Linux Kernel Compilation

Prof. Yongtae Kim

Computer Science and Engineering
Kyungpook National University

Linux Kernel Download

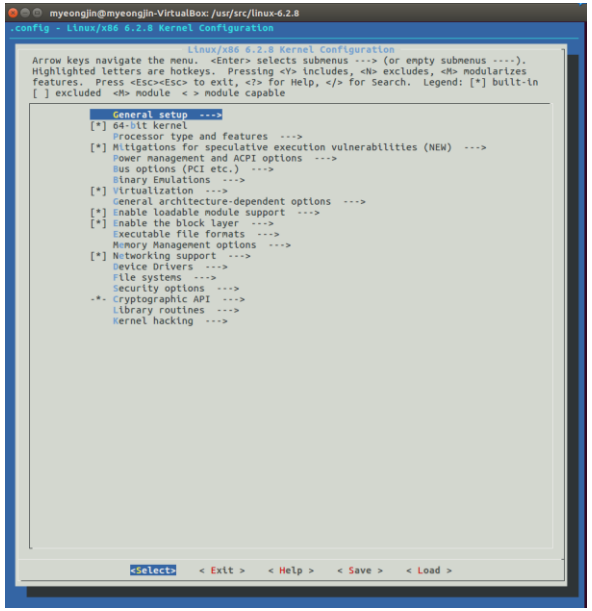
- **Please conduct the kernel compilation on the Linux installed in VirtualBox**

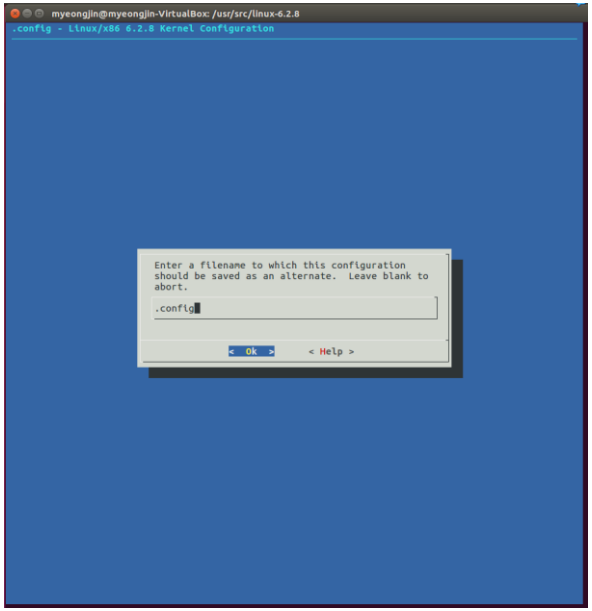
- **Install packages for kernel compilation**
  - $ sudo apt-get update
  - $ sudo apt-get install build-essential libncurses5-dev bison flex libssl-dev libelf-dev vim -y

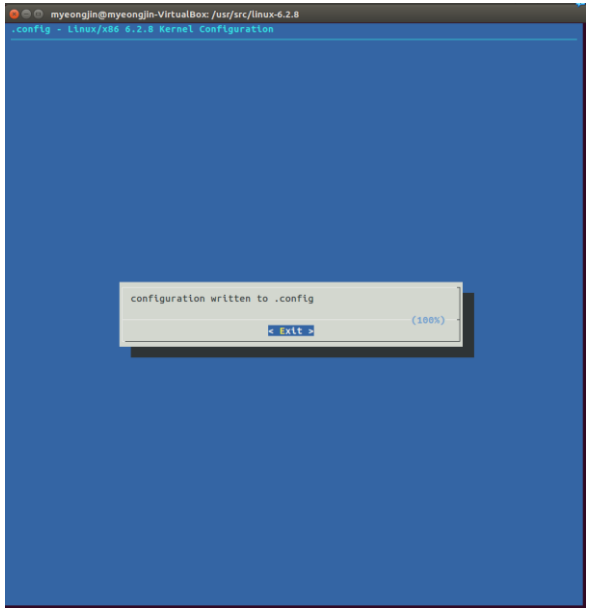
- **Download kernel source (6.2.8 – latest release, Mar. 23, 2022)**
  - $ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.2.8.tar.xz
  - $ sudo cp linux-6.2.8.tar.xz /usr/src
  - $ cd /usr/src

