**CONTENTS**

## 1. Introduction:

Through this project, we have tried to add a new system call to the linux kernel 4.10.10 on Ubuntu 16.04. This requires working as the root user, which can be done by the command: **"sudo -s"** as modifications involve kernel programming and processing. The implementation has to be done very carefully as any other changes done by accident could cause the operating system to crash.
We have chosen the Linux Operating System for this project as the functionality provided by it is simpler than others and is open source i.e freely available online.

## 2. Design and Implementation

### 2.1 System Requirements

- Ubuntu 16.04 – 64 bit
- Latest stable kernel version – In this case, it is 4.10.10

### 2.2 Module Description

Adding a new system call is not just adding a new module, as changes have to be made in the already existing files in the kernel. Adding a new module means adding one more functionality (for example, device drivers). But adding a new system call needs changes to be made in the kernel.

### 2.3 Implementation

The following steps were followed for the implementation of the project:

**Step 1:** Download the kernel source (downloaded in Downloads folder by default) and extract in /usr/src and change into the directory. Then we run all the other commands as root user.

```
$ cd Downloads/
$ sudo tar -xvf linux-4.10.10.tar.xz -C /usr/src/
$ cd /usr/src/linux-4.10.10/
$ sudo -s
```

**Step 2:** Now, we define the new syscall (sys_hello()).

Make a new directory called ‘hello’ under the kernel directory and create the source file for syscall.

```
# mkdir hello
# cd hello
# gedit hello.c
```

Add the following code to hello.c:

```
#include <linux/kernel.h>
asmlinkage long sys_hello(void) {
 printk(“Message to displayed”); //printk prints to the kernel’s log file.
 return 0;
}
```

**Step 3:** Create a “Makefile” in the hello folder

```
# gedit Makefile
```

Add the following line to it:-

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

This is to ensure that the hello.c file is compiled and included in the kernel source code.

**Step 4:** Change back into the linux-4.10.10 folder and open Makefile

```
# gedit Makefile
```

Go to the line which says :- “*core-y += kernel/certs/ mm/ fs/ ipc/ security/ crypto/ block/* ”

change this to “*core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ hello/*”

This is to tell the compiler that the source files of our new system call (sys_hello()) are in present in the hello directory.

**Step 5:** Now, we’ll have to alter the **syscall_64.tbl.** A neat way to figure out where this file is present is to use the ‘**find**’ command on the terminal from the linux-4.10.10 directory.

```
# find -name syscall_64.tbl
```

This is present in **/usr/src/linux-4.10.10/arch/x86/entry/syscalls/syscall_64.tbl**

Now, we need to edit the file to include the new system call number and its entry point: (Note the system call number for reference)

Before modification the file looked like this:

```
330   common      pkey_alloc         sys_pkey_alloc
331   common      pkey_free          sys_pkey_free
          <add your new system call here>
#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
```

The line to be added is:

```
332   common      hello              sys_hello
```

**Step 6:** Now we have to alter the **syscalls.h** file. Again, use **'find'** look for where the syscalls.h file is present.

```
# find -name syscalls.h
```

This is present in **/usr/src/linux-4.10.10/include/linux/**

Add the following line to the end of the file before the #endif:

```
asmlinkage long sys_hello(void)
```

**Step 7:** Configure, Recompile and Reboot

To integrate the system call and to be able to actually use it, we'll need to recompile the kernel.

```
# make menuconfig
```

The above command is used to configure the Linux kernel. Once you execute the command, you will get a pop up window with the list of menus and you can select the items for the new configuration. If you are unfamiliar with the configuration just check for the file systems menu and check whether ext4 is chosen or not, if not select it and save the configuration.

```
# make
# make modules_install
# make install
```

Once this is done restart the system.

**Step 8:** Testing the system call

To test the system call write a simple 'sysHello.c' function

```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
int main(int argc, char *argv[])
{
        long int a = syscall(332);
        printf("System call returned %ld\n", a);
        return 0;
}
```

This should be compiled and then executed.

```
$ cc sysHello.c
$./a.out
```

This should print 0

The message is written to the kernel log. To view this, we use the command:

```
$ dmesg
```

Message to be displayed

**Note:** In case there is a problem and the desired output is not obtained, follow these steps as root:

**Step 9**: $ sudo -s

```
# make menuconfig
```

After above command a pop up window will come up. Make sure that ext4 is selected and then save.

**Step 10**: Then create DEB file from new kernel:

```
# make KDEB_PKGVERSION=1.arbitrary-name deb-pkg
```

It will create some deb files in /usr/src/

**Step 11**: After that we need to install them:

```
# dpkg -i linux*.deb
```

**Step 12:** $ uname -r

This should display 4.10.10

```
$ cat /proc/kallsyms | grep <system call name>
```

In our case :

