



سؤال ۱. مشاهده فراخوانی‌های سیستمی تعریف شده

در این قسمت با توجه به این‌که ساختار سخت‌افزار ما با سوال فرق دارد، فایل asm را به کمک دستور find پیدا کردیم:

```
> cd /usr/include/  
> find . -name asm  
./x86_64-linux-gnu/asm  
> cd ./x86_64-linux-gnu/asm/  
> cat unistd.h
```

```
mostafa@mostafa-UX303UB: /usr/include  
File Edit View Search Terminal Help  
> cat unistd.h  
/* Copyright (C) 1991-2018 Free Software Foundation, Inc.  
   This file is part of the GNU C Library.  
  
   The GNU C Library is free software; you can redistribute it and/or  
   modify it under the terms of the GNU Lesser General Public  
   License as published by the Free Software Foundation; either  
   version 2.1 of the License, or (at your option) any later version.  
  
   The GNU C Library is distributed in the hope that it will be useful,  
   but WITHOUT ANY WARRANTY; without even the implied warranty of  
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
   Lesser General Public License for more details.  
  
   You should have received a copy of the GNU Lesser General Public  
   License along with the GNU C Library; if not, see  
   <http://www.gnu.org/licenses/>. */  
  
/*  
 *      POSIX Standard: 2.10 Symbolic Constants      <unistd.h>  
 */  
#ifndef _UNISTD_H  
#define _UNISTD_H      1
```

سؤال ۲. اجرای یک فراخوانی سیستمی

- دستورات زیر را درون terminal اجرا می‌کنیم.

```
> mkdir 2 && cd 2
```

```
> sudo nano testsyscall.cpp
```

- کد داده شده در سوال را در این قسمت کپی کرده و سپس با استفاده از دستور `ctrl + x` آن را ذخیره می‌کنیم.

- به کمک دستورات زیر کد را کامپایل کرده و اجرا می‌کنیم.

```
> sudo gcc -o testsyscall testsyscall.cpp
```

```
> ./testsyscall
```

- نتیجه‌ی اجرای آن ساخته شدن یک پوشه به نام testdir در مسیر^۱ فعلی است و در آخر پیام The result is 0. را چاپ می‌کند.
- همان‌طور که در توضیحات فراخوانی سیستمی آمده است، هر فراخوانی سیستم با یک شماره‌ی ثابت شناخته می‌شود. نقش __NR_mkdir (که به‌طور سراسری^۲ تعریف شده است) در این‌جا این است تا عدد مربوط به این فراخوانی جایگزین آن شود.

```

mostafa@mostafa-UX303UB: /usr/include/x86_64-linux-gnu/asm
File Edit View Search Terminal Help
#define __NR_semop 65
#define __NR_semctl 66
#define __NR_shmctl 67
#define __NR_msgget 68
#define __NR_msgsnd 69
#define __NR_msgrcv 70
#define __NR_msgctl 71
#define __NR_fcntl 72
#define __NR_flock 73
#define __NR_fsync 74
#define __NR_fdatasync 75
#define __NR_truncate 76
#define __NR_ftruncate 77
#define __NR_getdents 78
#define __NR_getcwd 79
#define __NR_chdir 80
#define __NR_fchdir 81
#define __NR_rename 82
#define __NR_mkdir 83
#define __NR_rmdir 84
#define __NR_creat 85
#define __NR_link 86
#define __NR_unlink 87
#define __NR_symlink 88

```

- تابع syscall() یک تابع کوچک است که رابط زبان assembly آن دارای شماره و آرگومان‌های مشخص است. تابع پر کاربردی است اگر از آن بدون wrapperها استفاده کنیم، رجسترهای پردازنده قبل از فراخوانی سیستم ذخیره می‌کند و بعداً آن‌ها را بازگردانی می‌کند. خروجی آن در صورتی که با موفقیت انجام گیرد، «۰» و در غیر این صورت «۱-» خواهد بود.

