1. **多进程拷贝目录**

具体代码如下:

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <string.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <sys/wait.h>

**int** mkdir(**const** **char** \*pathname, mode\_t mode);

**int** main(**int** argc, **char** \*argv[])

{

    pid\_t pid;

**char** \* sourceDir, \* destDir;

    DIR \* dir;

**struct** dirent \* ptr;

**int** processCnt = 0;

**if**(argc !=3){

        fprintf(stderr,"Wrong arguments");

        exit(-1);

    }**else**{

        sourceDir = argv[1];

        destDir = argv[2];

    }

**if**(NULL == opendir(destDir)){

        mkdir(destDir,0777);

    }

    dir = opendir(sourceDir);

**while**((ptr = readdir(dir)) != NULL){

**if**(strcmp(ptr->d\_name,".") == 0 || strcmp(ptr->d\_name,"..") == 0){

**continue**;

        }

        pid = fork();  //创建子进程

**if**(pid < 0){

            fprintf(stderr,"In parent process: fork failed");

            exit(-1);

        }**else** **if**(pid == 0){

**char** fromDir[256],toDir[256];

            strcpy(fromDir,sourceDir);

            strcpy(toDir,destDir);

            strcat(strcat(fromDir,"/\0"),ptr->d\_name);

            strcat(strcat(toDir,"/\0"),ptr->d\_name);

            execlp("/bin/cp","cp","-r","-p",fromDir,toDir,NULL);

        }

        processCnt++;

    }

**while**(processCnt--){

        wait(NULL);

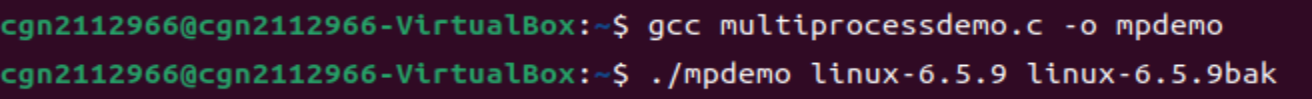
    }

    closedir(dir);

    exit(0);

}

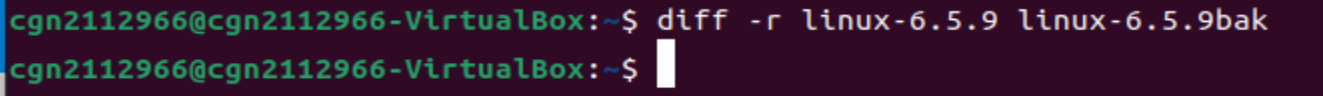
接着进行编译和运行



1. **比较拷贝的文件**

使用diff -r 命令来查看拷贝是否完整

如果所示拷贝完整



1. **测试拷贝的时间，比较多进程和单进程时间差别**

单进程代码如下：

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#include <dirent.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <string.h>

#define BUFFER\_SIZE 4096

**void** copy\_file(**const** **char** \*src, **const** **char** \*src2) {

**char** buffer[BUFFER\_SIZE];

**int** src\_fd = open(src, O\_RDONLY);

**int** src2\_fd = open(src2, O\_WRONLY | O\_CREAT, 0644);

**if** (src\_fd == -1 || src2\_fd == -1) {

        perror("文件打开失败");

        exit(1);

    }

    ssize\_t bytes\_read;

**while** ((bytes\_read = read(src\_fd, buffer, **sizeof**(buffer))) > 0) {

        write(src2\_fd, buffer, bytes\_read);

    }

    close(src\_fd);

    close(src2\_fd);

}

**void** copy\_symlink(**const** **char** \*src, **const** **char** \*dest) {

**char** target[512];

    ssize\_t len = readlink(src, target, **sizeof**(target) - 1);

**if** (len == -1) {

        perror("读取符号链接失败");

        exit(1);

    }

    target[len] = '\0'; // 确保字符串以 null 结尾

**if** (symlink(target, dest) == -1) {

        perror("创建符号链接失败");

        exit(1);

