



گزارش آزمایش ۲: آشنایی با فراخوانی‌های سیستمی

ملیکا علیزاده ۴۰۱۱۰۶۲۵۵

الینا هژبری ۴۰۱۱۷۰۶۶۱

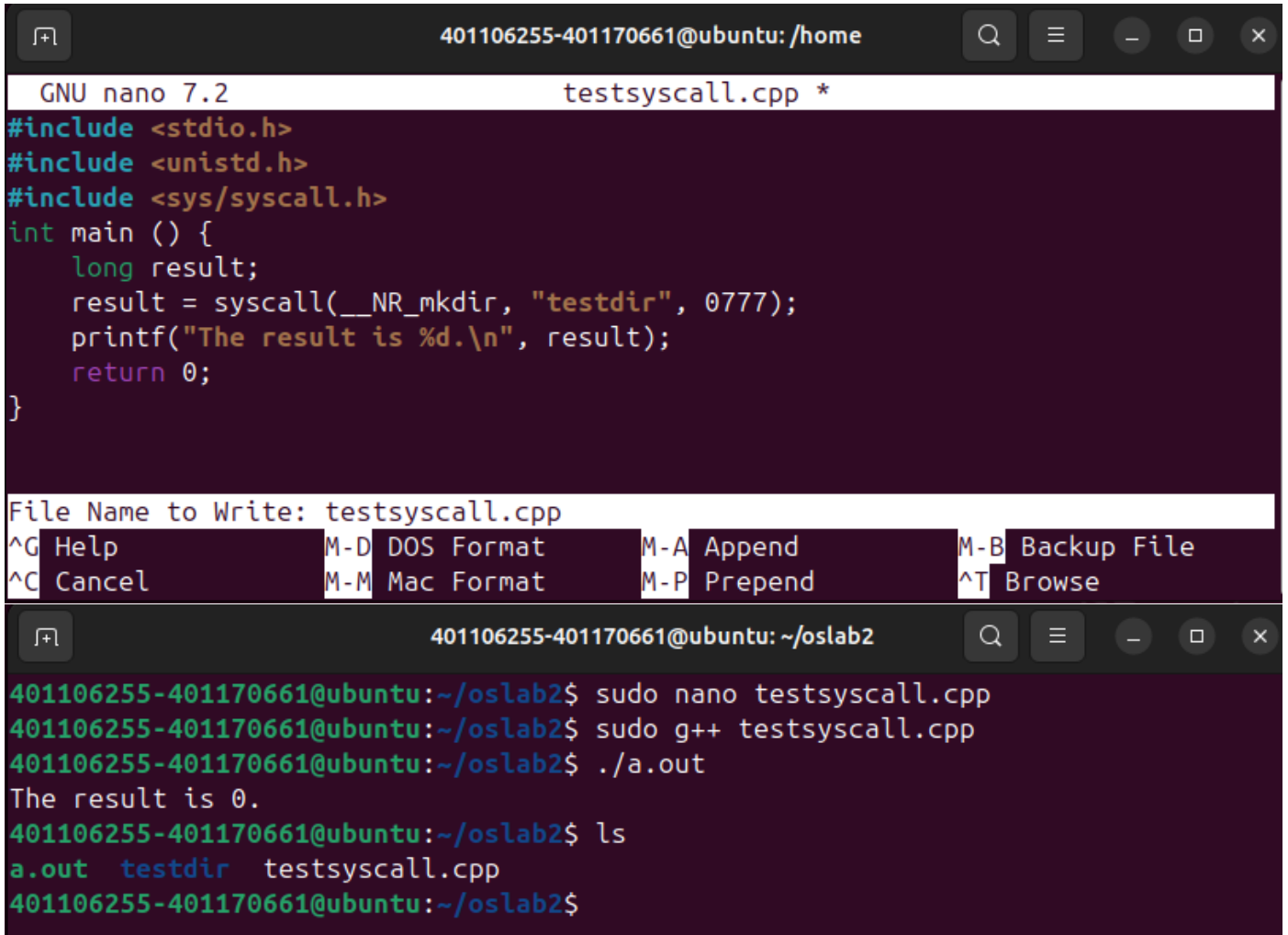
۱ مشاهده‌ی فراخوانی‌های سیستمی تعریف شده

```
401106255-401170661@ubuntu: ~  
401106255-401170661@ubuntu:~$ nano /usr/include/x86_64-linux-gnu/asm/unistd_64.h  
401106255-401170661@ubuntu:~$  
  
GNU nano 7.2 /usr/include/x86_64-linux-gnu/asm/unistd_64.h  
#ifndef _ASM_UNISTD_64_H  
#define _ASM_UNISTD_64_H  
  
#define __NR_read 0  
#define __NR_write 1  
#define __NR_open 2  
#define __NR_close 3  
#define __NR_stat 4  
#define __NR_fstat 5  
#define __NR_lstat 6  
#define __NR_poll 7  
#define __NR_lseek 8  
#define __NR_mmap 9  
#define __NR_mprotect 10  
#define __NR_munmap 11  
#define __NR_brk 12  
#define __NR_rt_sigaction 13  
#define __NR_rt_sigprocmask 14  
#define __NR_rt_sigreturn 15  
#define __NR_ioctl 16  
#define __NR_pread64 17  
#define __NR_pwrite64 18  
#define __NR_readv 19  
#define __NR_writev 20  
#define __NR_access 21  
#define __NR_pipe 22  
#define __NR_select 23  
#define __NR_sched_yield 24  
#define __NR_mremap 25  
#define __NR_msync 26  
#define __NR_mincore 27  
#define __NR_madvise 28  
  
^C Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location  
^X Exit      ^R Read File ^\ Replace  ^U Paste     ^J Justify  ^_ Go To Line
```