سؤال ۳. اجرای ساده‌تر فراخوانی‌های سیستمی

```

#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
int main() {
    long result;
    result = mkdir("testdir", 0777);
    printf("The result is %ld.\n", result);
    return 0;
}

```

سؤال ۴. آشنایی با چند فراخوان سیستمی پرکاربرد

- فراخوان سیستمی access

directory^۱
global^۲

```

#include <errno.h>
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    int result;
    char *path = argv[1];
    result = access(path, F_OK);
    if (result == 0)
    {
        printf("%s exists\n", path);
    }
    else
    {
        printf("%s doesn't exist\n", path);
    }
    result = access(path, R_OK);
    if (result == 0) {
        printf("read permission is granted\n");
    }
    else {
        printf("read permission isn't granted\n");
    }
    result = access(path, W_OK);
    if (result == 0) {
        printf("write permission is granted\n");
    }
    else {
        printf("write permission isn't granted\n");
    }
    result = access(path, X_OK);
    if (result == 0) {
        printf("execute permission is granted\n");
    }
    else {
        printf("execute permission isn't granted\n");
    }
    return 0;
}

```

• فراخوان‌های سیستمی open, close, write

```

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main() {
    int open_result;
    int write_result;
    open_result = open("oslab2.txt", O_CREAT | O_WRONLY, 0777);

```

```

write_result = write(open_result, "Mostafa Ghadimi\n", strlen("Mostafa Ghadimi\n"));
close(open_result);
return 0;
}

```

• فراخوان سیستمی sysinfo

```

#include <linux/kernel.h>
#include <stdio.h>
#include <sys/sysinfo.h>

int main()
{
    const double megabyte = 1024 * 1024;
    struct sysinfo si;
    sysinfo(&si);
    printf("total RAM: 1f.%5 MB\n", si.totalram / megabyte);
    printf("free RAM: 1f.%5 MB\n", si.freeram / megabyte);
    return 0;
}

```

• فراخوان سیستمی getrusage

```

#include <sys/time.h>
#include <sys/resource.h>
#include <stdio.h>

int main() {
    struct rusage ru;
    getrusage(RUSAGE_SELF, &ru);
    printf("maximum resident set size: %ld\n", ru.ru_maxrss);
    printf("integral shared memory size: %ld\n", ru.ru_ixrss);
    printf("integral unshared stack size: %ld\n", ru.ru_isrss);
}

```

سؤال ۵. اضافه کردن یک فراخوانی سیستمی به سیستم عامل

- با توجه به لینکی که در کلاس داده شد، می‌خواهیم یک فراخوانی سیستمی اضافه کنیم که تعداد page fault را نشان دهد.

۱. در ابتدا به ۲ فایل `arch/x86/entry/syscalls/syscall_32.tbl` و `arch/x86/entry/syscalls/syscall_64.tbl`، فراخوانی سیستمی دل‌خواه با شماره ۴۳۶ را اضافه می‌کنیم.

۲. به عنوان micro به فایل `include/uapi/asm-generic/unistd.h` آن را اضافه می‌کنیم.

۳. حال می‌خواهیم کد خود را در `kernel` اضافه کنیم. در فایل `include/linux/mm.h`، در انتهای فایل یک متغیر سراسری به نام `pfcount` که نشان‌دهنده تعداد page fault ها می‌باشد را اضافه می‌کنیم.