Configuration for Kernel Compilation (1)

- **Decompress the downloaded kernel source**
  - $ sudo tar -xvf linux-6.2.8.tar.xz
  - $ cd linux-6.2.8
- **Configure**
  - $ sudo make menuconfig
  - Just press enter with “Save → Ok → Exit → Exit”







Configuration for Kernel Compilation (2)

- **$ls –al**

```
myeongjin@myeongjin-VirtualBox:/usr/src/linux-6.2.8$ ls -al
합계 1056
drwxrwxr-x  26 root root   4096  3월 22 21:38 .
drwxr-xr-x   7 root root   4096  3월 31 20:15 ..
-rw-rw-r--   1 root root  20523  3월 22 21:38 .clang-format
-rw-rw-r--   1 root root     59  3월 22 21:38 .cocciconfig
-rw-rw-r--   1 root root    151  3월 22 21:38 .get_maintainer.ignore
-rw-rw-r--   1 root root     62  3월 22 21:38 .gitattributes
-rw-rw-r--   1 root root   2061  3월 22 21:38 .gitignore
-rw-rw-r--   1 root root  25665  3월 22 21:38 .mailmap
-rw-rw-r--   1 root root    369  3월 22 21:38 .rustfmt.toml
-rw-rw-r--   1 root root    496  3월 22 21:38 COPYING
-rw-rw-r--   1 root root 102088  3월 22 21:38 CREDITS
drwxrwxr-x  88 root root   4096  3월 22 21:38 Documentation
```

Before

  - .config file will be created

```
myeongjin@myeongjin-VirtualBox:/usr/src/linux-6.2.8$ ls -al
합계 1316
drwxrwxr-x  26 root root   4096  3월 31 20:20 .
drwxr-xr-x   7 root root   4096  3월 31 20:15 ..
-rw-rw-r--   1 root root  20523  3월 22 21:38 .clang-format
-rw-rw-r--   1 root root     59  3월 22 21:38 .cocciconfig
-rw-r--r--   1 root root 265400  3월 31 20:20 .config
-rw-rw-r--   1 root root    151  3월 22 21:38 .get_maintainer.ignore
-rw-rw-r--   1 root root     62  3월 22 21:38 .gitattributes
-rw-rw-r--   1 root root   2061  3월 22 21:38 .gitignore
-rw-rw-r--   1 root root  25665  3월 22 21:38 .mailmap
-rw-rw-r--   1 root root    369  3월 22 21:38 .rustfmt.toml
-rw-rw-r--   1 root root    496  3월 22 21:38 COPYING
-rw-rw-r--   1 root root 102088  3월 22 21:38 CREDITS
drwxrwxr-x  88 root root   4096  3월 22 21:38 Documentation
```

After

Compiling Linux Kernel

- **Configure**
  - $ sudo scripts/config --disable SYSTEM_TRUSTED_KEYS
  - $ sudo scripts/config --disable SYSTEM_REVOCATION_KEYS

- **Compile the kernel source**
  - $ sudo make –j6 # (6 = # of CPU core)

```
Additional X.509 keys for default system keyring (SYSTEM_TRUSTED_KEYS) [] (NEW)
```

    - If you see this kind of message (… “NEW”), just press the enter to proceed
    - This command makes you to compile the kernel and this process will take a long time (few tens of minutes to hours; it depends on your system)
  - $ sudo make modules
  - $ sudo make modules_install
  - $ sudo make install

Start Your System with Compiled Kernel

- **$ uname –r**
  - This command show you the current kernel version information

```
myeongjin@myeongjin:/usr/src/linux-6.2.8$ uname -r
5.19.0-38-generic
```

- **Grub update and environmental settings**
  - $ sudo update-grub
  - $ sudo vi /etc/default/grub
  - Comment out “GRUB_HIDDEN_TIMEOUT=0” if exists

```
GRUB_DEFAULT=0
#GRUB_TIMEOUT_STYLE=hidden
#GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
```

    - Use # to comment the line out

- **Reboot**
  - You will see the new kernel version!

```
myeongjin@myeongjin:~/Desktop$ uname -r
6.2.8
```

System Call (1)