شکل ۱: مشاهده لیست فراخوانی‌های سیستمی به همراه شماره‌ی آنها

۲ اجرای یک فراخوانی سیستمی

با اجرای این فراخوانی سیستمی خروجی `The result is 0.` چاپ شده است که با توجه به `۰` یعنی این عمل موفق بوده و پوشه `testdir` ساخته شده است. `__NR_mkdir` همان شماره‌ی مربوط به فراخوانی سیستمی `mkdir` است. با استفاده از دستور `syscall` می‌توان هر فراخوانی سیستمی موجود در سیستم را با شماره آن و ورودی‌های دیگرش، صدا زد. خروجی این تابع `۰` در صورت موفقیت و `۱-` در صورت بروز خطا است.



```
401106255-401170661@ubuntu: /home
GNU nano 7.2 testsyscall.cpp *
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
int main () {
    long result;
    result = syscall(__NR_mkdir, "testdir", 0777);
    printf("The result is %d.\n", result);
    return 0;
}
File Name to Write: testsyscall.cpp
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^T Browse
401106255-401170661@ubuntu: ~/oslab2
401106255-401170661@ubuntu:~/oslab2$ sudo nano testsyscall.cpp
401106255-401170661@ubuntu:~/oslab2$ sudo g++ testsyscall.cpp
401106255-401170661@ubuntu:~/oslab2$ ./a.out
The result is 0.
401106255-401170661@ubuntu:~/oslab2$ ls
a.out testdir testsyscall.cpp
401106255-401170661@ubuntu:~/oslab2$
```

شکل ۲: استفاده از فراخوانی سیستمی `mkdir`

The image consists of two screenshots of a terminal window. The top screenshot shows the GNU nano 7.2 text editor editing a file named testsyscall2.cpp. The code inside the file is as follows:

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

The bottom screenshot shows the terminal prompt where the user runs the following commands:

```
401106255-401170661@ubuntu: ~/oslab2$ sudo nano testsyscall2.cpp
401106255-401170661@ubuntu: ~/oslab2$ sudo g++ testsyscall2.cpp
401106255-401170661@ubuntu: ~/oslab2$ ./a.out
The result is 0.
401106255-401170661@ubuntu: ~/oslab2$
```

شکل ۳: بازنویسی کد بخش قبل با mkdir()

۴ آشنایی با چند فراخوانی سیستمی پرکاربرد

تست فراخوانی سیستمی `access`:

```
401106255-401170661@ubuntu: ~/oslab2
GNU nano 7.2 taccess.c
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Enter the address to the file.\n");
        return 1;
    }

    if(access(argv[1], R_OK) == 0 || access(argv[1], W_OK) == 0) {
        printf("File is accessible.\n");
    } else
        printf("The file does not exists.\n");

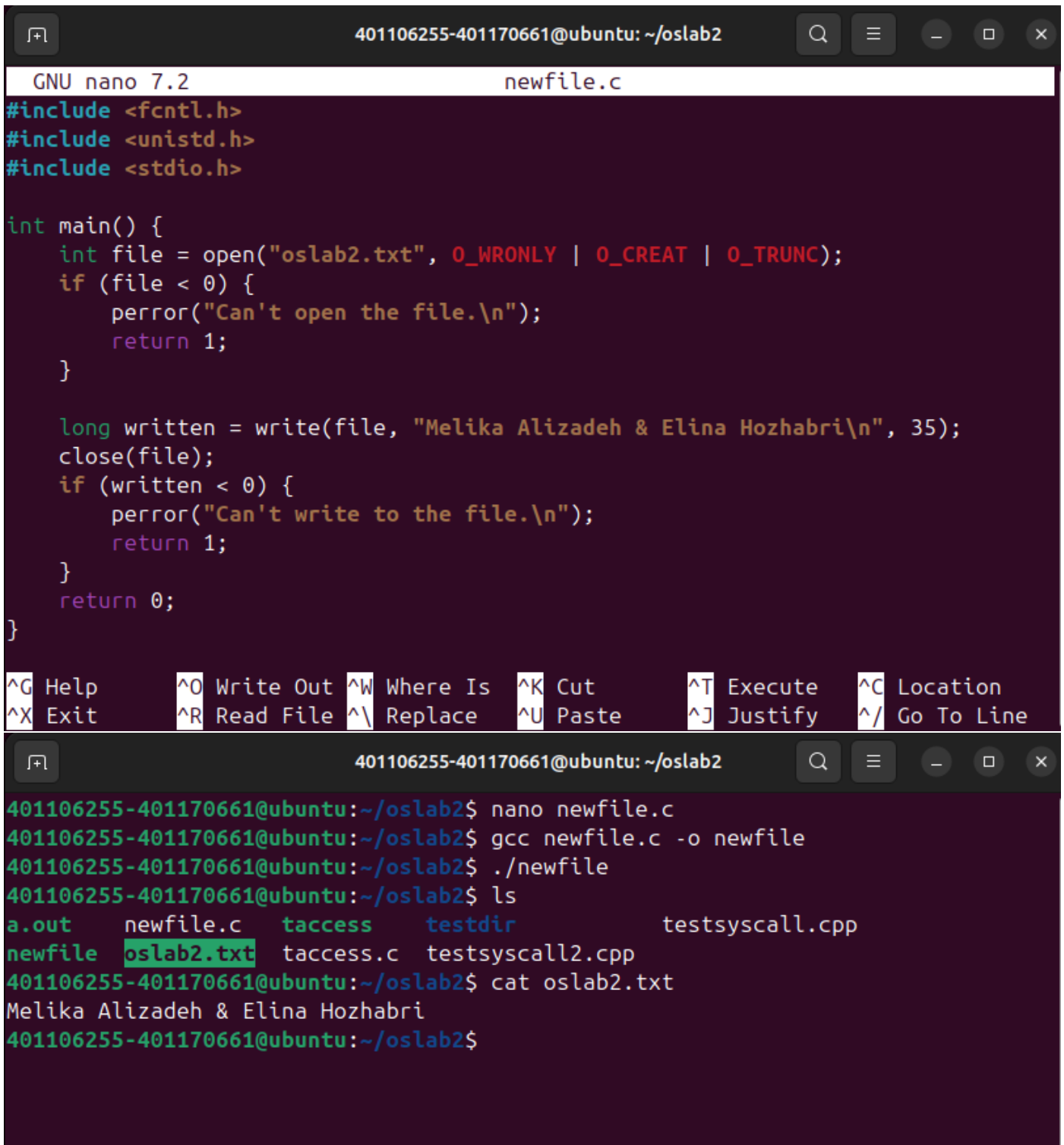
    return 0;
}

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line

401106255-401170661@ubuntu: ~/oslab2
401106255-401170661@ubuntu:~/oslab2$ nano taccess.c
401106255-401170661@ubuntu:~/oslab2$ gcc taccess.c -o taccess
401106255-401170661@ubuntu:~/oslab2$ ./taccess
Enter the address to the file.
401106255-401170661@ubuntu:~/oslab2$ ./taccess test
The file does not exists.
401106255-401170661@ubuntu:~/oslab2$ ./taccess testdir
File is accessible.
401106255-401170661@ubuntu:~/oslab2$
```

