






进程

 Courses	 操作系统
<input checked="" type="checkbox"/> Done	<input type="checkbox"/>
 Status	Done

标准输出的缓冲机制

什么是缓冲机制？

缓冲机制（Buffering）是一种用来优化数据传输的方法。它通过临时存储数据以减少 I/O 操作次数，从而提高性能。

类比：邮递员送信

假设你是一个邮递员，你需要每天送信。如果每收到一封信就马上去送，这样会非常低效，因为你会不断地来回奔波。然而，如果你先将收到的信件存储在一个邮包里，等到邮包装满或到达一定的时间点再一起去送，这样就会节省很多来回跑的时间和精力。这就是缓冲机制的核心思想。

标准输出中的缓冲机制

在标准输出中，缓冲机制通常有三种类型：

1. **全缓冲（Fully Buffered）**：只有在缓冲区满时，数据才会被写入到输出设备。这类似于我们前面提到的邮递员将邮包装满再去送信。
2. **行缓冲（Line Buffered）**：每当遇到换行符时，缓冲区中的数据会被立即写入到输出设备。这类似于邮递员每收到一行邮件（信）就去送一次。
3. **无缓冲（Unbuffered）**：每次输出操作都会立即写入到输出设备，不经过缓冲区。这类似于邮递员每收到一封信就立即去送。

如何选择缓冲机制？

选择哪种缓冲机制通常取决于输出设备和具体的应用场景。例如：

- **全缓冲**：适用于文件输出，因为文件写入操作比较耗时，通过缓冲可以减少实际写入的次数，提高性能。
- **行缓冲**：适用于终端输出，因为用户在终端上更容易逐行查看输出的结果。
- **无缓冲**：适用于需要实时输出的场景，比如日志记录或错误输出（stderr），这样可以保证每次输出操作都能立即看到结果。

`_exit()` 销毁状态机

多线程程序怎么办？

cmp

return、exit()都是c程序可以理解的退出

`_exit(0)`和`syscall`的区别？ strace

`exit(0)`会调用`group_exit()`

关于pipe()

C标准库和实现

系统调用之上一libc

可执行文件

什么是可执行文件？

一个数据结构

ELF是对人类不友好的

链接和加载中的核心概念：代码  (and 数据)、符号 、重定位 

拼接，两次遍历，mmap