

Sistemas Operativos

1er Semestre 2017

Laboratorio 2

Profesores:

Cristóbal Acosta (cristobal.acosta@usach.cl)

Miguel Cárcamo (miguel.carcamo@usach.cl)

Fernando Rannou (fernando.rannou@usach.cl)

Ayudantes:

Fernanda Estay (fernanda.estay@usach.cl)

Natalia Pérez (natalia.perez.g@usach.cl)

I. Objetivos

El presente laboratorio tiene por objetivo aplicar los conceptos y técnicas asociadas a *System Call* de un sistema operativo, mediante la creación de un llamado al sistema que entregue información acerca los procesos que se encuentran en ejecución. Para ello el estudiante debe modificar, compilar e instalar un kernel de Linux utilizando el lenguaje de programación C, tal que incorpore el nuevo system call.

Además, debe escribir un pequeño programa en C que utilice el llamado al sistema.

II. Enunciado

El laboratorio consiste en codificar un nuevo llamado al sistema `procinfo()` que liste y entregue información de todos los procesos que se encuentren en un estado particular. Un proceso en Linux puede estar en uno de los siguientes estados:

TASK_RUNNING	0
TASK_INTERRUPTIBLE	1
TASK_UNINTERRUPTIBLE	2
__TASK_STOPPED	4
__TASK_TRACED	8
EXIT_DEAD	16
EXIT_ZOMBIE	32
TASK_DEAD	64
TASK_WAKEKILL	128
TASK_WAKING	256
TASK_PARKED	512
TASK_NOLOAD	1024
TASK_NEW	2048
TASK_STATE_MAX	4096

Luego el único argumento del system call `procinfo()` es un entero que indique uno de los estados anteriores. En el anexo se explica con detalle cómo crear estos system calls y cómo incorporarlos al kernel de Linux. Por ahora, basta decir que el laboratorio se probará mediante la creación de un pequeño programa en C que invoque `procinfo()`. El programa debe recibir de línea de comandos el argumento con el estado que se consulta. Por ejemplo, si se desea conocer todos los procesos zombies, ejecutariamos el siguiente comando:

```
$ ./lab1 -s 32
```

El programa debe entonces imprimir por `stdout` la siguiente información para cada proceso en ese estado:

1. Nombre del proceso
2. PID
3. UID
4. Nombre de todos los hijos

Un ejemplo de salida podría ser:

```
Proceso gnome-sesion      pid:1446      uid:1000
Hijo compiz
Hijo nautilus
Hijo nm-applet
Hijo polkit-gnome-au
Hijo telepathy-indic
Proceso bash      pid:2068      uid:1000
Hijo prueba.o
```

Note que el programa `lab1` invoca al syscall `procinfo()`, y es el syscall quien realiza la impresión por pantalla.

El syscall debe detectar cuando el argumento no corresponde a uno de los listados, en cuyo caso debe retornar el valor `-1`. En caso de éxito retorna el mismo número del syscall. Resumiendo, el prototipo del syscall es:

```
int procinfo(int state)
```

III. Fecha de entrega

El laboratorio debe ser entregado en UsachVirtual 2.7 el 5 de mayo, a las 23:55.

Se descontará 1 punto por día de atraso. Copias detectadas serán calificadas con NOTA MÍNIMA.

IV. Detalles de la entrega

El laboratorio puede ser realizado en pareja. El día de la entrega debe subir un archivo comprimido a usachVirtual 2.7, el cual debe tener el formato "rut1-rut2".rar, los archivos que debe incluir son:

1. main.c: archivo de prueba en lenguaje C, que ejecute el syscall.
2. Makefile: para compilar
3. syscall_32.tbl
4. syscall_64.tbl
5. syscalls.h
6. sys.c

La implementación del syscall en sys.c debe estar debidamente comentada e implementada con estructuras básicas del kernel, por lo que solo debe utilizar funciones definidas como una macro de preprocesador. Luego NO puede utilizar funciones como `getpid()`, `find_process_by_pid()`, etc. En esta página pueden encontrar referencias en cuanto a funciones y estructuras del kernel de Linux <http://lxr.free-electrons.com/ident?v=3.16>.

El laboratorio será evaluado en presencia de los estudiantes, quienes traerán el kernel modificado y compilado en un laptop, el cual será booteado con dicho kernel.

