OS_Lab06_多进程拷贝目录

姓名	李懋
学号	2213189
邮箱	2213189@mail.nankai.edu.cn

1 实验题目

- 编写一个C/C++程序实现利用多进程拷贝文件夹

2 实验目的

1. 编写C/C++程序
2. 实现利用多进程拷贝一个文件夹及其所有子文件夹
    1. 采用最新的Linux Kernel作为测试文件夹
3. 验证拷贝结果是否正确
4. 比较多进程拷贝与单进程拷贝的效率

3 实验原理

1. **多进程拷贝**：通过`fork()`系统调用创建多个子进程，每个子进程负责拷贝目录中的部分文件或子目录。这样可以并行处理，提高拷贝效率。
2. **单进程拷贝**：使用单个进程递归地遍历目录树，逐个拷贝文件和子目录。
3. **效率比较**：通过`time`命令记录程序的执行时间，比较多进程和单进程拷贝的效率。
4. **验证拷贝结果**：使用`diff -r`命令比较源目录和目标目录的内容，确保拷贝结果正确。

3.1 单进程

1. **文件拷贝 (`copy_file` 函数)**
    - **功能**：将单个文件从源路径复制到目标路径。
    - **实现**：
        - 打开源文件和目标文件。
        - 读取源文件数据并写入目标文件。
        - 关闭文件描述符。
2. **目录拷贝 (`copy_directory` 函数)**
    - **功能**：递归地遍历源目录，并使用单进程来拷贝文件和子目录。
    - **实现**：

        - 打开源目录。
        - 创建目标目录。
        - 遍历源目录中的每个条目：
            - 如果是目录，递归调用 `copy_directory` 函数。
            - 如果是普通文件，调用 `copy_file` 函数进行拷贝。
            - 如果是符号链接，读取链接目标并创建新的符号链接。
        - 关闭目录。
3. **主函数 (`main` 函数)**
    - **功能**：解析命令行参数，并调用 `copy_directory` 函数开始拷贝过程。
    - **实现**：
        - 检查命令行参数是否正确。
        - 调用 `copy_directory` 函数进行目录拷贝1. **文件拷贝 (`copy_file` 函数)**

- **功能**：将单个文件从源路径复制到目标路径。
- **实现**：与单进程拷贝程序相同。

3.2 多进程

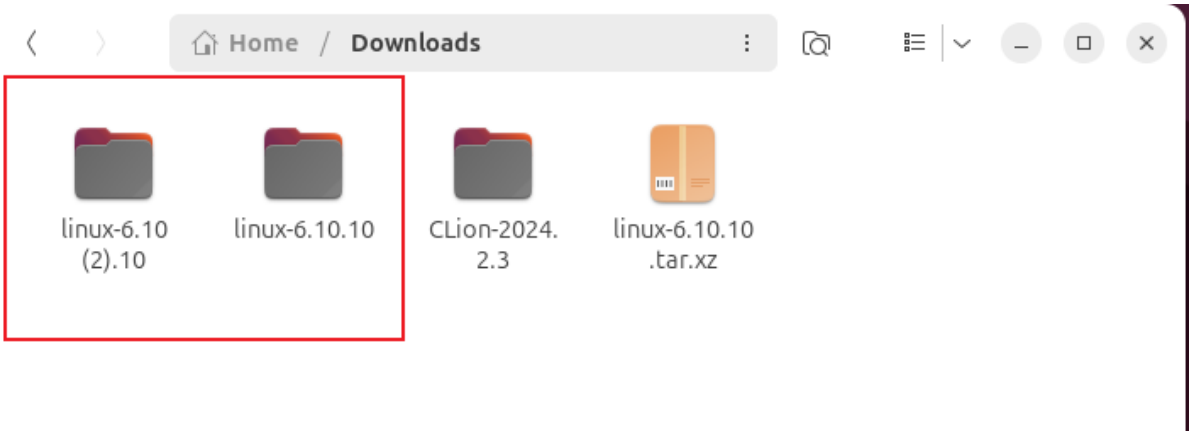
1. **文件拷贝 (`copy_file` 函数)**
    - **功能**：将单个文件从源路径复制到目标路径。
    - **实现**：与单进程拷贝程序相同。
2. **目录拷贝 (`copy_directory` 函数)**
    - **功能**：递归地遍历源目录，并使用多进程来拷贝文件和子目录。
    - **实现**：
        - 打开源目录。
        - 创建目标目录。
        - 遍历源目录中的每个条目：
            - 如果是目录，使用 `fork` 函数创建子进程，并在子进程中递归调用 `copy_directory` 函数。
            - 如果是普通文件，调用 `copy_file` 函数进行拷贝。
            - 如果是符号链接，读取链接目标并创建新的符号链接。
        - 关闭目录。
3. **主函数 (`main` 函数)**
    - **功能**：解析命令行参数，并调用 `copy_directory` 函数开始拷贝过程。
    - **实现**：
        - 检查命令行参数是否正确。
        - 调用 `copy_directory` 函数进行目录拷贝。

