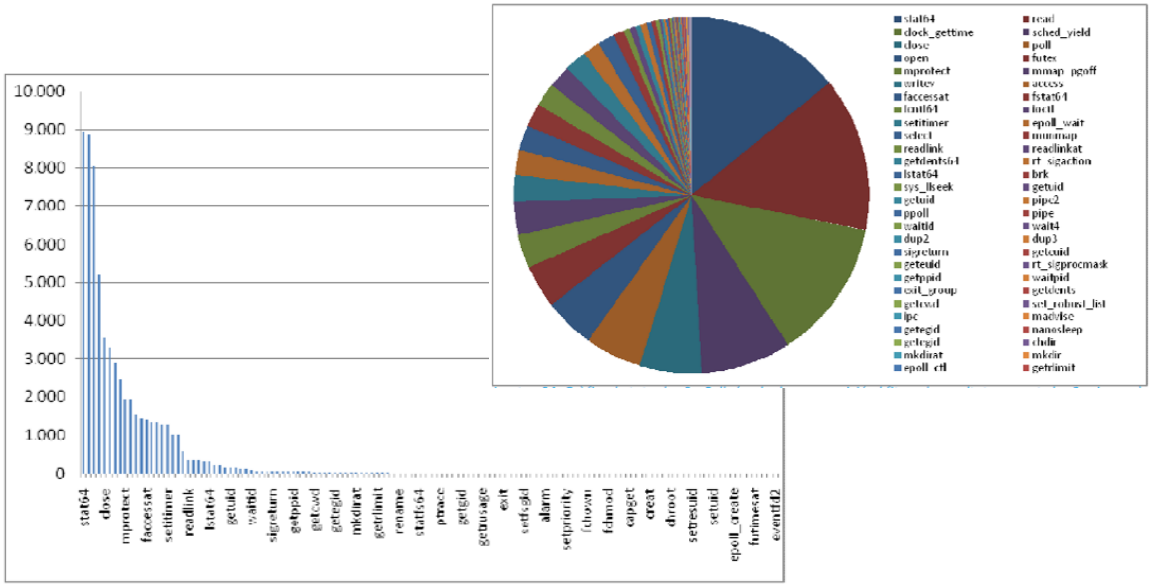
El fet d'escriure ('*write*') la data ('*time*') i hora ('*gettimeofday*') al fitxer de log, ja fa que s'executin aquestes syscalls per l'acció d'enregistrar altres crides al sistema. El número d'aparicions d'aquestes tres crides no és igual, cosa que indica que algunes aparicions són pròpies crides a aquestes syscalls. A més, la funció '*write*' apareix menys que '*time*' i '*gettimeofday*', cosa que fa veure que la informació s'ha escrit en blocs al fitxer, i no s'ha fet una crida a '*write*' per cada línia que s'enregistra.

En qualsevol cas, del total de 137 syscalls amb resultats, queda clar que les tres primeres són estadísticament descartables per tractar-se d'extrems que representen informació no adient. Així, vaig decidir separar '*time*', '*gettimeofday*' i '*write*' de l'estadística, per estudiar la informació correctament. De la mateixa manera, no faig menció a les crides al sistema que no tenen cap aparició als resultats.

Els valors extrems retirats de l'estadística (414.820, 354.551 i 167.434), sumaven un total de 936.805 resultats. Això representava un 93,68% de les dades. D'aquesta manera, la estadística final la he realitzat amb la resta de resultats: 63.195 dades en 134 crides al sistema. Tot i que aquesta quantitat és molt menor que la inicial, resulta suficient per a analitzar-la. Però el més important és que ara representa una informació correcta i sense esbiaix.

El problema de la varietat de les dades:

El següent problema que es presentava en intentar representar la informació en gràfics, és que les 134 syscalls restants són massa per mostrar-les i analitzar-les de forma clara i que aporti algun significat (especialment perquè els seus valors són molt diferents):



Imatge 10: Gràfiques amb massa valors diferents. No permeten una anàlisi de la informació.

Per aquest motiu, he fet servir el mètode estadístic de l'agrupació. He trobat un criteri que permet agrupar les diferents crides al sistema en tres grups, en funció del número d'aparicions als resultats ("més de 100", "més de 10", "més de 0"). El primer grup el formen 32 syscalls; el segon, també 32; i l'últim, 70.

D'aquesta manera, les agrupacions que he creat representen una separació entre les crides al sistema més habituals, les que tenen una aparició mitjana, i les puntuals o amb baixa aparició.

Ara, és possible fer una anàlisi d'aquesta informació, que exposo a continuació de forma detallada:

Conclusions:

A partir de les dades i gràfiques resultants, les estadístiques que ofereixo mostren tres grups de crides al sistema, en funció del nombre de vegades que s'executen en un inici normal del meu sistema GNU/Linux Ubuntu 11.04. Diferents versions de Linux i diferents distribucions, poden oferir resultats sensiblement diferents.

