Assignment I - C...

HackMD (https://hackmd.io?utm_source=view-page&utm_medium=logo-nav)

Assignment I - Compiling a Custom Linux Kernel & Adding New System Calls

NYCU [CSIC30015] Operating System (https://timetable.nycu.edu.tw/?r=main/crsoutline&Acy=113&Sem=1&CrsNo=535500&lang=zh-tw) by Prof. Chun-Feng Wu (https://www.cs.nycu.edu.tw/members/detail/cfwu417)

Due Date: 23:59, Monday, October 28, 2024

Table of Contents

Goal of the assignment

- Compile a custom Linux kernel from source.
- Add a new Linux system call.

- Create a kernel patch.

Introduction

This semester, all three hands-on projects will involve working with the Linux kernel.

Kernel Documentation

The Linux kernel is open source and well-documented. The documentation can be found in the `Documentation/` directory at the root of the kernel source tree. You can build the documentation using the `make htmldocs` or `make pdfdocs` commands. Alternatively, there is an online version for kernel **v6.1** available here (https://www.kernel.org/doc/html/v6.1/index.html#) for all our assignments. Documentation for many other kernel versions can be found here (https://www.kernel.org/doc/html/), with the latest version available here (https://www.kernel.org/doc/html/latest/).

Additionally, different kernel versions' online source code is provided by Bootlin (https://elixir.bootlin.com/linux/v6.11-rc6/source), which allows you to easily browse and trace the code through their website.

Please feel free to ask questions via email or on the forum if you have problem finding out the solution from the documentation provided.

Environment Set-Up

1. Installing a Linux Distribution

For this assignment, you will need to compile the Linux kernel on an **Ubuntu 24.04 AMD64** system. You can download the Ubuntu image from the official site (https://releases.ubuntu.com/noble/). Both desktop and server versions are available; you may choose either based on your preference.

> **For Mac ARM64 Users:** If you have no **X86** machine, we can create a remote VM for you. Please email `os.oscarlab@gmail.com` to request a remote VM.

You can work on a native Linux system or use a Linux guest on a hypervisor such as VMware (https://www.vmware.com/), VirtualBox (https://www.virtualbox.org/). Working within a guest VM adds an additional layer of isolation and safety, ensuring that your main system is protected from any potential issues.

If you have no experience on installing Linux before, you may want to check this tutorial (https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox#1-overview) to see how to run Ubuntu on a VM using VirtualBox.

> **Tip 1:** Ensure the disk size is at least **50GB** to avoid encountering errors such as "No space left on the device" during kernel compilation. For more informations, see Appendix

> **Tip2:** We also recommend setting the number of vCPU cores in the VM to at least 4, or you may spend a lots of time compiling the kernel.

> **Remark:** While you can use other Linux distributions, such as Fedora, openSUSE, Linux Mint, etc., we strongly recommend using **Ubuntu 24.04 AMD64**. Although this assignment should be Linux distribution-neutral, using Ubuntu 24.04 will ensure compatibility and prevent potential issues during grading. If the TA encounters compatibility issues with other distributions, it may result in **point deductions**.

2. Download the Linux kernel source

Here, We clone the mainline Linux Git repository (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git) (i.e., Linus Torvalds' Git tree), which can be downloaded using `git` as follows:

```
$ sudo apt update
$ sudo apt install git
$ git clone https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
```

To get the commit log of the tags in the kernel Git tree, use the `git log` command as follows:

```
$ cd linux
$ git log --oneline --tags --simplify-by-decoration
```