شکل ۴: تست فراخوانی سیستمی `access`

تست فراخوانی‌های سیستمی open, close, write:



```
401106255-401170661@ubuntu: ~/oslab2
GNU nano 7.2 newfile.c
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

int main() {
    int file = open("oslab2.txt", O_WRONLY | O_CREAT | O_TRUNC);
    if (file < 0) {
        perror("Can't open the file.\n");
        return 1;
    }

    long written = write(file, "Melika Alizadeh & Elina Hozhabri\n", 35);
    close(file);
    if (written < 0) {
        perror("Can't write to the file.\n");
        return 1;
    }
    return 0;
}

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line

401106255-401170661@ubuntu: ~/oslab2
401106255-401170661@ubuntu:~/oslab2$ nano newfile.c
401106255-401170661@ubuntu:~/oslab2$ gcc newfile.c -o newfile
401106255-401170661@ubuntu:~/oslab2$ ./newfile
401106255-401170661@ubuntu:~/oslab2$ ls
a.out      newfile.c  taccess    testdir     testsyscall.cpp
newfile    oslab2.txt taccess.c  testsyscall2.cpp
401106255-401170661@ubuntu:~/oslab2$ cat oslab2.txt
Melika Alizadeh & Elina Hozhabri
401106255-401170661@ubuntu:~/oslab2$
```

شکل ۵: تست فراخوانی‌های سیستمی open, close, write

```

401106255-401170661@ubuntu: ~/oslab2
GNU nano 7.2 meminfo.c
#include <stdio.h>
#include <sys/sysinfo.h>

int main() {
    struct sysinfo info;
    if (sysinfo(&info) == 0) {
        printf("Total RAM: %lu\nFree RAM: %lu\n", info.totalram, info.freeram);
    } else {
        printf("Can't access.\n");
        return 1;
    }
    return 0;
}

[ Wrote 13 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line

401106255-401170661@ubuntu: ~/oslab2
401106255-401170661@ubuntu:~/oslab2$ nano meminfo.c
401106255-401170661@ubuntu:~/oslab2$ gcc meminfo.c -o meminfo
401106255-401170661@ubuntu:~/oslab2$ ./meminfo
Total RAM: 8327151616
Free RAM: 4333957120
401106255-401170661@ubuntu:~/oslab2$

```

شکل ۶: تست فراخوانی سیستمی sysinfo

```

401106255-401170661@ubuntu: ~/oslab2
GNU nano 7.2 usage.c
#include <stdio.h>
#include <sys/time.h>
#include <sys/resource.h>

int main() {
    struct rusage usage;
    if (getrusage(RUSAGE_SELF, &usage) == 0) {
        printf("RAM usage: %lu KB\n", usage.ru_maxrss);
    } else {
        printf("Can't access.\n");
        return 1;
    }
    return 0;
}

[ Wrote 14 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line

401106255-401170661@ubuntu: ~/oslab2
401106255-401170661@ubuntu:~/oslab2$ nano usage.c
401106255-401170661@ubuntu:~/oslab2$ gcc usage.c -o usage
401106255-401170661@ubuntu:~/oslab2$ ./usage
RAM usage: 2324 KB
401106255-401170661@ubuntu:~/oslab2$

