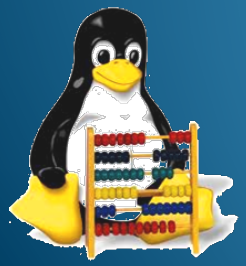
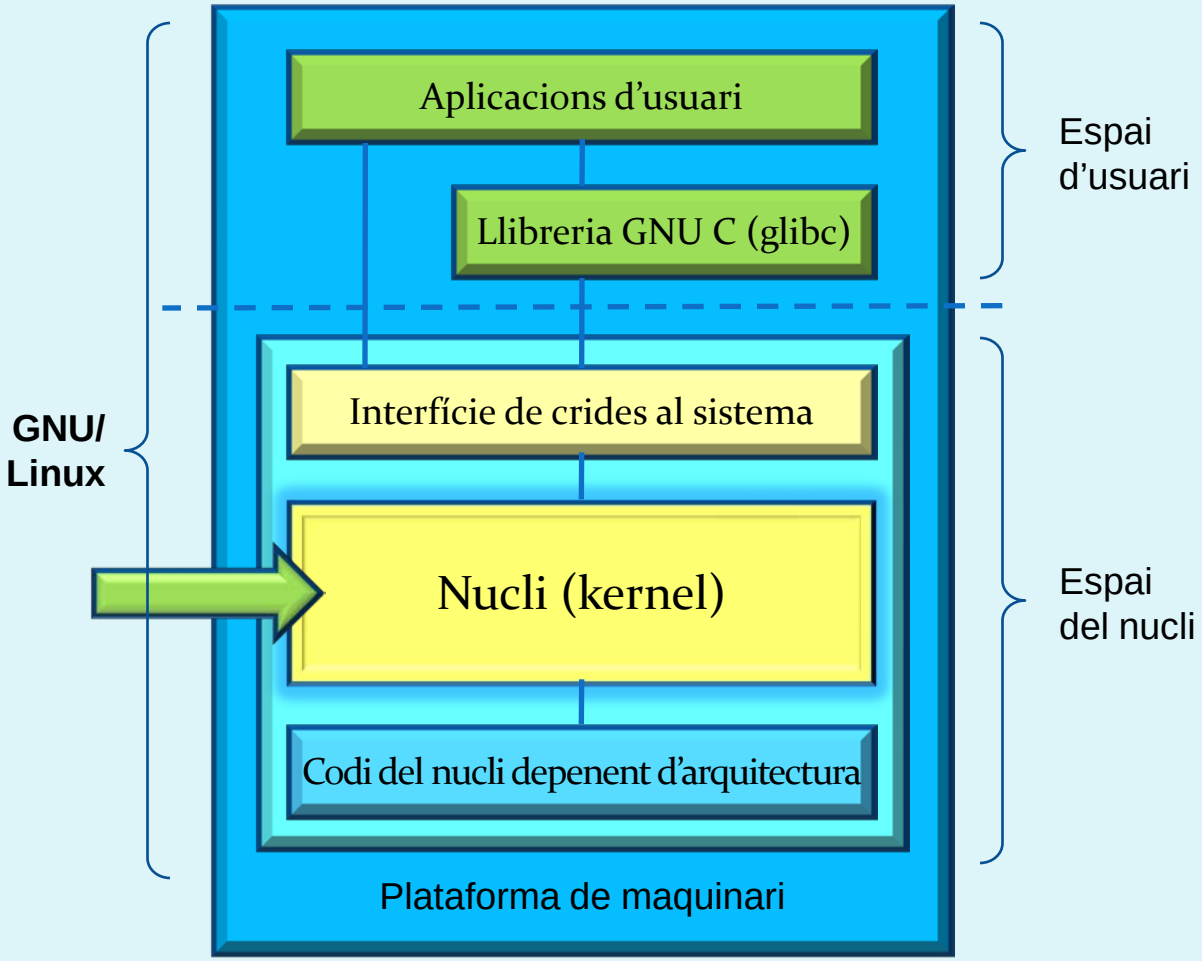
Estadístiques de crides al sistema en plataforma GNU/Linux

German Guirado
TFC ETIS (UOC)



INTRODUCCIÓ

Aquest projecte es basa en la **modificació del nucli del sistema operatiu GNU/Linux**, per dotar-lo de la capacitat d'extreure **estadístiques de crides al sistema (syscalls)**. A partir de la modificació, compilació i instal·lació del nou nucli, es registrarà informació sobre el **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.



PRESENTACIÓ

Context i justificació:

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.

