

**操作系统课程设计**

|  |  |
| --- | --- |
| **实验、进程控制** | **Experiment 5, Copy Files** |

学院：计算机学院

专业：计算机科学与技术

学生姓名：夏奇拉

学号：1820171025

班级：07111705

Table of Contents

[Purpose 3](#__RefHeading___Toc1766_1221548062)

[Problem Discussion 3](#__RefHeading___Toc1744_1221548062)

[Execution [Windows] 5](#__RefHeading___Toc1768_1221548062)

[Results and Analysis [Windows] 9](#__RefHeading___Toc1770_1221548062)

[Execution [Linux] 10](#__RefHeading___Toc604_992951594)

[Results and Analysis [Linux] 15](#__RefHeading___Toc1770_12215480621)

[Reference: 16](#__RefHeading___Toc606_992951594)

# Purpose

Complete a directory copy command mycp, including the files and subdirectories in the directory, and the running results are as follows:

|  |
| --- |
| [beta@bugs.com](mailto:beta@bugs.com) [~/]# ls -l sem |
| total 56  drwxr-xr-x 3 beta beta 4096 Dec 19 02:53 ./  drwxr-xr-x 8 beta beta 4096 Nov 27 08:49 ../   * rw-r--r-- 1 beta beta 128 Nov 27 09:31 Makefile * rwxr-xr-x 1 beta beta 5705 Nov 27 08:50 consumer\* * rw-r--r-- 1 beta beta 349 Nov 27 09:30 consumer.c   drwxr-xr-x 2 beta beta 4096 Dec 19 02:53 subdir/ |
| [beta@bugs.com](mailto:beta@bugs.com) [~/]# mycp sem target |
| [beta@bugs.com](mailto:beta@bugs.com) [~/]# ls -l target |
| total 56  drwxr-xr-x 3 beta beta 4096 Dec 19 02:53 ./  drwxr-xr-x 8 beta beta 4096 Nov 27 08:49 ../   * rw-r--r-- 1 beta beta 128 Nov 27 09:31 Makefile * rwxr-xr-x 1 beta beta 5705 Nov 27 08:50 consumer\* * rw-r--r-- 1 beta beta 349 Nov 27 09:30 consumer.c   drwxr-xr-x 2 beta beta 4096 Dec 19 02:53 subdir/ |

Description:

Linux: creat, read, write and other system calls

Windows: CreateFile(), ReadFile(), WriteFile(), CloseHandle() and other functions

Requires the ability to copy nested folders (there are folders in the folder), link files

# Problem Discussion

The problem asks that we

1. print the contents of the source directory in the command line
2. run the mycp command to copy the contents of the source directory to the target directory
3. print the contents of the target directory in the command line

# Execution [Windows]

<https://www.installsetupconfig.com/win32programming/windowsfileapis4_22.html>

## Results and Analysis [Windows]

# Execution [Linux]

**Step 1: List the files in the source directory**

In the terminal, list the files and folders in the source directory using the “ls -l” command which allows us to list files and directories in long listing format, one per line. The line shows the file or directory permission, owner and group name, file size, created/modified date and time, file/folder name.

|  |
| --- |
| $ ls -l source |

**Step 2: Create the mycp program**

In the current directory, create the mycp.c file by running the following command:

|  |
| --- |
| $ nano mycp.c |

In the file we write the following code:

|  |
| --- |
| // BIT 100073007 Operating Systems Course Lab 5: Copy File  #include <stdio.h> |

**About the code:**

*Line 132*: The code accepts command line arguments through the declaration of main which accepts two arguments: the number of command line arguments and a full list of all the command line arguments.

The if statement checks that the mycp command is being used correctly with 3 arguments.

*Line 152:* sprintf which stands for String print, stores the output on a char buffer instead of printing to the console. In this case sprintf stores the contents of argv[] to source & target which are of type char. This is being done for clarity in the code.

*Line 157*: opens the source directory using opendir(). Opendir() exists in the sys/types.h library and opens a directory stream corresponding to the directory name, and returns a pointer to the directory stream.

*Line 163*: creates the target directory using opendir() if it doesn’t exist. The new directory will be a copy of the source directory. Stat() retrieves the attributes of the source directory and stores the in the statbuf structure which was declared on *line 142*. This information is used to make the new target directory. These attributes include the file type and mode, the access time and the modification time. Utime() is used to change the file last access and modification times of the target directory

*Line 182*: calls the copyDirectory() function and passes character arrays source and target to is.

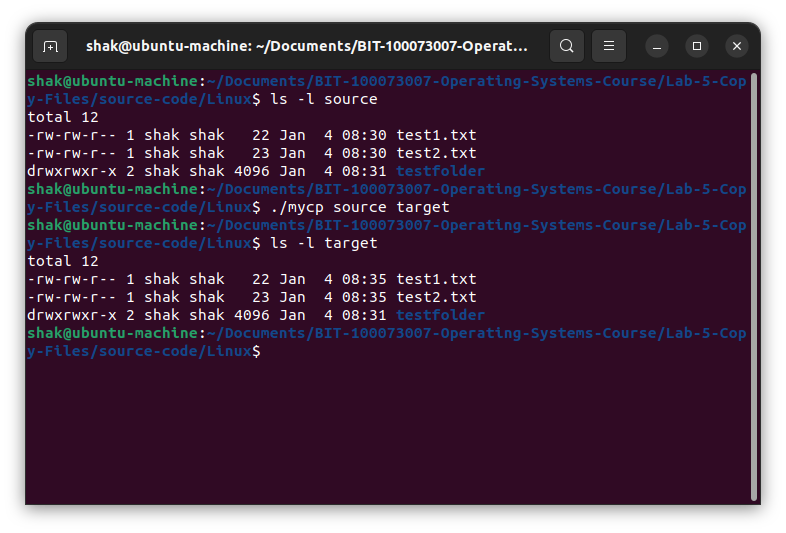
*Line 57*; declares the copyDirectory() function.