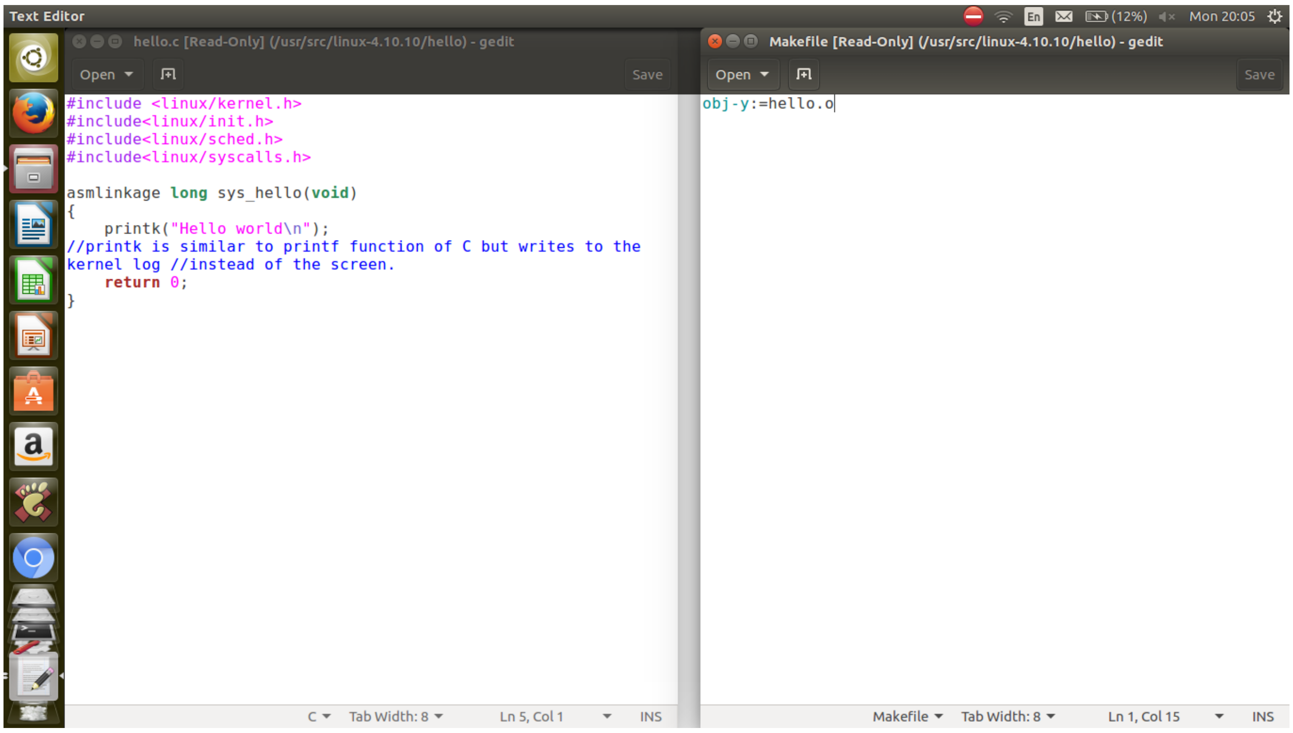
```
$ cat /proc/kallsyms | grep hello
```

Following output indicates that your System Call successfully added to the Kernel :

```
00000000 T sys_hello
```

**Step 13:** Check the “system.map” to see if your addition was included in the kernel.

## 3. Results

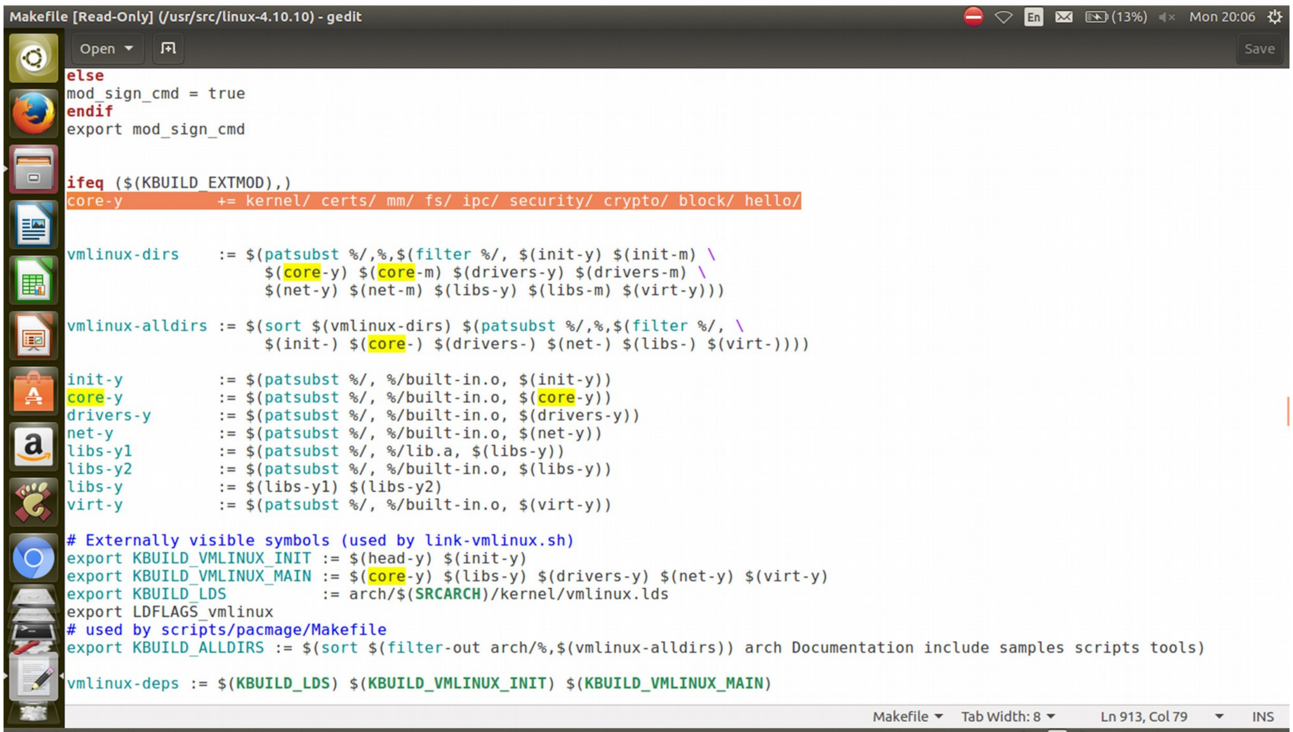


Text Editor
Mon 20:05
hello.c [Read-Only] (/usr/src/linux-4.10.10/hello) - gedit
Open
Save
Makefile [Read-Only] (/usr/src/linux-4.10.10/hello) - gedit
C
Tab Width: 8
Ln 5, Col 1
INS
Makefile
Ln 1, Col 15

```c
#include <linux/kernel.h>
#include<linux/init.h>
#include<linux/sched.h>
#include<linux/syscalls.h>

asmlinkage long sys_hello(void)
{
    printk("Hello world\n");
//printk is similar to printf function of C but writes to the
kernel log //instead of the screen.
    return 0;
}
```