Aquestes estadístiques, lluny de mostrar cap resultat sorprenent, ofereixen una visió aproximada de les syscalls més habituals (només durant la fase d'arrencada del sistema), i ordenades per quantitat d'aparicions.

Com ja he comentat anteriorment, he hagut de descartar les crides al sistema '*time*', '*gettimeofday*' i '*write*', perquè s'executen pel sol fet d'estar enregistrant informació a l'arxiu de *log*.

Després de obviar aquestes tres syscalls, i de fer les agrupacions pertinents, comento els resultats de cada un dels grups resultants. En el capítol "1.2.4.- Fase d'informació i resultats", s'ofereixen les taules, gràfiques i descripcions de les crides al sistema de cada grup.

Grup ">100" (32 crides amb més de 100 aparicions i menys de 10.000):

Com es pot comprovar a les taules, gràfiques i descripcions de crides al sistema d'aquest grup, les *syscalls* més recurrents en l'inici del sistema són aquelles que tenen a veure amb informació, accés i gestió de fitxers (stat64, read, close, open, writev, access), gestió de processos (sched_yield, poll, futex), gestió de memòria (mprotect, mmap_pgoff) i tasques de rellotge (clock_gettime).

Grup ">10" (32 crides amb més de 10 aparicions i menys de 100):

En el segon grup, format per les *syscalls* amb una aparició menys repetitiva que el primer, però encara important, es troben especialment crides al sistema relatives a esperes entre processos (waitid, wait4, waitpid), identificadors de processos i usuaris (geteuid, getppid), signals (sigreturn, rt_sigprocmask), i descriptors de fitxers i gestió de directoris (dup2, dup3, getdents, getcwd).

Grup ">0" (70 crides amb alguna aparició i menys de 10):

Per últim, en el grup de les *syscalls* amb molt poca aparició, però que s'executen alguna vegada durant l'arrencada del sistema operatiu, es poden trobar crides al sistema que gestionen configuracions del sistema (statfs64, sched_getparam, sched_getscheduler), les que obtenen identificadors del processos involucrats (getgroups, getpid, getgid) i crides que gestionen l'ús de recursos i permisos (ptrace, chmod, times, getrusage).

Al tractar-se d'un inici del sistema, és normal que algunes syscalls habituals i conegudes com '*kill*' no tinguin una presència marcada, que sí tindrien en altre fase d'una sessió d'usuari.

2.- GLOSSARI

Crida al sistema (*syscall*): Mecanisme utilitzat per una aplicació per a sol·licitar un servei al *kernel* del sistema operatiu. Normalment, fan servir una instrucció especial de la CPU que causa que el processador transfereixi el control a un codi especialitzat, prèviament especificat pel mateix codi. Això permet al codi privilegiat especificar on serà connectat, així com l'estat del processador. Quan una crida al sistema és invocada, l'execució del programa que invoca és interrompuda i les seves dades són guardades, per poder continuar executant-se després. El processador comença aleshores a executar instruccions d'alt nivell de privilegi, per a realitzar la tasca requerida. Quan això finalitza, es retorna al procés original i continua la seva execució. En sistemes operatius sota norma POSIX o similars, algunes crides al sistema molt utilitzades són: *open*, *read*, *write*, *close*, *wait*, *exec*, *fork*, *exit*, *kill*, etc. Els sistemes operatius actuals tenen centenars de crides al sistema (per exemple, Linux 2.x té més de 300).
http://es.wikipedia.org/wiki/Llamada_al_sistema

Distribució: (Col·loquialment, '*distro*'). Paquet de programari o conjunt d'aplicacions reunides. En el cas d'una distribució Linux, és una distribució de programari i/o un sistema operatiu basat en el nucli Linux, que inclou determinats paquets de programari per satisfer les necessitats d'un grup específic d'usuaris, donant així origen a edicions domèstiques, empresarials i per servidors. Generalment, estan compostes total o majoritàriament de programari lliure. A més del nucli Linux, les distribucions inclouen habitualment les biblioteques i eines del projecte GNU (en aquest cas, es coneix com a distribució GNU/Linux). Per a la realització d'aquest projecte, he fet servir la distribució de GNU/Linux Ubuntu 11.04.
http://es.wikipedia.org/wiki/Distribución_Linux

GNU: (Acrònim recursiu de: 'Gnu is Not Unix'). Projecte iniciat per Richard Stallman amb l'objectiu de crear un sistema operatiu completament lliure: el sistema GNU. El 27 de setembre de 1983 es va anunciar públicament el projecte per primera vegada al grup de notícies *net.unix-wizards*. A l'anunci original, van seguir altres assajos escrits per Richard Stallman com el 'Manifest GNU', que establiren les seves motivacions per a realitzar el projecte GNU, entre les que destaca "tornar a l'esperit de col·laboració que va prevaler en els temps inicials de la comunitat d'usuaris de computadores."
http://es.wikipedia.org/wiki/GNU

