

Laboratorio #1:

Llamadas al Sistema y Kernel

Ejercicio 1 (20 puntos):

1. Compile el primer programa y ejecútalo varias veces. Responda: ¿por qué aparecen números diferentes cada vez?
 - a. getpid() es una llamada de sistema que devuelve el número del proceso que hace dicha llamada. Es decir, que muestra el ID del proceso ejecutado, lo cual es el pid_t.
2. Proceda a compilar el segundo programa y ejecútalo una vez. ¿Por qué aparecen dos números distintos a pesar de que estamos ejecutando un único programa?
 - a. Esto quiere decir que son diferentes procesos, ya que cada ejecución fue diferente y lo sabemos gracias al id de proceso por el pid_t.
3. ¿Por qué el primer y el segundo números son iguales?
 - a. En este caso, es por que van representando la cantidad de procesos ejecutados en el programa. También, al ejecutarlo varias veces, claramente cambie el número, incluso dentro de la bifurcación, ya que son diferentes procesos.
4. En la terminal, ejecute el comando top (que despliega el top de procesos en cuanto a consumo de CPU) y note cuál es el primer proceso en la lista (con identificador 1). ¿Para qué sirve este proceso?
 - a. El PID 1 es el proceso de inicio principal responsable de iniciar y apagar el sistema.

PID	USER	PR	NI	VIRT	RES	SAR	S	%CPU	%MEM	TIME+	COMMAND
2048	os	20	0	82324	11m	9208	S	0.7	3.0	0:17.89	gnome-terminal
1197	root	20	0	43544	18m	7696	S	0.3	4.9	1:24.51	Xorg
1886	os	20	0	17388	6032	4828	S	0.3	1.6	0:01.59	gnome-screensav
1	root	20	0	2036	720	624	S	0.0	0.2	0:00.74	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:00.02	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
6	root	20	0	0	0	0	S	0.0	0.0	0:00.26	events/0
7	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuset
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khelper
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	netns
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	async/mgr
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pm
12	root	20	0	0	0	0	S	0.0	0.0	0:00.02	sync_supers

Ejercicio 2 (30 puntos):

1. Observe el resultado desplegado. ¿Por qué la primera llamada que aparece es `execve`?
 - a. Porque este se refiere al nombre de la ruta. En otras palabras, `execve()` ejecuta el programa al que se refiere el nombre de ruta.
2. Ubique las llamadas de sistema realizadas por usted. ¿Qué significan los resultados (números que están luego del signo '=')?
 - a. Es el orden en que se van ejecutando los procesos, ya que algunos tienen números binarios, y otros tienen números negativos. Si estoy en lo correcto, número negativos son llamadas del sistema canceladas por un error, y de 0 a n, siendo 0 prioridad, muestra el orden del llamado.

3. ¿Por qué entre las llamadas realizadas por usted hay un `read` vacío?

- a. Ese `read` quiere decir que omite los espacios y solo le intereso a los espacios con valores, lo cual en este caso lo salto.

```
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
545000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb76456c0, limit:10485
32bit:1, contents:0, read_exec_only:0, limit_in_pages:1, seg_not_present
ple:1}) = 0
mprotect(0xb7787000, 8192, PROT_READ)      = 0
mprotect(0xb77bb000, 4096, PROT_READ)      = 0
munmap(0xb778d000, 64828)                  = 0
open("prueba.txt", O_RDONLY)                = 3
creat("copia_prueba.txt", 0666)             = 4
read(3, "hola\n", 8192)                    = 5
write(4, "hola\n", 5)                      = 5
read(3, "", 8192)                          = 0
close(3)                                   = 0
close(4)                                   = 0
exit_group(0)                              = ?
os@debian:~/Desktop$ ./ejercicio2 'prueba.txt' 'copia_prueba.txt'
os@debian:~/Desktop$ top
```

4. Identifique tres servicios distintos provistos por el sistema operativo en este `strace`. Liste y explique brevemente las llamadas a sistema que corresponden a los servicios identificados (puede incluir `read`, `write`, `open` `close` que el sistema haga por usted, no los que usted haya producido directamente con su programa).
 - a. **Write:** escribe bytes del archivo original con el buffer al archivo copia o socket asociado.
 - b. **Open:** Abre un archivo para leer o escribir. También permite un archivo para ser bloqueado proporcionando acceso exclusivo.
 - c. **Close:** le dice al sistema operativo que ha terminado con un descriptor de archivo y cierra el archivo que señaló `fd`.