۴. در فایل `include/linux/sched.h`، به `task_struct` (که یک abstraction برای پردازنده‌های لینوکس می‌باشد) یک شمارنده page fault اضافه می‌کنیم.

```

349 425 common io_uring_setup      __x64_sys_io_uring_setup
350 426 common io_uring_enter      __x64_sys_io_uring_enter
351 427 common io_uring_register    __x64_sys_io_uring_register
352 428 common open_tree          __x64_sys_open_tree
353 429 common move_mount        __x64_sys_move_mount
354 430 common fsopen             __x64_sys_fsopen
355 431 common fsconfig           __x64_sys_fsconfig
356 432 common fsmount            __x64_sys_fsmount
357 433 common fspick             __x64_sys_fspick
358 434 common pidfd_open        __x64_sys_pidfd_open
359 435 common clone3            __x64_sys_clone3/ptregs
360 436 common msyscall         __x64_sys_msyscall
361 #
362 # x32-specific system call numbers start at 512 to avoid cache impact
363 # for native 64-bit operation. The __x32_compat_sys stubs are created
364 # on-the-fly for compat_sys_*( ) compatibility system calls if X86_X32
365 # is defined.
366 #
367 512 x32 rt_sigaction         __x32_compat_sys_rt_sigaction
368 513 x32 rt_sigreturn        sys32_x32_rt_sigreturn
369 514 x32 ioctl               __x32_compat_sys_ioctl
370 515 x32 readv               __x32_compat_sys_readv
371 516 x32 writev               __x32_compat_sys_writev
372 517 x32 recvmfrom           __x32_compat_sys_recvmfrom
373 518 x32 sendmsg              __x32_compat_sys_sendmsg
374 519 x32 recvmmsg              __x32_compat_sys_recvmmsg

```

```

431 424 i386 pidfd_send_signal    sys_pidfd_send_signal
__ia32_sys_pidfd_send_signal
432 425 i386 io_uring_setup      sys_io_uring_setup
__ia32_sys_io_uring_setup
433 426 i386 io_uring_enter      sys_io_uring_enter
__ia32_sys_io_uring_enter
434 427 i386 io_uring_register    sys_io_uring_register
__ia32_sys_io_uring_register
435 428 i386 open_tree          sys_open_tree
__ia32_sys_open_tree
436 429 i386 move_mount        sys_move_mount
__ia32_sys_move_mount
437 430 i386 fsopen             sys_fsopen
__ia32_sys_fsopen
438 431 i386 fsconfig           sys_fsconfig
__ia32_sys_fsconfig
439 432 i386 fsmount            sys_fsmount
sys_fsmount          __ia32_sys_fsmount
440 433 i386 fspick             sys_fspick
__ia32_sys_fspick
441 434 i386 pidfd_open        sys_pidfd_open
__ia32_sys_pidfd_open
442 435 i386 clone3            sys_clone3
__ia32_sys_clone3
443 436 i386 msyscall         sys_msyscall
__x64_sys_msyscall

```

۵. در فایل kernel/fork.c، کد `tsk->pf = 0` در جای مشخص اضافه می‌کنیم.
۶. حال برای اینکه هر بار page fault رخ داد، متغیر خود را افزایش دهیم، در فایل arch/x86/mm/fault.c خطوط ۱۵۱۵، ۱۵۲۱ و ۱۵۲۲ را اضافه می‌کنیم.
۷. برای پیاده سازی تابع systemcall، ابتدا کد زیر را به فایل kernel/sys.c اضافه می‌کنیم.
۸. تعریف تابع systemcall موردنظر را به فایل include/linux/syscalls.h اضافه می‌کنیم.
۹. در نهایت برای compatibility کد زیر را به فایل kernel/sys_ni.c اضافه می‌کنیم.
۱۰. حال با استفاده از دستور `make -j12 nohup &`، کرنل را کامپایل کرده و با استفاده از دستور `cat nohup.out` خروجی آن را ملاحظه می‌کنیم.
۱۱. با استفاده از فایل test.c می‌خواهیم خروجی آن را ببینیم که با اجرای فایل خروجی test حاصل از gcc با دستور `test.c -o test` آن را بدست می‌آوریم.


```

1260 #endif
1261 #ifdef CONFIG_SECURITY
1262     /* Used by LSM modules for access restriction: */
1263     void *security;
1264 #endif
1265
1266 #ifdef CONFIG_GCC_PLUGIN_STACKLEAK
1267     unsigned long lowest_stack;
1268     unsigned long prev_lowest_stack;
1269 #endif
1270 unsigned long pf;
1271 /*
1272  * New fields for task_struct should be added above here, so that
1273  * they are included in the randomized portion of task_struct.
1274  */
1275 randomized_struct_fields_end
1276
1277 /* CPU-specific state of this task: */
1278 struct thread_struct thread;
1279
1280 /*
1281  * WARNING: on x86, 'thread_struct' contains a variable-sized
1282  * structure. It *MUST* be at the end of 'task_struct'.
1283  *
1284  * Do not put anything below here!
1285  */