GNU/Linux: Terme utilitzat per referir-se a la combinació del nucli Linux amb eines de sistema GNU. Malgrat que Linux és, en sentit estricte, el sistema operatiu, es fa servir usualment amb eines del projecte GNU o d'altres projectes com GNOME. La contribució de GNU és la raó de la controvèrsia a l'hora d'utilitzar 'Linux' o 'GNU/Linux' per referir-se al sistema operatiu format per les eines de GNU i el nucli Linux en conjunt.
http://es.wikipedia.org/wiki/GNU/Linux

Kernel: Veure 'nucli'.

Linux: Nucli del sistema operatiu lliure de tipus Unix ('LINUX' podria ser un acrònim recursiu de "Linux Is Not UniX"). És un dels principals exemples de programari lliure (llicenciat sota la GPL v2 [*General Public License]),* i està desenvolupat per col·laboradors de tot el món. Va ser concebut pel llavors estudiant finlandès de Ciències de la Computació **Linus Torvalds**, al 1991. Va aconseguir ràpidament desenvolupadors i usuaris que adoptaren codi d'altres projectes de programari lliure pel seu ús en el nou sistema operatiu. El nucli Linux ha rebut contribucions de milers de programadors. Normalment, s'utilitza amb una distribució. La versió actual del nucli de Linux és la 2.6.
http://es.wikipedia.org/wiki/Núcleo_Linux

Maquinari (*hardware*): Conjunt de les parts físiques d'un ordinador. Es classifica principalment per situació (perifèric [ratolí, teclat, impressora] o central [memòria, targetes]) i funció (entrada [càmera web, escàner], sortida [monitor, altaveus], entrada-sortida o emmagatzematge [disc dur, memòria USB, disc òptic]. Són les parts tangibles d'un sistema informàtic. Els seus components són elèctrics, electrònics (placa mare [*Motherboard*], UCP/processador [CPU]), mecànics i electromecànics (disc dur, gravadora, font d'alimentació, etc.).
http://es.wikipedia.org/wiki/Hardware / *http://ca.wikipedia.org/wiki/Maquinari*

Nucli (*kernel*): Programari que constitueix la part més important del sistema operatiu. És el principal responsable de facilitar als diferents programes un accés segur al maquinari de la computadora. És l'encarregat de gestionar recursos, mitjançant serveis de crides al sistema. Degut al gran nombre de programes i la limitació en l'accés al maquinari, també s'encarrega de decidir quin programa pot fer ús d'un dispositiu de maquinari i durant quant de temps. En informàtica, el nucli és el programa que s'assegura de:
- La comunicació entre els programes que sol·liciten recursos, i el maquinari.
- La gestió dels diferents programes informàtics (tasques/processos) de la màquina.
- La gestió del maquinari (memòria, processador, perifèric, emmagatzematge, etc.).
http://es.wikipedia.org/wiki/Núcleo_(informática)

Programari (*software*): Equipament o suport **lògic** d'un sistema informàtic, que comprèn el conjunt de components lògics necessaris que fan possible la realització de tasques específiques (en contraposició als components **físics**, anomenats maquinari). Els components lògics inclouen, entre d'altres, les aplicacions informàtiques (processadors de textos, calculadora, navegador web, etc.) i el programari del sistema (sistema operatiu i eines del sistema).
http://es.wikipedia.org/wiki/Software

Programari lliure (*free/open software*): (Actualment, es tendeix a utilitzar el terme 'open software', per no confondre la paraula anglesa 'free', que pot significar 'gratis'). Programari que respecta la llibertat dels usuaris sobre el seu producte adquirit i, per tant, un cop adquirit, pot ser executat, copiat, distribuït, estudiat, modificat i redistribuït modificat lliurement. Acostuma a estar disponible gratuïtament, o al preu de cost de distribució, tot i que no és obligatori que sigui així. Per tant, no s'ha d'associar "programari lliure" a "programari gratuït" (denominat normalment "*freeware*"). Un dels màxims exponents d'aquest tipus de programari, és GNU/Linux, tot i que també són coneguts altres com el paquet 'OpenOffice'.
http://es.wikipedia.org/wiki/Software_libre

Sistema operatiu (SO): Programa o conjunt de programes que efectuen la gestió dels processos bàsics d'un sistema informàtic, i permet la normal execució de la resta d'operacions. Tot i que aquest terme es refereix estrictament al nucli, incorrectament se l'associa amb el conjunt d'eines com exploradors d'arxius, navegador i altres programes que permeten la interacció amb el sistema operatiu. L'exemple més clar d'aquesta diferència és el nucli de Linux, que és el kernel del sistema operatiu GNU/Linux (de GNU, existeixen nombroses distribucions).
http://es.wikipedia.org/wiki/Sistema_operativo

Syscall: Veure 'crida al sistema'.