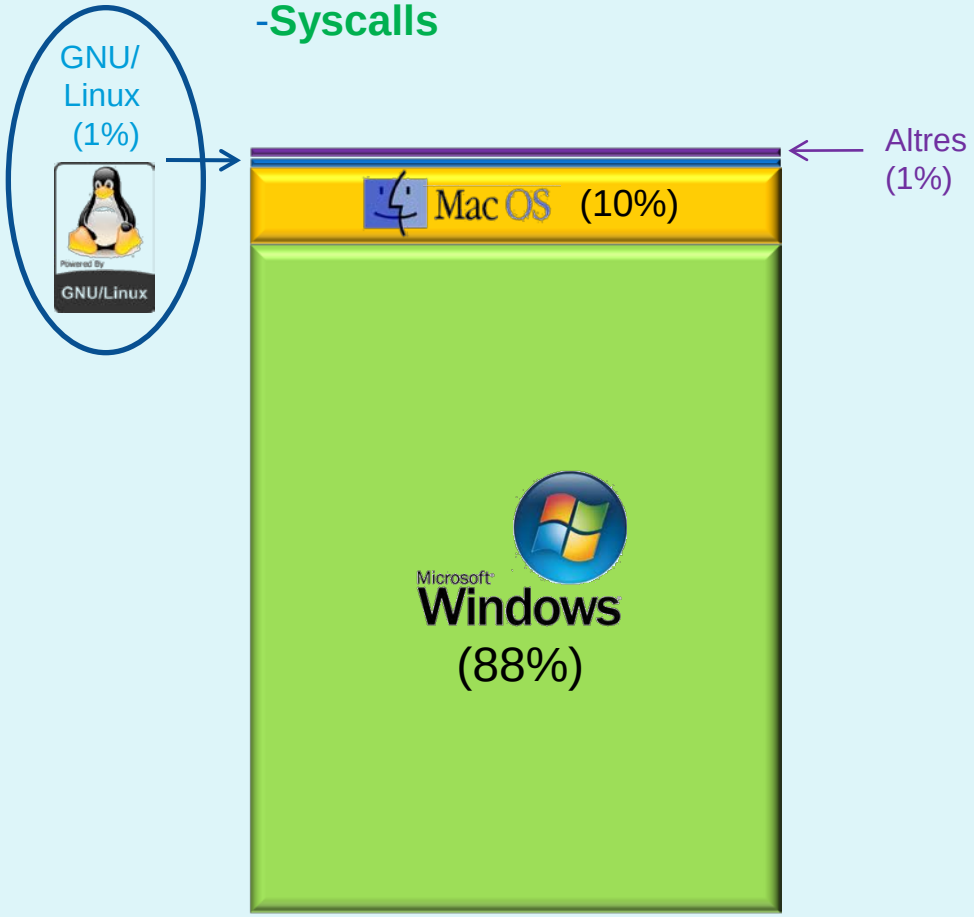
Tot i la seva potència, qualitat i fiabilitat (cosa que està fent que s'estengui de forma important per instal·lacions de servidors), el sistema encara no té gaire presència entre el usuaris finals de computadores.

Per a un usuari novell en aquest sistema operatiu, 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.

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.

Temes cabdals en GNU/Linux:

-Kernel
-Syscalls



OBJECTIUS

OBJECTIU DEL PROJECTE:

Modificar el kernel de Linux per obtenir estadístiques de les crides al sistema.

OBJECTIUS ACADÈMICS:

-Aprendre a modificar i compilar el nucli (kernel) del sistema GNU/Linux.

-Adquirir coneixements sobre les crides al sistema més utilitzades.

OBJECTIUS PÚBLICS:

-Documentar, de forma explicativa, el procés dut a terme.

-Oferir informació, en forma d'estadístiques, de les syscalls.

MÈTODE

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

- Instal·lació del sistema operatiu (Ubuntu 11.04).
- Instal·lació del codi font i eines de compilació.
- Compilació i instal·lació del nou kernel.

FASE 2: Modificació del kernel

- Programació per enregistrar les crides al sistema (*syscalls*).
- Editar fitxer .C que fa la funció de dispatching de les *syscalls*.

FASE 3: Informació i resultats

- Representació de la informació.
- Conclusions (anàlisi i valoracions).