V. Tutorial agregar *Syscall*

- Descargue el código fuente de Linux desde el Link <https://www.kernel.org/>, se recomienda la versión 3.16.43, en el sistema Ubuntu 14.04 LTS.

- Descomprimir el archivo linux-3.16.43.tar.xz con el comando:

```
$ sudo tar Jxvf linux-3.16.43.tar.xz
```

- Acceda a la carpeta recién descomprimida y luego ingrese a la carpeta arch/

```
$ cd linux-3.16.43/  
$ cd arch/
```

- Dentro de arch/ acceda a la carpeta x86 e ingrese a la carpeta syscalls/.

```
$ cd x86/  
$ cd syscalls/
```

- Dentro de syscalls/ se encuentran los archivos syscall_32.tbl y syscall_64, los cuales definen las llamadas al sistema, tanto para máquinas de arquitectura de 32 bits como para las de 64 bits.

Proceda a editar los archivos syscall_32.tbl y syscall_64, para ello debe insertar una linea que contenga, el número correlativo de syscalls, la arquitectura, y el nombre de su llamada a sistema.

1. Para el caso de syscall_32.tbl ingrese el comando.

```
$ sudo nano syscall_32.tbl
```

E inserte en el final del archivo la linea correspondiente a su syscall, a continuación se muestra un ejemplo con la syscall sys_hola_mundo.

347	i386	process_vm_readv	sys_process_vm_readv
	compat_sys_process_vm_readv		
348	i386	process_vm_writev	sys_process_vm_writev
	compat_sys_process_vm_writev		
349	i386	kcmp	sys_kcmp
350	i386	finit_module	sys_finit_module
351	i386	sched_setattr	sys_sched_setattr
352	i386	sched_getattr	sys_sched_getattr
353	i386	renameat2	sys_renameat2
354	i386	hola_mundo	sys_hola_mundo

2. Para el caso de syscall_64.tbl ingrese el comando:

```
$ sudo nano syscall_64.tbl
```

E inserte casi al final del archivo (aproximadamente la linea 326), la linea correspondiente a su syscall. A continuación se muestra un ejemplo con la syscall sys_hola_mundo.

313	common	finit_module	sys_finit_module
314	common	sched_setattr	sys_sched_setattr
315	common	sched_getattr	sys_sched_getattr
316	common	renameat2	sys_renameat2
317	64	hola_mundo	sys_hola_mundo
#			
#	x32-specific system call numbers start at 512 to avoid cache impact		
#	for native 64-bit operation.		
#			
512	x32	rt_sigaction	compat_sys_rt_sigaction
513	x32	rt_sigreturn	stub_x32_rt_sigreturn
514	x32	ioctl	compat_sys_ioctl

3. Para agregar su syscall, ingrese a la ruta /linux-3.16.43/include/linux/ y edite el archivo syscalls.h.

```
$ cd /linux-3.16.43/include/linux/  
$ sudo nano syscalls.h
```

Al final de este archivo agregue la cabecera del syscall. A continuación se muestra un ejemplo con el syscall sys_hola_mundo.

```
asmlinkage long sys_kcmp(pid_t pid1, pid_t pid2, int type,  
                         unsigned long idx1, unsigned long idx2);  
asmlinkage long sys_finit_module(int fd, const char __user *uargs, int flags);  
#endif  
asmlinkage int sys_hola_mundo(int arg);
```

- Para agregar el código del syscall ingrese a linux-3.16.43/kernel/ e ingrese al final del archivo sys.c la implementación de su syscall.

```
$ cd /linux-3.16.43/kernel/  
$ sudo nano sys.c
```

A continuación se muestra la implementación del ejemplo sys_hola_mundo:

```
        return -EFAULT;  
  
    return 0;  
}  
#endif /* CONFIG_COMPAT */  
asmlinkage int sys_hola_mundo(int arg){  
    printk(KERN_INFO "HOLA_MUNDO_");  
    return 0;  
}
```

VI. Tutorial Compilar Kernel

- Dentro de la carpeta /linux-3.16.43/ ingrese el siguiente comando:

```
$ make menuconfig
```

Al ejecutarlo aparecerá la ventana de la figura 1, debe seleccionar la opción “General Setup”