*Line 82*: copies the contents of character array sourceDir which is an argument of copyDirectory() and pastes it in character array source which was declared on *line 59*.

*Line 85*: opens the source directory.

*Line 87*: reads the source directory using readdir(). The readdir() function returns a pointer to a dirent structure representing the next directory entry in the directory stream pointed to by dirp. It returns NULL on reaching the end of the directory stream or if an error occurred.

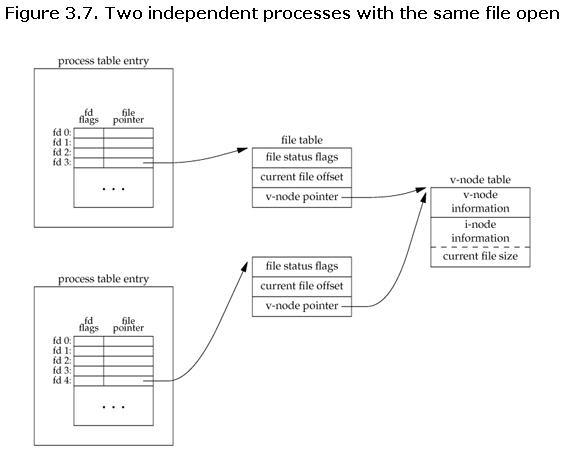
*Line 101*: uses the operator -> which is used to access the data elements of a structure that a pointer variable refers to. The syntax is: *(pointer variable) -> (variable) = value;*



To the kernel, all open files are referred to by File Descriptors. A file descriptor is a non-negative number.

**When we open an existing file or create a new file, the kernel returns a file descriptor to the process.**

The kernel maintains a table of all open file descriptors, which are in use. The allotment of file descriptors is generally sequential and they are allotted to the file as the next free file descriptor from the pool of free file descriptors. When we closes the file, the file descriptor gets freed and is available for further allotment.  
See this image for more details :

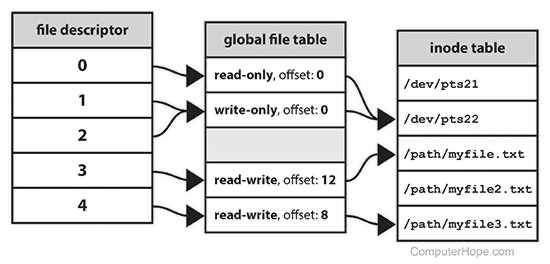
[](https://i.stack.imgur.com/KCjzm.jpg)

When we want to read or write a file, we identify the file with the file descriptor that was returned by **open()** or **create()** function call, and use it as an argument to either **read()** or **write()**.  
It is by convention that, UNIX System shells associates the file descriptor 0 with **Standard Input** of a process, file descriptor 1 with **Standard Output**, and file descriptor 2 with **Standard Error**.  
File descriptor ranges from 0 to OPEN\_MAX. File descriptor max value can be obtained with ulimit -n

A **file descriptor** is a number that uniquely identifies an open file in a computers operating system.

A **file descriptor table** is the collection of integer array indices that are file descriptors in which elements are pointers to file table entries. One unique file descriptors table is provided in an operating system for each process.

A **file table entry** is a structure in memory surrogate for an



<https://www.geeksforgeeks.org/input-output-system-calls-c-create-open-close-read-write/>

[https://profile.iiita.ac.in/bibhas.ghoshal/lab\_files/System%20calls%20for%20files%20and%20directories%20in%20Linux.html](https://profile.iiita.ac.in/bibhas.ghoshal/lab_files/System calls for files and directories in Linux.html)

## Results and Analysis [Linux]

# Reference:

* <https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-getsysteminfo>
* <https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/ns-sysinfoapi-system_info>
* <https://learn.microsoft.com/en-gb/windows/win32/sysinfo/getting-hardware-information?redirectedfrom=MSDN>
* <https://www.installsetupconfig.com/win32programming/windowsvolumeapis1_6.html>
* <https://learn.microsoft.com/en-us/windows/win32/winprog64/virtual-address-space>
* <https://learn.microsoft.com/en-us/windows/win32/memory/memory-management>
* <https://www.tutorialspoint.com/operating_system/os_memory_management.htm>#
* [How Random Access Memory (RAM) affects performance | Dell US](https://www.dell.com/support/kbdoc/en-us/000129805/how-random-access-memory-ram-affects-performance)
* [Physical and Virtual Memory in Windows 10 - Microsoft Community](https://answers.microsoft.com/en-us/windows/forum/all/physical-and-virtual-memory-in-windows-10/e36fb5bc-9ac8-49af-951c-e7d39b979938)
* [Page State - Win32 apps | Microsoft Learn](https://learn.microsoft.com/en-us/windows/win32/Memory/page-state)
* [MEMORY\_BASIC\_INFORMATION (winnt.h) - Win32 apps | Microsoft Learn](https://learn.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-memory_basic_information)
* <https://linuxhint.com/what-does-ls-l-command-do-in-linux/>
* <https://linuxhint.com/linux_file_permissions/>
* <https://man7.org/linux/man-pages/man3/opendir.3.html>
* <https://man7.org/linux/man-pages/man2/lstat.2.html>
* <https://man7.org/linux/man-pages/man3/strcpy.3.html>
* <https://man7.org/linux/man-pages/man2/open.2.html>
* <https://man7.org/linux/man-pages/man2/write.2.html>