```

```

854 {
855     unsigned long *stackend;
856
857     stackend = end_of_stack(tsk);
858     *stackend = STACK_END_MAGIC; /* for overflow detection */
859 }
860
861 static struct task_struct *dup_task_struct(struct task_struct *orig, int
node)
862 {
863     struct task_struct *tsk;
864     unsigned long *stack;
865     struct vm_struct *stack_vm_area __maybe_unused;
866     int err;
867
868     if (node == NUMA_NO_NODE)
869         node = tsk_fork_get_node(orig);
870     tsk = alloc_task_struct_node(node);
871     if (!tsk)
872         return NULL;
873     tsk->pf = 0;
874     stack = alloc_thread_stack_node(tsk, node);
875     if (!stack)
876         goto free_tsk;
877
878     if (memcg_charge_kernel_stack(tsk))
879         goto free_stack;

```

Activities Text Editor 21:20 3 آغۇست 2017 fa

Open fault.c Save

~/linux-5.4.0/arch/x86/mm

*AZ_OS2.txt x fault.c x

```
1504         perf_sw_event(PERF_COUNT_SW_PAGE_FAULTS_MIN, 1, regs,
1505             address);
1506     }
1507     check_v8086_mode(regs, address, tsk);
1508 }
1509 NOKPROBE_SYMBOL(do_user_addr_fault);
1510
1511 /*
1512  * Explicitly marked noline such that the function tracer sees this as
1513  * the
1514  * page_fault entry point.
1515  */
1516 unsigned long pfcnt;
1517 static noline void
1518 __do_page_fault(struct pt_regs *regs, unsigned long hw_error_code,
1519     unsigned long address)
1520 {
1521     pfcnt++;
1522     current->pf++;
1523     prefetchw(&current->mm->mmap_sem);
1524
1525     if (unlikely(kmmio_fault(regs, address)))
1526         return;
1527 }
```

C Tab Width: 8 Ln 1522, Col 9 INS

Activities Text Editor 20:20 3 آغۇست 2017 en

Open sys.c Save

~/linux-5.4.0/kernel

unistd.h x mm.h x sched.h x fork.c x fault.c x sys.c x

```
2657     __put_user(s->freeswap, &info->freeswap) ||
2658     __put_user(s->procs, &info->procs) ||
2659     __put_user(s->totalhigh, &info->totalhigh) ||
2660     __put_user(s->freehigh, &info->freehigh) ||
2661     __put_user(s->mem_unit, &info->mem_unit))
2662     return -EFAULT;
2663
2664     return 0;
2665 }
2666 #endif /* CONFIG_COMPAT */
2667 SYSCALL_DEFINE0(mysyscall)
2668 {
2669
2670     printk("Number of dirty pages per process: \n");
2671     struct task_struct *t;
2672
2673     // Dirty pages of each process
2674     for_each_process(t)
2675     {
2676         printk("%s [%d]: %lu\n", t->comm, t->pid, t->nr_dirtied);
2677     }
2678     printk("Number of page faults (total): %lu\n", pfcnt);
2679     printk("Number of page faults (current): %lu\n", current->pf);
2680
2681     return 0;
2682 }
```