PREPARACIÓ D'ENTORN i COMPILACIÓ DEL KERNEL

INSTAL·LAR CODI FONT i EINES DE COMPILACIÓ

CODI FONT DEL NUCLI

Descarregar el codi font del kernel

$ sudo apt-get install linux-source

Descomprimir el codi font del kernel

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

EINES DE COMPILACIÓ

Instal·lar eina de compilació 'gcc'

$ sudo apt-get install gcc

+

Instal·lar eina de compilació 'make'

$ sudo apt-get install make

COMPILACIÓ DEL NOU NUCLI

CONFIGURACIÓ DEL NUCLI

Copiar una configuració coneguda

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

Instal·lar el paquet 'libqt4-dev'

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

Configurar el nucli

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

COMPILACIÓ DEL NUCLI

Obtenir 'make-kpkg'i netejar config.:

$ sudo apt-get install kernel-package
/usr/src/linux-source-2.6.38$
sudo make-kpkg clean

Modificar l'arxiu '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
Descomentar la línia 160 i comentar la 161:
160: EXTRA_LDFLAGS += $(src)/sections.lds
161: #EXTRA_LDFLAGS += $(PWD)/ubuntu/omnibook/sections.lds

Compilar el nucli

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

INSTAL·LACIÓ DEL NOU NUCLI

INSTAL·LACIÓ NOU NUCLI

Arxiu d'imatge creat

/usr/src/**kernel-image-2.6.38.281011_10.00.Custom_i386.deb**

Instal·lar el nucli amb l'arxiu d'imatge

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

Al reiniciar, apareix una nova opció en la llista de sistemes operatius, que porta el sufix indicat al paràmetre "--append-to-version="

MODIFICACIÓ DEL KERNEL

FUNCIÓ DE DISPATCHING DE LES SYSCALLS

CREAR NOVA *SYSCALL*

Nom de la *syscall*

/usr/src/linux-2.6.38/arch/x86/kernel/syscall_table.S (**Afegir línia al final**):
.long sys_statistics

Número de la *syscall*

```
#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
```

Definició de la *syscall*

/usr/src/linux-2.6.38/include/linux/syscalls.h (**Afegir línia al final**):
asmlinkage long sys_statistics(int i);

DIRECTORIS I ARXIUS

Paràmetre "core-y"

/usr/src/linux-2.6.38/Makefile:
[...] core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ **statistics/** [...]

Directori per la *syscall*

Crear el directori:
/usr/src/linux-2.6.38/statistics

Codi de la *syscall*

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

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

Arxiu 'Makefile'

/usr/src/linux-2.6.38/statistics/Makefile:
obj-y := statistics.o

MODIFICACIÓ NUCLI

Funció de dispatching

/usr/src/linux-2.6.38/arch/x86/kernel/entry_32.S (punt d'entrada al kernel):

```
syscall_call:
  call *sys_call_table(,264,4)
  call *sys_call_table(,%eax,4)
```

Compilar el nucli

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

Instal·lar el nou nucli

```
/usr/src$
sudo dpkg –i linux-image-2.6.38.8.021111_2.6.38.8. 021111-10.00.Custom_i386.deb
```

ALTRES SOLUCIONS

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. Per tant, la primera solució plantejada no és vàlida. A continuació, es presenten altres solucions que he intentat per enregistrar les crides al sistema:

Solució 2: Cridar directament a la nova syscall '**sys_statistics**' des de 'entry_32.S':

```
syscall_call:
call *sys_statistics(,%eax)
call *sys_call_table(,%eax,4)
```

Solució 3: Cridar directament a la funció '**printk**' des de 'entry_32.S':

```
syscall_call:
call *printk(KERN_DEBUG, %eax)
call *sys_call_table(,%eax,4)
```

Solució 4: Obviar el paràmetre de 'printk' '**KERN_DEBUG**' a 'entry_32.S':

```
syscall_call:
call *printk(%eax)
call *sys_call_table(,%eax,4)
```

Solució 5: Utilitzar la **pila** per passar els paràmetres de 'printk' a 'entry_32.S':

```
syscall_call:
  push %eax
  call *printk
  pop %eax
```

Solució 6:

Diferents tècniques basades en crides a la funció '**sys_statistics**'

Solució 8:

Diferents tècniques basades en passar com a paràmetre el registre '**%eax**'

Solució 7:

Diferents tècniques basades en crides a la funció '**printk**'

Solució 9:

Altres tècniques i variacions de les solucions anteriors. **Cap d'elles funciona**.

SOLUCIÓ DEFINITIVA

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.