```

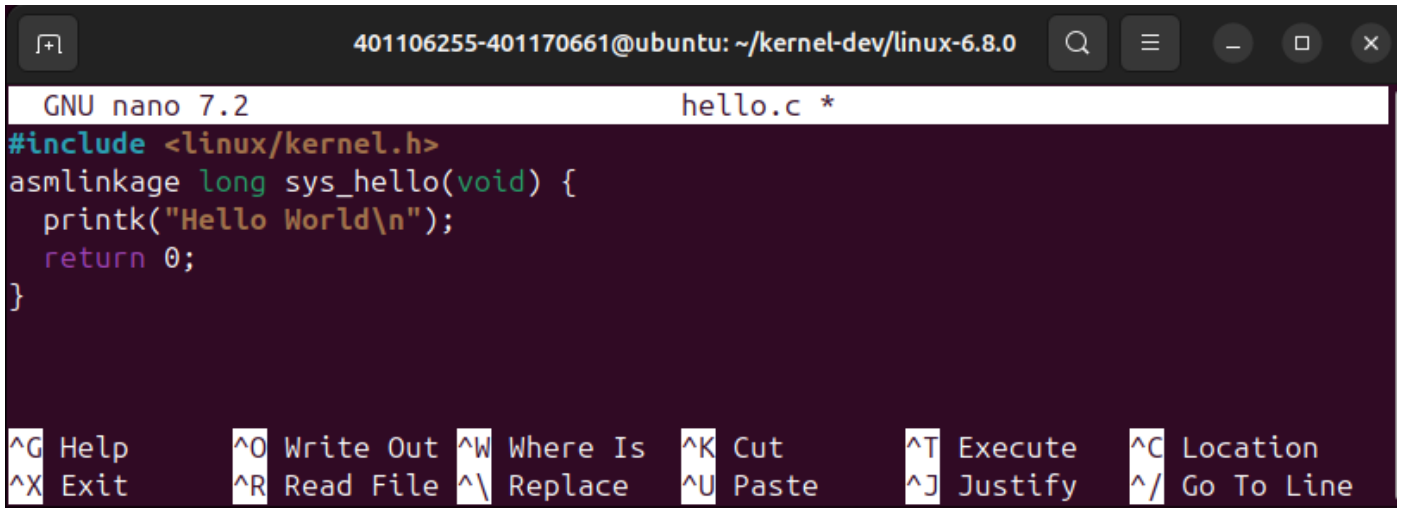
شکل ۷: تست فراخوانی سیستمی getrusage

۵ اضافه کردن فراخوانی سیستمی به سیستم عامل

افزون فراخوانی سیستم hello:

مراحل افزودن فراخوانی سیستمی:

۱. با دسترسی root وارد سیستم عامل شده
۲. وارد پوشه‌ی کد منبع هسته سیستم عامل شده
۳. ایجاد پوشه خالی hello
۴. در این پوشه ایجاد فایل hello.c که شامل کد فراخوانی سیستمی است.

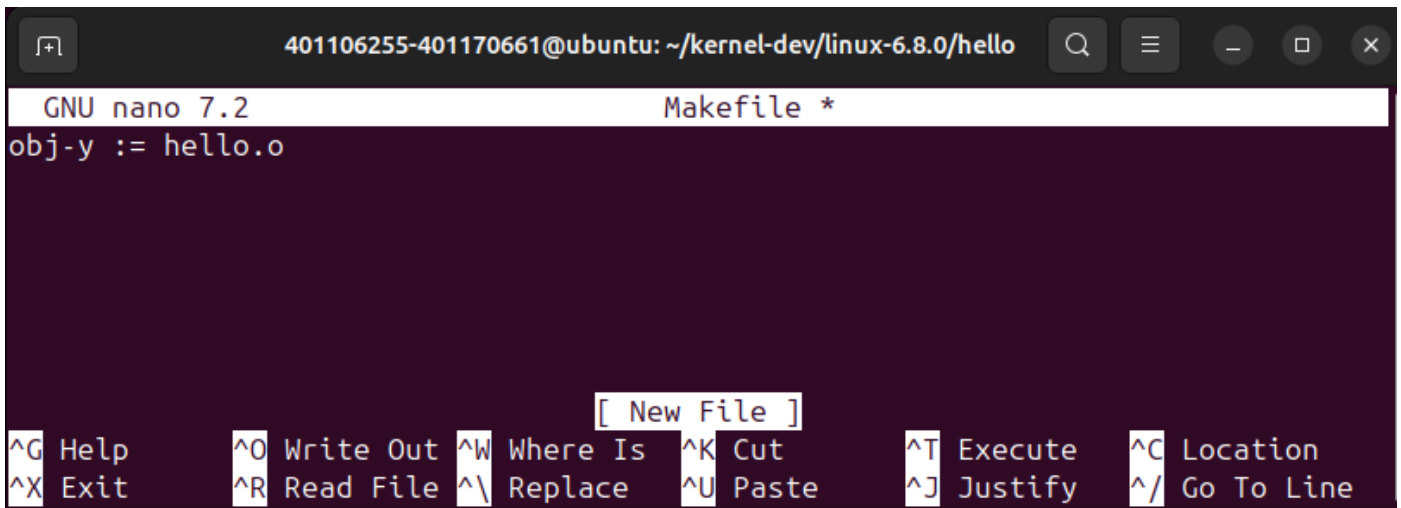


```
GNU nano 7.2 hello.c *
#include <linux/kernel.h>
asmlinkage long sys_hello(void) {
    printk("Hello World\n");
    return 0;
}
```

Help Write Out Where Is Cut Execute Location
Exit Read File Replace Paste Justify Go To Line

شکل ۸: کد hello.c

۵. ایجاد فایل Makefile




```
GNU nano 7.2 Makefile *
obj-y := hello.o
```

[New File]

Help Write Out Where Is Cut Execute Location
Exit Read File Replace Paste Justify Go To Line

