



دانشگاه جهرم

اضافه کردن یک فراخوان سیستمی به هسته‌ی لینوکس

نگارش:

امیررضا ارجمند

استاد:

دکتر محمدجواد پارسه

شماره دانشجویی:

۹۹۳۱۲۲۰۱

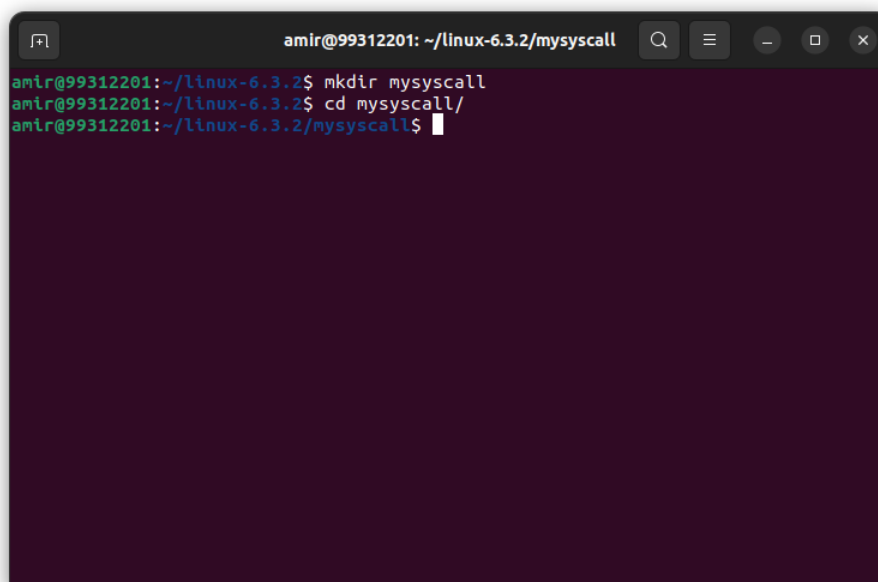
۱. آماده سازی

۱/۱. همانند گزارش کار قبلی، هسته‌ی لینوکس را دانلود کرده و پکیج‌های مورد نیاز در سیستم خود را به‌روز می‌کنیم. در اینجا از نسخه‌ی ۶/۳/۲ استفاده می‌کنیم.

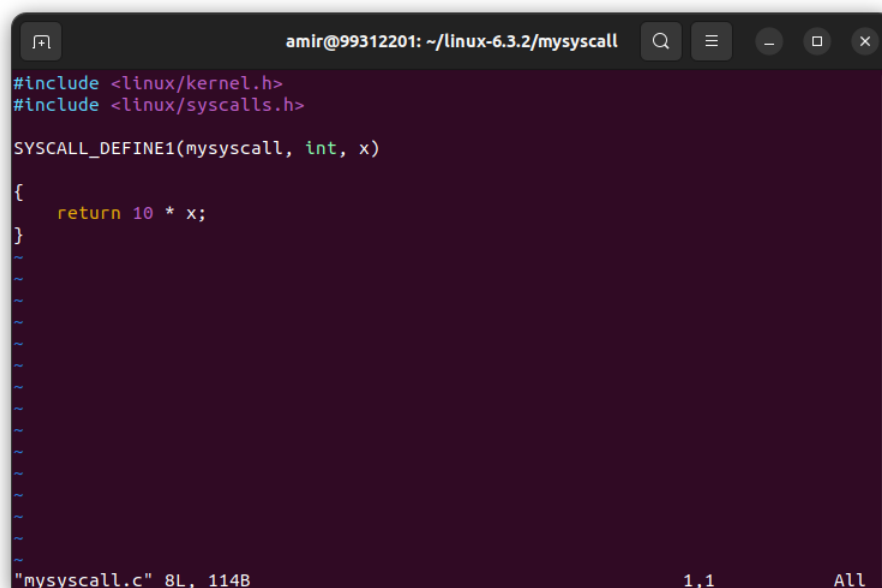


۲. اضافه کردن فراخوان سیستمی

۲/۱. پس از استخراج فایل‌های هسته با دستور `cd linux-6.3.2/` به شاخه‌ی استخراج شده می‌رویم و پوشه‌ی جدیدی می‌سازیم. این پوشه باید شامل ۲ فایل باشد که در آن‌ها کدهای مربوط به فراخوان سیستمی جدید قرار می‌گیرد. در اینجا نام فراخوان سیستمی جدید `mysyscall` در نظر گرفته شده است و نام پوشه جدید نیز به همین شکل نام گذاری شده است.