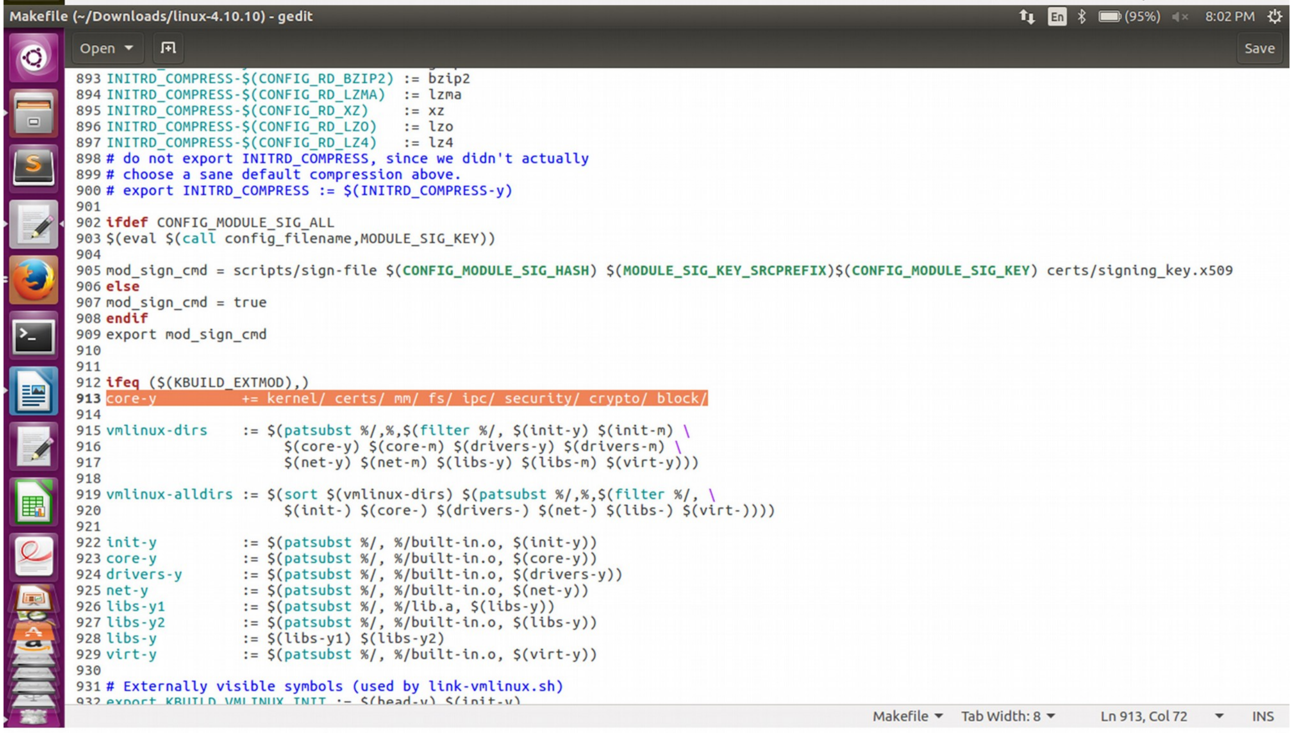
```makefile
obj-y:=hello.o
```

Fig1



Makefile [Read-Only] (/usr/src/linux-4.10.10) - gedit

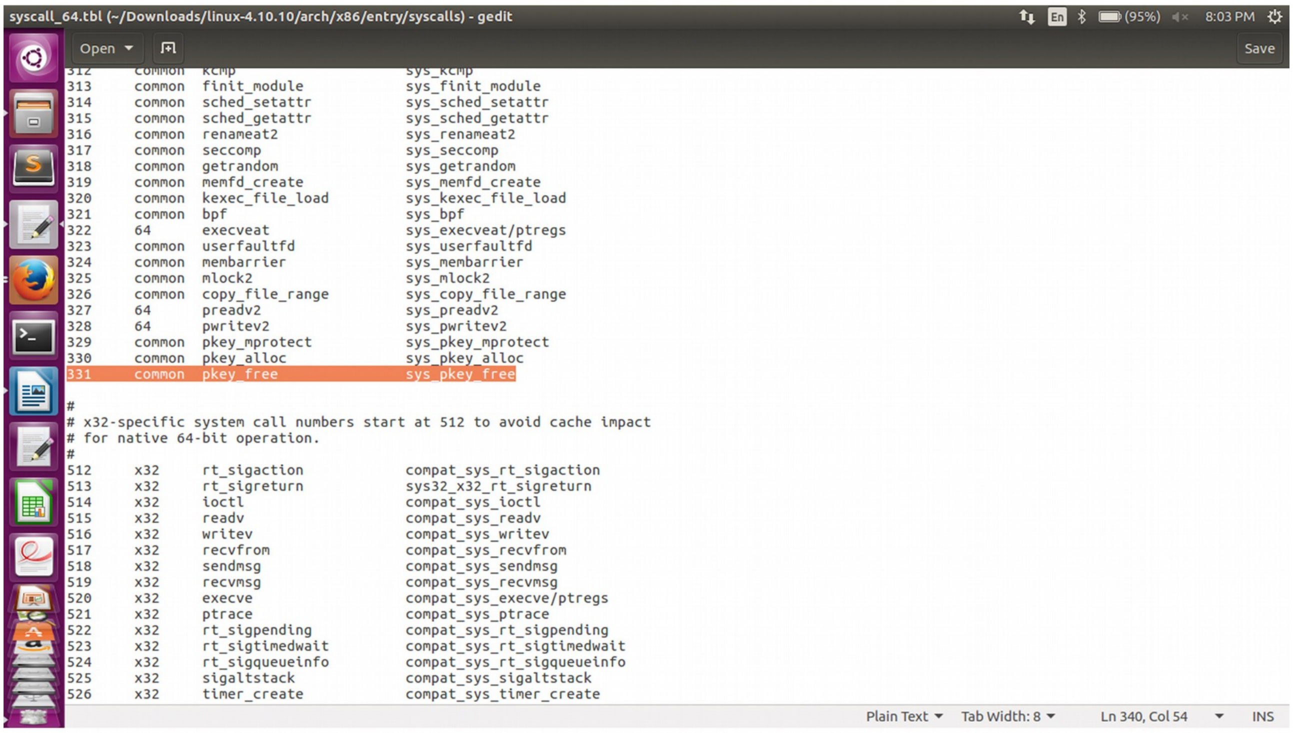
```
else
mod_sign_cmd = true
endif
export mod_sign_cmd


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


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

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

init-y          := $(patsubst %/, %/built-in.o, $(init-y))
core-y          := $(patsubst %/, %/built-in.o, $(core-y))
drivers-y       := $(patsubst %/, %/built-in.o, $(drivers-y))
net-y           := $(patsubst %/, %/built-in.o, $(net-y))
libs-y1         := $(patsubst %/, %/lib.a, $(libs-y))
libs-y2         := $(patsubst %/, %/built-in.o, $(libs-y))
libs-y          := $(libs-y1) $(libs-y2)
virt-y          := $(patsubst %/, %/built-in.o, $(virt-y))

# Externally visible symbols (used by link-vmlinux.sh)
export KBUILD_VMLINUX_INIT := $(head-y) $(init-y)
export KBUILD_VMLINUX_MAIN := $(core-y) $(libs-y) $(drivers-y) $(net-y) $(virt-y)
export KBUILD_LDS          := arch/$(SRCARCH)/kernel/vmlinux.lds
export LDFLAGS_vmlinux
# used by scripts/pacmage/Makefile
export KBUILD_ALLDIRS := $(sort $(filter-out arch/%,$(vmlinux-alldirs)) arch Documentation include samples scripts tools)

vmlinux-deps := $(KBUILD_LDS) $(KBUILD_VMLINUX_INIT) $(KBUILD_VMLINUX_MAIN)
```