SOLUCIÓ DEFINITIVA: Editar els 72 arxius de crides al sistema:

arch/i386/kernel/ioport.c	fs/dquot.c	fs/open.c	fs/xattr.c	kernel/module.c	mm/madvise.c
arch/i386/kernel/ldt.c	fs/eventfd.c	fs/pipe.c	ipc/mqueue.c	kernel/perf_event.c	mm/mempolicy.c
arch/i386/kernel/process.c	fs/eventpoll.c	fs/read_write.c	kernel/acct.c	kernel/posix-timers.c	mm/migrate.c
arch/i386/kernel/ptrace.c	fs/exec.c	fs/readdir.c	kernel/capability.c	kernel/printk.c	mm/mincore.c
arch/i386/kernel/signal.c	fs/fcntl.c	fs/select.c	kernel/exec_domain.c	kernel/sched.c	mm/mlock.c
arch/i386/kernel/sys_i386.c	fs/filesystems.c	fs/signalfd.c	kernel/exit.c	kernel/signal.c	mm/mmap.c
arch/i386/kernel/vm86.c	fs/ioctl.c	fs/splice.c	kernel/fork.c	kernel/sys.c	mm/mprotect.c
arch/mips/kernel/syscall.c	fs/ioprio.c	fs/stat.c	kernel/futex.c	kernel/sysctl.c	mm/mremap.c
cket.c	fs/locks.c	fs/statfs.c	kernel/groups.c	kernel/time.c	mm/swapfile.c
fs/notify/inotify/inotify_user.c	fs/namei.c	fs/super.c	kernel/info.c	kernel/timer.c	net/socket.c
fs/buffer.c	fs/namespace.c	fs/timerfd.c	kernel/itimer.c	mm/filemap.c	profile.c
fs/dcache.c	fs/aio.c	fs/utimes.c	kernel/kexec.c	mm/fremap.c	security/keys/keyctl.c

EXEMPLES

kernel/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();
[...]
```

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);
asmlinkage_protect(3, ret, filename, flags, mode); return
ret; } [...]
```

INFORMACIÓ i RESULTATS

printk(KERN_DEBUG “|NomDeLaSysCall”)

/var/log/syslog

```
[...]
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
[...]
```

Recompte complet de *syscalls* a l’inici

Selecció d’1.000.000 de resultats

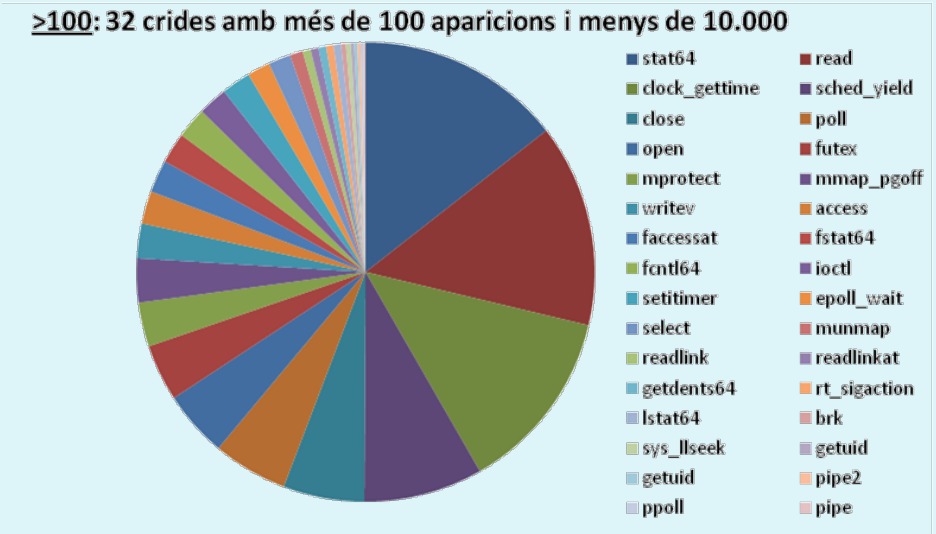
num	SysCall	Description	Source	1.000.000
13	**time**	get time in seconds	kernel/time.c	414.820
78	**gettimeofday**	get the date and time	kernel/time.c	354.551
4	**write**	write to a file descriptor	fs/read_write.c	167.434
195	stat64		fs/stat.c:358	8.936
3	read	read from a file descriptor	fs/read_write.c	8.867
265	clock_gettime		kernel/posix-timers.c	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

