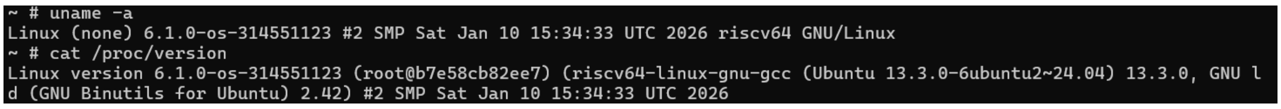
OS HW1 report

Student ID：314551123 / Name：蘇秭郁

Requirement 1

- Screenshot for uname -a and cat /proc/version



```
~ # uname -a
Linux (none) 6.1.0-os-314551123 #2 SMP Sat Jan 10 15:34:33 UTC 2026 riscv64 GNU/Linux
~ # cat /proc/version
Linux version 6.1.0-os-314551123 (root@b7e58cb82ee7) (riscv64-linux-gnu-gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0, GNU l
d (GNU Binutils for Ubuntu) 2.42) #2 SMP Sat Jan 10 15:34:33 UTC 2026
```

- Steps to compile a kernel
  1. make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- defconfig
     → The kernel will generate .config and the necessary auto.conf files required by the Makefile.
  2. make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- menuconfig
     → General setup -> Local version
  3. make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- -j$(nproc)
     → Start compiling
- Answer the Q&As
  1. What are the main differences between the RISC-V and x86 architectures?

     The main difference between RISC-V and x86 lies in their Instruction Set Architecture (ISA) types：

     x86 belongs to the CISC (Complex Instruction Set Computer). The objective of CISC is to perform complex tasks with a minimal number of instructions. As a result, CISC instructions vary in length and tend to have more complex functionalities.

     RISC-V belongs to the RISC (Reduced Instruction Set Computer). The objective of RISC are high performance and modularity, which is why RISC-V instructions have fixed lengths and simpler individual functions.

  2. Why do the architecture differences matter when building the kernel? What happens if you build the kernel without the correct RISC-V cross-compilation flag?

Since the registers, Memory Management Unit (MMU), and interrupt handling mechanisms vary significantly across different architectures, and the kernel communicates directly with the hardware, the kernel code must be compiled into machine code that the target hardware can understand.

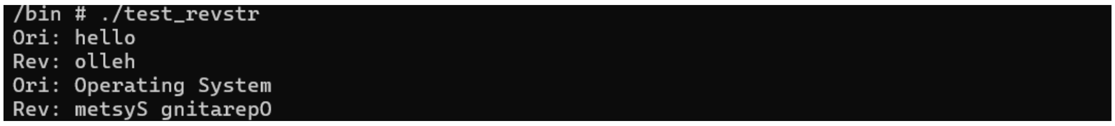
If the kernel is compiled without the correct RISC-V cross-compilation flags, the compiler will default to using the host's instruction set to parse the target code, leading to compilation failure. Even if the compilation succeeds, the resulting binary will only be compatible with the host computer and will fail to execute in the target environment.

3. Why is Docker used in this assignment, and what advantages does it provide? Please list at least two of them.
   (1) Isolation & Security
   Docker containers are isolated from the host OS. This ensures that experimental changes or potential system crashes within the container do not affect the host environment.
   (2) Portability
   By packaging all dependencies, libraries, and configurations into a single image, the development environment can be seamlessly moved across different platforms—such as Windows, macOS, or Linux—without worrying about compatibility issues.

Requirement 2

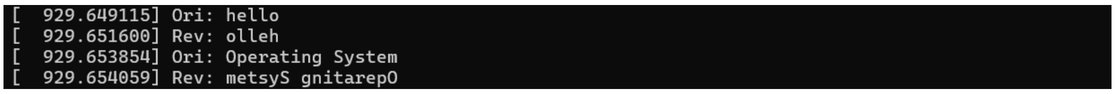
- Schreenshot

  The execution result of provided test_revstr user program for sys_revstr：



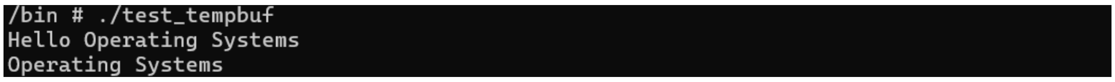
```
/bin # ./test_revstr
Ori: hello
Rev: olleh
Ori: Operating System
Rev: metsyS gnitarepO
```

The dmesg output of sys_revstr：



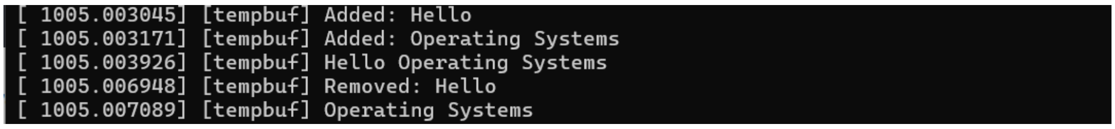
```
[  929.649115] Ori: hello
[  929.651600] Rev: olleh
[  929.653854] Ori: Operating System
[  929.654059] Rev: metsyS gnitarepO
```

The execution result of provided test_tempbuf user program for sys_tempbuf：



```
/bin # ./test_tempbuf
Hello Operating Systems
Operating Systems
```

The dmesg output of sys_tempbuf：



```
[ 1005.003045] [tempbuf] Added: Hello
[ 1005.003171] [tempbuf] Added: Operating Systems
[ 1005.003926] [tempbuf] Hello Operating Systems
[ 1005.006948] [tempbuf] Removed: Hello
[ 1005.007089] [tempbuf] Operating Systems
```

●