Saving file "/home/alireza/linux-5.4.0/kernel/... C Tab Width: 8 Ln 2682, Col 2 INS


```
1400     set_personality(personality);
1401     return old;
1402 }
1403 }
1404
1405 asmlinkage int sys_mysyscall(void);
1406
1407 /* for __ARCH_WANT_SYS_IPC */
1408 long ksys_semtimedop(int semid, struct sembuf __user *tsops,
1409                     unsigned int nsops,
1410                     const struct __kernel_timespec __user *timeout);
1411 long ksys_semget(key_t key, int nsems, int semflg);
1412 long ksys_old_semctl(int semid, int semnum, int cmd, unsigned long arg);
1413 long ksys_msgget(key_t key, int msgflg);
1414 long ksys_old_msgctl(int msqid, int cmd, struct msqid_ds __user *buf);
1415 long ksys_msgrcv(int msqid, struct msgbuf __user *msgp, size_t msgsz,
1416                 long msgtyp, int msgflg);
1417 long ksys_msgsnd(int msqid, struct msgbuf __user *msgp, size_t msgsz,
1418                 int msgflg);
1419 long ksys_shmget(key_t key, size_t size, int shmflg);
1420 long ksys_shmctl(char __user *shmaddr);
1421 long ksys_old_shmctl(int shmid, int cmd, struct shmid_ds __user *buf);
1422 long compat_ksys_semtimedop(int semid, struct sembuf __user *tsops,
1423                             unsigned int nsops,
1424                             const struct old_timespec32 __user *timeout);
1425
```

```
Activities Terminal 20:29 3 أغسطس en
alireza@alireza-VirtualBox: ~/linux-5.4.0
alireza@alireza-VirtualBox:~/linux-5.4.0$ nohup make -j12 &
[2] 8698
alireza@alireza-VirtualBox:~/linux-5.4.0$ nohup: ignoring input and appending o
utput to 'nohup.out'
^C
alireza@alireza-VirtualBox:~/linux-5.4.0$ cat nohup.out
SYSTBL arch/x86/include/generated/asm/syscalls_32.h
DESCEND objtool
SYSHDR arch/x86/include/generated/asm/unistd_32_ia32.h
SYSHDR arch/x86/include/generated/asm/unistd_64_x32.h
SYSTBL arch/x86/include/generated/asm/syscalls_64.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_32.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_x32.h
CALL scripts/atomic/check-atomics.sh
CC arch/x86/kernel/asm-offsets.s
DESCEND objtool
UPD include/generated/asm-offsets.h
CALL scripts/checksyscalls.sh
AS arch/x86/entry/entry_64.o
CC arch/x86/ia32/sys_ia32.o
CC certs/system_keyring.o
CC arch/x86/events/core.o
CC arch/x86/hyperv/hv_init.o
CC arch/x86/mm/init.o
CC [M] arch/x86/kvm/../../../../virt/kvm/kvm_main.o
CC init/main.o
AS arch/x86/entry/thunk_64.o
CC arch/x86/crypto/crc32c-intel_glue.o
```