Ejercicio 3 (50 puntos):

Screenshots del proceso:

```
os@debian:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
os@debian:~$ cd ..
os@debian:/home$ ls
os
os@debian:/home$ cd ~
os@debian:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
os@debian:~$ cd ~
os@debian:~$ sudo cp -a /usr/src/linux-2.6.39.4 .
[sudo] password for os:
os@debian:~$
```

Screenshot #1: Primer paso es copiar el código fuente del sistema operativo.

```
GNU nano 2.2.4      File: syscall_table_32.S      Modified:
.
.
.
.long sys_inotify_init1
.long sys_preadv
.long sys_pwritev
.long sys_rt_tgsigqueueinfo      /* 335 */
.long sys_perf_event_open
.long sys_recvmmsg
.long sys_fanotify_init
.long sys_fanotify_mark
.long sys_prlimit64              /* 340 */
.long sys_name_to_handle_at
.long sys_open_by_handle_at
.long sys_clock_adjtime
.long sys_syncfs
.long sys_mycall                 /* 345 */

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^L UnCut Text ^T To Spell
```

Screenshot #2: Agregar sys_mycall con comentario.

```
os@debian: ~/linux-2.6.39.4/arch/x86/include/asm
File Edit View Terminal Help
GNU nano 2.2.4 File: unistd_32.h Modified

#define __NR_rt_tsigqueueinfo 335
#define __NR_perf_event_open 336
#define __NR_recvmmsg 337
#define __NR_fanotify_init 338
#define __NR_fanotify_mark 339
#define __NR_prlimit64 340
#define __NR_name_to_handle_at 341
#define __NR_open_by_handle_at 342
#define __NR_clock_adjtime 343
#define __NR_syncfs 344
#define __NR_mycall 345

#ifdef __KERNEL__
#define NR_syscalls 346
#define __ARCH_WANT_IPC_PARSE_VERSION

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

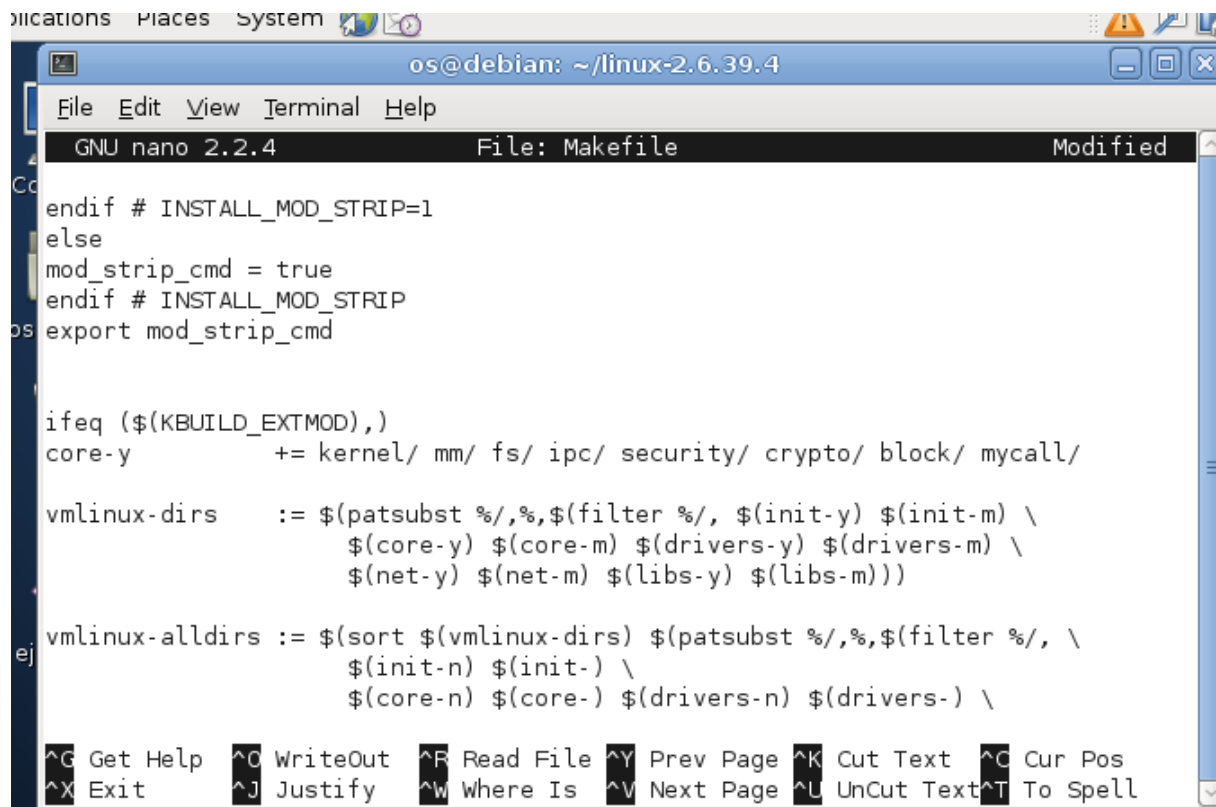
Screenshot #3: Se agrega el mycall y se modifica la cantidad de llamadas del sistema.