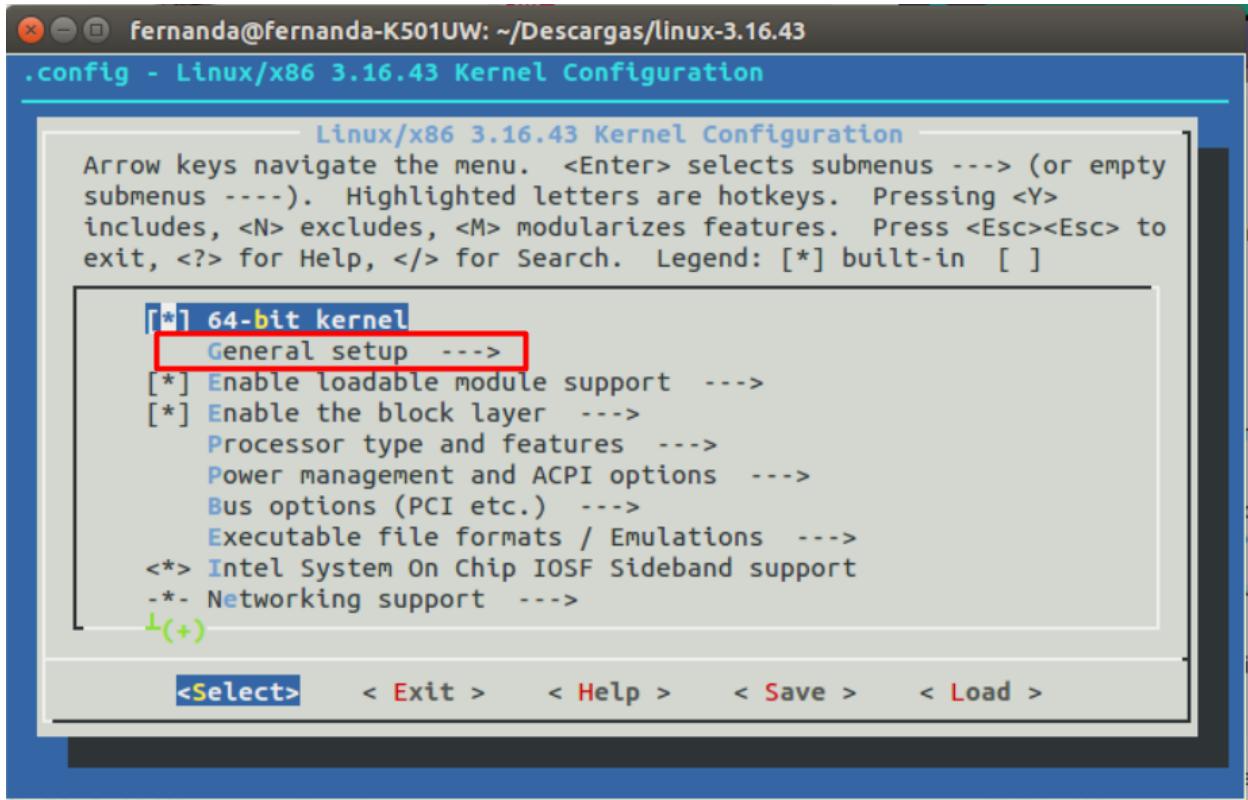


Figure 1. make menuconfig

*ERROR: Unable to find the ncurses libraries

Solución: Ejecutar el siguiente comando:

```
$ sudo apt-get install libncurses-dev
```

- Al seleccionar “General Setup”, aparecerá la ventana de la figura 2, deberá seleccionar la opción “Local version - append to kernel release”.

