《操作系统原理》实验报告

实验名称	系统调用基础	实验序号	1
实验日期	2023/03/20	实验人	盖乐

一、实验题目

实现一个将一个文件的内容复制到另外一个文件的程序,并且程序通过用 户传入的第一个参数作为目的地,第二个参数作为数据来源,完成这个程序 时请使用 Posix 或者 windows api 来实现,并且做好相应所有必要的错误的 检测。

二、相关原理与知识

通过用户键入的命令行参数来获得目标文件和工作源文件的地址, 计算出文 件的大小,再通过相应的 open read write 系统调用来实现文件的打开读写,就 能实现文件的复制了

三、实验过程

```
1. 读取参数,参数判断并提示
if(argc != 3) {
   usage(argv[0]);
    return 0;
}
static void usage(char *name)
{
   fprintf(stderr, "Usage: %s source [dest]\n", name);
2. 打开源文件并检查错误
  int src = open(argv[1], 0 RDONLY);
  if(src < 0) {
      fprintf(stderr, "Error: cant open source file.\n");
      return 0;
3. 打开目标文件并检查错误
 int dest = open(argv[2], 0 WRONLY|0 CREAT|0 TRUNC, 0644);
 if(dest < 0) {
     fprintf(stderr, "Error: cant open dest file.\n");
     return 0;
```

4. 使用循环读取到缓冲区并写到新地址

```
int pos = 0;
while((pos = read(src, buf, 1024)) > 0)
{
    write(dest, buf, pos);
}
```

四、实验结果与分析

1. 使用 gcc 连接并编译可执行文件

```
gaile@gaile-virtual-machine:~/Code/operation system experiment$ gcc op1.c -o op1
```

2. Usage 提示

```
gaile@gaile-virtual-machine:~/Code/operation system experiment$ ./op1
Usage: ./op1 source [dest]
```

3. 然后使用运行程序,进行文件复制操作

```
gaile@gaile-virtual-machine:~/Code/operation system experiment$ ./op1 op1.c cp1.txt
```

4.通过 vi 查看目标文件复制结果

gaile@gaile-virtual-machine:-/Code/operation system experiment\$ vi cp1.txt

```
#include <fcntl.h>
#include <unistd.h>
static void usage(char*);
int main(int argc, char* argv[])
       char buf[1024];
       if(argc != 3) {
              usage(argv[0]);
               return 0;
       int src = open(argv[1], 0_RDONLY);
       if(src < 0) {
              fprintf(stderr, "Error: cant open source file.\n");
               return 0;
       int dest = open(argv[2], 0_WRONLY|0_CREAT|0_TRUNC, 0644);
              fprintf(stderr, "Error: cant open dest file.\n");
               return 0:
       int pos = 0;
       while((pos = read(src, buf, 1024)) > 0)
               write(dest, buf, pos);
      close(src);
       close(dest);
       return 0;
static void usage(char *name)
       fprintf(stderr, "Usage: %s source [dest]\n", name);
```

此程序使用了Linux系统调用来打开、读取、写入文件,并且在必要时关闭文件。 Linux系统调用是操作系统提供给用户程序的接口,可以访问底层系统资源(如 文件、网络、进程、内存等),并且可以在用户空间和内核空间之间进行数据传输。

在此程序中使用的 Linux 系统调用包括:

- ① open(): 打开文件,并返回一个文件描述符(file descriptor),表示对该文件的访问权限和状态。它接收文件路径、访问模式和文件权限作为参数。本程序使用该系统调用打开源文件和目标文件。
- ② read():从文件描述符指定的文件中读取数据,并将数据存储在指定的缓冲区中。它接收文件描述符、缓冲区和要读取的字节数作为参数。本程序使用该系统调用从源文件中读取数据。
- ③ write(): 向文件描述符指定的文件中写入数据。它接收文件描述符、 缓冲区和要写入的字节数作为参数。本程序使用该系统调用将从源文件中读取的 数据写入目标文件中。
- ④ close():关闭文件描述符,并释放与该文件描述符相关的系统资源。它接收文件描述符作为参数。本程序使用该系统调用关闭打开的源文件和目标文件。

五、问题总结

1.源文件错误报错,无法找到源文件地址

gaile@gaile-virtual-machine:~/Code/operation system experiment\$./op1 o1.c cp
Error: cant open source file.

2.目标文件错误报错,无法找到目标文件地址

gaile@gaile-virtual-machine:~/Code/operation system experiment\$./op1 op1.c ./
Error: cant open dest file.

3.使用 strace 分析系统调用

```
nt$ strace ./op1 op1.c cp1.txt
execve("./op1", ["./op1", "op1.c", "cp1.txt"], 0x7fff23723c80 /* 46 vars */) = 0
                                    = 0x5654a1ed2000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc2de0f510) = -1 EINVAL (无效的参数)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdf6f673000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (没有那个文件或目录)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=66031, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 66031, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fdf6f662000
close(3)
                                    = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\0\1\0\0\P\237\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\00U\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0GNU\0i8\235HZ\227\223\333\350s\360\352,\223\340."..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2216304, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\4\0\0\0@\0\0\0\0\0\0@\0\0\0\0\0\0@\0\0\0\0\0\0"..., 784, 64) = 784
nmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fdf6f43a000
nmap(0x7fdf6f462000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fdf6f462000
nmap(0x7fdf6f5f7000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7fdf6f5f7000
nmap(0x7fdf6f64f000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7fdf6f64f000
nmap(0x7fdf6f655000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fdf6f655000
nmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdf6f437000
arch_prctl(ARCH_SET_FS, 0x7fdf6f437740) = 0
set_tid_address(0x7fdf6f437a10)
set_robust_list(0x7fdf6f437a20, 24)
                                    = 0
rseq(0x7fdf6f4380e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7fdf6f64f000, 16384, PROT_READ) = 0
mprotect(0x5654a04d0000, 4096, PROT_READ) = 0
nprotect(0x7fdf6f6ad000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
nunmap(0x7fdf6f662000, 66031)
openat(AT_FDCWD, "op1.c", O_RDONLY)
openat(AT_FDCWD, "cp1.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 4
read(3, "#include <stdio.h>\n#include <fcn"..., 1024) = 663
write(4, "#include <stdio.h>\n#include <fcn"..., 663) = 663
read(3, "", 1024)
close(3)
                                     = 0
close(4)
                                    = 0
exit_group(0)
+++ exited with 0 +++
六、源代码
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
static void usage(char*);
int main(int argc, char* argv[])
{
     char buf[1024];
     if(argc != 3) {
           usage(argv[0]);
           return 0;
```

```
}
   int src = open(argv[1], O_RDONLY);
    if(src < 0) {
        fprintf(stderr, "Error: cant open source file.\n");
        return 0;
   }
   int dest = open(argv[2], O_WRONLY|O_CREAT|O_TRUNC, 0644);
   if(dest < 0) {
        fprintf(stderr, "Error: cant open dest file.\n");
        return 0;
   }
   int pos = 0;
   while((pos = read(src, buf, 1024)) > 0)
    {
        write(dest, buf, pos);
   }
   close(src);
    close(dest);
    return 0;
static void usage(char *name)
{
   fprintf(stderr, "Usage: %s source [dest]\n", name);
```