UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

LABORATORIO Nº2
SISTEMAS OPERATIVOS

Compilación del Kernel de Linux y creación de un Syscall

Profesor: Fernando Rannou **Fecha de entrega:** Sábado 25 de Junio de 2016 (23:59 hrs)

Ayudante: Cristian Jeldes J. cristian.jeldes @usach.cl

OBJETIVOS

Este laboratorio tiene como objetivo que los estudiantes apliquen los conceptos de interrupciones, llamadas al sistema, contexto de usuario y contexto de Kernel, mediante el soporte de un sistema operativo basado en el núcleo de Linux y el lenguaje de programación C. Se busca que los estudiantes sean capaces de compilar, modificar e instalar un Kernel de Linux en una distribución de Linux.

ENUNCIADO

Contexto del laboratorio

En el ámbito de la computación, en el desarrollo de la misma, la evolución de los sistemas operativos a sido fundamental, dando la posibilidad de existencia de diferentes aplicaciones apoyadas en el soporte que el sistema operativo les otorga. Es por lo anterio que el objetivo de este laboratorio es que lo estudiantes aprendan y puedan manipular un Kernel de Linux para que puedan agregarle funcionalidades según se requiera.

Syscall a agregar al Kernel

La funcionalidad que se debe agregar es un syscall que dado el PID de un proceso, entregue el usuario que lo ejecutó, el nombre del programa en ejecución y el estado del proceso como String, además para cada uno de los hijos del proceso debe entregar la misma información, incluyendo a los hijos de los hijos y así sucesivamente (definición recursiva).

Como agregar un Syscall al Kernel

1) Descargue el código fuente del Kernel de Linux de las versiones 3.1 a 3.6 (Son las versiones soportadas por este tutorial, en particular se recomienda hacer este laboratorio en ubuntu 14.04 LTS)
2) El archivo debería llamarse algo así como “linux-3.16.tar.gz”
3) Ocupar el comando tar para descomprimirlo:

```
linux-3.16  linux-3.16.tar.gz
```

4) Entramos a la carpeta y usamos ls –a:

```
.              firmware     MAINTAINERS          signing_key.priv
..             fs           Makefile             signing_key.x509
arch           .gitignore   .missing-syscalls.d  sound
block          include      mm                   .tmp_versions
.config        init         modules.order        tools
.config.old    ipc          net                  usr
COPYING        Kbuild       README               .version
CREDITS        Kconfig      REPORTING-BUGS       virt
crypto         kernel       samples              x509.genkey
Documentation  lib          scripts
drivers        .mailmap     security
```

5) Como el syscall que vamos a agregar es para computadores x86 y x64, nos dirigimos a la carpeta arch, dentro de esta buscamos la carpeta x86 y hacemos ls:

```
boot        Kbuild         lguest          mm             power     video
built-in.o  Kconfig        lib             modules.order  realmode  xen
configs     Kconfig.cpu    Makefile        net            syscalls
crypto      Kconfig.debug  Makefile_32.cpu oprofile       tools
ia32        kernel         Makefile.um     pci            um
include     kvm            math-emu        platform       vdso
```

6) Ahora entramos en la carpeta syscalls y hacemos ls:

```
Makefile  syscall_32.tbl  syscall_64.tbl  syscallhdr.sh  syscalltbl.sh
```

7) En particular los archivos importantes son syscall_32.tbl y syscall_64.tbl, en estos agregaremos un registro del syscall para poder usarlo posteriormente, hacemos nano en syscall_32.tbl, vamos hasta el final del archivo y agregamos una entrada con el entero que sigue a la última entrada existente y le

ponemos el nombre que le vamos a dar a nuestro syscall y lo guardamos:

```
  GNU nano 2.2.6               Archivo: syscall_32.tbl                          Modificado

309     i386    ppoll                   sys_ppoll                       compat_sys_ppoll
310     i386    unshare                 sys_unshare
311     i386    set_robust_list         sys_set_robust_list             compat_sys_set_robus$
312     i386    get_robust_list         sys_get_robust_list             compat_sys_get_robus$
313     i386    splice                  sys_splice
314     i386    sync_file_range         sys_sync_file_range             sys32_sync_file_range
315     i386    tee                     sys_tee
316     i386    vmsplice                sys_vmsplice                    compat_sys_vmsplice
317     i386    move_pages              sys_move_pages                  compat_sys_move_pages
318     i386    getcpu                  sys_getcpu
319     i386    epoll_pwait             sys_epoll_pwait
320     i386    utimensat               sys_utimensat                   compat_sys_utimensat
321     i386    signalfd                sys_signalfd                    compat_sys_signalfd
322     i386    timerfd_create          sys_timerfd_create
323     i386    eventfd                 sys_eventfd
324     i386    fallocate               sys_fallocate                   sys32_fallocate
325     i386    timerfd_settime         sys_timerfd_settime             compat_sys_timerfd_s$
326     i386    timerfd_gettime         sys_timerfd_gettime             compat_sys_timerfd_g$
327     i386    signalfd4               sys_signalfd4                   compat_sys_signalfd4
328     i386    eventfd2                sys_eventfd2
329     i386    epoll_create1           sys_epoll_create1
330     i386    dup3                    sys_dup3
331     i386    pipe2                   sys_pipe2
332     i386    inotify_init1           sys_inotify_init1
333     i386    preadv                  sys_preadv                      compat_sys_preadv
334     i386    pwritev                 sys_pwritev                     compat_sys_pwritev
335     i386    rt_tgsigqueueinfo       sys_rt_tgsigqueueinfo           compat_sys_rt_tgsigq$
336     i386    perf_event_open         sys_perf_event_open
337     i386    recvmmsg                sys_recvmmsg                    compat_sys_recvmmsg
338     i386    fanotify_init           sys_fanotify_init
339     i386    fanotify_mark           sys_fanotify_mark               compat_sys_fanotify_$
340     i386    prlimit64               sys_prlimit64
341     i386    name_to_handle_at       sys_name_to_handle_at
342     i386    open_by_handle_at       sys_open_by_handle_at           compat_sys_open_by_h$
343     i386    clock_adjtime           sys_clock_adjtime               compat_sys_clock_adj$
344     i386    syncfs                  sys_syncfs
345     i386    sendmmsg                sys_sendmmsg                    compat_sys_sendmmsg
346     i386    setns                   sys_setns
347     i386    process_vm_readv        sys_process_vm_readv            compat_sys_process_v$
348     i386    process_vm_writev       sys_process_vm_writev           compat_sys_process_v$
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    taskreader              sys_taskreader



^G Ver ayuda  ^O Guardar    ^R Leer Fich  ^Y RePág.     ^K Cortar Texto^C Pos actual
^X Salir      ^J Justificar ^W Buscar     ^V Pág. Sig.  ^U PegarTxt    ^T Ortografía
```

