CS 332/532 – 1G- Systems Programming

HW 3

Deadline: 10/27/2024 Sunday 11:59pm

Objectives

- To practice creating, managing, and terminating processes using system calls in C.
- To manipulate files and directories with system calls.
- To demonstrate error handling in system-level programming.

Description

In this homework, you will implement a program that performs file manipulation and manages multiple child processes to perform tasks on files in a given directory. The program will:

1. Accept a directory as a command-line argument.
2. For each file in the directory, create a child process that prints the file name, its size, and the number of words in the file.
3. Ensure the parent process waits for all child processes to complete before exiting.
4. Implement basic error handling: check for invalid directories, failed process creation, and inaccessible files.

Detailed Requirements:

- **Command-line Arguments:**
  - The program should accept one command-line argument, which is the directory name.
  - If no argument is provided, the program should print an error message and exit.
  - If the directory is invalid or doesn't exist, print an error message and exit.

- **File and Directory Traversal:**
  - Traverse the directory and list all files (excluding subdirectories).
  - For each file, create a child process using fork().
  - The child process will:
  - Print the file name.
  - Print the file size (in bytes).
  - Count and print the number of words in the file (assume words are separated by spaces).
  - Use stat() or lstat() to get the file size.
  - You may use standard I/O functions like fopen(), fscanf(), or system calls like read() for word counting.

- **Process Management:**
  - Ensure that the parent process waits for all child processes to complete using wait() or waitpid().
  - Handle any failed fork() calls by printing an error message and terminating the program.

- **Error Handling:**
  - Handle cases where the file cannot be opened or read (e.g., permissions issues) and print appropriate error messages.
  - Make sure the program can handle edge cases such as empty files, very large files, or directories with no files.

- **CS 532 Students Only**
  - In addition to printing file names, file sizes, and word counts, the program use additional command-line options to filter and display files based on their ownership

    Command-line options:

    -u <username>: List only the files owned by the specified user.

Program Documentation and Testing

1. Use appropriate names for variables and functions.
2. Use a Makefile to compile your program.
3. Include meaningful comments to indicate various operations performed by the program.
4. Programs must include the following header information within comments:

```
/*
Name:
BlazerId:
Project #:
To compile: <instructions for compiling the program>
To run: <instructions to run the program>
*/
```

4. Test your program with the sample test cases provided as well as your own test cases.
5. You can include any comments you may have about testing in the README.txt file.

Examples

Command	*Description*
./hw3 /path/to/directory	File: file1.txt | Size: 1234 bytes | Words: 200 File: file2.txt | Size: 5678 bytes | Words: 500 File: file3.txt | Size: 345 bytes | Words: 50 ...
If the directory does not exist:	Error: Directory not found.
If no argument is provided:	Usage: ./hw3 <directory_name>
*** graduate students ./hw3 -u username /path/to/directory	File: file1.txt | Size: 1234 bytes | Words: 200 | Owner: username File: file2.txt | Size: 5678 bytes | Words: 500 | Owner: username File: file3.txt | Size: 345 bytes | Words: 50 | Owner: username ...

Submission Guidelines

- **Makefile:** Include a Makefile that compiles the program.
- **README.txt:** Provide instructions for compiling and running the program. Include any known issues or edge cases you handled.
- **Source Code:** Submit your source code and ensure it follows best coding practices, with appropriate comments and documentation.

Grading Rubrics

Description	Points
Correct handling of command-line arguments and directory validation	20 points
Correct traversal of directory and file listing	20 points
Creation and management of child processes	30 points
File size and word count calculation	20 points
Error handling and code documentation	10 points