- **Make a user defined system call**
  - $ cd /usr/src/linux-6.2.8/kernel
  - $ sudo vi **proj1.c** //open a new file for system call implementation

```
#include <linux/kernel.h>
#include <linux/linkage.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(proj1call)
{
        printk("COMP0312_OS_PROJ1_2018000001_GildongHong: Hello Kernel");
        return 0;

}
```

  - Here, SYSCALL_DEFINE0 ➔ "0" indicates # of parameters
  - $ sudo vi Makefile
  - Add **proj1.o** as follows:

```
obj-y     = fork.o exec_domain.o panic.o \
            cpu.o exit.o softirq.o resource.o \
            sysctl.o capability.o ptrace.o user.o \
            signal.o sys.o umh.o workqueue.o pid.o task_work.o \
            extable.o params.o \
            kthread.o sys_ni.o nsproxy.o \
            notifier.o ksysfs.o cred.o reboot.o \
            async.o range.o smpboot.o ucount.o regset.o proj1.o
```

System Call (2)

- **Make a user defined system call (cont'd)**
  - $ cd /usr/src/linux-6.2.8/arch/x86/entry/syscalls
  - $ sudo vi syscall_64.tbl
    - Add your system call
    - SysCall# common **NameOfSysCall** functionName

```
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  proj1call               sys_proj1call
```

System Call (3)

- **Make a user defined system call (cont'd)**
  - $ cd /usr/src/linux-6.2.8/include/linux/
  - $ sudo vi syscalls.h
  - Add system call after #endif /*CONFIG_ARCH_HAS_SYSCALL_WRAPPER */

```
#include <linux/types.h>
#include <linux/aio_abi.h>
#include <linux/capability.h>
#include <linux/signal.h>
#include <linux/list.h>
#include <linux/bug.h>
#include <linux/sem.h>
#include <asm/siginfo.h>
#include <linux/unistd.h>
#include <linux/quota.h>
#include <linux/key.h>
#include <linux/personality.h>
#include <trace/syscall.h>

#ifdef CONFIG_ARCH_HAS_SYSCALL_WRAPPER
/*
 * It may be useful for an architecture to override the definitions of the
 * SYSCALL_DEFINE0() and __SYSCALL_DEFINEx() macros, in particular to use a
 * different calling convention for syscalls. To allow for that, the prototypes
 * for the sys_*() functions below will *not* be included if
 * CONFIG_ARCH_HAS_SYSCALL_WRAPPER is enabled.
 */
#include <asm/syscall_wrapper.h>
#endif /* CONFIG_ARCH_HAS_SYSCALL_WRAPPER */

asmlinkage long proj1call(void);
```

System Call (3)

- **Re-compile kernel; option #1 (slow)**
  - $ cd /usr/src/linux-6.2.8/
  - $ sudo make –j6
  - $ sudo make install

- **Re-compile kernel; option #2 (fast)**
  - $ cd /usr/src/linux-6.2.8/
  - $ sudo make bzImage –j6
  - $ sudo cp /usr/src/linux-6.2.8/arch/x86/boot/bzImage /boot/vmlinuz-6.2.8

System Call (4)

- **Reboot and write test code**
  - $ vi test_proj1.c

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

#define SYSCALL_NUM 451

int main()
{
        long int res = syscall(SYSCALL_NUM);
        printf("sys_proj1call returned: %ld\n", res);
        return 0;
}
```

  - $ gcc test_proj1.c -o test
  - $ ./test

```
myeongjin@myeongjin:~/Desktop$ ./test
sys_proj1call returned: 0
```

  - $ sudo dmesg (to confirm your system call works well)
    - If you don't see the kernel message by dmesg, run ./test once more and check

```
[   99.044356] COMP0312_OS_PROJ1_2018000001_GildongHong: Hello Kernel
```

OS Project 1

- **Linux Kernel Compilation**
  - Install Linux on VirtualBox
  - Compile a new kernel
  - Writing your own system call
- **Submission Due**
  - Due: 4/30, Sunday 23:59
  - No late submission is allowed
- **What to Submit**
  - System call files: proj1.c, test_proj1.c, syscall_64.tbl, syscalls.h
    - Please make a single tarball (.tgz or .tar) to include these files in your Linux
  - Screen captures: system call test and dmesg results
    - please see previous slide and make a single PDF to include both captures
    - System call's kernel message must be:
      COMP0312_OS_PROJ1_yourStudentID_yourName: Hello Kernel
- **Grading**
  - Total: 30 pts