8) Hacemos lo mismo en syscall_64.tbl:

```
  GNU nano 2.2.6               Archivo: syscall_64.tbl

289     common  signalfd4               sys_signalfd4
290     common  eventfd2                sys_eventfd2
291     common  epoll_create1           sys_epoll_create1
292     common  dup3                    sys_dup3
293     common  pipe2                   sys_pipe2
294     common  inotify_init1           sys_inotify_init1
295     64      preadv                  sys_preadv
296     64      pwritev                 sys_pwritev
297     64      rt_tgsigqueueinfo       sys_rt_tgsigqueueinfo
298     common  perf_event_open         sys_perf_event_open
299     64      recvmmsg                sys_recvmmsg
300     common  fanotify_init           sys_fanotify_init
301     common  fanotify_mark           sys_fanotify_mark
302     common  prlimit64               sys_prlimit64
303     common  name_to_handle_at       sys_name_to_handle_at
304     common  open_by_handle_at       sys_open_by_handle_at
305     common  clock_adjtime           sys_clock_adjtime
306     common  syncfs                  sys_syncfs
307     64      sendmmsg                sys_sendmmsg
308     common  setns                   sys_setns
309     common  getcpu                  sys_getcpu
310     64      process_vm_readv        sys_process_vm_readv
311     64      process_vm_writev       sys_process_vm_writev
312     common  kcmp                    sys_kcmp
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      taskreader              sys_taskreader

#
# 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
515     x32     readv                   compat_sys_readv
516     x32     writev                  compat_sys_writev
517     x32     recvfrom                compat_sys_recvfrom
518     x32     sendmsg                 compat_sys_sendmsg
519     x32     recvmsg                 compat_sys_recvmsg
520     x32     execve                  stub_x32_execve
521     x32     ptrace                  compat_sys_ptrace
522     x32     rt_sigpending           compat_sys_rt_sigpending
523     x32     rt_sigtimedwait         compat_sys_rt_sigtimedwait
524     x32     rt_sigqueueinfo         compat_sys_rt_sigqueueinfo
525     x32     sigaltstack             compat_sys_sigaltstack
                              [ 364 líneas escritas ]
^G Ver ayuda   ^O Guardar     ^R Leer Fich   ^Y RePág.      ^K Cortar Texto^C Pos actual
^X Salir       ^J Justificar  ^W Buscar      ^V Pág. Sig.   ^U PegarTxt    ^T Ortografía
```

9) Lo siguiente que haremos es agregar nuestro prototipo de syscall al header de syscalls de linux, en particular este archivo se encuentra en linux-3.16/include/linux/syscalls.h, vamos hasta el fondo del

archivo y agregamos nuestro syscall:

```
                        int __user *, int);
#else
asmlinkage long sys_clone(unsigned long, unsigned long, int __user *,
              int __user *, int);
#endif
#endif

asmlinkage long sys_execve(const char __user *filename,
                const char __user *const __user *argv,
                const char __user *const __user *envp);

asmlinkage long sys_perf_event_open(
                struct perf_event_attr __user *attr_uptr,
                pid_t pid, int cpu, int group_fd, unsigned long flags);

asmlinkage long sys_mmap_pgoff(unsigned long addr, unsigned long len,
                        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_setns(int fd, int nstype);
asmlinkage long sys_process_vm_readv(pid_t pid,
                                     const struct iovec __user *lvec,
                                     unsigned long liovcnt,
                                     const struct iovec __user *rvec,
                                     unsigned long riovcnt,
                                     unsigned long flags);
asmlinkage long sys_process_vm_writev(pid_t pid,
                                      const struct iovec __user *lvec,
                                      unsigned long liovcnt,
                                      const struct iovec __user *rvec,
                                      unsigned long riovcnt,
                                      unsigned long flags);

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_taskreader(int pid);

                                [ 872 líneas escritas ]
^G Ver ayuda    ^O Guardar      ^R Leer Fich    ^Y RePág.       ^K Cortar Texto ^C Pos actual
^X Salir        ^J Justificar   ^W Buscar       ^V Pág. Sig.    ^U PegarTxt     ^T Ortografía
```

