Write a c/c++ program to implement copy one directory with multi-processes

多进程拷贝目录，并与单进程拷贝目录做效率比较

Target

1. Write a c/c++ program
2. To implement copy one directory and it's subdiretories with multi-processes
3. GCC
4. Test directory: 使用最新的Linux Kernel来测试(从www.kernel.org下载最新的linux内核)
    1. https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.5.9.tar.xz
    2. extract linux-6.5.9.tar.xz to linux-6.5.9 directory,
    3. and copy linux-6.5.9 directory to linux-6.5.9bak directory
5. Verify that the directory copy is correct

Tools

Install GCC Software Collection

```
sudo apt-get install build-essential
```

How to use GCC

- gcc and make

比较目录是否相同

```
diff -r DirA DirB
```

get the total time of program execution

```
$ time pwd
/mnt/test2linux

real    0m0.000s
user    0m0.000s
sys 0m0.000s


$ time tar xvJf  linux-6.5.9.tar.xz

real    0m28.554s
```

```
user    0m7.738s
sys 0m3.554s
```

structure of directory

```
struct dirent
{
    ino_t d_ino; //d_ino 此目录进入点的inode
    ff_t d_off; //d_off 目录文件开头至此目录进入点的位移
    signed short int d_reclen; //d_reclen _name 的长度, 不包含NULL 字符
    unsigned char d_type; //d_type d_name 所指的文件类型 d_name 文件名
    har d_name[256];
};

the value returned in d_type:
              DT_BLK      This is a block device.
              DT_CHR      This is a character device.
              DT_DIR      This is a directory.
              DT_FIFO     This is a named pipe (FIFO).
              DT_LNK      This is a symbolic link.
              DT_REG      This is a regular file.
              DT_SOCK     This is a UNIX domain socket.
              DT_UNKNOWN  The file type could not be determined.

opendir()
readdir()
closedir()
```

Create a symbol link file

```
#include <fcntl.h>           /* Definition of AT_* constants */
#include <unistd.h>
int link(const char *oldpath, const char *newpath);
```

create process and execute one program

fork(): clone a new instance of current process

```
#include <sys/types.h>
#include <unistd.h>

pid_t fork(void);
```

exec():

```
#include <unistd.h>

int execl(const char *path, const char *arg, ...);
int execlp(const char *file, const char *arg, ...);
int execle(const char *path, const char *arg, ..., char *const envp[]);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
int execve(const char *path, char *const argv[], char *const envp[]);
```

命令行参数

```
int main(int argc, char* argv[]){
    int i;
    for (i = 0; i < argc; i++)
    {
      printf ("%3d %s\n", i, argv[i]);
    }
}
```

How to do

write a c program to support mulit processes copy one diretory and it's subdiretories, and the program also verifies the result多进程拷贝目录

1. Example of multi-processes and command line arguments

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int
main (int argc, char *argv[])
{
  int i;
  pid_t pid;
  int status;
  printf ("Parent %d: begin\n", getpid ));
  printf ("arguments list of %s:\n", argv[0]);
  for (i = 0; i < argc; i++)
    {
      printf ("%3d %s\n", i, argv[i]);
    }

  for (i = 1; i < argc; i++)
    {
      pid = fork ();
```

```
      if (pid < 0)
    {
      fprintf (stderr, "Fork Failed\n");
      break;
    }
      if (pid == 0)
    {
      execlp ("/bin/ls", "ls", argv[i], NULL);
    }
      else
    {
      printf ("Parent %d: Create Child Process %d\n", getpid (), pid);
    }
    }

  while (1)
    {
      pid = wait (&status);
      if (pid == -1)
    {
      break;
    }
      else
    {
      printf ("Parent %d: Child %d exited with %d code\n", getpid (), pid,
          WEXITSTATUS (status));
    }
    }
  printf ("Parent %d: exited\n", getpid ());
  return 0;
}
```

Compiling:

```
gcc     multiprocessdemo.c   -o mpdemo
./mpdemo
```

Result:

```
$./mpdemo /home /usr abc
Parent 5062: begin
arguments list of ./mpdemo:
  0 ./mpdemo
  1 /home
  2 /usr
  3 abc
Parent 5062: Create Child Process 5063
Parent 5062: Create Child Process 5064
Parent 5062: Create Child Process 5065
```

```
albert
bin    include  lib32  libexec  local  share
games  lib  lib64  libx32   sbin   src
ls: Parent 5062: Child 5063 exited with 0 code
无法访问 'abc'Parent 5062: Child 5064 exited with 0 code
: 没有那个文件或目录
Parent 5062: Child 5065 exited with 2 code
Parent 5062: exited
$
```

2. Example of traverse one directory

```
#include <dirent.h>
#include <unistd.h>
#include <stdlib.h>

int main()
{
    DIR * dir;
    struct dirent * ptr;
    /*open dir*/
    dir = opendir("/home");
    /*read dir entry*/
    while((ptr = readdir(dir)) != NULL)
    {
        printf("d_name : %s", ptr->d_name);
        if (ptr->d_type==DT_DIR){
            printf("\tDir");
        }
            printf("\n");
    }
    /*close dir*/
    closedir(dir);
    exit(0);
}
```

Compiling:

```
gcc     listdir.c   -o listdir
./listdir
```