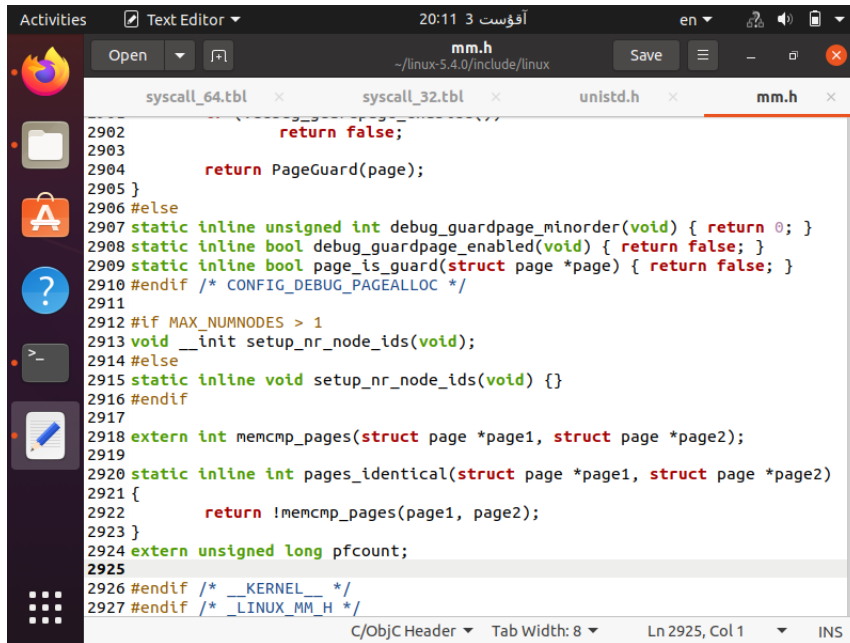
```
Activities Terminal 20:29 3 أغسطس en
alireza@alireza-VirtualBox: ~/linux-5.4.0
SYSHDR arch/x86/include/generated/uapi/asm/unistd_32.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_x32.h
CALL scripts/atomic/check-atomics.sh
CC arch/x86/kernel/asm-offsets.s
DESCEND objtool
UPD include/generated/asm-offsets.h
CALL scripts/checksyscalls.sh
AS arch/x86/entry/entry_64.o
CC arch/x86/ia32/sys_ia32.o
CC certs/system_keyring.o
CC arch/x86/events/core.o
CC arch/x86/hyperv/hv_init.o
CC arch/x86/mm/init.o
CC [M] arch/x86/kvm/../../../../virt/kvm/kvm_main.o
CC init/main.o
AS arch/x86/entry/thunk_64.o
CC arch/x86/crypto/crc32c-intel_glue.o
CALL scripts/atomic/check-atomics.sh
CALL scripts/checksyscalls.sh
CC arch/x86/entry/syscall_64.o
CC mm/filemap.o
CC kernel/fork.o
CC init/main.o
CC arch/x86/kernel/process_64.o
CC certs/blacklist.o
CHK include/generated/compile.h
CC init/do_mounts.o
CC init/do_mounts_initrd.o
```

```
Activities Terminal 20:29 3 أغسطس en alireza@alireza-VirtualBox: ~/linux-5.4.0
CC arch/x86/mm/intel_b4.o
CC arch/x86/net/bpf_jit_comp.o
CC arch/x86/events/intel/core.o
AR init/built-in.a
CC arch/x86/entry/vdso/vma.o
CC arch/x86/entry/vdso/vma.o
CHK include/generated/compile.h
CC kernel/exec_domain.o
CC fs/open.o
CC kernel/panic.o
CC [M] arch/x86/kvm/../../../../virt/kvm/coalesced_mmio.o
CC arch/x86/hyperv/hv_apic.o
CC init/version.o
CC mm/oom_kill.o
CC fs/open.o
CC fs/read_write.o
AS [M] arch/x86/crypto/twofish-x86_64-asm_64.o
fixdep: error opening file: arch/x86/crypto/.des3_edg_glue.o.d: No such file or
directory
make[2]: *** [scripts/Makefile.build:275: arch/x86/crypto/des3_edg_glue.o] Error
2
make[2]: *** Deleting file 'arch/x86/crypto/des3_edg_glue.o'
make[1]: *** [scripts/Makefile.build:522: arch/x86/crypto] Error 2
make[1]: *** Waiting for unfinished jobs....
CC kernel/panic.o
CC fs/file_table.o
CC mm/oom_kill.o
CC kernel/cpu.o
CC [M] arch/x86/crypto/twofish_glue.o
CC arch/x86/kernel/signal_compat.o
```

```
Activities Terminal 20:29 3 أغسطس en alireza@alireza-VirtualBox: ~/linux-5.4.0
arch/x86/hyperv/hv_apic.c: In function 'hv_send_apic_ack_acknowledged':
arch/x86/hyperv/hv_apic.c:226:1: warning: the frame size of 1032 bytes is large
r than 1024 bytes [-Wframe-larger-than=]
226 | }
    | ^
CC security/apparmor/apparmorfs.o
CC kernel/exit.o
CC arch/x86/hyperv/hv_spinlock.o
CC arch/x86/kernel/idt.o
CC [M] arch/x86/kvm/../../../../virt/kvm/eventfd.o
AR arch/x86/hyperv/built-in.a
CC ipc/util.o
CC kernel/softirq.o
CC kernel/exit.o
CC fs/super.o
CC fs/read_write.o
CC arch/x86/mm/fault.o
CC fs/super.o
AR arch/x86/net/built-in.a
CC arch/x86/events/intel/bts.o
CC fs/char_dev.o
CC arch/x86/events/intel/ds.o
CC fs/char_dev.o
CC [M] arch/x86/crypto/serpent_sse2_glue.o
AR arch/x86/events/amd/built-in.a
CC fs/stat.o
ar: arch/x86/events/amd/core.o: No such file or directory
make[3]: *** [scripts/Makefile.build:395: arch/x86/events/amd/built-in.a] Error
1
make[2]: *** [scripts/Makefile.build:522: arch/x86/events/amd] Error 2
```