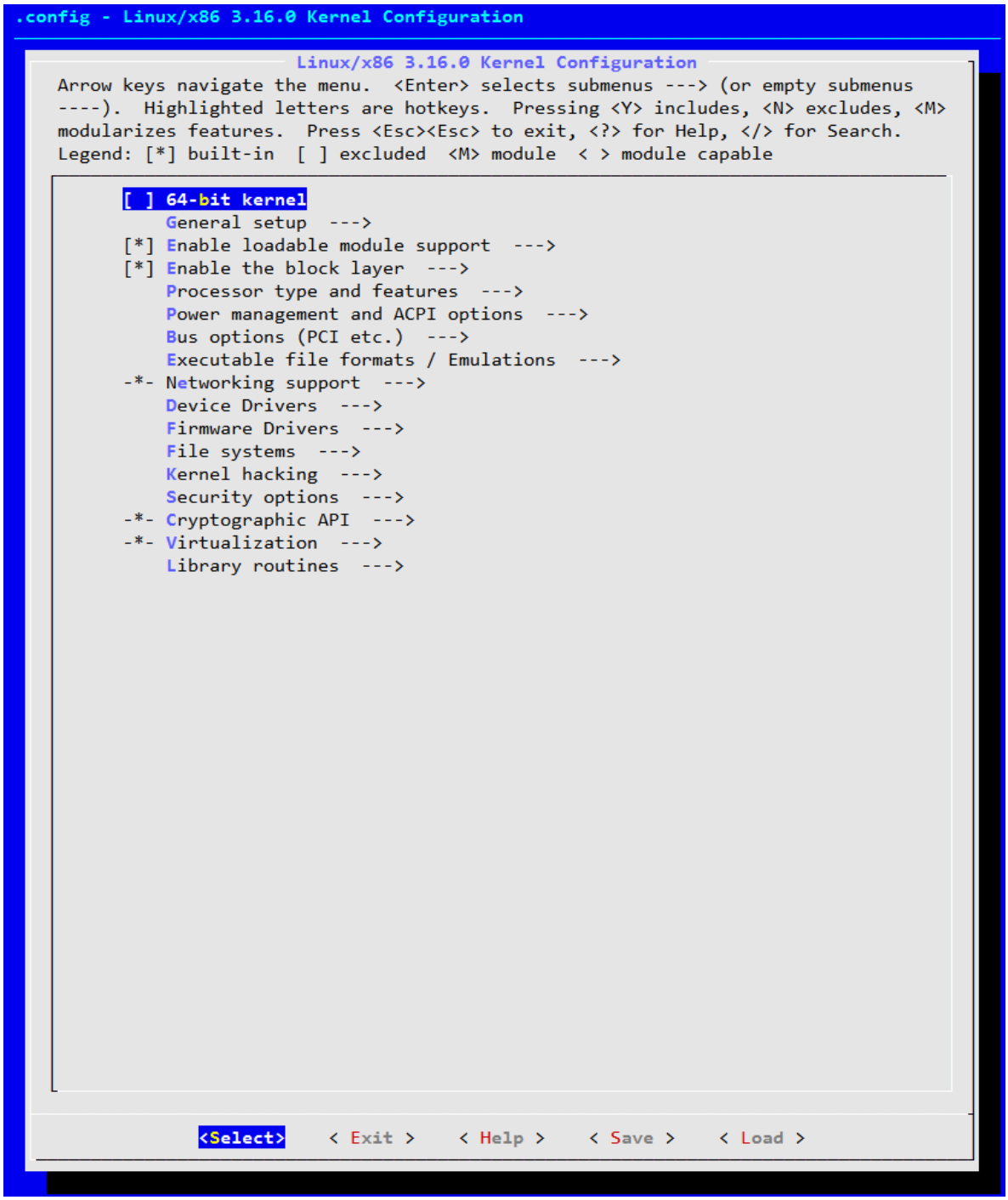
10) Ahora agregaremos nuestro código de syscall al kernel de linux, el archivo que modificaremos es sys.c y se encuentra en kernel/, bajamos hasta el fondo y agregamos nuestro código:

```
asmlinkage int sys_taskreader(int pid){
        //El
        //codigo
        //del
        //Syscall
        return 0;
}
```

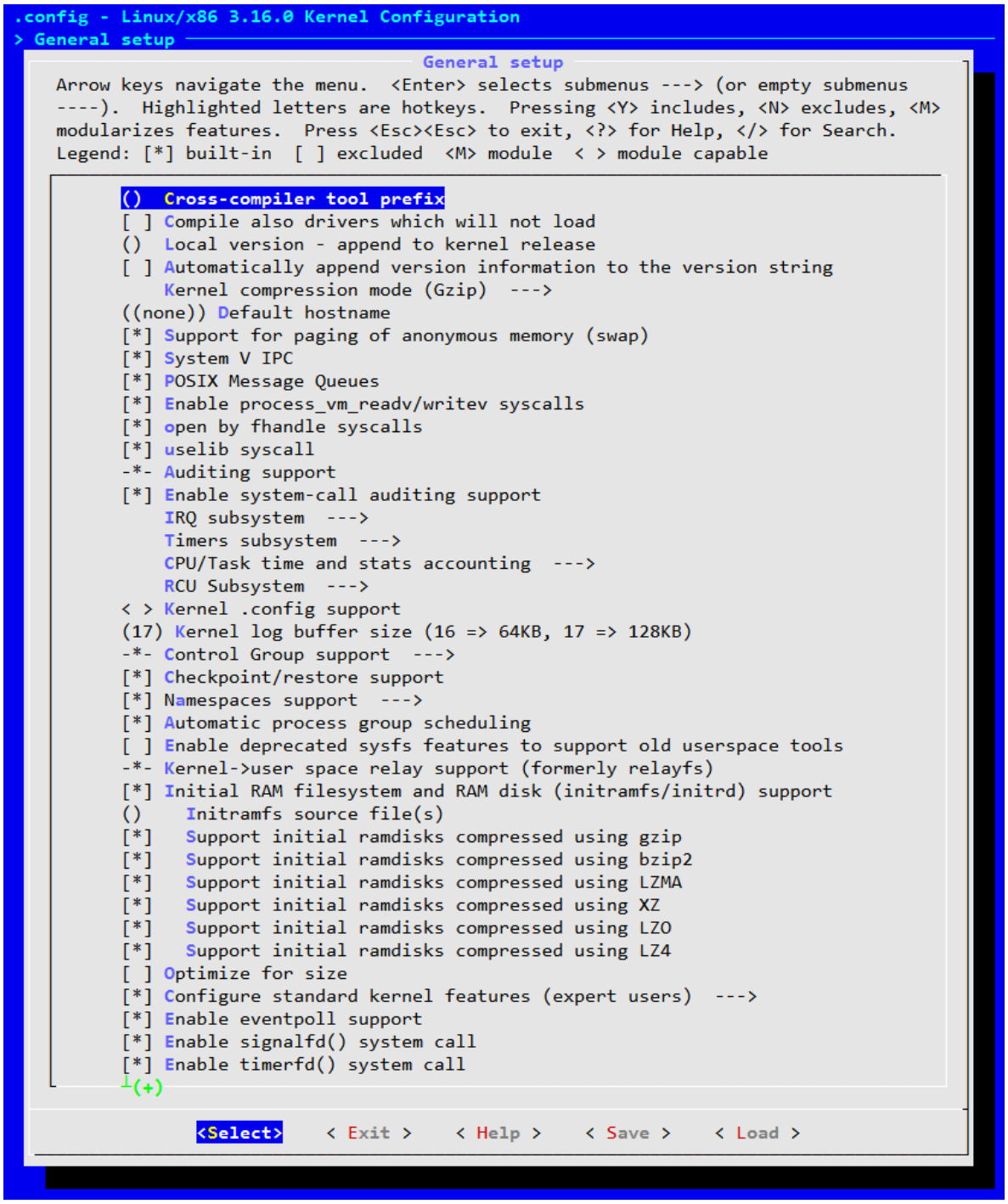
11) Con esto ya agregamos nuestro syscall al código del Kernel