3.- BIBLIOGRAFIA

Per al desenvolupament d'aquest projecte s'han fet servir només fonts d'Internet. En alguns casos, no es disposa de l'autor, la data o alguna altra dada. En qualsevol cas, sempre se'n facilita l'adreça al recurs utilitzat, i les dades disponibles sobre la font. A data 28/12/2011, tots els enllaços estan en funcionament.

- M. Tim Jones, Consultant Engineer, Emulex Corp. (06/06/2007). *Anatomy of the Linux kernel. History and architectural decomposition.* IBM developerWorks. [en línia]: http://www.ibm.com/developerworks/linux/library/l-linux-kernel/

- Andries Brouwer, *aeb@cwi.nl* (01/02/2003). *The Linux kernel*. Capítol 4: *System Calls*. [en línia]: http://www.win.tue.nl/~aeb/linux/lk/lk-4.html

- Jorge Manjarez (01/12/1999). *Implementing Linux System Calls*. [en línia]: http://www.linuxjournal.com/article/3326

- Die.Net (??/??/??). *syscalls(2) – Linux man page*. [en línia]: http://linux.die.net/man/2/syscalls

- Stephen Pape (20/01/2010). *Linux Kernel: System call hooking example*. Stackoverflow. [en línia]: http://stackoverflow.com/questions/2103315/linux-kernel-system-call-hooking-example

- Telemachus (07/05/2009). *Newb question regarding MAKE*. Forums.debian.net. [en línia]: http://forums.debian.net/viewtopic.php?f=10&t=38885

- Thangamaniarum (08/07/2010): *How to quickly build custom kernel on Ubuntu 10.04?*. [en línia]: http://thangamaniarun.wordpress.com/2010/07/08/how-to-quickly-build-custom-kernel-on-ubuntu-10-04/

- Panaud0lordv (10/01/2010). *Won't compile if build is remote called (wrong path setting in ubuntu/omnibook/Makefile)*. [en línia]: https://bugs.launchpad.net/ubuntu/+source/linux/+bug/505420

- Caribdis (??/??/??). Cómo compilar el kernel de Ubuntu. [en línia]: http://www.ubuntu-es.org/node/431

- Al Dev [Alavoor Vasudevan] (04/05/2003). The Linux kernel HOWTO. [en línia]: http://www.faqs.org/docs/Linux-HOWTO/Kernel-HOWTO.html

- Amit Choudhary (27/10/2006). *Implementing a System Call on Linux 2.6 for i386*. [en línia]: http://tldp.org/HOWTO/html_single/Implement-Sys-Call-Linux-2.6-i386/

- Greg Ose (??/??/??). Linux Syscall Reference. [en línia]: http://syscalls.kernelgrok.com/

- AUTOR/A DESCONEGUT/DA [WIKIPEDIA.ORG] (Nov. 2010). System call. [en línia]: *http://en.wikipedia.org/wiki/System_call*

- TOM ZANUSSI (27/01/2010). perf trace: Phyton scripting support. [en línia]: *http://lwn.net/Articles/371448/*

- DON DOMINGO / WILLIAM COHEN (2010). *SystemTap Beginners Guide*: *4.3.5. Tracking most Frequently Used System Calls*. (Edition 2.0). [en línia]: *http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/SystemTap_Beginners_Guide/topsyssect.html*

- WIKIPEDIA.ORG: GLOSSARI del projecte. *http://es.wikipedia.org*

4.- ANNEXOS

Documentació de la fase de preparació de l'entorn i compilació del kernel

- Tota la documentació d'aquesta fase es pot trobar entre les pàgines 12 i 22, al capítol: "1.2.2.- Fase de preparació de l'entorn i compilació del kernel".

Documentació de la fase de modificació del kernel

- Tota la documentació d'aquesta fase es pot trobar entre les pàgines 23 i 31, al capítol: "1.2.3.- Fase de modificació del kernel".

- De la mateixa manera, adjunto un arxiu de tipus patch, amb les modificacions realitzades, perquè estiguin disponibles de manera lliure per qui vulgui aplicar-les.

Documentació de la fase d'informació i resultats

- Tota la documentació d'aquesta fase es pot trobar entre les pàgines 32 i 45, al capítol: "1.2.4.- Fase d'informació i resultats".

- D'igual forma, adjunto el full de càlcul amb el recompte de crides al sistema, ordenades per número d'ocurrències.