۲/۲. در اینجا نیاز به ساخت دو فایل داریم که یکی شامل کد نوشته شده به زبان C می‌باشد و دیگری یک Makefile است که در آن نحوه‌ی کامپایل شدن فایل C درج شده است. فایل C را با نام `mysyscall.c` می‌سازیم و در آن کد زیر را قرار می‌دهیم.

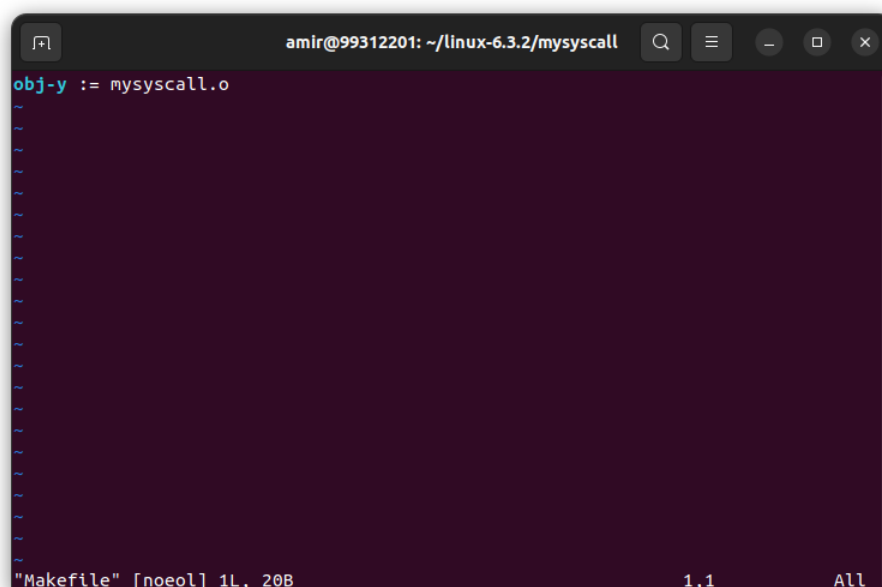


```
amir@99312201: ~/linux-6.3.2/mysyscall
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE1(mysyscall, int, x)
{
    return 10 * x;
}

"mysyscall.c" 8L, 114B 1,1 All
```

۲/۳. بعد از ساخته شدن فایل C، Makefile را می‌سازیم و در آن نیز دستورات مورد نیاز زیر را برای کامپایل شدن کد C قرار می‌دهیم.



```
amir@99312201: ~/linux-6.3.2/mysyscall
obj-y := msyscall.o

"Makefile" [noeol] 1L, 20B 1,1 All
```

۲/۴. حال نوبه ایجاد تغییر در فایل‌های موجود در سورس هسته می‌باشد. در شاخه‌ی اصلی فایل Kbuild را باز می‌کنیم و در انتهای آن کد زیر را درج می‌کنیم.

```
amir@99312201: ~/linux-6.3.2
obj-y      += init/
obj-y      += usr/
obj-y      += arch/${SRCARCH}/
obj-y      += $(ARCH_CORE)
obj-y      += kernel/
obj-y      += certs/
obj-y      += mm/
obj-y      += fs/
obj-y      += ipc/
obj-y      += security/
obj-y      += crypto/
obj-$(CONFIG_BLOCK) += block/
obj-$(CONFIG_IO_URING) += io_uring/
obj-$(CONFIG_RUST) += rust/
obj-y      += $(ARCH_LIB)
obj-y      += drivers/
obj-y      += sound/
obj-$(CONFIG_SAMPLES) += samples/
obj-$(CONFIG_NET) += net/
obj-y      += virt/
obj-y      += $(ARCH_DRIVERS)
obj-y      += msyscall/
```