```
os@debian: ~/linux-2.6.39.4/include/linux
File Edit View Terminal Help
GNU nano 2.2.4 File: syscalls.h Modified

unsigned long prot, unsigned long flags,
unsigned long fd, unsigned long pgoff);
asmlinkage long sys_old_mmap(struct mmap_arg_struct __user *arg);
asmlinkage long sys_name_to_handle_at(int dfd, const char __user *name,
struct file_handle __user *handle,
int __user *mnt_id, int flag);
asmlinkage long sys_open_by_handle_at(int mountdirfd,
struct file_handle __user *handle,
int flags);
asmlinkage long sys_mycall(int i);
#endif

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Screenshot #4: Se crea la función sys_mycall como llamado dentro del SO.



```
os@debian: ~/linux-2.6.39.4
GNU nano 2.2.4 File: Makefile Modified

endif # INSTALL_MOD_STRIP=1
else
mod_strip_cmd = true
endif # INSTALL_MOD_STRIP
export mod_strip_cmd

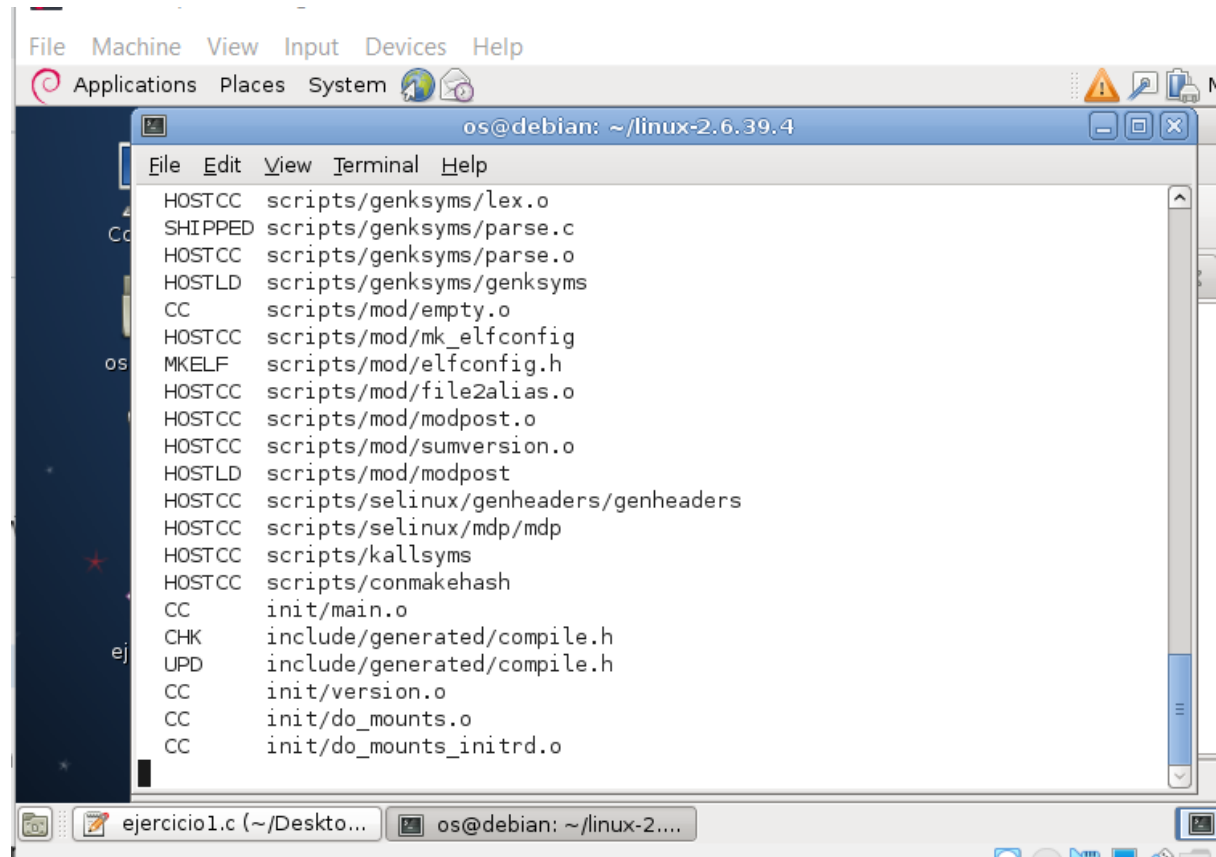
ifeq ($(KBUILD_EXTMOD),)
core-y      += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ mycall/

vmlinux-dirs := $(patsubst %/,%,$(filter %/, $(init-y) $(init-m) \
$(core-y) $(core-m) $(drivers-y) $(drivers-m) \
$(net-y) $(net-m) $(libs-y) $(libs-m)))

vmlinux-alldirs := $(sort $(vmlinux-dirs) $(patsubst %/,%,$(filter %/, \
$(init-n) $(init-) \
$(core-n) $(core-) $(drivers-n) $(drivers-) \

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Screenshot #5: Modificación de core-y.



```
os@debian: ~/linux-2.6.39.4
GNU nano 2.2.4 File: Makefile Modified

HOSTCC scripts/genksyms/lex.o
SHIPPED scripts/genksyms/parse.c
HOSTCC scripts/genksyms/parse.o
HOSTLD scripts/genksyms/genksyms
CC scripts/mod/empty.o
HOSTCC scripts/mod/mk_elfconfig
MKELF scripts/mod/elfconfig.h
HOSTCC scripts/mod/file2alias.o
HOSTCC scripts/mod/modpost.o
HOSTCC scripts/mod/symversion.o
HOSTLD scripts/mod/modpost
HOSTCC scripts/selinux/genheaders/genheaders
HOSTCC scripts/selinux/mdp/mdp
HOSTCC scripts/kallsyms
HOSTCC scripts/conmakehash
CC init/main.o
CHK include/generated/compile.h
UPD include/generated/compile.h
CC init/version.o
CC init/do_mounts.o
CC init/do_mounts_initrd.o
```

Screenshot #6: Configuración del menuconfig.