شکل ۹: محتوای Makefile

۶. تغییر فایل Makefile موجود در ریشه‌ی کد منبع را باز کنید.



```
GNU nano 7.2 Makefile *
# Just "make" or "make all" shall build modules as well

ifneq ($(filter all modules nsdeps %compile_commands.json clang-%,$(MAKECMDGOALS))
    KBUILD_MODULES := 1
endif

ifeq ($(MAKECMDGOALS),)
    KBUILD_MODULES := 1
endif

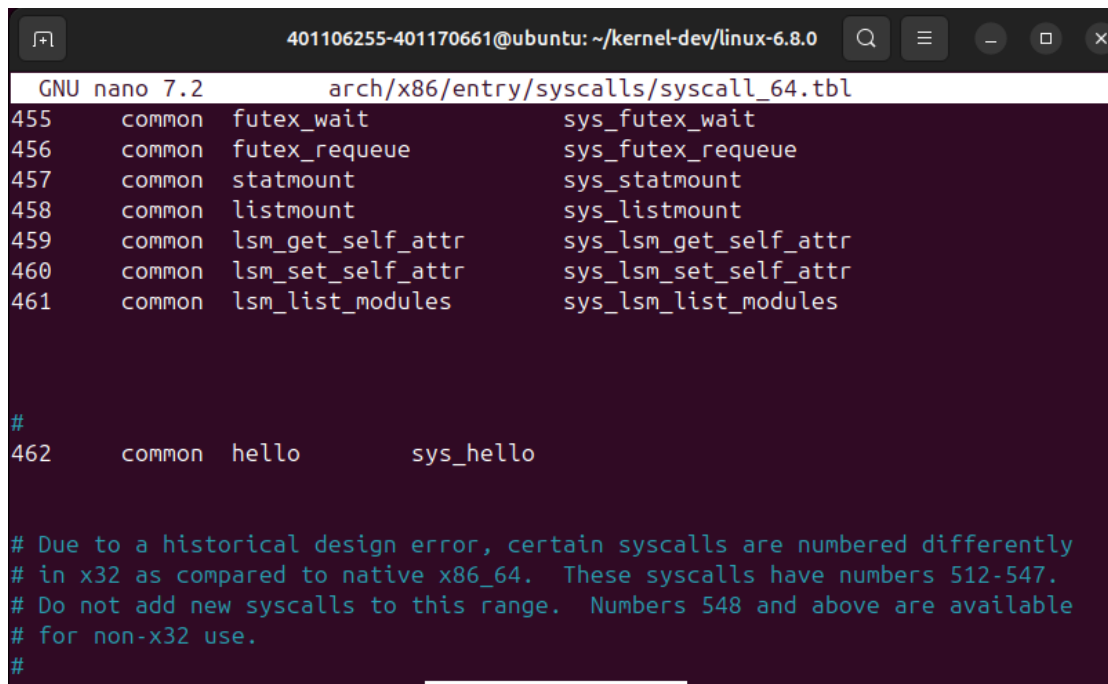
export KBUILD_MODULES KBUILD_BUILTIN

ifdef need-config
include include/config/auto.conf
endif

ifeq ($(KBUILD_EXTMOD),)
# Objects we will link into vmlinux / subdirs we need to visit
core-y      += kernel/ certs/ hello/
```

شکل ۱۰: محتوای Makefile

۷. به روزرسانی جدول فراخوانی سیستمی nano arch/x86/entry/syscalls/syscall_64.tbl



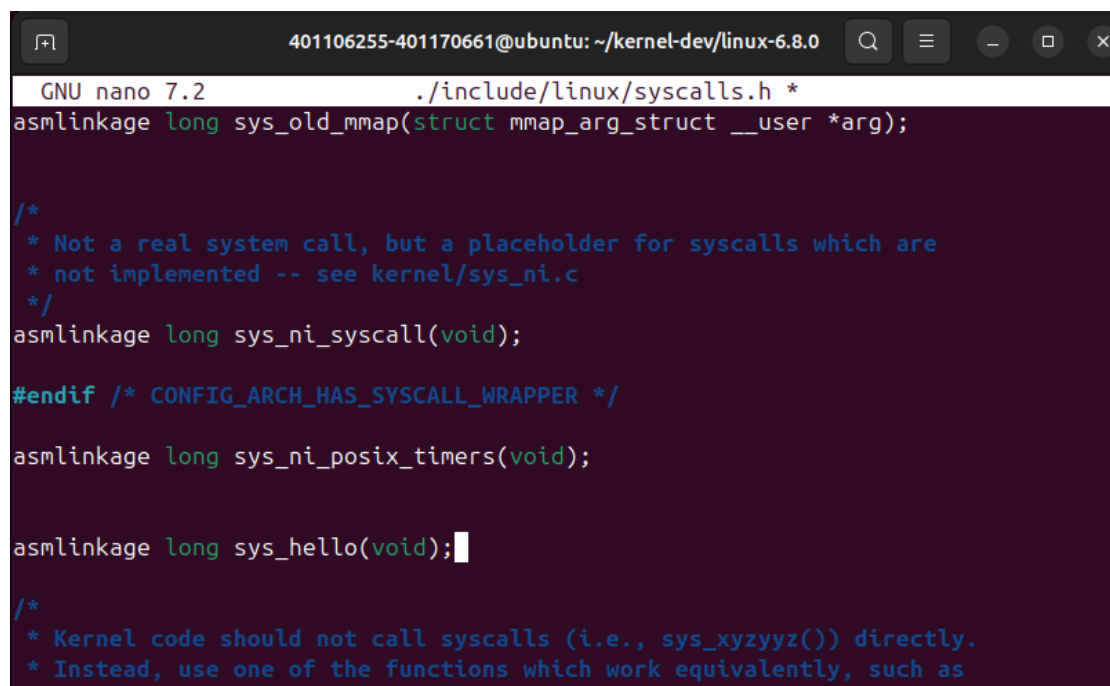
```
GNU nano 7.2 arch/x86/entry/syscalls/syscall_64.tbl
455 common futex_wait sys_futex_wait
456 common futex_requeue sys_futex_requeue
457 common statmount sys_statmount
458 common listmount sys_listmount
459 common lsm_get_self_attr sys_lsm_get_self_attr
460 common lsm_set_self_attr sys_lsm_set_self_attr
461 common lsm_list_modules sys_lsm_list_modules

#
462 common hello sys_hello

# Due to a historical design error, certain syscalls are numbered differently
# in x32 as compared to native x86_64. These syscalls have numbers 512-547.
# Do not add new syscalls to this range. Numbers 548 and above are available
# for non-x32 use.
#
```