The result will look like:

```
431c1646e1f8 (HEAD, tag: v6.11-rc6) Linux 6.11-rc6
5be63fc19fca (tag: v6.11-rc5) Linux 6.11-rc5
47ac09b91bef (tag: v6.11-rc4) Linux 6.11-rc4
7c626ce4bae1 (tag: v6.11-rc3) Linux 6.11-rc3
de9c2c66ad8e (tag: v6.11-rc2) Linux 6.11-rc2
8400291e289e (tag: v6.11-rc1) Linux 6.11-rc1
8400291e289e (tag: v6.11-rc1) Linux 6.11-rc1
0c3836482481 (tag: v6.10) Linux 6.10
256abd8e550c (tag: v6.10-rc7) Linux 6.10-rc7
22a40d14b572 (tag: v6.10-rc6) Linux 6.10-rc6
f2661062f16b (tag: v6.10-rc5) Linux 6.10-rc5
6ba59ff42279 (tag: v6.10-rc4) Linux 6.10-rc4
83a7eefedc9b (tag: v6.10-rc3) Linux 6.10-rc3
c3f38fa61af7 (tag: v6.10-rc2) Linux 6.10-rc2
1613e604df0c (tag: v6.10-rc1) Linux 6.10-rc1
a38297e3fb01 (tag: v6.9) Linux 6.9
dd5a440a31fa (tag: v6.9-rc7) Linux 6.9-rc7
e67572cd2204 (tag: v6.9-rc6) Linux 6.9-rc6
[...]
88603b6dc419 (tag: v6.2-rc2) Linux 6.2-rc2
1b929c02afd3 (tag: v6.2-rc1) Linux 6.2-rc1
830b3c68c1fb (tag: v6.1) Linux 6.1
76dcd734eca2 (tag: v6.1-rc8) Linux 6.1-rc8
b7b275e60bcd (tag: v6.1-rc7) Linux 6.1-rc7
[...]
```

For this assignment, you need to compile the release of the **6.1 LTS kernel**. Use the `git checkout` command to switch to the specific commit tagged as v6.1:

```
$ git checkout v6.1
```

> **Alternative:** Downloading the full Linux git tree is time-consuming, so you can download a specific version of the kernel using the following options: `--depth=1 --branch v6.1 --single-branch` passed to the `git clone` command to speed up the download time and save disk space.

To verify the kernel version you are working with, check the first few lines of the Makefile in the root of the kernel source tree (also known as the *top-level Makefile*):

```
$ head Makefile -n 5
```

The result should be:

```
# SPDX-License-Identifier: GPL-2.0
VERSION = 6
PATCHLEVEL = 1
SUBLEVEL = 0
EXTRAVERSION =
```

For more details about Git, please refer to Git Official website (https://git-scm.com/).

> **Remark:** You must use the above method to download the Linux source. Do not download the specific kernel version tarball, as it may cause issues when you submit the Linux patch.

Requirements

I. Compiling the Linux Kernel

1. Kernel Compilation (Do It Yourself !)

Now, begin your journey of compiling the kernel!
The required kernel source version is **v6.1 LTS kernel** (or **v6.1.0**), which is a Long-Term Support (LTS) maintained until December 2026. Moreover, the Civil Infrastructure Project (CIP) has adopted v6.1 as an SLTS (Super LTS) and plan to maintain it for 10 years, until August 2033.

We will not go through the kernel compilation steps here since there are numerous resources and documents available online.

2. Change kernel local version

Before compiling the kernel, you need to modify the kernel local version suffix to `-os-<student ID>` (e.g. `-os-312551000`). This will serve as evidence that the kernel was built by you. A sample screenshot is provided below for your reference.

3. Requirement

- Here is an example of a successful kernel compilation by showing the results of `uname -a` and `cat /etc/os-release` commands:

```
osta@osta-VM:~$ uname -a
Linux osta-VM 6.1.0-os-312551000 #1 SMP PREEMPT_DYNAMIC Wed Sep  4 20:09:31 CST 2024 x86_64 x86_64 x86_64 GNU/Linux
osta@osta-VM:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04.1 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.1 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
```

- **Include the above screenshot in your report** and describe the steps you followed during the kernel compilation process.

Before you actually install the kernel you build in to the VM, you may want use `make kernelrelease` to check whether your image is meet our requirement. The result would be like:

```
6.1.0-os-312551000
```

> **Remark:** If you do not include the above screenshot in your report, this item may receive a score of **zero**.