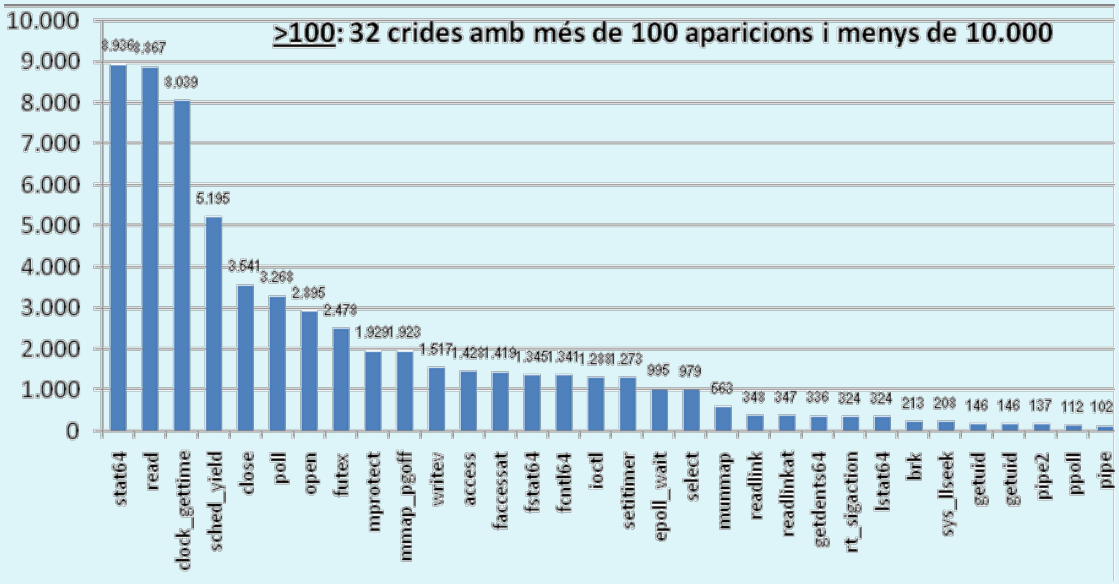
...

‘**time**’, ‘**gettimeofday**’ i ‘**write**’ són, amb diferència, les crides més habituals. Però 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*). **Les descarto**. A més, estudiar les 134 syscalls restants de forma conjunta no aporta bona informació. En faig **3 grups**.