```
os@debian: ~/linux-2.6.39.4
File Edit View Terminal Help
H16T0FW firmware/edgeport/boot2.fw
H16T0FW firmware/edgeport/down.fw
H16T0FW firmware/edgeport/down2.fw
IHEX    firmware/edgeport/down3.bin
IHEX2FW firmware/whiteheat_loader.fw
IHEX2FW firmware/whiteheat.fw
IHEX2FW firmware/keyspan_pda/keyspan_pda.fw
IHEX2FW firmware/keyspan_pda/xircom_pgs.fw
IHEX    firmware/cpia2/stv0672_vp4.bin
IHEX    firmware/yam/1200.bin
IHEX    firmware/yam/9600.bin
IHEX    firmware/sb16/mulaw_main.csp
IHEX    firmware/sb16/alaw_main.csp
IHEX    firmware/sb16/ima_adpcm_init.csp
IHEX    firmware/sb16/ima_adpcm_playback.csp
IHEX    firmware/sb16/ima_adpcm_capture.csp
os@debian:~/linux-2.6.39.4$ sudo make modules
[sudo] password for os:
CHK      include/linux/version.h
CHK      include/generated/utsrelease.h
CALL     scripts/checksyscalls.sh
```

Screenshot #7: Compilación con el nuevo recurso creado en los pasos anteriormente.

```
os@debian: ~/linux-2.6.39.4