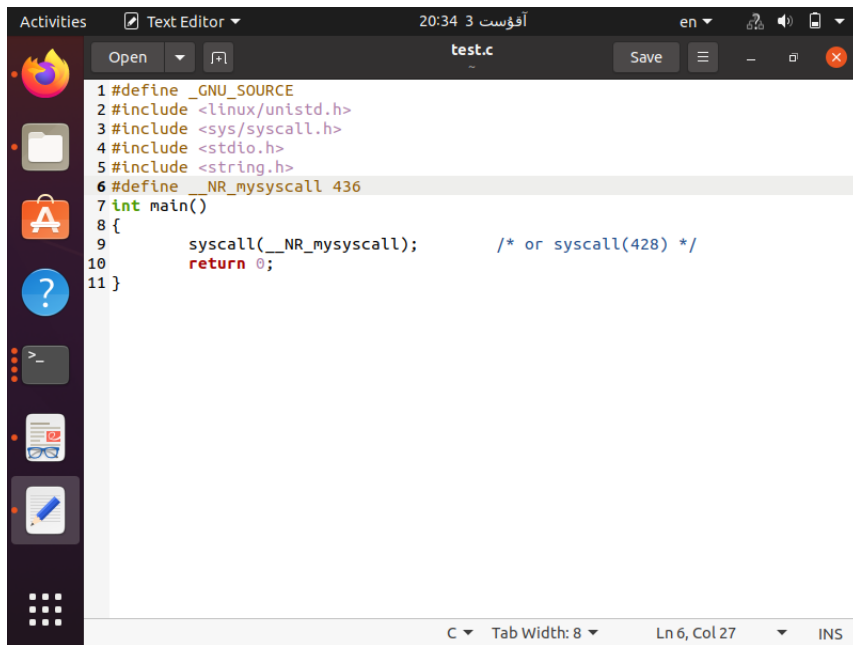
```
Activities Terminal 20:29 3 أغسطس en
alireza@alireza-VirtualBox: ~/linux-5.4.0
make[2]: *** [scripts/Makefile.build:522: arch/x86/events/and/p4.o] Error 1
make[2]: *** Waiting for unfinished jobs....
CC fs/exec.o
CC arch/x86/kernel/irq.o
CC mm/fadvise.o
CC arch/x86/kernel/irq_64.o
CC [M] arch/x86/crypto/aesni-intel_glue.o
CC ipc/msgutil.o
CC mm/fadvise.o
CC kernel/softirq.o
CC [M] arch/x86/kvm/../../../../virt/kvm/irqchip.o
CC fs/pipe.o
CC arch/x86/kernel/dumpstack_64.o
CC kernel/resource.o
CC mm/maccess.o
CC arch/x86/events/intel/knc.o
CC arch/x86/events/intel/lbr.o
CC fs/namei.o
CC fs/fcntl.o
CC fs/stat.o
CC fs/exec.o
CC kernel/sysctl.o
CC arch/x86/mm/ioremap.o
kernel/exit.o: warning: objtool: gelf_getehdr: invalid 'Elf' handle
make[1]: *** [scripts/Makefile.build:273: kernel/exit.o] Error 1
make[1]: *** Deleting file 'kernel/exit.o'
make[1]: *** Waiting for unfinished jobs....
CC arch/x86/events/intel/p4.o
```

```
Activities Terminal 20:29 3 أغسطس en
alireza@alireza-VirtualBox: ~/linux-5.4.0
CC mm/maccess.o
CC arch/x86/events/intel/knc.o
CC arch/x86/events/intel/lbr.o
CC fs/namei.o
CC fs/fcntl.o
CC fs/stat.o
CC fs/exec.o
CC kernel/sysctl.o
CC arch/x86/mm/ioremap.o
kernel/exit.o: warning: objtool: gelf_getehdr: invalid 'Elf' handle
make[1]: *** [scripts/Makefile.build:273: kernel/exit.o] Error 1
make[1]: *** Deleting file 'kernel/exit.o'
make[1]: *** Waiting for unfinished jobs....
CC arch/x86/events/intel/p4.o
CC fs/ioctl.o
fixdep: error opening file: kernel/.exit.o.d: No such file or directory
make[1]: *** [scripts/Makefile.build:275: kernel/exit.o] Error 2
make[1]: *** Waiting for unfinished jobs....
CC arch/x86/mm/extable.o
CC ipc/msg.o
CC fs/pipe.o
CC fs/namei.o
CC mm/maccess.o
CC [M] arch/x86/crypto/ghash-clmulni-intel_glue.o