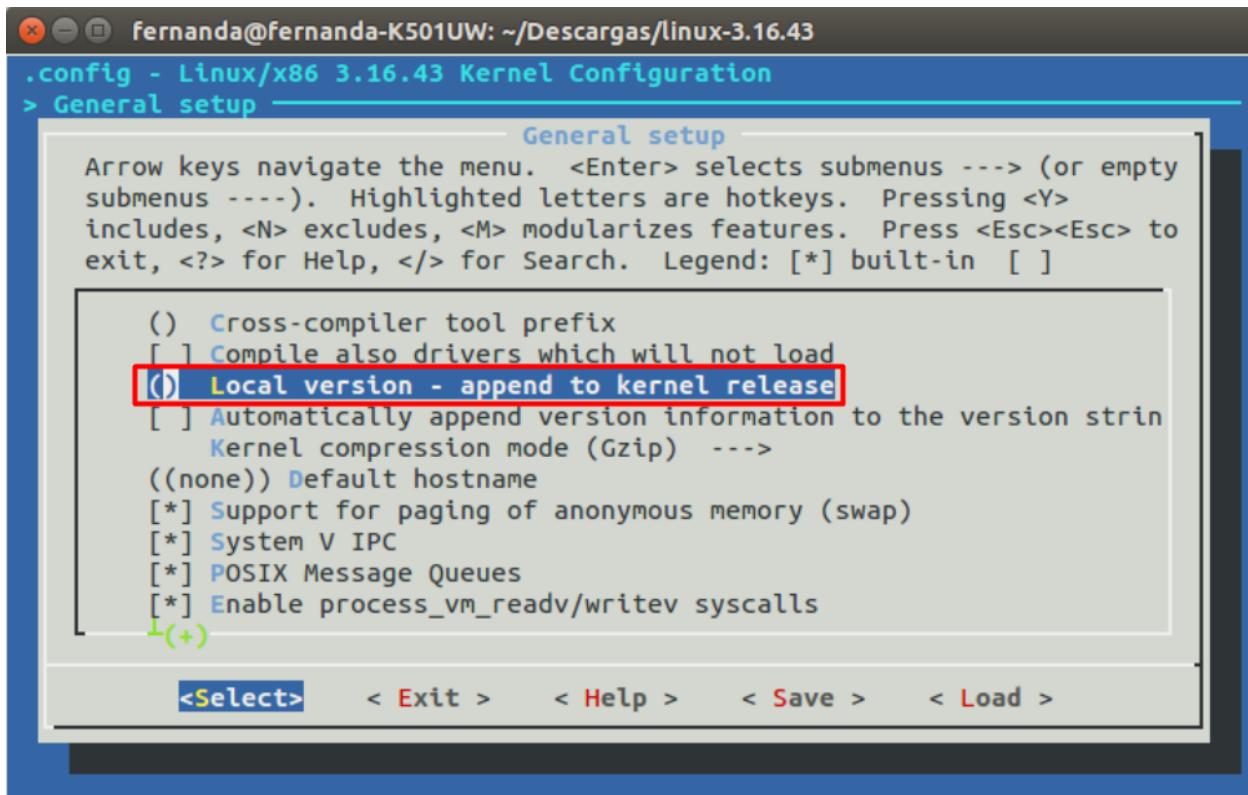


Figure 2. make menuconfig: General Setup

- A continuación se le solicitará un nombre para el kernel, como recomendación no utilice caracteres especiales ni letras mayúsculas, la figura 3 se puede apreciar un ejemplo.