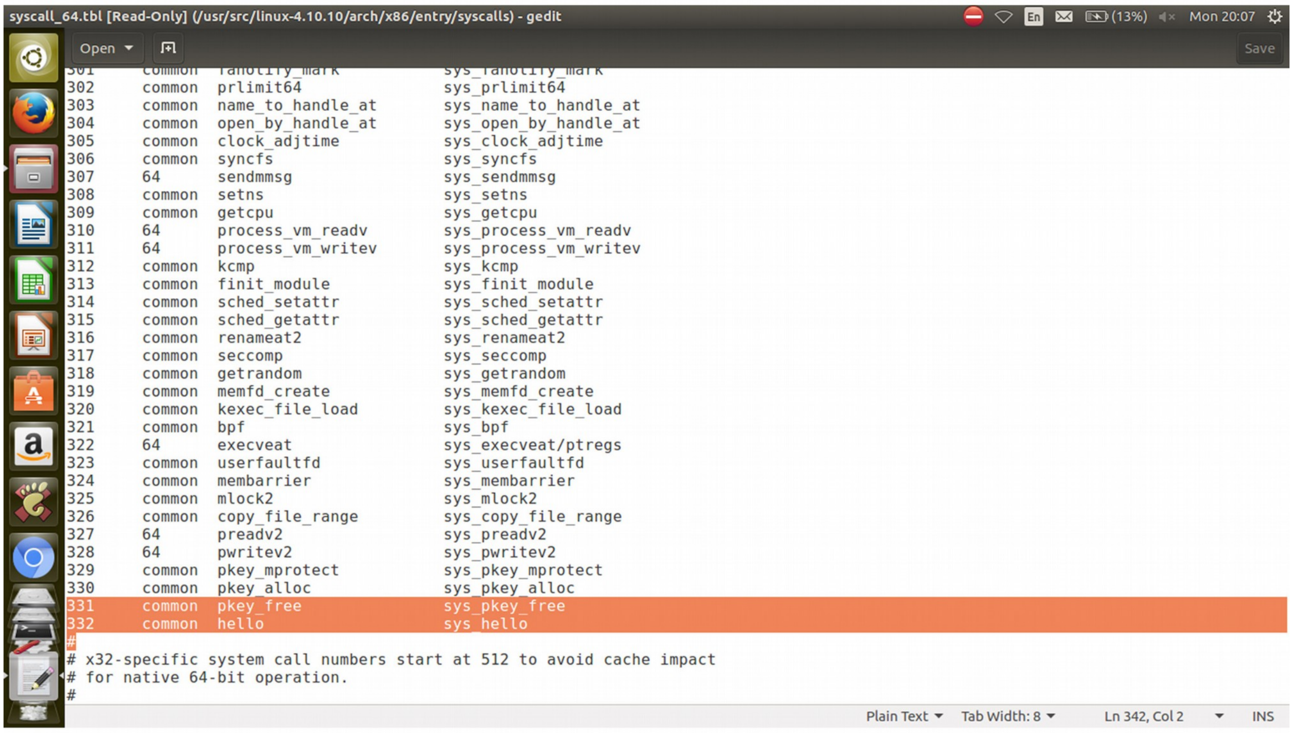
Makefile (~/Downloads/linux-4.10.10) - gedit

```
INITRD_COMPRESS-$(CONFIG_RD_BZIP2) := bzip2
INITRD_COMPRESS-$(CONFIG_RD_LZMA)  := lzma
INITRD_COMPRESS-$(CONFIG_RD_XZ)    := xz
INITRD_COMPRESS-$(CONFIG_RD_LZO)   := lzo
INITRD_COMPRESS-$(CONFIG_RD_LZ4)   := lz4
# do not export INITRD_COMPRESS, since we didn't actually
# choose a sane default compression above.
# export INITRD_COMPRESS := $(INITRD_COMPRESS-y)

ifdef CONFIG_MODULE_SIG_ALL
$(eval $(call config_filename,MODULE_SIG_KEY))

mod_sign_cmd = scripts/sign-file $(CONFIG_MODULE_SIG_HASH) $(MODULE_SIG_KEY_SRCPREFIX)$(CONFIG_MODULE_SIG_KEY) certs/signing_key.x509
else
mod_sign_cmd = true
endif
export mod_sign_cmd


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

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

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

init-y          := $(patsubst %/, %/built-in.o, $(init-y))
core-y          := $(patsubst %/, %/built-in.o, $(core-y))
drivers-y       := $(patsubst %/, %/built-in.o, $(drivers-y))
net-y           := $(patsubst %/, %/built-in.o, $(net-y))
libs-y1         := $(patsubst %/, %/lib.a, $(libs-y))
libs-y2         := $(patsubst %/, %/built-in.o, $(libs-y))
libs-y          := $(libs-y1) $(libs-y2)
virt-y          := $(patsubst %/, %/built-in.o, $(virt-y))

# Externally visible symbols (used by link-vmlinux.sh)
export KBUILD_VMLINUX_INIT := $(head-y) $(init-y)
```