    }

}

**void** copy\_directory(**const** **char** \*src, **const** **char** \*dest) {

    DIR \*dir = opendir(src);

**if** (dir == NULL) {

        perror("打开文件夹失败");

        exit(1);

    }

    mkdir(dest, 0755);  // rwxr-xr-x

**struct** dirent \*entry;

**struct** stat statbuf;

**while** ((entry = readdir(dir)) != NULL) {

**char** src\_path[512];

**char** dest\_path[512];

**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) == -1) {

            perror("获取文件状态失败");

**continue**;

        }

**if** (S\_ISDIR(statbuf.st\_mode)) {

            copy\_directory(src\_path, dest\_path);

        } **else** **if** (S\_ISLNK(statbuf.st\_mode)) {

            copy\_symlink(src\_path, dest\_path);

        } **else** {

            copy\_file(src\_path, dest\_path);

        }

    }

    closedir(dir);

}

**int** main(**int** argc, **char** \*argv[]) {

**if**(argc !=3){

        fprintf(stderr,"Wrong arguments");

        exit(-1);

    }**else**{

    copy\_directory(argv[1], argv[2]);

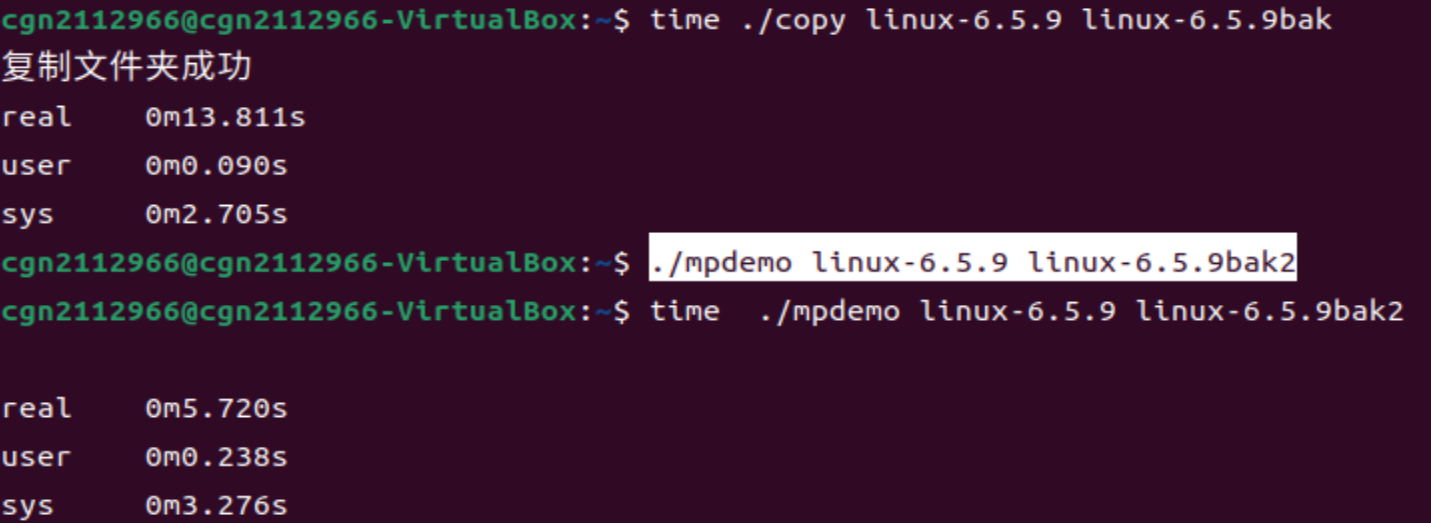
    }

    printf("复制文件夹成功");

**return** 0;

}

单进程和多进程运行时间如下(第一个为单进程，第二个为多进程）：



由图可知，多进程运行时间比单进程运行时间短。

1. **心得体会**

本次实验我学会了如何用多进程来拷贝文件目录，并与单进程拷贝做时间比较，以此来看出多进程的优势。