شکل ۱۱: به روزرسانی جدول فراخوانی سیستمی

۸. تغییر محتوای فایل `./include/linux/syscalls.h`



```
GNU nano 7.2 ./include/linux/syscalls.h *
asmlinkage long sys_old_mmap(struct mmap_arg_struct __user *arg);

/*
 * Not a real system call, but a placeholder for syscalls which are
 * not implemented -- see kernel/sys_ni.c
 */
asmlinkage long sys_ni_syscall(void);

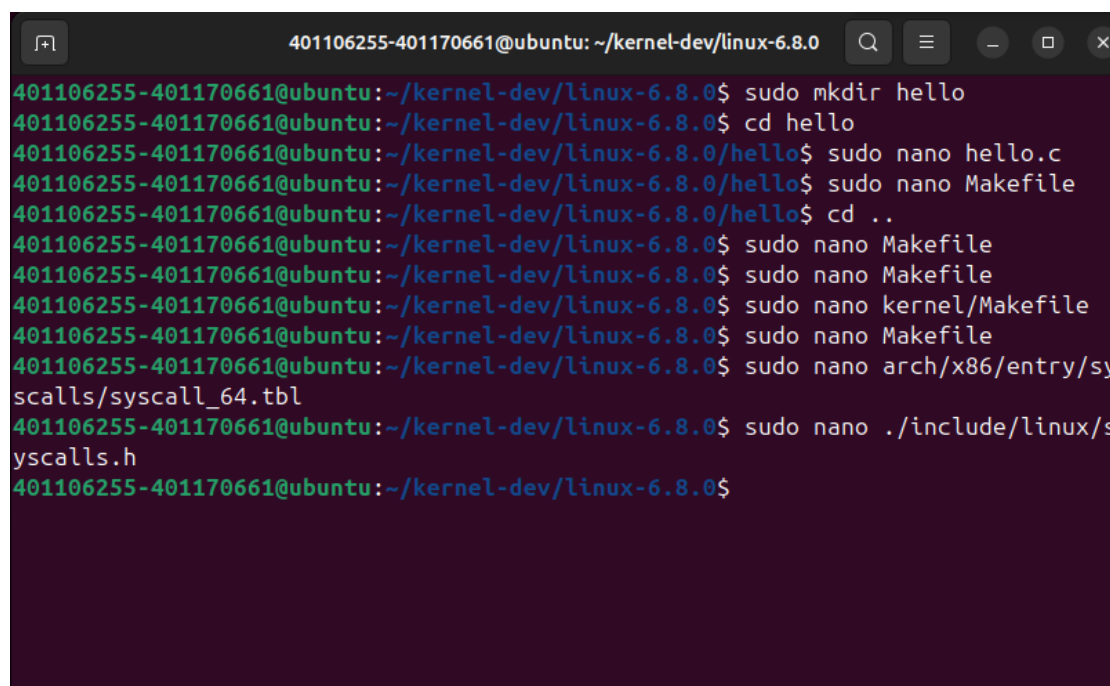
#endif /* CONFIG_ARCH_HAS_SYSCALL_WRAPPER */

asmlinkage long sys_ni_posix_timers(void);

asmlinkage long sys_hello(void);

/*
 * Kernel code should not call syscalls (i.e., sys_xyzzyz()) directly.
 * Instead, use one of the functions which work equivalently, such as
```

شکل ۱۲: افزودن خط بالا در فایل `syscalls.h`



```
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo mkdir hello
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ cd hello
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0/hello$ sudo nano hello.c
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0/hello$ sudo nano Makefile
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0/hello$ cd ..
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano Makefile
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano Makefile
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano kernel/Makefile
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano Makefile
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano arch/x86/entry/syscalls/syscall_64.tbl
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano ./include/linux/syscalls.h
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$
```

شکل ۱۳: کل دستورات استفاده شده در این بخش

۹. کامپایل و نصب مجدد هسته

make olddefconfig

make -j6

sudo make modules_install

sudo make install

```
401106255-401170661@ubuntu: ~/kernel-dev/linux-6.8.0
yscalls.h
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo make oldconfig
#
# No change to .config
#
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo make -j6
mkdir -p /home/401106255-401170661/kernel-dev/linux-6.8.0/tools/objtool && make
O=/home/401106255-401170661/kernel-dev/linux-6.8.0 subdir=tools/objtool --no-print-directory -C objtool
mkdir -p /home/401106255-401170661/kernel-dev/linux-6.8.0/tools/bpf/resolve_btfids && make O=/home/401106255-401170661/kernel-dev/linux-6.8.0 subdir=tools/bpf/resolve_btfids --no-print-directory -C bpf/resolve_btfids
SYSHDR arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_x32.h
SYSHDR arch/x86/include/generated/asm/unistd_64_x32.h
SYSTBL arch/x86/include/generated/asm/syscalls_64.h
INSTALL libsubcmd_headers
INSTALL libsubcmd_headers
CC arch/x86/kernel/asm-offsets.s
CALL scripts/checksyscalls.sh
```