Fig 2

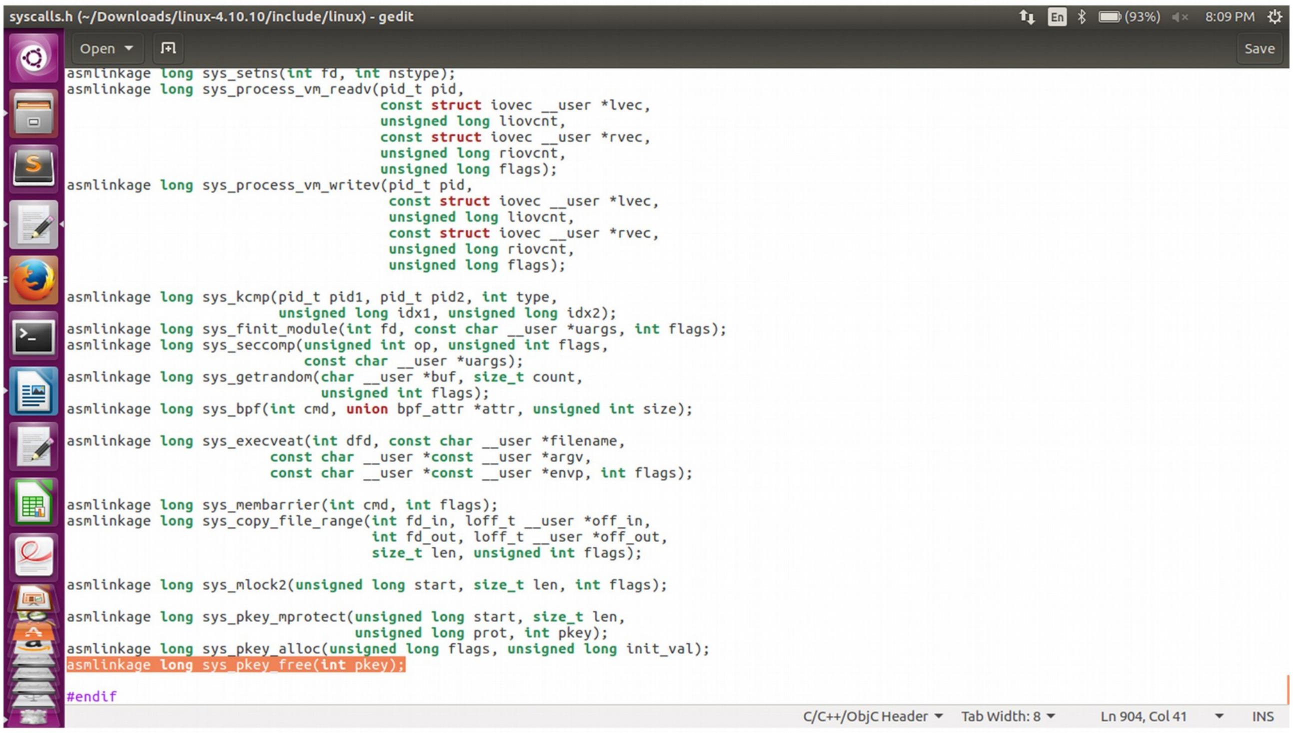

syscall_64.tbl (~/Downloads/linux-4.10.10/arch/x86/entry/syscalls) - gedit
(95%)
8:03 PM
Open
Save
Plain Text
Tab Width: 8
Ln 340, Col 54
INS

```
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     common  seccomp                 sys_seccomp
318     common  getrandom               sys_getrandom
319     common  memfd_create            sys_memfd_create
320     common  kexec_file_load         sys_kexec_file_load
321     common  bpf                     sys_bpf
322     64      execveat                sys_execveat/ptregs
323     common  userfaultfd             sys_userfaultfd
324     common  membarrier              sys_membarrier
325     common  mlock2                  sys_mlock2
326     common  copy_file_range         sys_copy_file_range
327     64      preadv2                 sys_preadv2
328     64      pwritev2                sys_pwritev2
329     common  pkey_mprotect           sys_pkey_mprotect
330     common  pkey_alloc              sys_pkey_alloc
331     common  pkey_free               sys_pkey_free

#
# 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            sys32_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                  compat_sys_execve/ptregs
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
526     x32     timer_create            compat_sys_timer_create
```

