第零章：操作系统概述

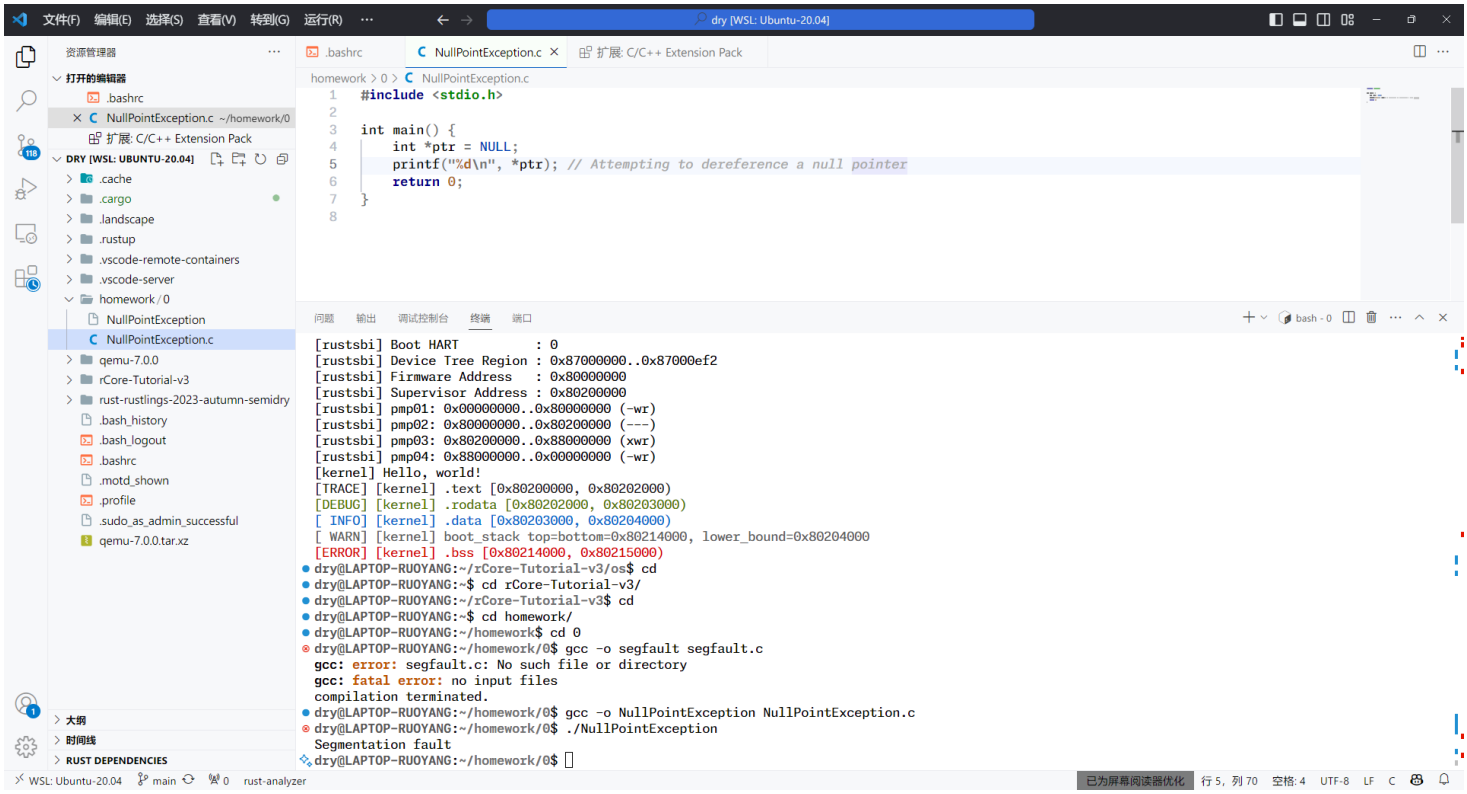
编程题

1. 在你日常使用的操作系统环境中安装并配置好实验环境。简要说明你碰到的问题/困难和解决方法。

配置WSL2：和以前配置过的电脑环境出现冲突。各种方法都试回来了，不知道哪种操作起作用了。
原先是在codespace上做的，结果这个月的额度用完了，阿里云服务器网络又不会又开不了代理，遂作罢。又重新在本地配置过环境。