شکل ۱۴: کامپایل و نصب مجدد هسته

```
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>

#define __NR_hello 462

int main() {
    long res = syscall(__NR_hello);
    printf("Syscall returned: %ld\n", res);
    return 0;
}
```

```
401106255-401170661@ubuntu:~$ gcc test.c -o test
401106255-401170661@ubuntu:~$ ./test
Syscall returned: 0
401106255-401170661@ubuntu:~$ sudo nano test.c
[sudo] password for 401106255-401170661:
401106255-401170661@ubuntu:~$
```

```

[ 38.566421] audit: type=1400 audit(1753260963.601:189): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="snap.snapd-desktop-integration.snapd-desktop-integration" pid=1825 comm="apparmor_parser"
[ 41.746566] systemd-journald[295]: /var/log/journal/f4cfc388105d41648a3aa43e38bd239d/user-1000.journal: Journal file uses a different sequence number ID, rotating.
[ 43.744911] rfkill: input handler enabled
[ 45.112522] systemd-journald[295]: Time jumped backwards, rotating.
[ 47.570192] rfkill: input handler disabled
[ 81.545301] Hello World!
401106255-401170661@ubuntu:~$
```

شکل ۱۵: تست hello

افزون فراخوانی سیستم adder:

مراحل افزودن فراخوانی سیستمی:

۱. با دسترسی root وارد سیستم عامل شده
۲. وارد پوشه‌ی کد منبع هسته سیستم عامل شده
۳. ایجاد پوشه خالی adder
۴. در این پوشه ایجاد فایل adder.c که شامل کد فراخوانی سیستمی است.

```
GNU nano 7.2                                adder.c
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE2(adder, int, a, int, b)
{
    return a + b;
}
```

شکل ۱۶: کد adder.c

۵. ایجاد فایل Makefile

۶. تغییر فایل Makefile موجود در ریشه‌ی کد منبع را باز کنید.

```
GNU nano 7.2                                Makefile *
endif

export KBUILD_MODULES KBUILD_BUILTIN

ifdef need-config
include include/config/auto.conf
endif

ifeq ($(KBUILD_EXTMOD),)
# Objects we will link into vmlinux / subdirs we need to visit
_o_uring/ hello/ adder/
drivers-y      :=
libs-y         := lib/
endif # KBUILD_EXTMOD

# The all: target is the default when no target is given on the
# command line.
# This allow a user to issue only 'make' to build a kernel including modules
# Defaults to vmlinux, but the arch makefile usually adds further targets
all: vmlinux
File Name to Write: Makefile
^G Help      ^M-D DOS Format  ^M-A Append      ^M-B Backup File
^C Cancel    ^M-M Mac Format  ^M-P Prepend     ^T Browse
```

شکل ۱۷: محتوای Makefile

۷. به روزرسانی جدول فراخوانی سیستمی nano arch/x86/entry/syscalls/syscall_64.tbl

```
GNU nano 7.2 arch/x86/entry/syscalls/syscall_64.tbl *
448 common process_mrelease sys_process_mrelease
449 common futex_waitv sys_futex_waitv
450 common set_mempolicy_home_node sys_set_mempolicy_home_node
451 common cachestat sys_cachestat
452 common fchmodat2 sys_fchmodat2
453 64 map_shadow_stack sys_map_shadow_stack
454 common futex_wake sys_futex_wake
455 common futex_wait sys_futex_wait
456 common futex_requeue sys_futex_requeue
457 common statmount sys_statmount
458 common listmount sys_listmount
459 common lsm_get_self_attr sys_lsm_get_self_attr
460 common lsm_set_self_attr sys_lsm_set_self_attr
461 common lsm_list_modules sys_lsm_list_modules

#
462 common hello sys_hello
463 common adder sys_adder

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

شکل ۱۸: به روزرسانی جدول فراخوانی سیستمی

۸. تغییر محتوای فایل ./include/linux/syscalls.h

```
GNU nano 7.2 ./include/linux/syscalls.h
* not implemented -- see kernel/sys_ni.c
*/
asmlinkage long sys_ni_syscall(void);

#endif /* CONFIG_ARCH_HAS_SYSCALL_WRAPPER */

asmlinkage long sys_ni_posix_timers(void);

asmlinkage long sys_hello(void);
asmlinkage long sys_adder(void);
```

شکل ۱۹: افزودن خط بالا در فایل syscalls.h