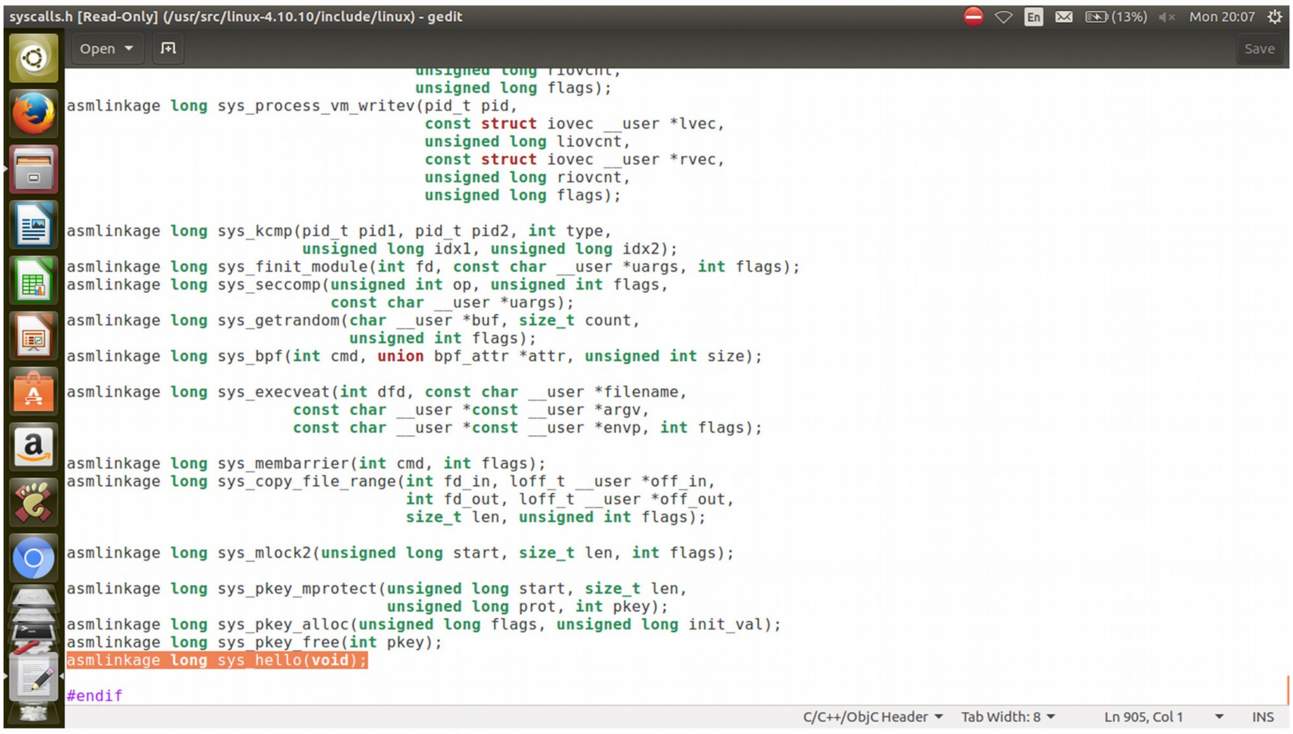

syscall_64.tbl [Read-Only] (/usr/src/linux-4.10.10/arch/x86/entry/syscalls) - gedit
(13%)
Mon 20:07
Open
Save
Plain Text
Tab Width: 8
Ln 342, Col 2
INS

```
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     common  seccomp                 sys_seccomp
318     common  getrandom               sys_getrandom
319     common  memfd_create            sys_memfd_create
320     common  kexec_file_load         sys_kexec_file_load
321     common  bpf                     sys_bpf
322     64      execveat                sys_execveat/ptregs
323     common  userfaultfd             sys_userfaultfd
324     common  membarrier              sys_membarrier
325     common  mlock2                  sys_mlock2
326     common  copy_file_range         sys_copy_file_range
327     64      preadv2                 sys_preadv2
328     64      pwritev2                sys_pwritev2
329     common  pkey_mprotect           sys_pkey_mprotect
330     common  pkey_alloc              sys_pkey_alloc
331     common  pkey_free               sys_pkey_free
332     common  hello                   sys_hello
#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
```

Fig 3

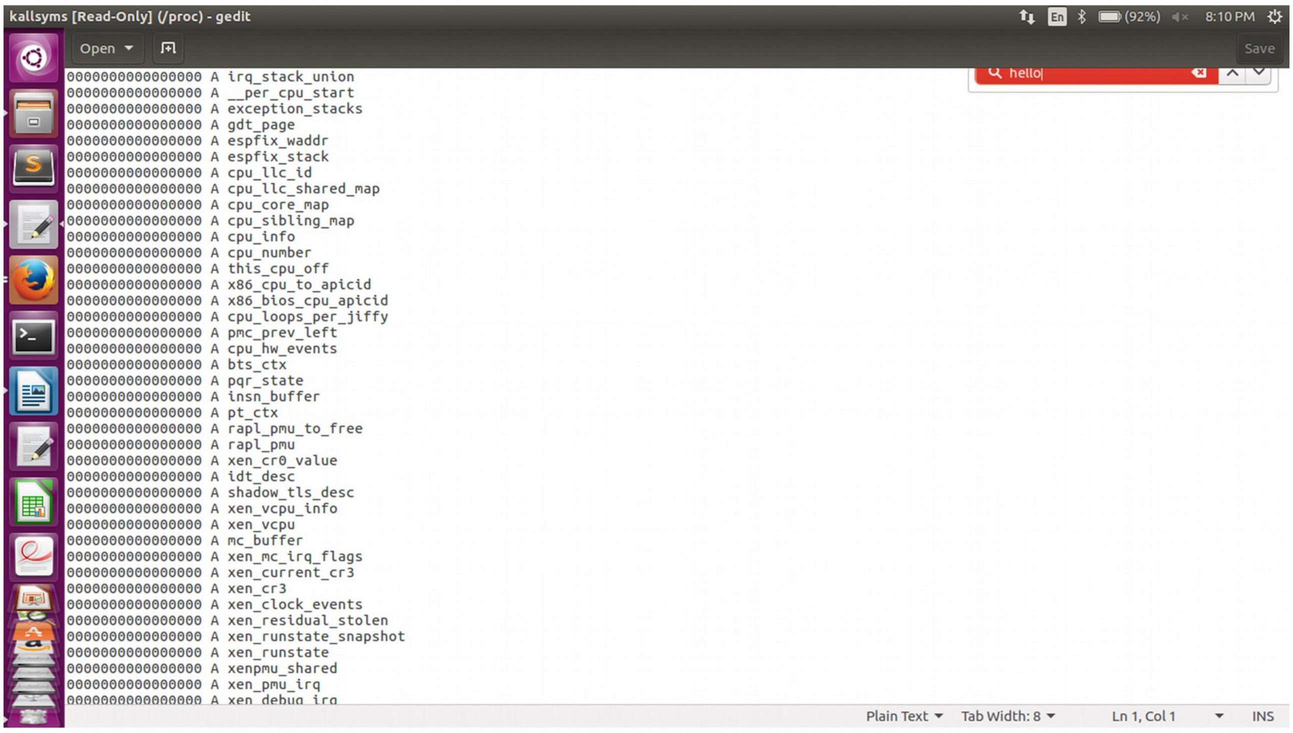


```
asmlinkage long sys_setns(int fd, int nstype);
asmlinkage long sys_process_vm_readv(pid_t pid,
                                     const struct iovec __user *lvec,
                                     unsigned long liovcnt,
                                     const struct iovec __user *rvec,
                                     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);
asmlinkage long sys_seccomp(unsigned int op, unsigned int flags,
                            const char __user *uargs);
asmlinkage long sys_getrandom(char __user *buf, size_t count,
                              unsigned int flags);
asmlinkage long sys_bpf(int cmd, union bpf_attr *attr, unsigned int size);

asmlinkage long sys_execveat(int dfd, const char __user *filename,
                        const char __user *const __user *argv,
                        const char __user *const __user *envp, int flags);

asmlinkage long sys_membarrier(int cmd, int flags);
asmlinkage long sys_copy_file_range(int fd_in, loff_t __user *off_in,
                                    int fd_out, loff_t __user *off_out,
                                    size_t len, unsigned int flags);

asmlinkage long sys_mlock2(unsigned long start, size_t len, int flags);

asmlinkage long sys_pkey_mprotect(unsigned long start, size_t len,
                                  unsigned long prot, int pkey);
asmlinkage long sys_pkey_alloc(unsigned long flags, unsigned long init_val);
asmlinkage long sys_pkey_free(int pkey);

#endif
```