2. 在Linux环境下编写一个会产生异常的应用程序，并简要解释操作系统的处理结果。

```c
#include <stdio.h>

int main() {
    int *ptr = NULL;
    printf("%d\n", *ptr);
    return 0;
}
```

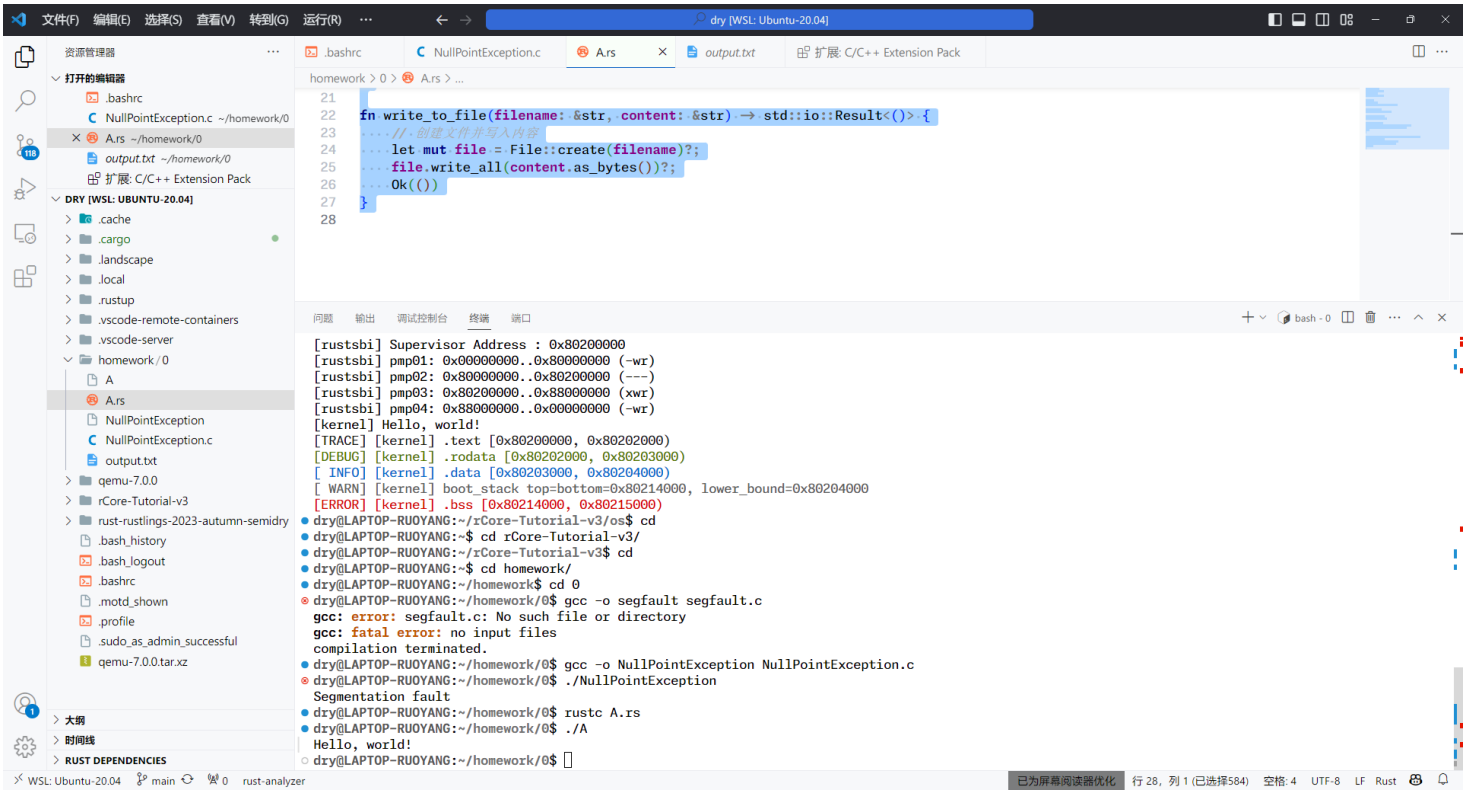


操作系统会向产生异常的进程发送一个信号，以通知进程发生了严重的错误。当进程接收到 SIGSEGV 信号时，操作系统会立即终止该进程。

3. 在Linux环境下编写一个可以睡眠5秒后打印出一个字符串，并把字符串内容存入一个文件中的应用程序A。(基于C或Rust语言)

```rust
use std::fs::File;
use std::io::prelude::*;
use std::thread;
```

```rust
use std::time::Duration;

fn main() {
    // 睡眠 5 秒
    thread::sleep(Duration::from_secs(5));

    // 要打印的字符串
    let message = "Hello, world!";

    // 打印字符串到控制台
    println!("{}", message);

    // 将字符串写入文件
    if let Err(err) = write_to_file("output.txt", message) {
        eprintln!("Failed to write to file: {}", err);
    }
}

fn write_to_file(filename: &str, content: &str) -> std::io::Result<()> {
    // 创建文件并写入内容
    let mut file = File::create(filename)?;
    file.write_all(content.as_bytes())?;
    Ok(())
}
```



4. 在Linux环境下编写一个应用程序B，简要说明此程序能够体现操作系统的并发性、异步性、共享性和持久性。(基于C或Rust语言)

注： 在类Linux环境下编写尝试用GDB等调试工具调试应用程序A，能够设置断点，单步执行，显示变量信息。

