myfind

Bo Xuan Yang, Barnabás Körömi

Overview

This C++ program searches for files in a directory and its subdirectories (if enabled) using parallel processes. Each filename is searched in a separate child process, utilizing fork(). The program can perform case-insensitive searches and handle recursion with -R and -i flags. The parent process waits for all child processes to finish before terminating.

Key Parallelization Concepts

Forking Processes:

The program creates a new child process for each filename search using fork(). This allows multiple filenames to be searched in parallel.

Process Synchronization:

The parent process waits for all child processes to complete using wait(), preventing orphaned "zombie" processes.

Output Handling:

Each child process prints its search results to stdout. Since multiple processes output concurrently, the results may be interleaved, but each result includes the PID, making it identifiable.

Functions Explanation

toLowerCase(std::string &str):

Converts the input string to lowercase for case-insensitive comparison using std::transform().

findFile(const std::string &directory, const std::string &filename, bool recursive, bool caseInsensitive):

Searches for a filename in a specified directory. It handles recursion, case-insensitive comparison, and prints matching files' paths along with the process ID.

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

Handles command-line arguments (-R for recursion, -i for case-insensitivity), then forks a child process for each filename. The parent process waits for all children to finish.

Execution Flow

Input Parsing:

getopt() processes flags (-R, -i) and gathers the directory and filenames.

Forking:

The program forks a child process for each filename to search in parallel.

File Search:

Each child process calls findFile() to search the directory. It handles recursion and case-insensitivity as needed.

Synchronization:

The parent waits for all child processes to complete using wait().

Output:

Each child process outputs its results with the format: <pid>: <filename>: <full path>.

Challenges

Concurrency in Output:

Since multiple processes write to stdout simultaneously, the output can be interleaved, but each result includes a PID to identify the source process.

Fork Overhead:

Forking a large number of processes may incur overhead, especially if the directory structure is not large.

Build and Run:

Build it within the WSL

g++ -o myfind myfind.cpp

Test cases

Test Case 1 (pass): Search for a specific file in a directory (non-recursive, case-sensitive):

./myfind ./test_dir Test.txt test.doc test

Test Case 1 (fail – no output): Search for a specific file in a directory (non-recursive, case-sensitive):

./myfind ./test_dir test.txt test.doc test

Test Case 2: Search for a file recursively in subdirectories (-R flag):

./myfind -R ./test_dir testing.txt

Test Case 3: Search for a file with case-insensitive comparison (-i flag):

./myfind -i ./test_dir test.txt test.doc test

Test Case 4: Combine both -R and -i flags for a recursive, case-insensitive search:

./myfind -Ri ./test_dir test.txt test.doc Testing