

操作系统实验报告

基本信息

- 09023321 巩皓锴
- 报告日期: 2025.3.12
- 实验名称: 基于系统调用的文件复制程序

一、实验内容

使用系统调用，用C或C++写一个程序，实现如下功能：

- 从一个文件中读出数据，写入另一个文件中。
- 要求：
 - 具有良好的交互性
 - 使用者可输入源文件和目的文件的路径和文件名。
 - 具有完善的错误处理机制,要有相应的错误提示输出,
 - 在Linux操作系统上调试并运行

二、实验目的

- 通过实验，加深对系统调用概念的理解，了解其实现机制以及使用方式。
- 通过在Linux操作系统上编写和调试简单程序，进一步熟悉Linux操作系统的使用，初步掌握Linux环境下的C或C++编译和调试工具，为进一步理解和学习Linux操作系统的内核结构和核心机制作准备。

三、设计思路和流程图

1. 设计思路

- 用户输入源文件路径，检查是否存在，若不存在，则提示错误并重新输入。
- 用户输入目标文件路径，无需检查是否存在（若文件已存在，则覆盖）。
- 以 `O_RDONLY` 方式打开源文件，以 `O_WRONLY | O_CREAT | O_TRUNC` 方式打开目标文件（权限 `0644`）。
- 采用 **固定大小的缓冲区** 进行 `read` 和 `write` 操作，循环读取直到文件复制完成。
- 发生错误时输出错误信息，并释放相关资源。
- 复制完成后输出成功信息。

2. 流程图



```

30      |                               |
31      (是) 输出错误信息      (否) 继续写入
32      |
33      终止程序
34      |
35      +-----+
36      |  关闭文件, 释放资源  |
37      +-----+
38      |
39      +-----+
40      |  输出“文件复制成功”信息  |
41      +-----+

```

四、关键系统调用解析

本实验主要涉及以下 Linux 系统调用：

1. open()

```
1 | int open(const char *pathname, int flags, mode_t mode);
```

- 作用: 打开或创建文件，并返回文件描述符（文件操作的唯一标识）。
- 参数
 - `pathname`：文件路径。
 - `flags`：操作模式，如 `O_RDONLY`（只读）、`O_WRONLY`（只写）、`O_CREAT`（创建文件）、`O_TRUNC`（清空文件）。
 - `mode`（可选）：创建文件时的权限，如 `0644`（用户可读写，组和其他用户可读）。
- 返回值: 成功返回文件描述符（`fd`），失败返回 `-1`，可通过 `errno` 获取错误信息。

2. read()

```
1 | ssize_t read(int fd, void *buf, size_t count);
```

- 作用: 从文件描述符 `fd` 读取 `count` 个字节的数据到 `buf` 缓冲区。
- 返回值: 成功返回实际读取的字节数，失败返回 `-1`，并设置 `errno`。

3. write()

```
1 | ssize_t write(int fd, const void *buf, size_t count);
```

- 作用: 将 `count` 个字节的数据从 `buf` 缓冲区写入文件描述符 `fd` 。
 - 返回值: 成功返回写入的字节数, 失败返回 `-1`, 并设置 `errno` 。
-

4. close()

```
1 | int close(int fd);
```

- 作用: 关闭文件描述符 `fd`, 释放资源。
-

5. access()

```
1 | int access(const char *pathname, int mode);
```

- 作用: 检查文件是否存在, `mode = F_OK` 用于判断文件是否存在。
-

五、实验代码

```
1 | #include <iostream>
2 | #include <fcntl.h>
3 | #include <unistd.h>
4 | #include <cerrno>
5 | #include <cstring>
6 |
7 | // 定义缓冲区大小
8 | constexpr size_t BUFFER_SIZE = 4096;
9 |
10 | /**
11 |  * @brief 复制文件
12 |  * @param source 源文件路径
13 |  * @param destination 目标文件路径
14 |  */
15 | void copy_file(const std::string &source, const std::string &destination)
16 | {
17 |     // 以只读模式打开源文件
18 |     int src_fd = open(source.c_str(), O_RDONLY);
19 |     if (src_fd == -1)
20 |     {
21 |         std::cerr << "打开源文件失败: " << strerror(errno) << std::endl;
22 |         return;
23 |     }
```

```
24
25 // 以写入模式打开目标文件, 若文件不存在则创建
26 int dest_fd = open(destination.c_str(), O_WRONLY | O_CREAT | O_TRUNC, 0644);
27 if (dest_fd == -1)
28 {
29     std::cerr << "打开目标文件失败: " << strerror(errno) << std::endl;
30     close(src_fd);
31     return;
32 }
33
34 char buffer[BUFFER_SIZE];
35 ssize_t bytes_read, bytes_written;
36
37 // 循环读取源文件内容并写入目标文件
38 while ((bytes_read = read(src_fd, buffer, BUFFER_SIZE)) > 0)
39 {
40     bytes_written = write(dest_fd, buffer, bytes_read);
41     if (bytes_written == -1)
42     {
43         std::cerr << "写入目标文件失败: " << strerror(errno) << std::endl;
44         close(src_fd);
45         close(dest_fd);
46         return;
47     }
48 }
49
50 if (bytes_read == -1)
51 {
52     std::cerr << "读取源文件失败: " << strerror(errno) << std::endl;
53 }
54
55 close(src_fd);
56 close(dest_fd);
57 std::cout << "文件复制成功! " << std::endl;
58 }
59
60 /**
61  * @brief 判断文件是否存在
62  * @param path 文件路径
63  * @return 若文件存在返回 true, 否则返回 false
64  */
65 bool isFileExists(const std::string &path)
66 {
67     return access(path.c_str(), F_OK) != -1;
68 }
69
70 int main()
71 {
72     std::string source, destination;
73
74     // 让用户输入源文件路径
75     while (true)
```

```

76     {
77         std::cout << "请输入源文件路径: ";
78         std::getline(std::cin, source);
79         if (source[0] == '\\')
80             source = source.substr(1, source.size() - 2);
81
82         if (!isFileExists(source))
83         {
84             std::cerr << "文件不存在, 请重新输入! " << std::endl;
85             continue;
86         }
87         break;
88     }
89
90     std::cout << "请输入目标文件路径: ";
91     std::getline(std::cin, destination);
92     if (destination[0] == '\\')
93         destination = destination.substr(1, destination.size() - 2);
94
95     copy_file(source, destination);
96     return 0;
97 }
98

```

六、实验结果

实验程序在 Windows Subsystem for Linux 上运行, 系统环境为 Ubuntu24.04, 编译器使用 g++-14.2.0, 编译指令为

```

1 | /usr/bin/g++-14 /home/hackerhand/cpp/OS/exp1/exp1.cpp -o
   | /home/hackerhand/cpp/OS/exp1/exp1 -std=c++23

```

程序运行截图如下

```

hackerhand@HAIR15ARE:~$ /home/hackerhand/cpp/OS/exp1/exp1
请输入源文件路径: '/home/hackerhand/cpp/OS/exp1/exp1.cpp'
请输入目标文件路径: '/home/hackerhand/cpp/OS/exp1/exp1.txt'
文件复制成功!

```

检查目标文件后发现成功拷贝, 符合预期。

七、实验总结

1. 掌握了文件操作相关的系统调用, 熟悉了 `open`、`read`、`write`、`close` 的使用。
2. 理解了文件描述符的概念。

3. 通过缓冲区优化了文件读写性能，减少了 `read` 和 `write` 的调用次数。
4. 学习了错误处理方法，提高了程序的健壮性。