۲/۵. حال فایل موجود در آدرس include/linux/syscalls.h را باز می‌کنیم و امضای فراخوان سیستمی خود را به انتهای آن قبل از خط #endif اضافه می‌کنیم.

```
amir@99312201: ~/linux-6.3.2
long ksys_msgget(key_t key, int msgflg);
long ksys_old_msgctl(int msqid, int cmd, struct msqid_ds __user *buf);
long ksys_msgrcv(int msqid, struct msgbuf __user *msgp, size_t msgsz,
    long msgtyp, int msgflg);
long ksys_msgsnd(int msqid, struct msgbuf __user *msgp, size_t msgsz,
    int msgflg);
long ksys_shmget(key_t key, size_t size, int shmflg);
long ksys_shmctl(char __user *shmaddr);
long ksys_old_shmctl(int shmid, int cmd, struct shmid_ds __user *buf);
long compat_ksys_semtimeop(int semid, struct sembuf __user *tsems,
    unsigned int nsops,
    const struct old_timespec32 __user *timeout);
long __do_semtimedop(int semid, struct sembuf *tsems, unsigned int nsops,
    const struct timespec64 *timeout,
    struct ipc_namespace *ns);

int __sys_getsockopt(int fd, int level, int optname, char __user *optval,
    int __user *optlen);
int __sys_setsockopt(int fd, int level, int optname, char __user *optval,
    int optlen);

asmlinkage long sys_msyscall(int x);
#endif
```

۲/۶. سپس فایل موجود در آدرس arch/x86/entry/syscalls/syscall_64.tbl را باز می‌کنیم. وظیفه‌ی این فایل نگه داشتن جدولی از فراخوان‌های سیستمی موجود در سیستم می‌باشد. به انتهای آن قبل از بخش سیستم‌کال‌های x32 مشخصات سیستم‌کال خود را اضافه می‌کنیم. این مشخصات شامل آی‌دی سیستم‌کال، معماری مورد استفاده‌ی آن، نام آن در سیستم و نام تابع آن می‌باشد. مطمئن شوید که این مشخصات با استفاده از کاراکتر tab از یکدیگر جدا شده‌اند.

```
amir@99312201: ~/linux-6.3.2
441 common epoll_pwait2 sys_epoll_pwait2
442 common mount_setattr sys_mount_setattr
443 common quotactl_fd sys_quotactl_fd
444 common landlock_create_ruleset sys_landlock_create_ruleset
445 common landlock_add_rule sys_landlock_add_rule
446 common landlock_restrict_self sys_landlock_restrict_self
447 common memfd_secret sys_memfd_secret
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 msyscall sys_msyscall

#
# 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.
#
512 x32 rt_sigaction compat_sys_rt_sigaction
513 x32 rt_sigreturn compat_sys_x32_rt_sigreturn
514 x32 ioctl compat_sys_ioctl
515 x32 readv sys_readv
516 x32 writev sys_writev

387,1 91%
```

۳. کامپایل و نصب هسته‌ی جدید

۳/۱. پس از اضافه کردن فراخوان سیستمی جدید، همانند گزارش کار قبلی هسته را کانفیگ، کامپایل و سپس نصب می‌کنیم. در صورت موفقیت آمیز بودن نصب پس از راه‌اندازی مجدد سیستم باید سیستم‌کال جدید به آن اضافه شده باشد.

```
amir@99312201: ~
amir@99312201:~$ uname -a
Linux 99312201 6.3.2 #1 SMP PREEMPT_DYNAMIC Wed May 17 00:06:46 +0330 2023 x86_64
4 x86_64 x86_64 GNU/Linux
amir@99312201:~$
```

۴. تست فراخوان سیستمی

۴/۱. با استفاده از تابع `syscall` که در `<unistd.h>` تعریف شده می‌توانیم سیستم‌کال جدید را تست کنیم. این تابع آیدی عددی فراخوان را علاوه بر همه‌ی ورودی‌ها دریافت می‌کند و خروجی آن را به ما برمی‌گرداند.