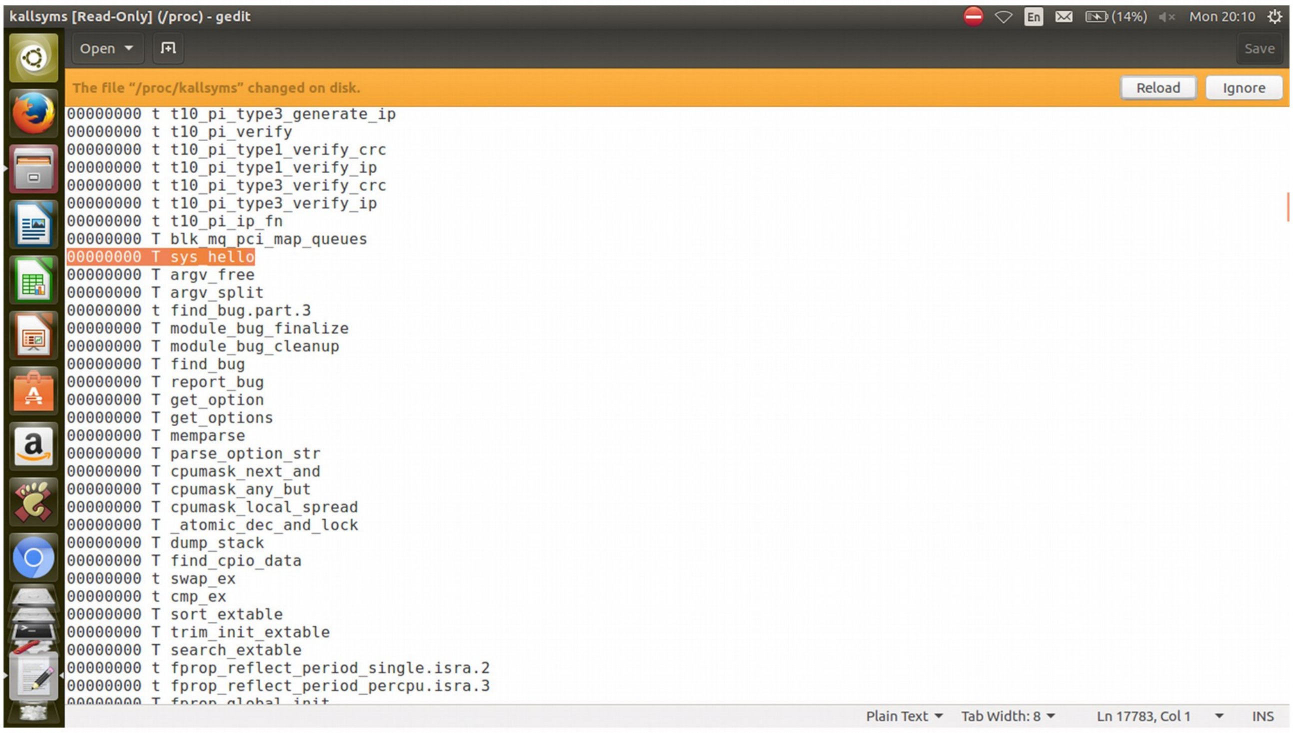

syscalls.h [Read-Only] (/usr/src/linux-4.10.10/include/linux) - gedit
En
(13%)
Mon 20:07
Open
Save
C/C++/ObjC Header
Tab Width: 8
Ln 905, Col 1
INS

```
                                     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);
asmlinkage long sys_seccomp(unsigned int op, unsigned int flags,
                            const char __user *uargs);
asmlinkage long sys_getrandom(char __user *buf, size_t count,
                              unsigned int flags);
asmlinkage long sys_bpf(int cmd, union bpf_attr *attr, unsigned int size);

asmlinkage long sys_execveat(int dfd, const char __user *filename,
                        const char __user *const __user *argv,
                        const char __user *const __user *envp, int flags);

asmlinkage long sys_membarrier(int cmd, int flags);
asmlinkage long sys_copy_file_range(int fd_in, loff_t __user *off_in,
                                    int fd_out, loff_t __user *off_out,
                                    size_t len, unsigned int flags);

asmlinkage long sys_mlock2(unsigned long start, size_t len, int flags);

asmlinkage long sys_pkey_mprotect(unsigned long start, size_t len,
                                  unsigned long prot, int pkey);
asmlinkage long sys_pkey_alloc(unsigned long flags, unsigned long init_val);
asmlinkage long sys_pkey_free(int pkey);
asmlinkage long sys_hello(void);

#endif
```

