CSCI 4061 – Introduction to Operating Systems
Fall 2017

Assignment 2

Due date: October 15, 2017

This assignment can be done either individually or in a team of up to two students.

Instruction: This assignment contains 2 parts. Solution of part 1 must be saved in a PDF file and put in the folder named 'Theory'. Solution of part 2 must be put in the folder 'Project'. Create another file named teaminfo.txt in the same directory containing the name and student IDs of the team members. These 2 folders must be zipped (.zip or .tar or .tar.gz) in a single file and submitted using the Homework Submission link on the moodle web page.

PART - 1 : Theory (20 points)

1) Explain the different process states in the lifecycle of a process. (5)

2) Describe the various options for the *wait* system call used by a process to wait for the completion of its child processes. (5)

3) Describe the actions taken by a kernel to context switch between processes. (5)

4) What are the resources that are common between a parent and its child processes? (5)

PART – 2 : Programming (100 points)

1. Objectives

The objective of this assignment is to learn the basic file and directory handling functions (system calls) in the C programming language. You will learn how to read or write a file, and how to read the contents of a directory, and how to read the attributes of the files present in a directory. You will also learn how to create a new directory in a C program. With the knowledge of these functions, you can understand how programs such as “ls”, “cp”,”mkdir” or “find” are implemented.

Through this assignment you will learn various file system related system calls such as: `open, close, read, write, mkdir, opendir, readdir, stat, lstat, link, symlink, rename.`
You will need to use string manipulation functions such as `strcat` etc.
For generating date/time, you will need to use functions such as: `time, strftime`

Before you start working on the assignment, familiarize yourself with these system calls by going through and understanding the following **example programs on the course webpage**:

- File copy program (filecopy.c) (Shows how to make a copy of a given file.)
- Different versions of the myls program (Shows how ls command is implemented.)
- statdemo.c (Shows use of stat system call to sum sizes of files in a directory .)
- lstatdemo.c (Shows how to read permissions for a file from directory entry.)
- timeinfo.c (Shows how to use `time` and `strftime` functions to generate current date/time string in the format such as `Feb-13-2013-16-59-08`.

Given to you is a skeleton C program to serve as a starting point for your code. Use the following link to download this code skeleton. Assignment2_skeleton.c

2. Important Things to Note

- You must do this assignment using the **C programming language**.
- You are **not allowed** to use any shell commands such as **cp, ls, find, mkdir, mv** etc.
- Assume that the directory specified to your program as input to any of the options will contain **only regular files, directories, or soft links**. It will not contain any device files.

3. Group Size

If you are working in a group, then **only one student should make the submission** and make sure that code document contains names of both the students. Please also include a short r**eadme** file with the names of both the students and steps to run the program (if any) and other useful information for the TAs.

4. Assignment Statement

In this assignment you are required to write a C program that will provide several useful functions for you. These functions include

1. Find the 3 largest files in a given directory tree.
2. List all files of zero length in the given directory tree.
3. In a directory tree, find all the files that have permission specified by the user as 3 octal digits (For example: 777 (or rwxrwxrwx)).
4. Make a complete backup copy of a directory tree.
5. Exit program

You can assume that the directory names and file names will not be longer than 255 characters.

Your program will prompt the user to make a selection from a menu by responding with a single character. A sample menu is shown below based on the skeleton code provided to you.

Example:

```
SELECT THE FUNCTION YOU WANT TO EXECUTE:
  1.Search for largest 3 files in a given directory tree
  2.Find zero length file in a given directory tree
  3.Search for all files with permission specified by user (e.g. : 777) in a
    given directory tree
  4.Create directory back in a given directory tree
  5.Exit
ENTER YOUR CHOICE:
```

The user should enter either 1, 2, 3, 4 or 5.

```
ENTER A DIRECTORY PATH:
```

The user should enter a directory path, either fullpath or relative path

Your program should do error checking to make sure that the specified directory exists

Case 1:

If the user enters 1, your program should then list the 3 largest files (by their length) in the given directory tree (i.e. looking into all nested directories). Your program should print the path names (either full-pathname or relative pathname) and sizes of these files.

Case 2:

If the user enters 2, your program should then list all files of 0 byte length in the given directory tree. Your program should print the path names (either full-path or relative path) of all such zero length files.

Case 3:

If the user enters 3, your program should prompt the user to provide 3 octal digit permissions. According to the user's input, your program should then list all the files in the given directory tree that have the specified permission bits set. For each such file, your program should print its path name (either full-path or relative path).

Case 4:

If the user enters 4, your program will create a back directory of the specified directory by copying the entire directory tree. For example, if the user entered a directory name "CS4061", then the program will create a new directory at the same level with name "CS4061.bak". Your program needs to create a new directory with extension ".bak" for the specified directory. If a directory by that name exists, then move it to "`directoryName.bak`" appended with current date-time string in the format such as `Feb-13-2013-16-59-08.` Please see `timeinfo.c` program on the examples page on the course webpage. For example, if a directory with name "CS4061.bak" already exists, then rename it to "CS4061.bak. Feb-13-2013-16-59-08".

Your program will then copy all files in the source directory tree to the new backup directory. The directory structure in the source tree and the backup tree must be identical. If the source directory contains any soft links, the backup directory should also have those soft links. The soft link in the source directory and the corresponding one in the backup directory will be pointing to the same file.

In this part you can assume the directory specified by the user will contain only regular files, directories, or symbolic links. It will not contain any device files.

<u>Important:</u> When copying a directory tree, ignore directory entries with file names "." and "..".

5. Things to submit

Submit your solution using the Homework Submission link on the class moodle page. Please include the information given in the guidelines on your submission. Create a tar with all you code files such as C program files and header files, a makefile, and a Readme file.

6. Notes on comments

Please comment your code. It is a good practice to comment your code so that it is understandable to you and others. Proper commenting will save you lot of time while debugging. Also see to it that you do not have excessive comments. Comments should be crisp clear and to the point.

7. Grading guidelines

Grading would be broken up into the following main parts:

- 15% Case 1: Search for largest 3 files
- 15% Case 2: for finding all zero length files
- 20% Case 3: Search for files with permission 777(or rwxrwxrwx)
- 40% Case 4: Creating directory backup correctly
  - Check for existing backup directory and move it correctly to a different name
  - Copying files correctly
  - Making soft links correctly
- **For each** of the four cases, 10% of the assigned points are for proper error checking in your code. For example, you must check the return status of all system calls, and in case of errors you must check **`errno`** and use **`perror`** to print the error message. In case of program exit due to un-recoverable errors, such as non-existent directory specified as input by the user or permissions problems, your program should **`exit`** with some non zero status code and write an error message to stderr.
- 5% for Makefile for your code
- 5% Documentation and readability: Comments in the program code, instructions if any for using the program, any assumptions in the program, corner cases in which the program would not work, and any other general comments that the graders should be aware of.

8. Errors and omissions

If there are any errors/omissions in this document, the TAs will post such errors/omissions or other changes to the Class Forum page. It is the responsibility of the students to check the class forum regularly to make sure you do not miss such updates. Moreover, if you have any questions or having difficulty understanding the assignments, you should contact the instructor or the TAs.