Como compilar un Kernel

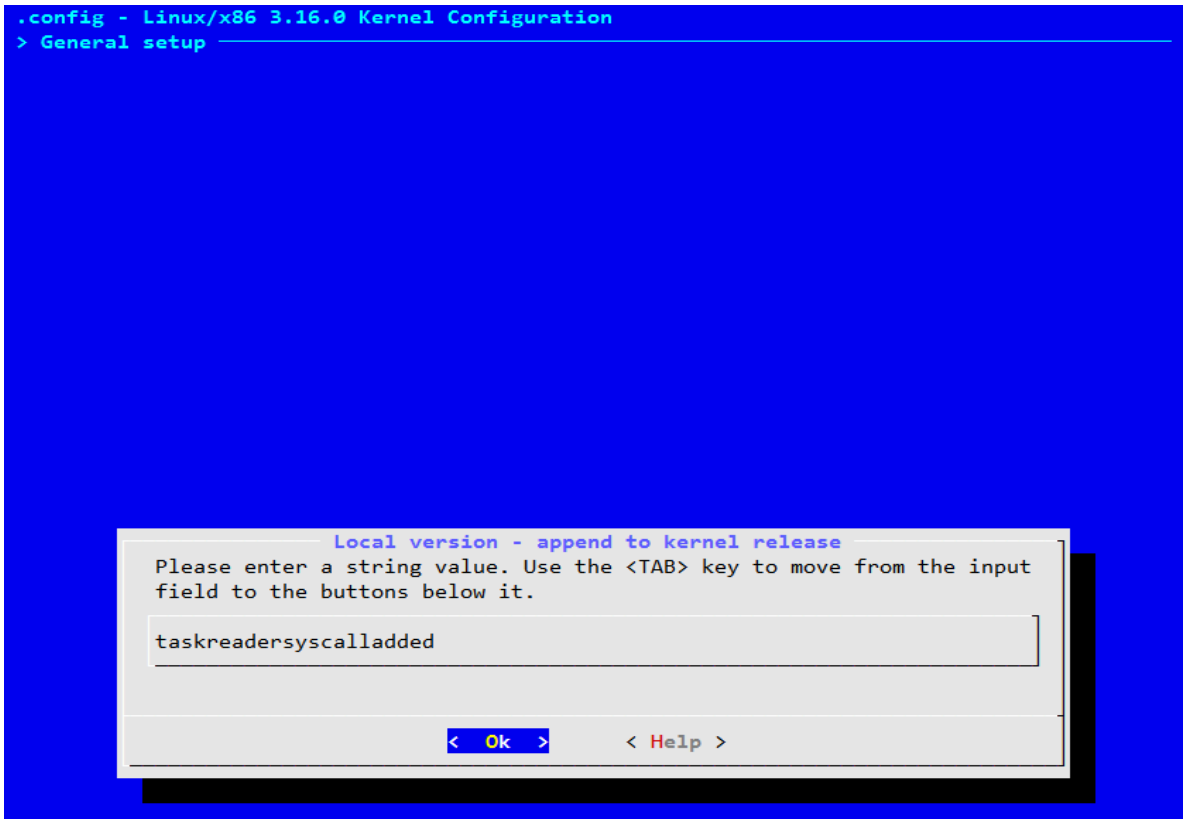
1) Hay que ir a la carpeta donde se encuentra el código fuente y escriba “make menuconfig”:



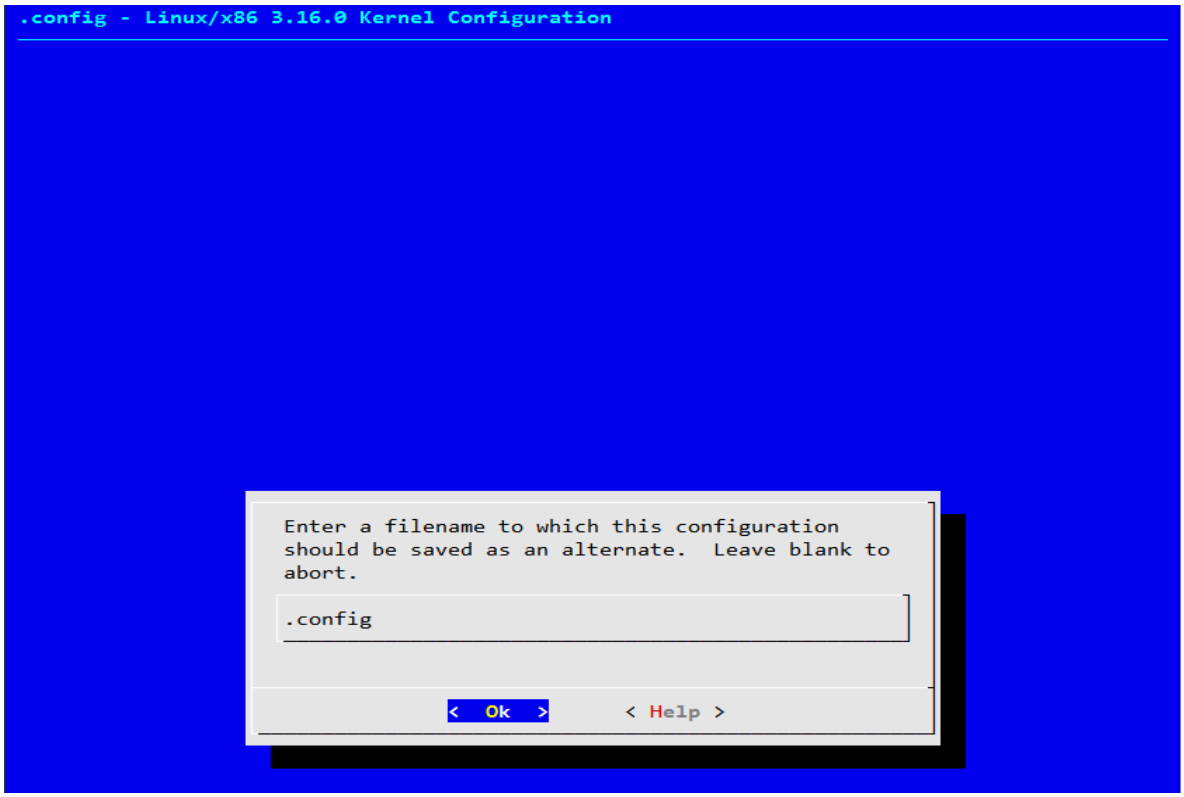
2) Entrar a General Setup --->:



3) Hay que ir donde dice Local Version –apend to kernel realease, y agrega el nombre extra que le dará a su Kernel y ok:



4) Posteriormente apretar save y ok, ok y después exit:



5) El resultado después de esto:

```
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/zconf.tab.o
In file included from scripts/kconfig/zconf.tab.c:2537:0:
scripts/kconfig/menu.c: In function ‘get_symbol_str’:
scripts/kconfig/menu.c:590:18: warning: ‘jump’ may be used uninitialized in this function [-W
maybe-uninitialized]
     jump->offset = strlen(r->s);
                  ^
scripts/kconfig/menu.c:551:19: note: ‘jump’ was declared here
  struct jump_key *jump;
                   ^
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf Kconfig


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```

6) Ahora veremos de cuantos nucleos de procesamiento disponemos para poder compilar nuestro Kernel, escribimos lscpu y donde dice CPU(s): número, es la cantidad de nucleos que podemos ocupar para compilar:

```
Arquitectura:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Orden de bytes:        Little Endian
CPU(s):                24
On-line CPU(s) list:   0-23
Hilo(s) por núcleo:    2
Núcleo(s) por zócalo:6
Socket(s):             2
Nodo(s) NUMA:          2
ID del vendedor:       GenuineIntel
Familia de CPU:        6
Modelo:                62
Model name:            Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz
Stepping:              4
CPU MHz:               1200.000
CPU max MHz:           2600,0000
CPU min MHz:           1200,0000
BogoMIPS:              5190.30
Virtualización:       VT-x
caché L1d:            32K
caché L1i:            32K
caché L2:             256K
caché L3:             15360K
NUMA node0 CPU(s):     0-5,12-17
NUMA node1 CPU(s):     6-11,18-23
```

7) Posteriormente, nos moveremos a la carpeta raíz del código fuente y procederemos a compilar el kernel con el comando “make –jNúmeroDeNucleos deb-pkg”, para el caso de este tutorial sería “make –

j24 deb-pkg”, se recomienda anteponer el comando time para saber cuanto tiempo se demoró en compilar el Kernel, este es un proceso largo y pesado para el computador por tanto es bueno saber cuanto tiempo se va a demorar, más que nada para debugging en caso de que nuestro syscall no haga lo que se espera, es importante notar que para cada prueba del código del syscall se va a tener que re-compilar el Kernel, por lo tanto se recomienda pensar/investigar 80% del tiempo y 20% de tiempo usar-lo en programar, una vez terminada la compilación verá algo similar a esto:

```
  INSTALL debian/headertmp/usr/include/rdma/ (6 files)
  INSTALL debian/headertmp/usr/include/scsi/fc/ (4 files)
  INSTALL debian/headertmp/usr/include/scsi/ (3 files)
  INSTALL debian/headertmp/usr/include/sound/ (11 files)
  INSTALL debian/headertmp/usr/include/video/ (3 files)
  INSTALL debian/headertmp/usr/include/xen/ (4 files)
  INSTALL debian/headertmp/usr/include/uapi/ (0 file)
  INSTALL debian/headertmp/usr/include/asm/ (64 files)
dpkg-gencontrol: aviso: -isp is deprecated; it is without effect
dpkg-deb: construyendo el paquete `linux-firmware-image-3.16.0taskreadersyscalladded' en `../linux-firmw
are-image-3.16.0taskreadersyscalladded_3.16.0taskreadersyscalladded-1_i386.deb'.
dpkg-gencontrol: aviso: -isp is deprecated; it is without effect
dpkg-deb: construyendo el paquete `linux-headers-3.16.0taskreadersyscalladded' en `../linux-headers-3.16
.0taskreadersyscalladded_3.16.0taskreadersyscalladded-1_i386.deb'.
dpkg-gencontrol: aviso: -isp is deprecated; it is without effect
dpkg-deb: construyendo el paquete `linux-libc-dev' en `../linux-libc-dev_3.16.0taskreadersyscalladded-1_
i386.deb'.
dpkg-gencontrol: aviso: -isp is deprecated; it is without effect
dpkg-deb: construyendo el paquete `linux-image-3.16.0taskreadersyscalladded' en `../linux-image-3.16.0ta
skreadersyscalladded_3.16.0taskreadersyscalladded-1_i386.deb'.