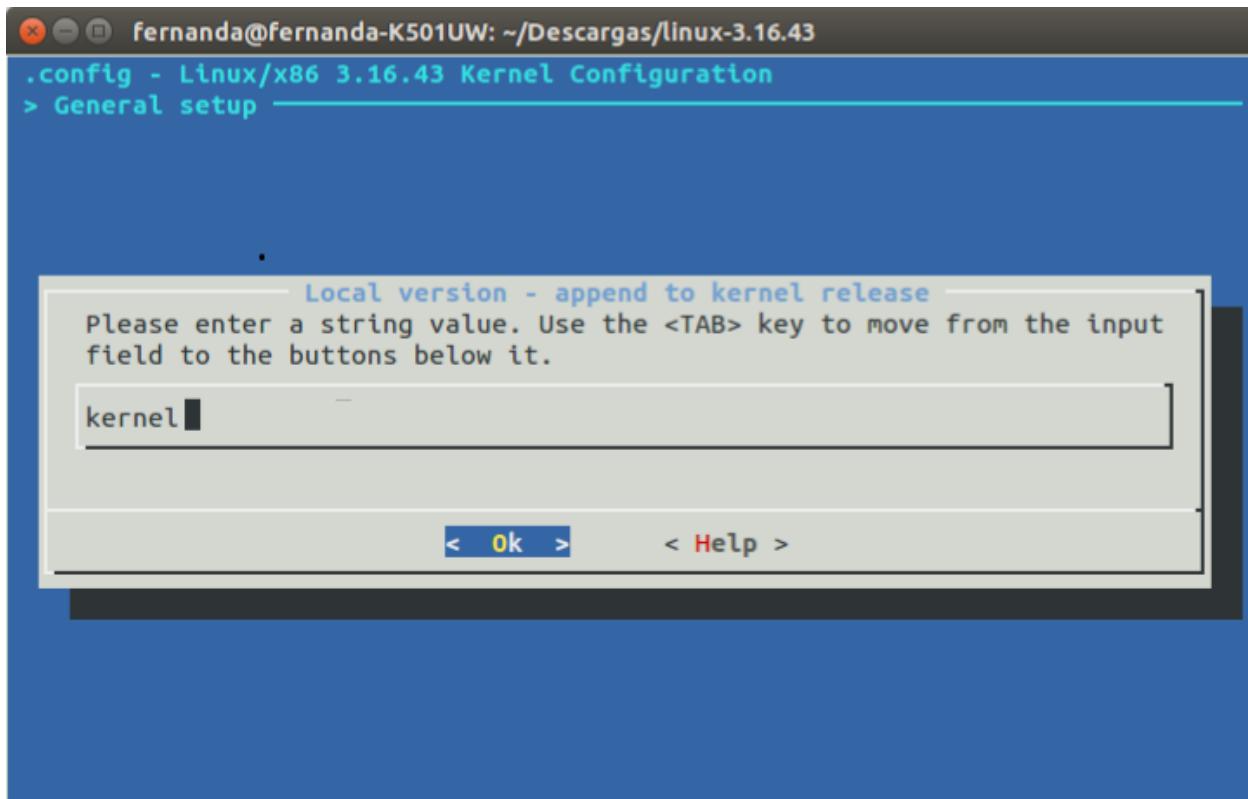


Figure 3. Local Version release

- Luego de presionar “OK”, aparecerá la ventana de la figura 4, debe volver a seleccionar “OK”, luego “EXIT” y “EXIT” nuevamente.