32 SYSCALLS “>100”

32 crides al sistema amb més de 100 aparicions i menys de 10.000

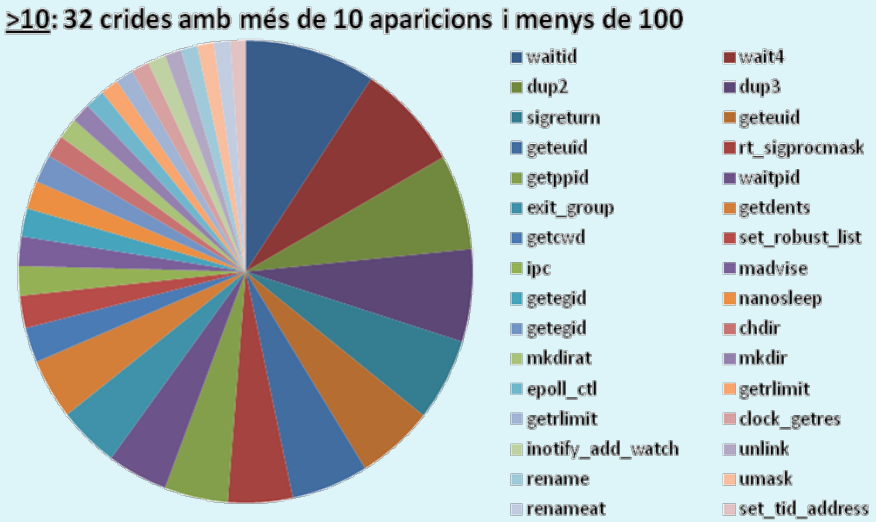


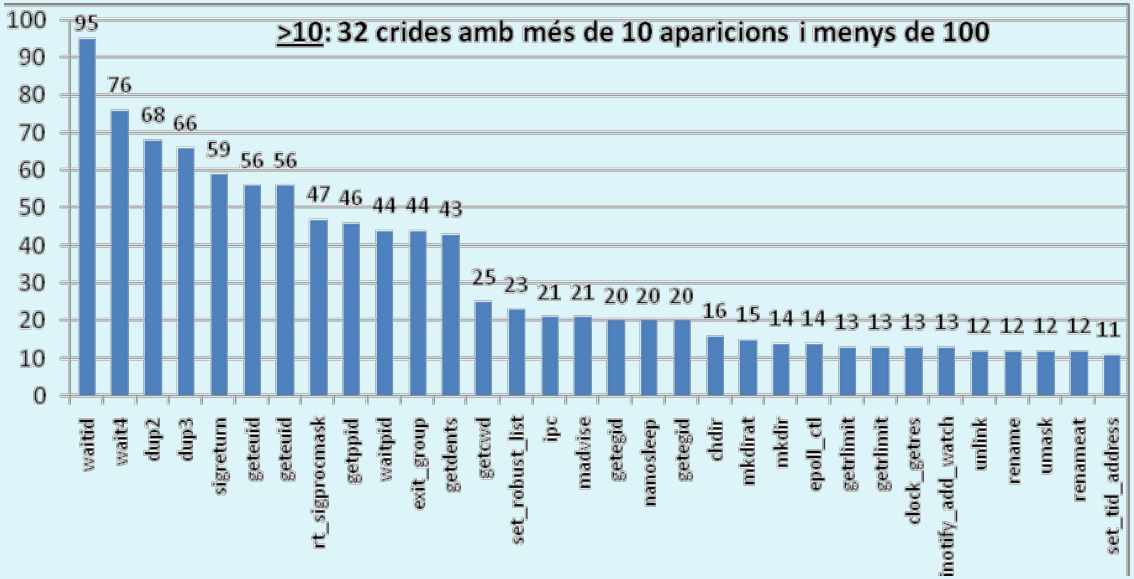


stat64, read, clock_gettime, sched_yield, close, poll, open, futex, mprotect, mmap_pgoff, writev, access, faccessat, fstat64, fcntl64, ioctl, setitimer, epoll_wait, select, munmap, readlink, readlinkat, getdents64, rt_sigaction, lstat64, brk, sys_llseek, getuid, getuid, pipe2, ppoll, pipe.