File Edit View Terminal Help
INSTALL drivers/parport/parport.ko
INSTALL drivers/parport/parport_ax88796.ko
INSTALL drivers/parport/parport_cs.ko
INSTALL drivers/parport/parport_pc.ko
INSTALL drivers/parport/parport_serial.ko
INSTALL drivers/pci/hotplug/acpihp.ko
INSTALL drivers/pci/hotplug/acpihp_ibm.ko
INSTALL drivers/pci/hotplug/cpcihp_generic.ko
INSTALL drivers/pci/hotplug/cpcihp_zt5550.ko
INSTALL drivers/pci/hotplug/cpqphp.ko
INSTALL drivers/pci/hotplug/fakephp.ko
INSTALL drivers/pci/hotplug/ibmphp.ko
INSTALL drivers/pci/hotplug/pci_hotplug.ko
INSTALL drivers/pci/hotplug/pciehp.ko
INSTALL drivers/pci/hotplug/shpchp.ko
INSTALL drivers/pci/pci-stub.ko
INSTALL drivers/pci/pcie/aer/aer_inject.ko
INSTALL drivers/pcmcia/i82092.ko
INSTALL drivers/pcmcia/i82365.ko
INSTALL drivers/pcmcia/pcmcia.ko
INSTALL drivers/pcmcia/pcmcia_core.ko
```

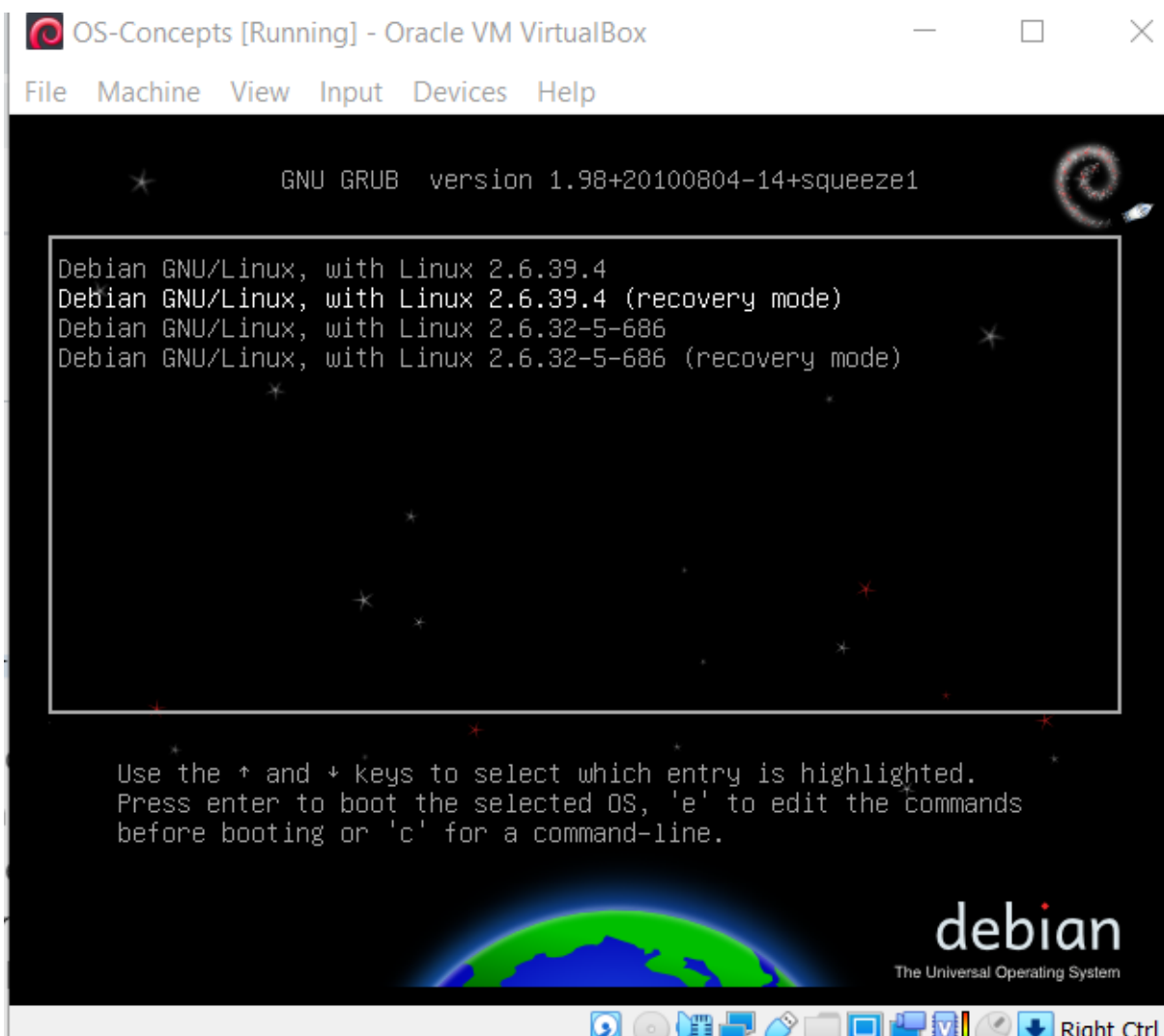
Screenshot #8: Instalación del nuevo kernel.