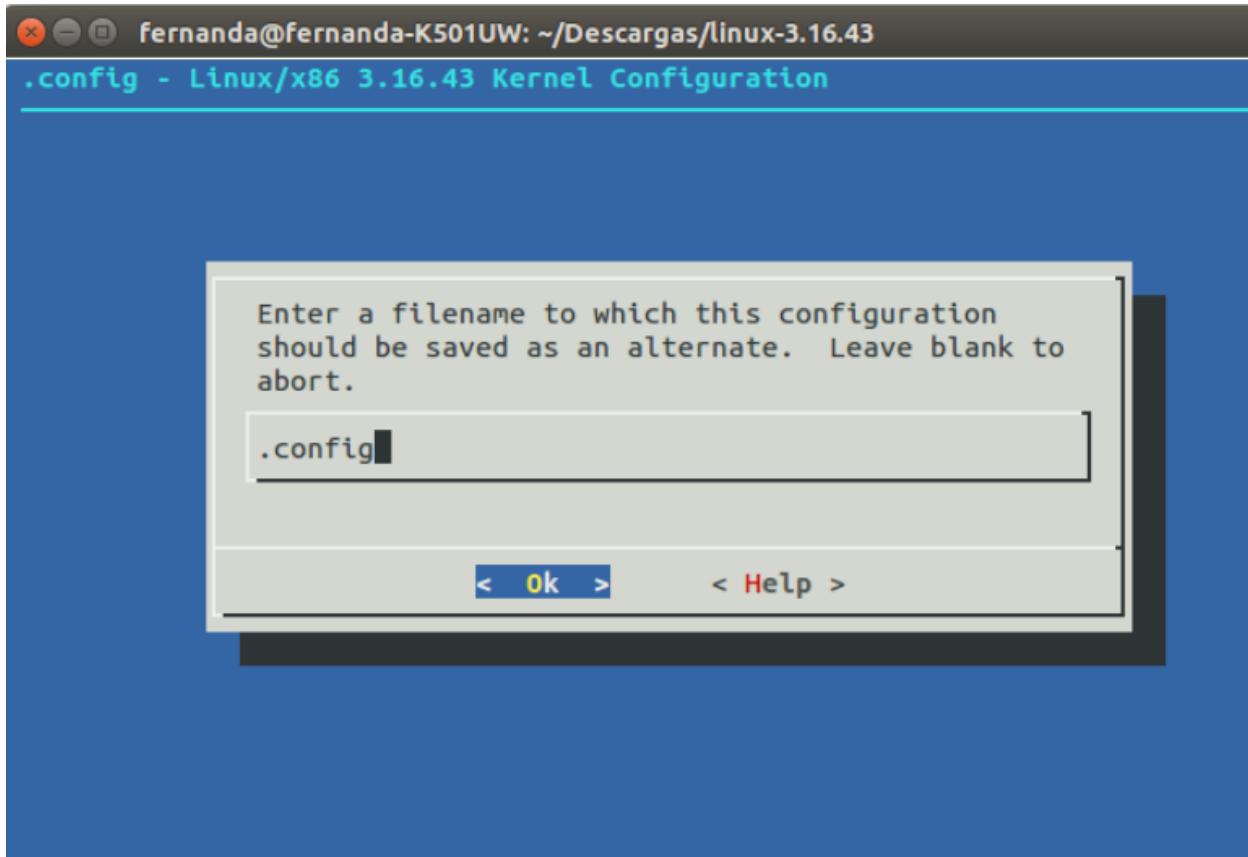


Figure 4. Local Version release: .config

- Al terminar este procedimiento volverá a la consola de Linux. Para compilar el kernel deberá asignar una cantidad de procesadores a la compilación, si no sabe cuantas CPUS tiene su computador ingrese el siguiente comando.

```
$ lscpu
```

Y como se puede apreciar en la figura 5 la línea 4 le dirá la cantidad de CPUS que posee su computador.

```
fernanda@fernanda-K501UW:~$ lscpu
Arquitectura:          x86_64
modo(s) de operación de las CPUs:32-bit, 64-bit
Orden de bytes:        Little Endian
CPU(s):                4
On-line CPU(s) list:  0-3
Hilo(s) de procesamiento por núcleo:2
Núcleo(s) por «socket»:2
Socket(s):              1
Modo(s) NUMA:           1
ID de fabricante:      GenuineIntel
Familia de CPU:         6
Modelo:                 78
Revisión:               3
CPU MHz:                2431.500
BogoMIPS:               4799.89
Virtualización:        VT-x
Caché L1d:               32K
Caché L1i:               32K
Caché L2:                 256K
Caché L3:                 3072K
NUMA node0 CPU(s):      0-3
fernanda@fernanda-K501UW:~$
```

Figure 5. lscpu

- A continuación dentro de linux-3.16.43/ y ejecute el siguiente comando:

```
$ make -j4 deb-pkg
```

En donde -j4 corresponde al número de procesadores de su máquina, revisado anteriormente con “lscpu”.

- Cuando termine la compilación deberá moverse a la carpeta padre de linux-3.16.43/ y verá que se habrán creado alrededor de 5 archivos “.deb”, como los que se pueden apreciar en la figura 6.

```

fernanda@fernanda-K501UW:~/Descargas/linux-3.16.43$ cd ..
fernanda@fernanda-K501UW:~/Descargas$ ls
linux-3.16.43
linux-3.16.43.tar.xz
linux-firmware-image-3.16.43kernel-hola-mundo_3.16.43kernel-hola-mundo-4_amd64.deb
linux-headers-3.16.43kernel-hola-mundo_3.16.43kernel-hola-mundo-4_amd64.deb
linux-image-3.16.43kernel-hola-mundo_3.16.43kernel-hola-mundo-4_amd64.deb
linux-image-3.16.43kernel-hola-mundo-dbg_3.16.43kernel-hola-mundo-4_amd64.deb
linux-libc-dev_3.16.43kernel-hola-mundo-4_amd64.deb

```

Figure 6. Archivos .deb

VII. Tutorial Instalar Kernel

- En la carpeta donde se encuentran los archivos “.deb” ejecute el siguiente comando:

```
$ sudo dpkg-i linux *.deb
```

- Posteriormente debe actualizar su GRUB para que se pueda iniciar desde el kernel instalado, para ello ejecute el siguiente comando:

```
$ sudo update-grub
```

VIII. Tutorial Desinstalar Kernel

- Si debe compilar nuevamente su kernel, se recomienda eliminar la versión anterior, para ello debe iniciar el sistema en su sistema operativo base, es decir, no debe desinstalar el sistema sobre el que se encuentra actualmente. Para ello ejecute el siguiente comando.

```
$ dpkg -l | tail -n +6 | grep -E 'linux-image-[0-9]+'
```

Este comando listará todos los Kernels instalados.

*Si no sabe sobre que Kernel se encuentra actualmente, ejecute el comando:

```
$ uname -r
```

- Finalmente con el siguiente comando eliminamos el kernel.

```
$ sudo dpkg --purge kernel
```

En donde “kernel” corresponde al nombre del kernel que desea eliminar, listado en el primer paso de esta sección.