4 实验具体步骤

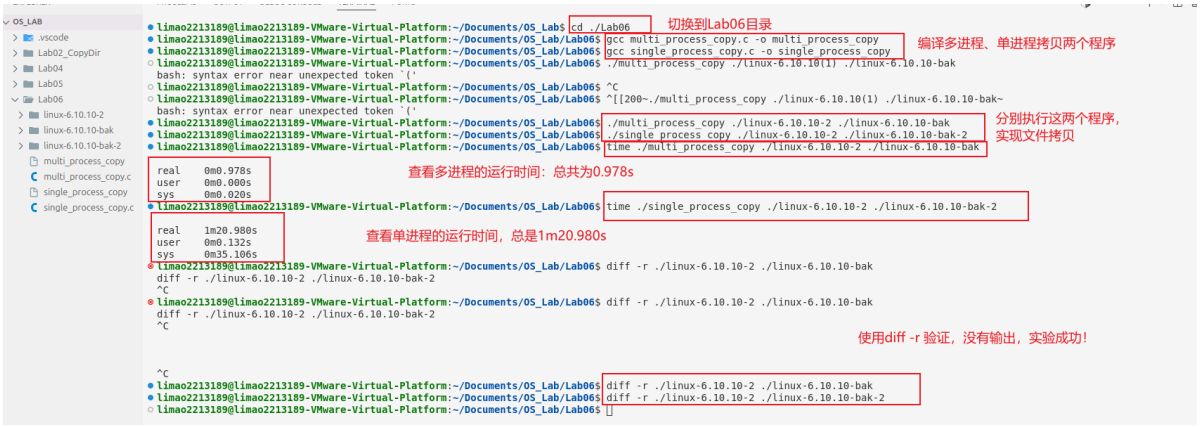
1. Install GCC

```
limao2213189@limao2213189-VMware-Virtual-Platform:~$ sudo apt-get install build-
essential
[sudo] password for limao2213189:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.10ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
limao2213189@limao2213189-VMware-Virtual-Platform:~$
```

2. 下载并解压最新的Linux内核，之前已经下好，直接展示



- 本次实验所有步骤在下图做了详细的解释，请仔细查看!!!!



3. 编写多进程、单进程拷贝目录程序，在实验原理部分已经解释了，这里不多赘述
4. 编译程序
5. 运行程序，拷贝目录
6. 分别执行time命令查看时间
7. 使用 diff -r 验证拷贝是否成功

5 实验总结

- 实验发现多进程比单进程快的多，这一点我问了几个同学，结果都和我不一样，可能是我设置虚拟机的时候，处理器数量选择的8核有关，多进程运行效率非常高，后面再尝试了一遍也是这样
- 通过本次实验，我对多进程的编程有了更深的理解，更加了解了进程的启动和调度