dpkg-gencontrol: aviso: -isp is deprecated; it is without effect
dpkg-deb: construyendo el paquete `linux-image-3.16.0taskreadersyscalladded-dbg' en `../linux-image-3.16
.0taskreadersyscalladded-dbg_3.16.0taskreadersyscalladded-1_i386.deb'.

real    26m13.234s
user    45m42.504s
sys     6m58.199s
```

8) Ahora vamos a la carpeta anterior a la raíz osea “cd ..” y hacemos ls:

```
linux-3.16
linux-3.16.tar.gz
linux-firmware-image-3.16.0taskreadersyscalladded_3.16.0taskreadersyscalladded-1_i386.deb
linux-headers-3.16.0taskreadersyscalladded_3.16.0taskreadersyscalladded-1_i386.deb
linux-image-3.16.0taskreadersyscalladded_3.16.0taskreadersyscalladded-1_i386.deb
linux-image-3.16.0taskreadersyscalladded-dbg_3.16.0taskreadersyscalladded-1_i386.deb
linux-libc-dev_3.16.0taskreadersyscalladded-1_i386.deb
```

Aquí está nuestro Kernel compilado

Como instalar y desinstalar un Kernel

Instalar

1) Una vez tenemos nuestro Kernel compilado vamos a la carpeta donde están nuestros .deb y ejecutamos “sudo dpkg –i linux*.deb”:

```
cjeldes@ubuntu:~/LAB1SO2016$ sudo dpkg -i linux*.deb
Seleccionando el paquete linux-firmware-image-3.16.0taskreadersyscalladded previamente no seleccionado.
(Leyendo la base de datos ... 209905 ficheros o directorios instalados actualmente.)
Preparing to unpack linux-firmware-image-3.16.0taskreadersyscalladded_3.16.0taskreadersyscalladded-1_i386.deb ...
Unpacking linux-firmware-image-3.16.0taskreadersyscalladded (3.16.0taskreadersyscalladded-1) ...
Seleccionando el paquete linux-headers-3.16.0taskreadersyscalladded previamente no seleccionado.
Preparing to unpack linux-headers-3.16.0taskreadersyscalladded_3.16.0taskreadersyscalladded-1_i386.deb ...
Unpacking linux-headers-3.16.0taskreadersyscalladded (3.16.0taskreadersyscalladded-1) ...
Seleccionando el paquete linux-image-3.16.0taskreadersyscalladded previamente no seleccionado.
Preparing to unpack linux-image-3.16.0taskreadersyscalladded_3.16.0taskreadersyscalladded-1_i386.deb ...
Unpacking linux-image-3.16.0taskreadersyscalladded (3.16.0taskreadersyscalladded-1) ...
Seleccionando el paquete linux-image-3.16.0taskreadersyscalladded-dbg previamente no seleccionado.
Preparing to unpack linux-image-3.16.0taskreadersyscalladded-dbg_3.16.0taskreadersyscalladded-1_i386.deb ...
Unpacking linux-image-3.16.0taskreadersyscalladded-dbg (3.16.0taskreadersyscalladded-1) ...
dpkg: aviso: desactualizando linux-libc-dev de 3.16.0.syscalltestholamundo2-16 a 3.16.0taskreadersyscalladded-1
Preparing to unpack linux-libc-dev_3.16.0taskreadersyscalladded-1_i386.deb ...
Unpacking linux-libc-dev (3.16.0taskreadersyscalladded-1) over (3.16.0.syscalltestholamundo2-16) ...
Configurando linux-firmware-image-3.16.0taskreadersyscalladded (3.16.0taskreadersyscalladded-1) ...
Configurando linux-headers-3.16.0taskreadersyscalladded (3.16.0taskreadersyscalladded-1) ...
Configurando linux-image-3.16.0taskreadersyscalladded (3.16.0taskreadersyscalladded-1) ...
update-initramfs: Generating /boot/initrd.img-3.16.0taskreadersyscalladded
Generando archivo de configuración grub...
Aviso: Ya no se permite establecer GRUB_TIMEOUT a un valor distinto de cero cuando GRUB_HIDDEN_TIMEOUT está activado.
Se encontró una imagen linux: /boot/vmlinuz-3.16.0-30-generic
Se encontró una imagen initrd: /boot/initrd.img-3.16.0-30-generic
Se encontró una imagen linux: /boot/vmlinuz-3.16.0taskreadersyscalladded
Se encontró una imagen initrd: /boot/initrd.img-3.16.0taskreadersyscalladded
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
hecho
Configurando linux-image-3.16.0taskreadersyscalladded-dbg (3.16.0taskreadersyscalladded-1) ...
Configurando linux-libc-dev (3.16.0taskreadersyscalladded-1) ...
```

2) Una vez hecho esto, tenemos instalado nuestro kernel, ahora para poder iniciar el sistema con el, usaremos grub, que es un gestor de arranques múltiples y nos permitirá iniciar con nuestro nuevo kernel
3) Si no tenemos instalado grub hacemos “sudo apt-get install grub2”
4) Para actualizar nuestros puntos de arranque del sistema operativo usamos “sudo update-grub” y con esto, una vez que reiniciemos el computador podremos entrar con nuestro nuevo Kernel