II. Implementing a new System Calls

In this section, you are required to implement a new Linux system calls in the kernel **v6.1** you have built: `sys_revstr` .

For guidance on adding system calls, refer to the kernel documentation (https://www.kernel.org/doc/html/latest/process/adding-syscalls.html#generic-system-call-implementation). You may check syscall(2) (https://man7.org/linux/man-pages/man2/syscall.2.html) to learn how to use `syscall` in linux system.

1. `sys_revstr`

- **SYNOPSIS:**

```
#include <unistd.h>

#define __NR_revstr 451
int syscall(__NR_revstr, char *str, size_t n);
```

- **DESCRIPTION:**
  You are required to implement a system call named `sys_revstr` , which accepts two arguments: a string and its length. This system call should reverse the string and perform the following actions:
  1. Print a line "The origin string: `string` " to the kernel ring buffer, where `string` is the string the user passed in.
  2. Reverse the given string.
  3. Print a line "The reversed string: `rev_string` " to the kernel ring buffer, where `rev_string` is the resulting string you reversed.
  4. copy the result string to the user's space.
  5. Return zero to indicate successful execution.

You don't need to handle invalid arguments such as a null pointer or mismatched string length.

- **SAMPLE EXPLANATION:**
  Ensure the following user-space code can be compiled and executed on your system.

```
#include <unistd.h>

#include <string.h>
#include <stdio.h>
#include <assert.h>

#define __NR_revstr 451

int main(int argc, char *argv[]) {

    char str1[20] = "hello";
    printf("Ori: %s\n", str1);
    int ret1 = syscall(__NR_revstr, str1, strlen(str1));
    assert(ret1 == 0);
    printf("Rev: %s\n", str1);

    char str2[20] = "Operating System";
    printf("Ori: %s\n", str2);
    int ret2 = syscall(__NR_revstr, str2, strlen(str2));
    assert(ret2 == 0);
    printf("Rev: %s\n", str2);

    return 0;
}
```

The output of the above user-space program may be looks like:

```
osta@VM:~/Desktop/user_prog$ ./test_revstr
Ori: hello
Rev: olleh
Ori: Operating System
Rev: metsyS gnitarepO
```

Check the kernel ring buffer for the output messages using `dmesg` command，you may see the following output

```
[  360.951315] The origin string: hello
[  360.951318] The reversed string: olleh
[  360.951322] The origin string: Operating System
[  360.951323] The reversed string: metsyS gnitarepO
```

> 💡 **Hint:**
> The data from the user space are not shared with the kernel space (and vice versa), so you need to figure out how to transfer data between them.

4. Requirement

- **Include screenshots** of the above outputs in your machine **to your report**. Additionally, describe the implementation details for the system call.

> **Remark:** If you do not include the above two screenshots in your report, this item may receive a score of **zero**.

III. Patch

In Linux kernel development, the **email format patch** is widely used. Now it's time to create your own patch, which we will use to build the system call you implemented in the kernel.

> 💡 **Hint:**
> You can refer to the git-format-patch(1) (https://man7.org/linux/man-pages/man1/git-format-patch.1.html) documentation for more information.

> **Note:** It's fine if you have many `.patch` files, depending on the number of commits. Just simply place all the `.patch` files in the `HW1_<student ID>/` folder.

> **Warning:** Using an incorrect patch format may cause the TA to be unable to build your patch, resulting in a zero grade. Therefore, please make sure the format is correct.

IV. Package list

Since the `.config` file you set may require some packages to build the kernel you configured, use the following command to list the packages installed on your system:

```
$ dpkg --get-selections > packages_list_<studtent ID>.txt
```

Grading

Total (100%)

- Report (30%)
  - Kernel Compiliation (10%)
  - System call implementation (20%)

- Implementations (70%):
  - Kernel Local Version (20%):
    - we will use the `.config` you provide to build the kernel, and we will check whether you set the correct local version specified here.
  - Public testcase (20%):
  - Hidden testcase (30%):

> **Note:** Hidden testcase is nothing special, it just want to prevent someone from simply printing the answer.