32 SYSCALLS “>10”

32 crides al sistema amb més de 10 aparicions i menys de 100

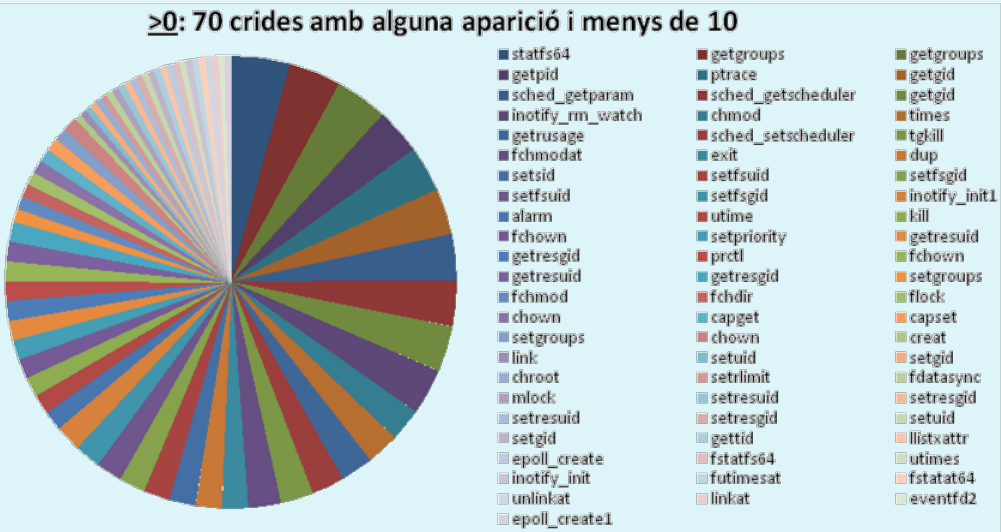


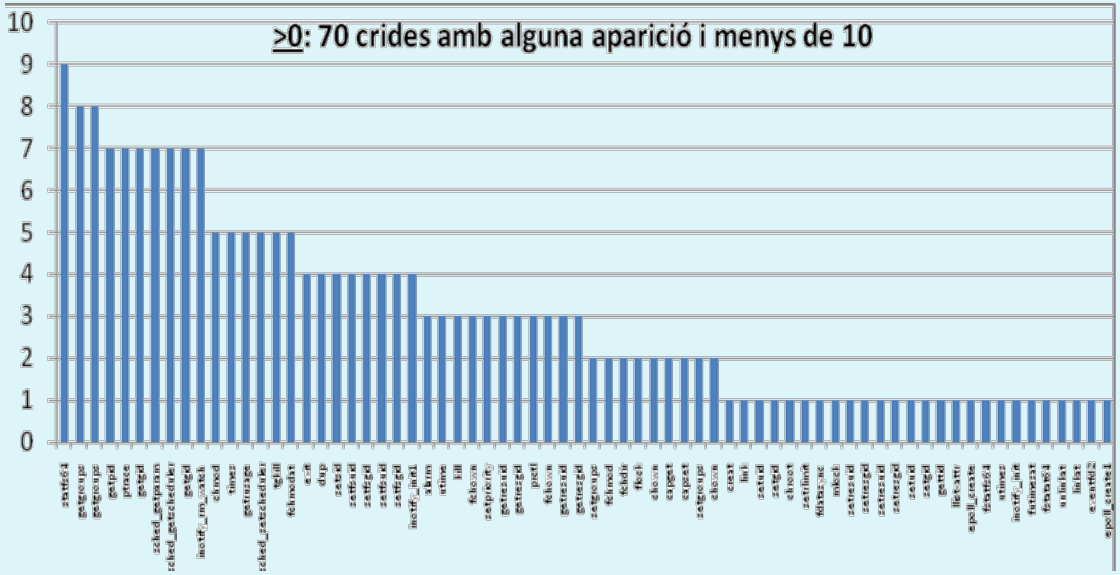


exit_group, getdents, getcwd, set_robust_list, ipc, madvise, getegid, nanosleep, getegid, chdir, mkdirat, mkdir, epoll_ctl, getrlimit, getrlimit, clock_getres, inotify_add_watch, unlink, rename, umask, renameat, set_tid_address.

70 SYSCALLS “>0”

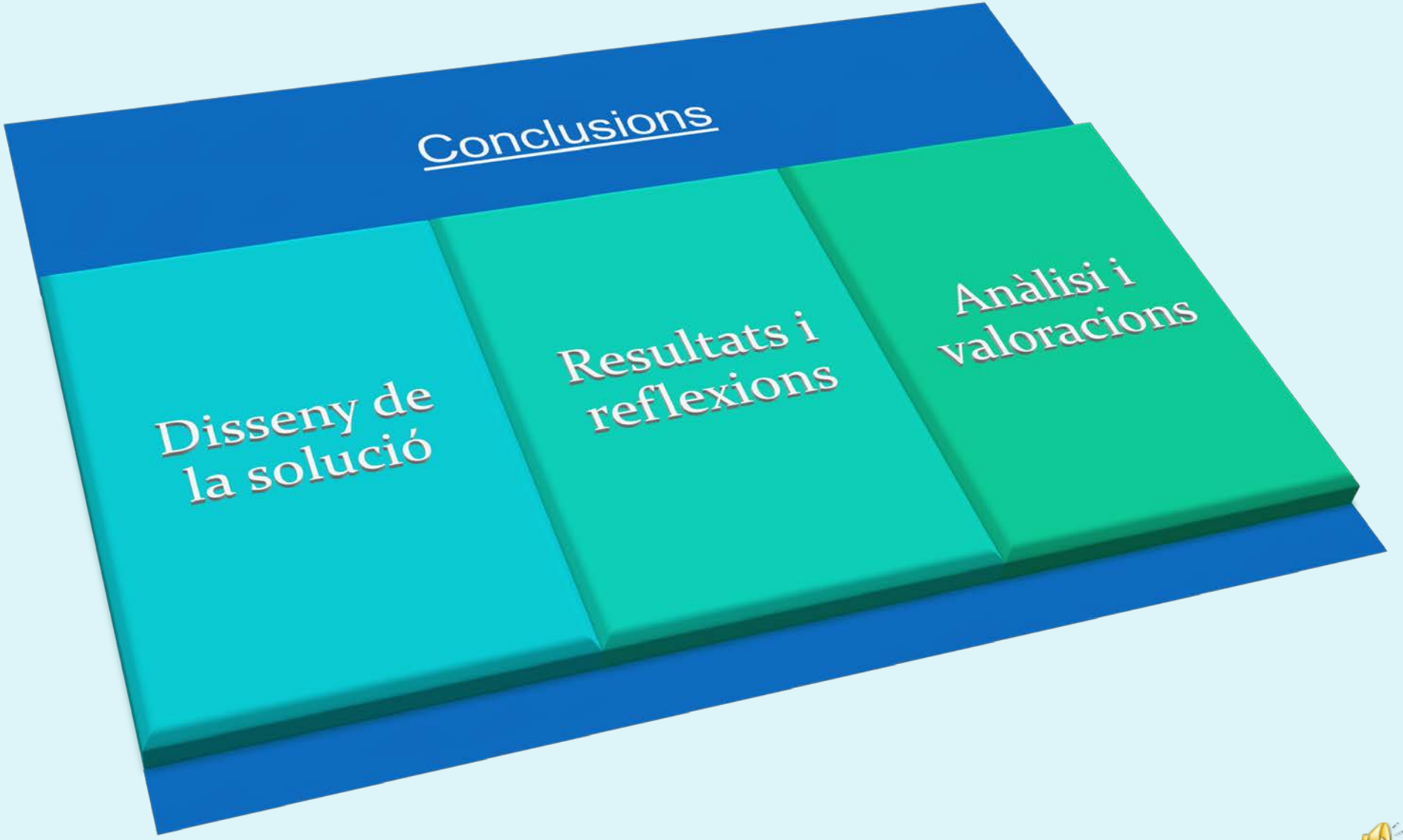
70 crides al sistema amb alguna aparició (>0) i menys de 10