```
(gdb) break main
Breakpoint 1 at 0xa4e0
(gdb) run
Starting program: /home/dry/homework/0/A
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x000055555555e4e0 in main ()
(gdb) next
Single stepping until exit from function main,
which has no line number information.
Hello, world!
__libc_start_main (main=0x55555555e4e0 <main>, argc=1, argv=0x7fffffffdbd8, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>,
    stack_end=0x7fffffffdbc8) at ../csu/libc-start.c:342
342     ../csu/libc-start.c: No such file or directory.
```

问答题

1. 什么是操作系统？操作系统的主要目标是什么？

操作系统是一种管理计算机硬件和软件资源的系统软件。它提供了一组基本的服务和功能，以便其他软件（应用程序）能够有效地运行。操作系统的主要目标是提供一个方便、有效且一致的计算机环境，使用户和其他软件能够轻松地访问硬件资源，并提供良好的性能、可靠性、安全性和可扩展性。

2. 面向服务器的操作系统与面向手机的操作系统在功能上有何异同？

- 面向服务器的操作系统通常设计用于运行在大型服务器上，主要用于支持网络服务、数据库、存储等大型应用。它们通常需要具有高性能、高可靠性和可扩展性。
- 面向手机的操作系统则主要用于移动设备，如智能手机和平板电脑。它们需要具有低能耗、高度优化的资源利用、优秀的用户体验和强大的移动网络支持等特性。

3. 对于目前的手机或桌面操作系统而言，操作系统是否应该包括网络浏览器？请说明理由。

对于现代手机和桌面操作系统来说，通常会包括网络浏览器。这是因为互联网已经成为人们生活和工作的重要组成部分，用户经常需要通过浏览器来访问网页、查找信息和进行在线交流。因此，将网络浏览器集成到操作系统中可以提供更好的用户体验和方便性。