```

os@debian:~/linux-2.6.39.4$ sudo make install
sh /home/os/linux-2.6.39.4/arch/x86/boot/install.sh 2.6.39.4 arch/x86/boot/bzIm
ge \
    System.map "/boot"
os@debian:~/linux-2.6.39.4$ sudo update-initramfs -c -k 2.6.39.4
update-initramfs: Generating /boot/initrd.img-2.6.39.4
os@debian:~/linux-2.6.39.4$ sudo update-grub
Generating grub.cfg ...
Found background image: /usr/share/images/desktop-base/desktop-grub.png
Found linux image: /boot/vmlinuz-2.6.39.4
Found initrd image: /boot/initrd.img-2.6.39.4
Found linux image: /boot/vmlinuz-2.6.32-5-686
Found initrd image: /boot/initrd.img-2.6.32-5-686
done
os@debian:~/linux-2.6.39.4$ █

```

Screenshot #9: Iniciar el nuevo kernel antes de reiniciar el VMI.

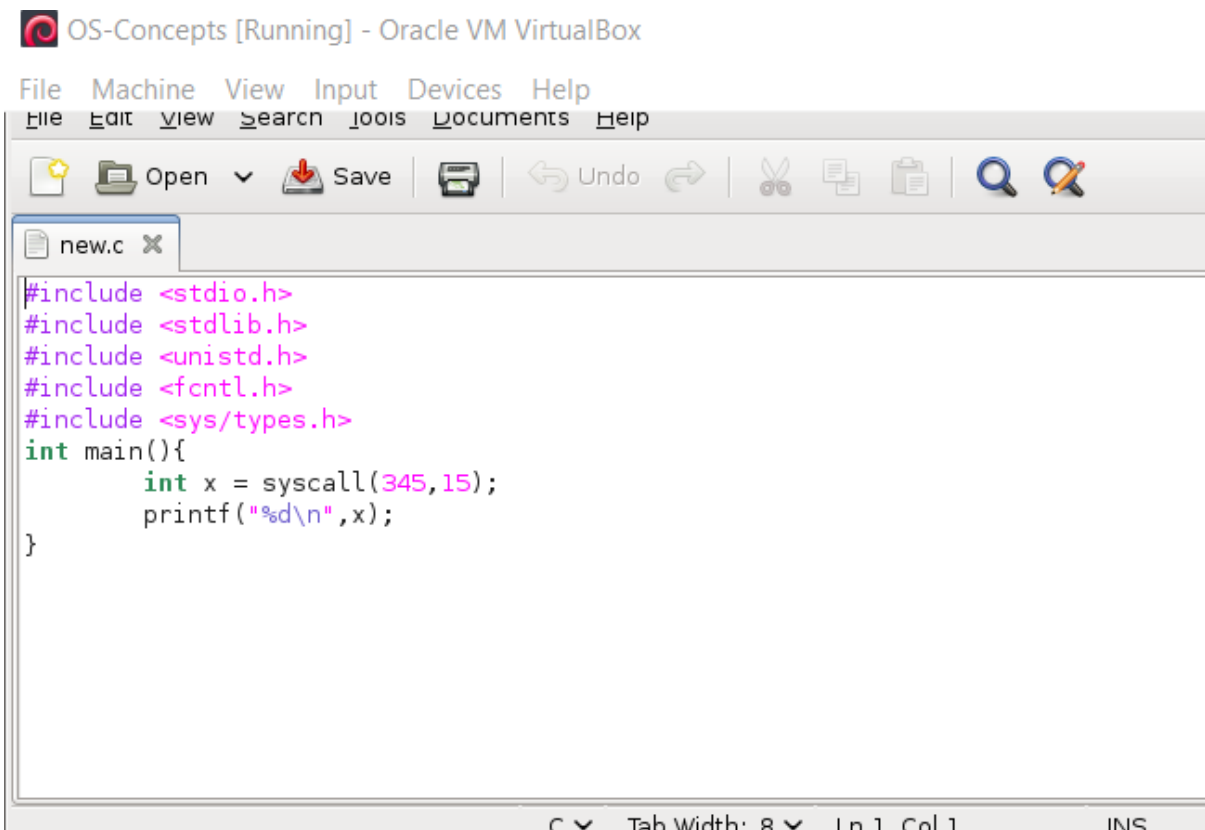


Screenshot #10: Cuatro nuevas opciones en el SO.

1. ¿Qué ha modificado aquí, la interfaz de llamadas de sistema o el API? Justifique su respuesta.
 - a. Aquí en mi opinión, y basado en lo que se hizo, fue el API. El sistema operativo actúa como intermediario, ofreciendo un API que el programa puede usar en cualquier

momento para solicitar recursos gestionados por el sistema operativo. En este caso, como generamos un nuevo recurso que ofrece el sistema operativo, podemos llamarlo en cualquier momento de esa versión que modificamos.