```

401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo mkdir adder
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ cd adder
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0/adder$ sudo nano adder.c
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0/adder$ sudo nano Makefile
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0/adder$ sudo nano kernel/Makefile
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0/adder$ cd ..
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano kernel/Makefile
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano arch/x86/entry/syscalls/syscall_64.tbl
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano Makefile
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo nano ./include/linux/syscalls.h

```

شکل ۲۰: کل دستورات استفاده شده در این بخش

۹. کامپایل و نصب مجدد هسته

make olddefconfig

make -j6

sudo make modules_install

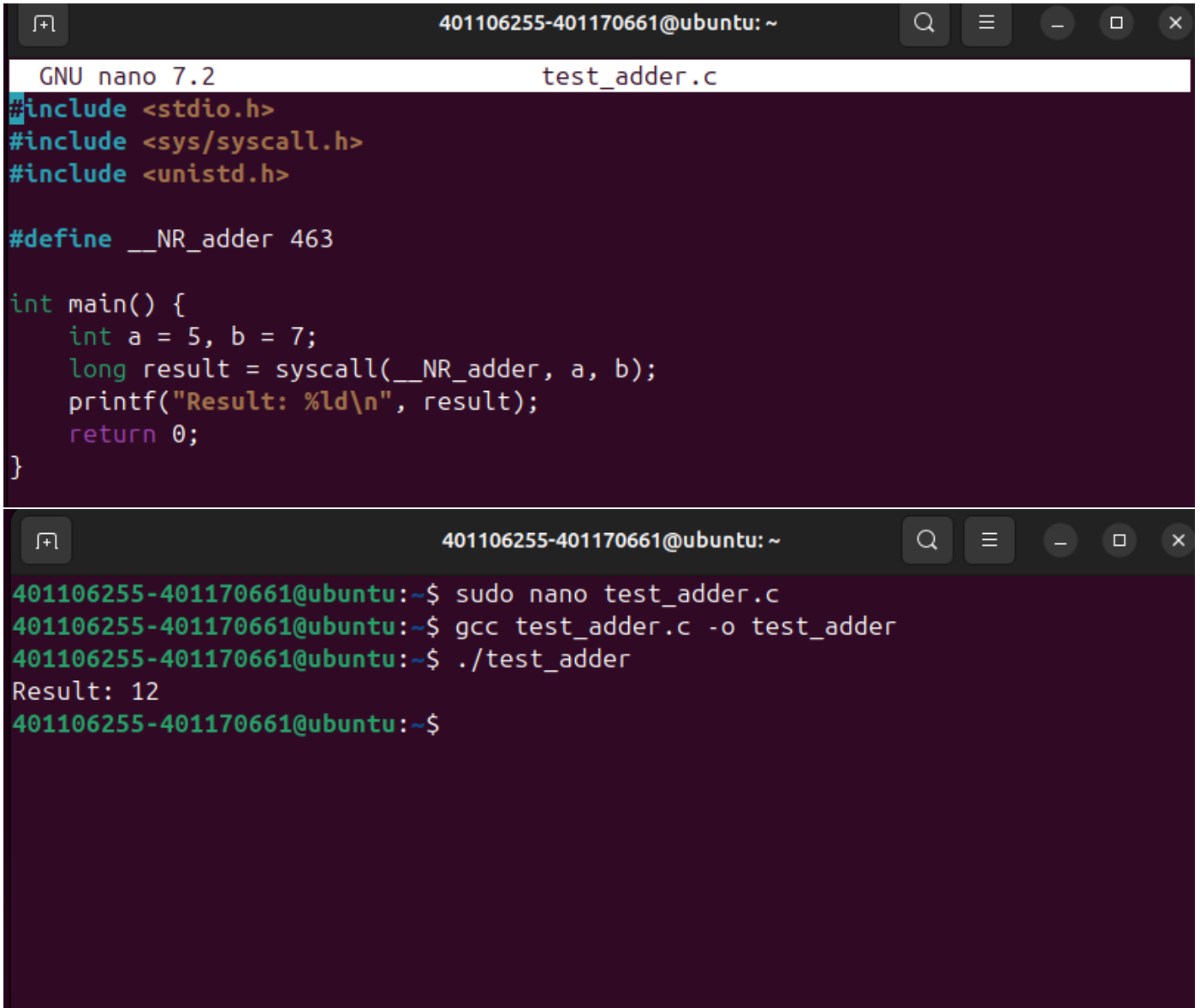
sudo make install

```

401106255-401170661@ubuntu: ~/kernel-dev/linux-6.8.0
yscalls.h
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo make oldconfig
#
# No change to .config
#
401106255-401170661@ubuntu:~/kernel-dev/linux-6.8.0$ sudo make -j6
mkdir -p /home/401106255-401170661/kernel-dev/linux-6.8.0/tools/objtool && make
O=/home/401106255-401170661/kernel-dev/linux-6.8.0 subdir=tools/objtool --no-print-directory -C objtool
mkdir -p /home/401106255-401170661/kernel-dev/linux-6.8.0/tools/bpf/resolve_btfids && make O=/home/401106255-401170661/kernel-dev/linux-6.8.0 subdir=tools/bpf/resolve_btfids --no-print-directory -C bpf/resolve_btfids
SYSHDR arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_x32.h
SYSHDR arch/x86/include/generated/asm/unistd_64_x32.h
SYSTBL arch/x86/include/generated/asm/syscalls_64.h
INSTALL libsubcmd_headers
INSTALL libsubcmd_headers
CC arch/x86/kernel/asm-offsets.s
CALL scripts/checksyscalls.sh

```

شکل ۲۱: کامپایل و نصب مجدد هسته



```
GNU nano 7.2 test_adder.c
#include <stdio.h>
#include <sys/syscall.h>
#include <unistd.h>

#define __NR_adder 463

int main() {
    int a = 5, b = 7;
    long result = syscall(__NR_adder, a, b);
    printf("Result: %ld\n", result);
    return 0;
}
```

```
401106255-401170661@ubuntu: ~$ sudo nano test_adder.c
401106255-401170661@ubuntu: ~$ gcc test_adder.c -o test_adder
401106255-401170661@ubuntu: ~$ ./test_adder
Result: 12
401106255-401170661@ubuntu: ~$
```

شکل ۲۲: تست adder