Submissions

1. Required Files

- Report: Only `.pdf` file is accepted
  - Make sure your report meets Requirement above.
  - Filename
    - `<student ID>_report.pdf` (e.g `312551000_report.pdf`)
- Config File
  - `.config` you used to build your kernel.
- Package List
  - Filename
    - `packages_list_<studtent ID>.txt` (e.g `packages_list_312551000.txt`)
- Patch file
  - Filename
    - `0001-xxxx.patch` (e.g. `0001-Modify-Makefile.patch`)
    - `0002-xxxx.patch` (e.g. `0002-Add-the-revstr-syscall-definition.patch`)
    - `0003-xxxx.patch` (e.g. `0003-Implement-revstr-syscall.patch`)
    - ... ...
    - ... ...

Ensure you include all necessary files above for this assignment.

2. Pack the required file

All your files should be organized in the following hierarchy and zipped into a `.zip` file, named `HW1_<student ID>.zip` (e.g. `HW1_312551000.zip`)

The structure inside the zipped file should be as the following:

```
HW1_<student ID>.zip
    |- HW1_<student ID>/
        |- <student ID>_report.pdf
        |- .config
        |- packages_list_<studtent ID>.txt
        |- 0001-xxxx.patch
        |- 0002-xxxx.patch
        |- ......
        |- ......
        |- ......
```

e.g.

```
HW1_312551000.zip
    |- HW1_312551000/
        |- 312551000_report.pdf
        |- .config
        |- packages_list_312551000.txt
        |- 0001-Modify-Makefile.patch
        |- 0002-Add-the-revstr-syscall-definition.patch
        |- 0003-Implement-revstr-syscall.patch
```

> **Note:** It's fine if you have many `.patch` files, depending on the number of commits. Just simply place all the `.patch` files in the `HW1_<student ID>/` folder.

> **DO NOT** include any files or directories not explicitly specified above in your archive, such as `__MACOSX` , `.git` , `.vscode` , `.vscode-server` ,etc., as they may potentially cause our automated build system to crash.

3. Submit the archive through the E3 platform before deadline.

If you have any questions about this assignment, please feel free to contact the TA or make an appointment by emailing os.oscarlab@gmail.com (mailto:os.oscarlab@gmail.com). You may also reach out to the TA during class.

> **Warning:** Incorrect file formats will result in a 20-point deduction. Additionally, if a formatting error prevents the TA from building your kernel, you may not receive any points.

> **Plagiarism Warning:** Plagiarism is strictly prohibited. Homework assignments must be completed **individually**. Assignments found to be plagiarized will receive reduced credit or, in most cases, **no credit**.

Additonal Note

In formal kernel development, you should write a Kconfig file to allow users to decide whether to build your custom system call using menuconfig or by configuring the kernel via `scripts/config`. The configuration option might be named `NYCU_OS_AS1` or whatever you prefer. However, for this assignment, you are not required to do this. This task is an additional exercise for those interested.

For more detail about Kconfig, please refer this Kernel Document (https://www.kernel.org/doc/html/v6.1/kbuild/kconfig-language.html).

Appendix

Error: No space left on device

By default, the Ubuntu installer allocates the logical volume of the root filesystem (`/dev/ubuntu-vg/ubuntu-lv`) for only half of space of the underlying physical volume `ubuntu-vg`.

To addres this issue, you can extend the default LVM to 100%.

```
sudo lvextend -l +100%FREE /dev/ubuntu-vg/ubuntu-lv
sudo resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv
```

Please refer to https://packetpushers.net/ubuntu-extend-your-default-lvm-space/ (https://packetpushers.net/ubuntu-extend-your-default-lvm-space/) for more information.