CC arch/x86/kernel/ioport.o
CC mm/page-writeback.o
CC [M] arch/x86/kvm/../../../../virt/kvm/vfio.o
CC arch/x86/events/intel/p6.o
alireza@alireza-VirtualBox:~/linux-5.4.0$
```



```
2902         return false;
2903
2904         return PageGuard(page);
2905     }
2906 #else
2907 static inline unsigned int debug_guardpage_minorder(void) { return 0; }
2908 static inline bool debug_guardpage_enabled(void) { return false; }
2909 static inline bool page_is_guard(struct page *page) { return false; }
2910 #endif /* CONFIG_DEBUG_PAGEALLOC */
2911
2912 #if MAX_NUMNODES > 1
2913 void __init setup_nr_node_ids(void);
2914 #else
2915 static inline void setup_nr_node_ids(void) {}
2916 #endif
2917
2918 extern int memcmp_pages(struct page *page1, struct page *page2);
2919
2920 static inline int pages_identical(struct page *page1, struct page *page2)
2921 {
2922     return !memcmp_pages(page1, page2);
2923 }
2924 extern unsigned long pfcount;
2925
2926 #endif /* __KERNEL__ */
2927 #endif /* _LINUX_MM_H */
```

C/ObjC Header Tab Width: 8 Ln 2925, Col 1 INS



The screenshot shows a Linux desktop environment. On the left is a vertical dock with icons for the Dash, Home, Applications, and several other utilities. The main window is a text editor titled 'Text Editor' with a file named 'test.c'. The code in the editor is a C program that defines a custom syscall number and uses it in the main function. The code is as follows:

```
1 #define _GNU_SOURCE
2 #include <linux/unistd.h>
3 #include <sys/syscall.h>
4 #include <stdio.h>
5 #include <string.h>
6 #define __NR_mysyscall 436
7 int main()
8 {
9     syscall(__NR_mysyscall);    /* or syscall(428) */
10    return 0;
11 }
```

The status bar at the bottom of the text editor shows 'C', 'Tab Width: 8', 'Ln 6, Col 27', and 'INS'.