- Finalmente actualizamos el GRUB del sistema con el comando:

```
$ sudo update-grub
```

- Como recomendación, diríjase a la carpeta donde se encuentran los archivos “.deb” compilados y elimínelos manualmente.

IX. Tutorial probar syscall

- Para probar su syscall, deberá reiniciar el sistema y entrar a las “Opciones avanzadas para Ubuntu”, como se puede apreciar en la figura 7:

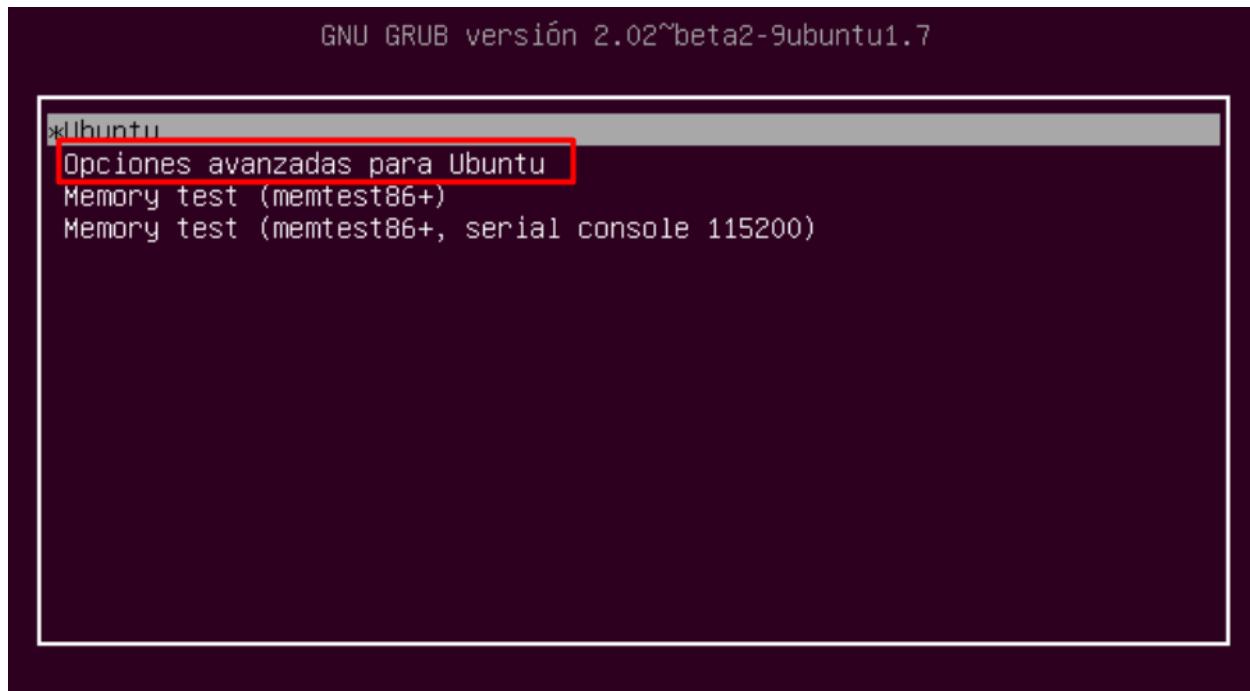


Figure 7. Arranque de Ubuntu

- Posteriormente debe ingresar a la opción que tenga el nombre de su kernel compilado, en la figura 8 se puede apreciar un ejemplo