statfs64, 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.

CONCLUSIONS



RESULTATS i REFLEXIONS

Preparació de l'entorn i compilació del kernel

Instal·lació del Stma. Op.

Ubuntu 11.04 compta amb molt de suport. Instal·lació **senzilla i ràpida**.

Instal·lació del codi font i eines de compilació

Aprenentatge d'algunes ordres com "**apt-get install**", "**sudo**" o "**tar**". Instal·lació del codi font de Linux.

Compilació i instal·lació del nou nucli

Configurar, compilar i instal·lar un nou nucli ha resultat una tasca **senzilla** però **molt llarga**.

El disseny de la solució ha estat el correcte, excepte l'apartat de modificació del nucli

Modificació del kernel (nucli)

Idea inicial: no vàlida

A partir de la versió **2.6** del nucli de Linux, la taula de crides al sistema (syscall table) ja **no és exportada.**

Solució nova syscall: no vàlida

Per motius de seguretat, els desenvolupadors del kernel de Linux han fet més difícil afegir noves crides al sistema.

Altres solucions: no vàlides

El **compilador** no acceptava el codi que havia insertat, o el sistema es **bloquejava**, o bé les modificacions no generaven **cap resultat** d'enregistrament de crides al sistema

Solució definitiva i funcional

Modificació dels **arxius ".c"** que contenen funcions de crides al sistema. Mitjançant l'ordre "**printk**", he pogut crear un '*log*' de les syscalls

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

Estructurar la informació

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'aparicions**, amb el que he situat en les primeres posicions les syscalls executades amb més freqüència.

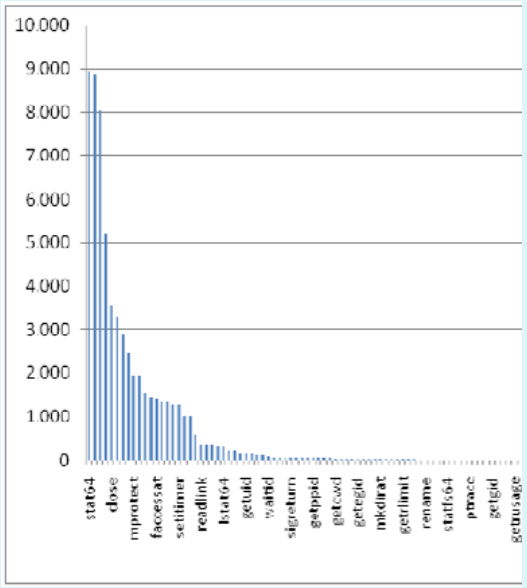
ANÀLISI i VALORACIONS

El problema de les dades extremes

num	SysCall	1.000.000
13	time	414.820
78	gettimeofday	354.551
4	write	167.434
195	stat64	8.936
3	read	8.867
265	clock_gettime	8.039
158	sched_yield	5.195
6	close	3.541
168	poll	3.268

Les tres primeres syscalls ('time', 'gettimeofday' i 'write') són estadísticament descartables per tractar-se d'extrems que representen informació no adient. Així, vaig decidir separar-les de l'estadística, per estudiar la informació correctament.

El problema de la varietat de les dades



Les 134 syscalls restants són massa per mostrar-les i analitzar-les de forma clara i que aporti algun significat (perquè els seus valors són molt diferents). He fet servir el mètode estadístic de l'agrupació, en funció del nº d'aparicions als resultats.

CONCLUSIÓ

Objectius acadèmics

-Aprendre a modificar i compilar el nucli del sistema GNU/Linux.

-Adquirir coneixements sobre les crides al sistema més utilitzades.

Objectius públics

-Documentar, de forma explicativa, el procés dut a terme.

-Oferir informació, en forma d'estadístiques, de les *syscalls*.

OBJECTIU DEL PROJECTE

Modificar el kernel de Linux per obtenir estadístiques de les crides al sistema.