```
cjeldes@ubuntu:~/LAB1SO2016$ sudo update-grub
Generando archivo de configuración grub...
Aviso: Ya no se permite establecer GRUB_TIMEOUT a un valor distinto de cero cuando GRUB_HIDDEN_TIMEOUT está activado.
Se encontró una imagen linux: /boot/vmlinuz-3.16.0-30-generic
Se encontró una imagen initrd: /boot/initrd.img-3.16.0-30-generic
Se encontró una imagen linux: /boot/vmlinuz-3.16.0taskreadersyscalladded
Se encontró una imagen initrd: /boot/initrd.img-3.16.0taskreadersyscalladded
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
hecho
```

Desinstalar

1) Puede que necesitemos desintalar el kernel que hemos instalado, para esto usaremos el comando “dpkg -l | tail -n +6 | grep -E 'linux-image-[0-9]+’” Para saber que kernels tenemos instalados
2) Para cada kernel que queramos desinstalar usamo “sudo dpkg –P ‘kernel a liminar’”, es muy importante tener en cuenta que no debemos desintalar el kernel sobre el que estamos trabajando, para saber sobre que kernel estamos trabajando podemos usar “uname –r”

```
cjeldes@ubuntu:~$ dpkg -l | tail -n +6 | grep -E 'linux-image-[0-9]+'
ii  linux-image-3.16.0-30-generic                       3.16.0-30.40~14.04.1
i386         Linux kernel image for version 3.16.0 on 32 bit x86 SMP
ii  linux-image-3.16.0.syscalltestholamundo2            3.16.0.syscalltestholamundo2-16
i386         Linux kernel, version 3.16.0.syscalltestholamundo2
ii  linux-image-3.16.0.syscalltestholamundo2-dbg        3.16.0.syscalltestholamundo2-16
i386         Linux kernel debugging symbols for 3.16.0.syscalltestholamundo2
cjeldes@ubuntu:~$ uname -r
3.16.0-30-generic
cjeldes@ubuntu:~$
```

3) Entonces eliminamos los kernel que queramos eliminar, en este caso será el Kernel “linux-image-3.16.0.syscalltestholamundo2” y “linux-image-3.16.0.syscalltestholamundo-dbg”, para eso hacemos “sudo dpkg -P ‘linux-image-3.16.0.syscalltestholamundo2’”

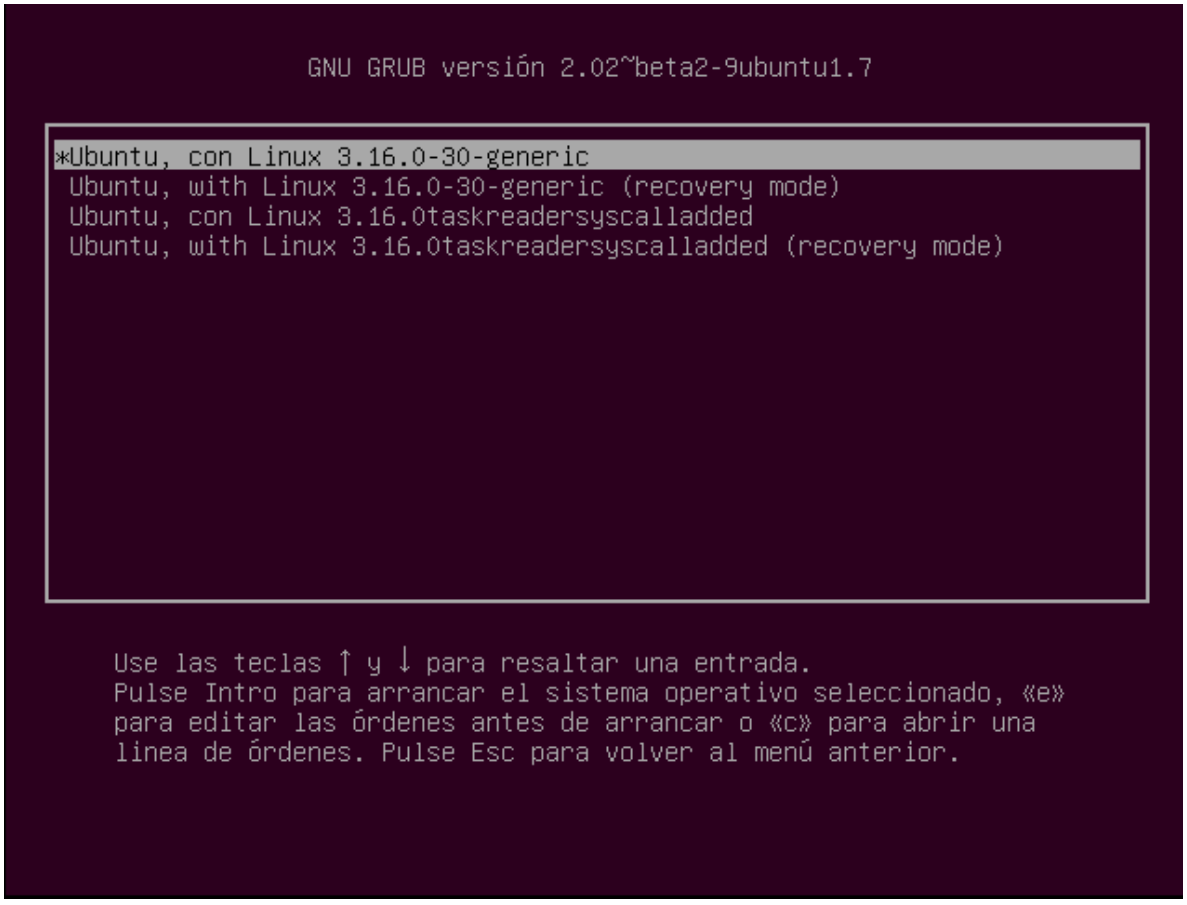
```
cjeldes@ubuntu:~$ sudo dpkg -P linux-image-3.16.0.syscalltestholamundo2
(Leyendo la base de datos ... 220262 ficheros o directorios instalados actualmente.)
Removing linux-image-3.16.0.syscalltestholamundo2 (3.16.0.syscalltestholamundo2-16) ...
update-initramfs: Deleting /boot/initrd.img-3.16.0.syscalltestholamundo2
Generando archivo de configuración grub...
Aviso: Ya no se permite establecer GRUB_TIMEOUT a un valor distinto de cero cuando GRUB_HIDDEN_TIMEOUT está activado.
Se encontró una imagen linux: /boot/vmlinuz-3.16.0-30-generic
Se encontró una imagen initrd: /boot/initrd.img-3.16.0-30-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
hecho
Purging configuration files for linux-image-3.16.0.syscalltestholamundo2 (3.16.0.syscalltestholamundo2-16) ...
cjeldes@ubuntu:~$ sudo dpkg -P linux-image-3.16.0.syscalltestholamundo2-dbg
(Leyendo la base de datos ... 209913 ficheros o directorios instalados actualmente.)
Removing linux-image-3.16.0.syscalltestholamundo2-dbg (3.16.0.syscalltestholamundo2-16) ...
```

4) Una vez hecho esto ejecutamos “sudo update-grub” para actualizar los kernel disponibles para partir l sistema