Fig 4



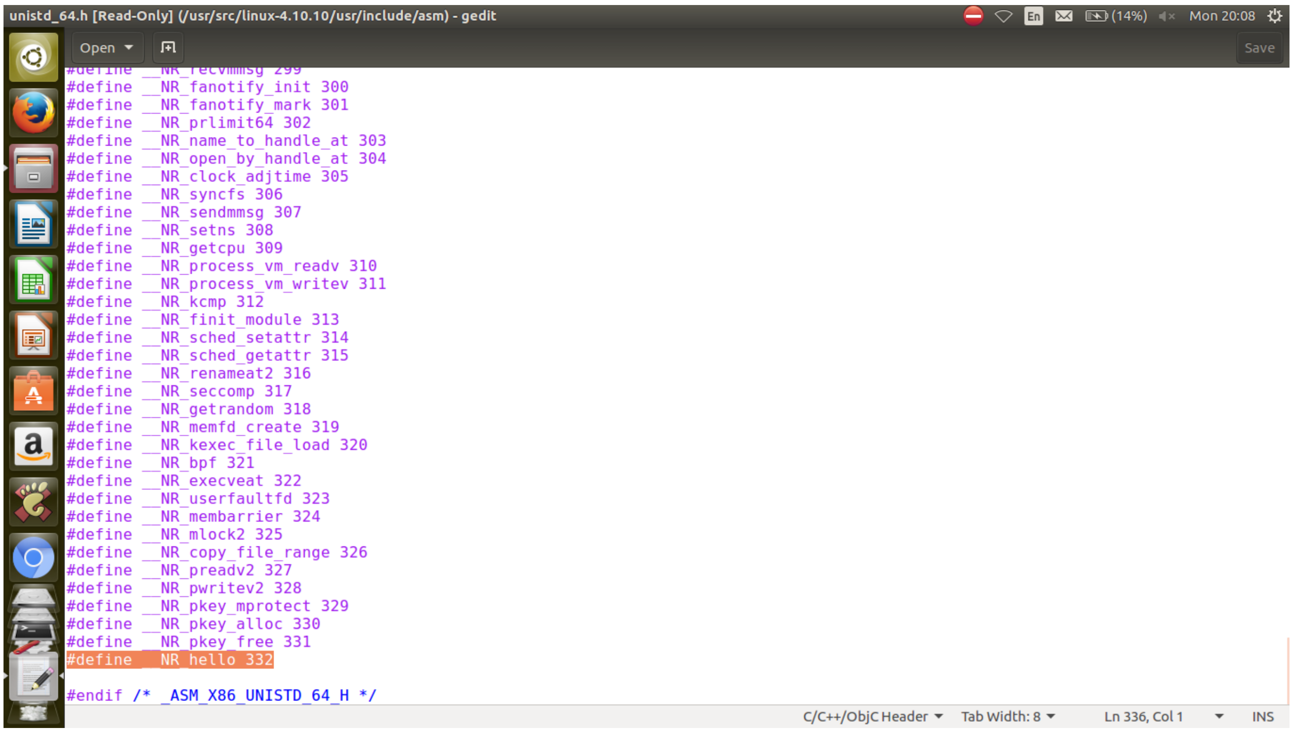
```
0000000000000000 A irq_stack_union
0000000000000000 A __per_cpu_start
0000000000000000 A exception_stacks
0000000000000000 A gdt_page
0000000000000000 A espfix_waddr
0000000000000000 A espfix_stack
0000000000000000 A cpu_llc_id
0000000000000000 A cpu_llc_shared_map
0000000000000000 A cpu_core_map
0000000000000000 A cpu_sibling_map
0000000000000000 A cpu_info
0000000000000000 A cpu_number
0000000000000000 A this_cpu_off
0000000000000000 A x86_cpu_to_apicid
0000000000000000 A x86_bios_cpu_apicid
0000000000000000 A cpu_loops_per_jiffy
0000000000000000 A pmc_prev_left
0000000000000000 A cpu_hw_events
0000000000000000 A bts_ctx
0000000000000000 A pqr_state
0000000000000000 A insn_buffer
0000000000000000 A pt_ctx
0000000000000000 A rapl_pmu_to_free
0000000000000000 A rapl_pmu
0000000000000000 A xen_cr0_value
0000000000000000 A idt_desc
0000000000000000 A shadow_tls_desc
0000000000000000 A xen_vcpu_info
0000000000000000 A xen_vcpu
0000000000000000 A mc_buffer
0000000000000000 A xen_mc_irq_flags
0000000000000000 A xen_current_cr3
0000000000000000 A xen_cr3
0000000000000000 A xen_clock_events
0000000000000000 A xen_residual_stolen
0000000000000000 A xen_runstate_snapshot
0000000000000000 A xen_runstate
0000000000000000 A xenpmu_shared
0000000000000000 A xen_pmu_irq
0000000000000000 A xen_debug_irq
```



The file "/proc/kallsyms" changed on disk.

```
00000000 t t10_pi_type3_generate_ip
00000000 t t10_pi_verify
00000000 t t10_pi_type1_verify_crc
00000000 t t10_pi_type1_verify_ip
00000000 t t10_pi_type3_verify_crc
00000000 t t10_pi_type3_verify_ip
00000000 t t10_pi_ip_fn
00000000 T blk_mq_pci_map_queues
00000000 T sys_hello
00000000 T argv_free
00000000 T argv_split
00000000 t find_bug.part.3
00000000 T module_bug_finalize
00000000 T module_bug_cleanup
00000000 T find_bug
00000000 T report_bug
00000000 T get_option
00000000 T get_options
00000000 T memparse
00000000 T parse_option_str
00000000 T cpumask_next_and
00000000 T cpumask_any_but
00000000 T cpumask_local_spread
00000000 T _atomic_dec_and_lock
00000000 T dump_stack
00000000 T find_cpio_data
00000000 t swap_ex
00000000 t cmp_ex
00000000 T sort_extable
00000000 T trim_init_extable
00000000 T search_extable
00000000 t fprop_reflect_period_single.isra.2
00000000 t fprop_reflect_period_percpu.isra.3
00000000 T fprop_global_init
```

Fig 5


unistd_64.h [Read-Only] (/usr/src/linux-4.10.10/usr/include/asm) - gedit
Open
Save
Mon 20:08
(14%)

```
#define __NR_recvmmsg 299
#define __NR_fanotify_init 300
#define __NR_fanotify_mark 301
#define __NR_prlimit64 302
#define __NR_name_to_handle_at 303
#define __NR_open_by_handle_at 304
#define __NR_clock_adjtime 305
#define __NR_syncfs 306
#define __NR_sendmmsg 307
#define __NR_setns 308
#define __NR_getcpu 309
#define __NR_process_vm_readv 310
#define __NR_process_vm_writev 311
#define __NR_kcmp 312
#define __NR_finit_module 313
#define __NR_sched_setattr 314
#define __NR_sched_getattr 315
#define __NR_renameat2 316
#define __NR_seccomp 317
#define __NR_getrandom 318
#define __NR_memfd_create 319
#define __NR_kexec_file_load 320
#define __NR_bpf 321
#define __NR_execveat 322
#define __NR_userfaultfd 323
#define __NR_membarrier 324
#define __NR_mlock2 325
#define __NR_copy_file_range 326
#define __NR_preadv2 327
#define __NR_pwritev2 328
#define __NR_pkey_mprotect 329
#define __NR_pkey_alloc 330
#define __NR_pkey_free 331
#define __NR_hello 332

#endif /* _ASM_X86_UNISTD_64_H */
```

```
C/C++/ObjC Header   Tab Width: 8   Ln 336, Col 1   INS
```

Fig 6