6 源代码

6.1 单进程

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>

void copy_file(const char *src, const char *dest) {
    int fd_src, fd_dest;
    char buffer[4096];
    ssize_t bytes_read;

    fd_src = open(src, O_RDONLY);
    if (fd_src == -1) {
        perror("open src");
        return;
    }

    fd_dest = open(dest, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd_dest == -1) {
        perror("open dest");
        close(fd_src);
        return;
    }

    while ((bytes_read = read(fd_src, buffer, sizeof(buffer))) > 0) {
        write(fd_dest, buffer, bytes_read);
    }

    close(fd_src);
    close(fd_dest);
}

void copy_directory(const char *src, const char *dest) {
    DIR *dir;
    struct dirent *entry;
    struct stat statbuf;
    char src_path[1024], dest_path[1024];
```

```
    if ((dir = opendir(src)) == NULL) {
        perror("opendir");
        return;
    }

    mkdir(dest, 0755);

    while ((entry = readdir(dir)) != NULL) {
        if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
{
            continue;
        }

        snprintf(src_path, sizeof(src_path), "%s/%s", src, entry->d_name);
        snprintf(dest_path, sizeof(dest_path), "%s/%s", dest, entry->d_name);

        if (lstat(src_path, &statbuf) < 0) {
            perror("lstat");
            continue;
        }

        if (S_ISDIR(statbuf.st_mode)) {
            copy_directory(src_path, dest_path);
        } else if (S_ISREG(statbuf.st_mode)) {
            copy_file(src_path, dest_path);
        } else if (S_ISLNK(statbuf.st_mode)) {
            char link_target[1024];
            ssize_t len = readlink(src_path, link_target, sizeof(link_target) -
1);
            if (len != -1) {
                link_target[len] = '\0';
                symlink(link_target, dest_path);
            }
        }
    }

    closedir(dir);
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <source_dir> <dest_dir>\n", argv[0]);
        return 1;
    }

    copy_directory(argv[1], argv[2]);

    return 0;
}
```

6.2 多进程

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>

void copy_file(const char *src, const char *dest) {
    int fd_src, fd_dest;
    char buffer[4096];
    ssize_t bytes_read;

    fd_src = open(src, O_RDONLY);
    if (fd_src == -1) {
        perror("open src");
        return;
    }

    fd_dest = open(dest, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd_dest == -1) {
        perror("open dest");
        close(fd_src);
        return;
    }

    while ((bytes_read = read(fd_src, buffer, sizeof(buffer))) > 0) {
        write(fd_dest, buffer, bytes_read);
    }

    close(fd_src);
    close(fd_dest);
}

void copy_directory(const char *src, const char *dest) {
    DIR *dir;
    struct dirent *entry;
    struct stat statbuf;
    char src_path[1024], dest_path[1024];

    if ((dir = opendir(src)) == NULL) {
        perror("opendir");
        return;
    }

    mkdir(dest, 0755);

    while ((entry = readdir(dir)) != NULL) {
        if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
{
            continue;
        }
```

```
        snprintf(src_path, sizeof(src_path), "%s/%s", src, entry->d_name);
        snprintf(dest_path, sizeof(dest_path), "%s/%s", dest, entry->d_name);

        if (lstat(src_path, &statbuf) < 0) {
            perror("lstat");
            continue;
        }

        if (S_ISDIR(statbuf.st_mode)) {
            pid_t pid = fork();
            if (pid == 0) {
                copy_directory(src_path, dest_path);
                exit(0);
            } else if (pid < 0) {
                perror("fork");
            }
        } else if (S_ISREG(statbuf.st_mode)) {
            copy_file(src_path, dest_path);
        } else if (S_ISLNK(statbuf.st_mode)) {
            char link_target[1024];
            ssize_t len = readlink(src_path, link_target, sizeof(link_target) -
1);
            if (len != -1) {
                link_target[len] = '\0';
                symlink(link_target, dest_path);
            }
        }
    }

    closedir(dir);
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <source_dir> <dest_dir>\n", argv[0]);
        return 1;
    }

    copy_directory(argv[1], argv[2]);

    return 0;
}
```