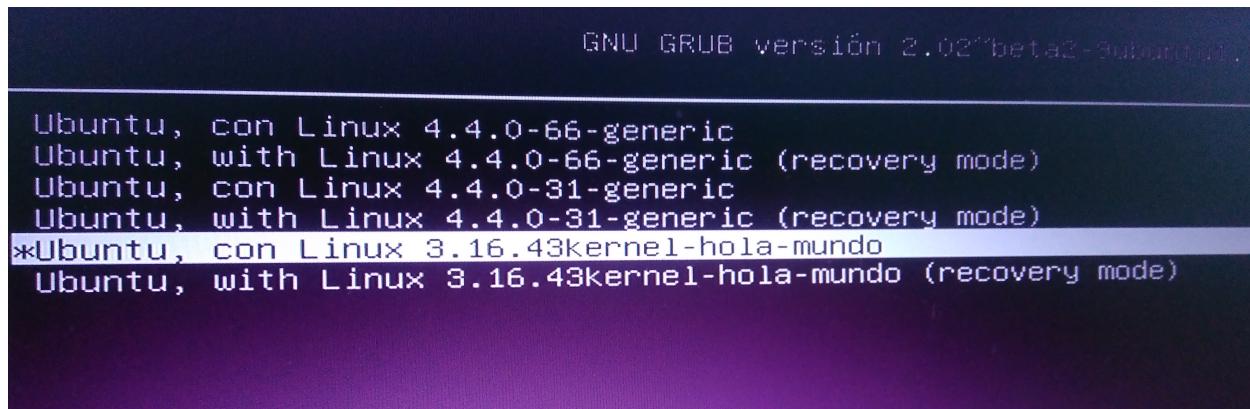


Figure 8. Arranque de Ubuntu

- Una vez dentro del sistema, cree un archivo “.c”, que llame a su syscall, un ejemplo de esto se puede apreciar en el siguiente programa:

prueba.c

```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
#define SYS_hola_mundo_64 317
#define SYS_hola_mundo_32 354

int main(){
    int opcion, arg = 0, salida;
    do{
        printf("1.-_MAQUINA_32_BITS\n2.-MAQUINA_64_BITS");
        scanf("%d", &opcion);
    }while(opcion != 1 && opcion != 2);

    if(opcion == 1){
        salida = syscall(SYS_hola_mundo_32, arg);
    }
    else{
        salida = syscall(SYS_hola_mundo_64, arg);
    }
    printf("salida_%d", salida);
    return 0;
}
```

- posteriormente compile y ejecute su programa, al utilizar en la syscall la función “printk”, los resultados de la llamada al sistema serán visibles luego de ejecutar el comando:

\$ dmesg

En la figura 9 se puede apreciar la ejecución de la llamada al sistema con el programa prueba.c.

```
root@ian-HP-Pavilion-Notebook: /home/ian/Descargas
root@ian-HP-Pavilion-Notebook: /home/ian/Descargas 112x24
hostname!
root@ian-HP-Pavilion-Notebook:/home/ian/Desktop# dmesg > salida.txt
root@ian-HP-Pavilion-Notebook:/home/ian/Desktop# gedit salida.txt
root@ian-HP-Pavilion-Notebook:/home/ian/Desktop# gedit prueba.c

(gedit:2531): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files

(gedit:2531): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files
root@ian-HP-Pavilion-Notebook:/home/ian/Desktop# gcc -o prueba.o prueba.c
root@ian-HP-Pavilion-Notebook:/home/ian/Desktop# ./prueba.o
1.- MAQUINA 32 BITS
2.-MAQUINA 64 BITS1
salida -lroot@ian-HP-Pavilion-Notebook:/home/ian/Desktop# gcc -o prueba.o prueba.c
root@ian-HP-Pavilion-Notebook:/home/ian/Desktop# ./prueba.o
1.- MAQUINA 32 BITS
2.-MAQUINA 64 BITS2
salida 0 root@ian-HP-Pavilion-Notebook:/home/ian/Desktop# dmesg > salida.txt
root@ian-HP-Pavilion-Notebook:/home/ian/Desktop# gedit salida.txt
```

Figure 9. Prueba de syscall

Al final del archivo “salida.txt”, como se muestra en la figura 10, se puede apreciar el resultado correcto de la llamada a sistema.

```
[ 61.147046] sd 4:0:0:0: [sdb] 30728832 512-byte logical blocks: (15.7 GB/14.6 GiB)
[ 61.147164] sd 4:0:0:0: [sdb] Write Protect is off
[ 61.147167] sd 4:0:0:0: [sdb] Mode Sense: 23 00 00 00
[ 61.147289] sd 4:0:0:0: [sdb] No Caching mode page found
[ 61.147291] sd 4:0:0:0: [sdb] Assuming drive cache: write through
[ 61.173526] sdb: sdb1
[ 61.174612] sd 4:0:0:0: [sdb] Attached SCSI removable disk
[ 61.552036] FAT-fs (sdb1): Volume was not properly unmounted. Some data may be corrupted.
[ 63.971241] systemd-hostnamed[2317]: Warning: nss-myhostname is not installed. Change /etc/nsswitch.conf to "hosts: files myhostname" and install nss-myhostname.
[ 120.487049] HOLA MUNDO
[ 223.242726] HOLA MUNDO
```

Figure 10. Resultado prueba de syscall