4. 操作系统的核心抽象有哪些？它们应对的对象是啥？

操作系统的核心抽象包括进程、线程、地址空间、文件系统和设备。它们应对的对象分别是正在执行的程序、并发执行的任务、内存空间、数据存储和外部设备。

5. 操作系统与应用程序之间通过什么来进行互操作和数据交换？

操作系统与应用程序之间通过系统调用来进行互操作和数据交换。系统调用是操作系统提供的一组接口，允许应用程序请求操作系统提供的服务，如文件操作、进程管理、内存管理等。

6. 操作系统的特征是什么？请结合你日常使用的操作系统的具体运行情况来进一步说明操作系统的特征。

操作系统的特征包括并发性、共享性、虚拟性、抽象性和持久性。例如，在日常使用的操作系统中，多个应用程序可以同时运行（并发性），它们可以共享资源（共享性），操作系统提供了对硬件的虚拟访问（虚拟性），并隐藏了硬件的具体细节（抽象性），并且数据通常会持久保存在存储设备上（持久性）。

7. 请说明基于C语言应用的执行环境与基于Java语言应用的执行环境的异同。

基于C语言的应用通常会直接编译为本地机器码，因此在执行时需要与特定的硬件架构相匹配。而基于Java语言的应用则是通过Java虚拟机（JVM）执行的，它们编译为字节码，可以在不同的平台上运行。这意味着基于

Java的应用具有更好的跨平台性，但可能会牺牲一些性能。

8. 请简要列举操作系统的系统调用的作用，以及简要说明与程序执行、内存分配、文件读写相关的Linux系统调用的大致接口和含义。

操作系统的系统调用用于向操作系统请求服务。例如：

- `fork()`：创建一个新进程。
- `exec()`：执行一个新程序。
- `brk()`：调整进程的数据段大小。
- `open()`：打开文件。
- `read()`：从文件中读取数据。
- `write()`：向文件写入数据。

9. 以你编写的可以睡眠5秒后打印出一个字符串的应用程序A为例，说明什么是控制流？什么是异常控制流？什么是进程、地址空间和文件？并简要描述操作系统是如何支持这个应用程序完成其工作并结束的。

控制流是程序执行的路径或顺序。异常控制流是程序执行过程中由于错误、异常或外部事件导致的控制流的改变。进程是操作系统中的一个运行中的程序的实例，具有自己的内存空间和资源。地址空间是进程可用于存储其代码、数据和堆栈的虚拟内存空间。文件是持久性存储设备上的数据单元。
对于应用程序 A，操作系统支持它完成工作的方式如下：

- 当应用程序 A 启动时，操作系统为其分配一个新的进程，并为其分配内存空间。
- 应用程序 A 调用了一个睡眠函数，这会使得该进程暂时挂起，不再占用 CPU 资源，直到指定的时间过去后再继续执行。
- 在睡眠期间，操作系统可能会调度其他进程运行，以充分利用系统资源。
- 睡眠结束后，应用程序 A 被重新调度到 CPU 上运行，并执行打印字符串的操作。
- 操作系统通过系统调用提供了文件操作的支持，应用程序 A 调用文件写入函数将字符串写入文件。
- 应用程序 A 完成其任务后，操作系统终止该进程，并释放其占用的资源，包括内存空间和文件描述符。

10. 请简要描述支持单个应用的OS、批处理OS、多道程序OS、分时共享OS的特点。

- 单个应用的操作系统：专门为运行单个应用程序而设计，只能运行一个应用程序，例如嵌入式系统。
- 批处理操作系统：按顺序执行一批作业，无需用户干预，例如早期的批处理系统。
- 多道程序操作系统：允许多个程序同时进入内存并轮流执行，提高了资源利用率，例如多道程序批处理系统。
- 分时共享操作系统：允许多个用户通过终端同时访问计算机系统，系统会在用户之间快速切换，以便每个用户感觉到他们独占了整个系统，例如现代的交互式操作系统如 Linux 和 Windows。