Protocol MyFind: Parallelization and Output

Mohamad Aissa, Noah Howadt

Synchronization

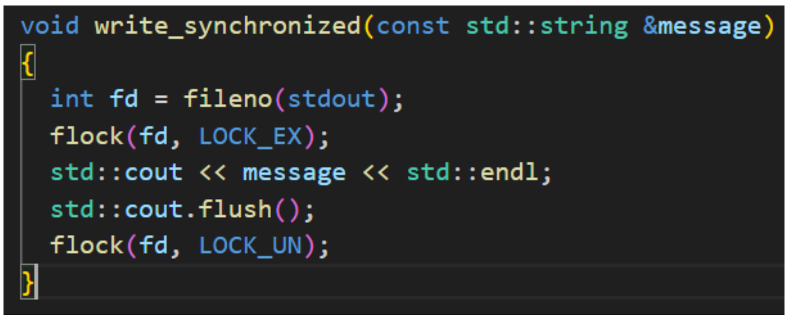
Parallelization Overview

To parallelize the program, we used fork() to create a separate processes for each filename. For every child process, the function searchFolder() is called, which searches for the file.

Synchronization Mechanism

One of the challenges with parallelization is ensuring that multiple processes do not produce interleaved output when writing to the standard output. To prevent this from happening, synchronization mechanisms are necessary to ensure that only one process writes to stdout at a time. We decided to use flock() for this purpose. This function allows processes to acquire a lock on the file descriptor before writing. By acquiring a lock before writing and releasing it afterward, processes ensure that their output is not interleaved with others.

Additionally, after each write operation, it is important to flush the output buffer using cout.flush() to ensure that all buffered data is written immediately to the console, rather than waiting for the buffer to fill up. This guarantees that the output appears in the correct order.



```
void write_synchronized(const std::string &message)
{
  int fd = fileno(stdout);
  flock(fd, LOCK_EX);
  std::cout << message << std::endl;
  std::cout.flush();
  flock(fd, LOCK_UN);
}
```