```
cjeldes@ubuntu:~$ sudo update-grub
Generando archivo de configuración grub...
Aviso: Ya no se permite establecer GRUB_TIMEOUT a un valor distinto de cero cuando GRUB_HIDDEN_TIMEOUT está activado.
Se encontró una imagen linux: /boot/vmlinuz-3.16.0-30-generic
Se encontró una imagen initrd: /boot/initrd.img-3.16.0-30-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
hecho
```

Como pruebo el Syscall creado

1) Debemos iniciar el sistema con nuestro nuevo Kernel a través de grub, vamos a “Opciones avanzadas para ubuntu” e iniciamos con nuestro nuevo syscall


GNU GRUB versión 2.02~beta2-9ubuntu1.7
*Ubuntu, con Linux 3.16.0-30-generic
Ubuntu, with Linux 3.16.0-30-generic (recovery mode)
Ubuntu, con Linux 3.16.0taskreadersyscalladded
Ubuntu, with Linux 3.16.0taskreadersyscalladded (recovery mode)
Use las teclas ↑ y ↓ para resaltar una entrada.
Pulse Intro para arrancar el sistema operativo seleccionado, «e»
para editar las órdenes antes de arrancar o «c» para abrir una
linea de órdenes. Pulse Esc para volver al menú anterior.


GNU GRUB versión 2.02~beta2-9ubuntu1.7
Ubuntu, con Linux 3.16.0-30-generic
Ubuntu, with Linux 3.16.0-30-generic (recovery mode)
*Ubuntu, con Linux 3.16.0taskreadersyscalladded
Ubuntu, with Linux 3.16.0taskreadersyscalladded (recovery mode)
Use las teclas ↑ y ↓ para resaltar una entrada.
Pulse Intro para arrancar el sistema operativo seleccionado, «e»
para editar las órdenes antes de arrancar o «c» para abrir una
linea de órdenes. Pulse Esc para volver al menú anterior.

2) Debemos crear un programa en C que use nuestro syscall

Que es lo que usted debe entregar como resultado de este laboratorio

Usted para este laboratorio debe entregar lo siguiente en un archivo comprimido:

- Archivo Sys_64.tbl
- Archivo Sys_32.tbl
- Código de su Syscall
- Código de ejecución de su Syscall
- Sys.c
- Syscalls.h
- Informe en PDF con formato tesis

El informe debe contar con Portada, Indice, Introduccion, Marco Teórico, Desarrollo y Conclusiones, todo con sus respectivas referencias. Además en el capítulo de desarrollo tiene que explicar paso a paso su código tanto de ejecución de Syscall como el de Syscall, además de un paso a paso detallado de cómo integró su Syscall al Kernel y posteriormente se debe hacer un análisis explicativo de cada uno de los archivos que modificó para llevar a cabo este laboratorio.

Preguntas Frecuentes

¿Cuánto tiempo tarda en promedio compilar un Kernel?

R: En un servidor con 24 nucleos alrdedor de 7-15 min y en un pc con procesador i5 3XXX alrededor de 30-60 minutos. Por favor tome este laboratorio con calma y sabiduría, empiece a probar desde la entrega de este, sino, no tendrá tiempo de terminarlo.

Con el paso a paso entregado en este documento, ¿Existe posibilidad de tener problemas con mi computador?

R: No, los archivos que se modifican no tienen impacto en la estabilidad del sistema operativo, siempre y cuando se sigan las instrucciones. De todas formas este paso a paso fue probado en cuatro computadores distintos sin causar ningún problema.

Detalles sobre la revisión

- Su código de Syscall será probado en una máquina virtual con Ubuntu 14.04 LTS, para verificar su correcto funcionamiento.
- El informe tendrá dos fases de revisión, una interna y otra presencial donde se le pedirá que lleve su computador para revisar el proceso que siguió en el informe presentado. Se espera para la revisión presencial que lleve compilado el Kernel resultante de su laboratorio.
- Cualquier copia será evaluada con un 1 para ambas partes, incluso las sospechas de copia. Evidentemente existe derecho a reclamo una vez entregadas las notas, pero es deber del estudiante buscar al ayudante para hacer este reclamo.

Detalles de la entrega

- Este laboratorio debe ser entregado el Sabado 25 de Junio a las 23:59 en el moodle, con el nombre “Laboratorio2-SO-<RUT>-<Nombre Apellido>”.
- Se debe entregar un solo archivo comprimido con extensión .tar.gz con todos los archivos que corresponden al laboratorio.
- Por cada día de atraso se descontará 1 punto, con 3 días de atraso la nota será un 1.0.
- El programa será revisado en la distribución de Linux Ubuntu versión 14.04 LTS de 32 bit.
- El proceso de compilación del programa para ejecutar el Syscall debe ser realizado mediante un MakeFile que deberá compilar el programa.
- Debe crear un archivo de texto README con instrucciones de compilación y ejecución.