2. ¿Por qué usamos el número de nuestra llamada de sistema en lugar de su nombre?
 - a. Porque la llamada al sistema proporciona los servicios del sistema operativo a los programas de usuario a través de la API, y cómo solicitamos un servicio del kernel del sistema operativo, se ejecuta llamando desde el API.
3. ¿Por qué las llamadas de sistema existentes como read o fork se pueden llamar por nombre?
 - a. Porque del parte del kernel, son funciones de llamadas al sistema por procesos, es decir, Linux proporciona unas llamadas definidas en el sistema por un nombre, en vez de números.
4. Incluya entre sus respuestas una captura de pantalla con el resultado de la ejecución de su llamada a sistema.



OS-Concepts [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

File Edit View Search Tools Documents Help

new.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
int main(){
    int x = syscall(345,15);
    printf("%d\\n",x);
}
```

C Tab Width: 8 Ln 1, Col 1 INS


```
File Edit View Terminal Help
os@debian:~$ cd Desktop/
os@debian:~/Desktop$ sudo touch new.c
[sudo] password for os:
os@debian:~/Desktop$ sudo gedit new.c
^[[A^[[A^Cos@debian:~/Desktop$ sudo gcc -o test new.c
os@debian:~/Desktop$ sudo ./test
22
os@debian:~/Desktop$ █
```