HACETTEPE UNIVERSITY

COMPUTER ENGINEERING

BBM342 OPERATING SYSTEMS

EXPERIMENT 2

Name:	Kürşat AKTAŞ
Student No:	21227949
Contact:	kursat.ce@gmail.com

Subject:	Multi-Thread Programming, Inter Process Communication, File System Calls

CONTENT

1- Aim Of The Experiment

In this experiment I have tried to implement multi-threaded string searcher program with ANSI C. For this purpose I have used pipe for inter-processes communication between parent and child processes, mutex for protecting critical sections and File system calls for traversing directories.

2- Usage

Firstly compile the two necessary C files like below;

```
gcc minion.c -o minion -Wall -ansi -lpthread
gcc main.c -o main -Wall -ansi -lpthread
```

Then you can run the program with below parameter explanation;

```
./main <minion_count> <buffer_size> <search_string> <root_directory>
```

3- Program Explanation

Although there are enough comments inside the code, explaining again in this document may helps the reader.

When the programs starting it firstly initialize mutexes. There are two mutexes, one for protecting the file buffer (which stores the paths of the .txt files) and one for protecting the log file (which stores to program logs such as threads activities). I have also used mutex condition variables for both the producer and consumer operation.

After mutex initializing is done program firstly creates file searcher thread (which is producer). This thread searches given directory and if it founds a file with .txt extension, then adds to the FIFO queue (FIFO queue implementation can be seen in main.h file). If the file buffer is full then searcher waits until buffer is available. When searcher thread traverse whole directory, it enables the "finished" flag.

After that all necessary controller threads created respectively. When creating a new controller thread, the following steps are followed respectively. It firstly creates 2 pipes. One pipe("fd_path_pipe") is used for sending path informations of .txt files from controllers to minion(child) processes and the other one("fd_log_pipe") is used for sending log information of minions processes from child process to controller thread. After creating these files, controller forks and creates a new child process. Then child process replaces its content with executable version of “minion.c” file. Minions takes the file pathes from pipe and starts to search given string in it. When string is found it writes to necessary informations to its output file and informs its controller thread about that.

Lastly, when all the files searched and the file buffer completely consumed, controllers sends “finito” message to minions and kills themselves. Also received “finito” message causes to killing of child processes.

4- Bugs

- **Bug 1:** The program can not traverse the subdirectories it can only traverse root directory.
- **Bug 2:** The program can not find all the occurrences of search key in the same line. It can only finds the first occurrence of key for each line.
- **Bug 3:** When the minion counts is greater than 3, the program doesn't terminate itself. But it produces output files.

5- Resources

1. **http://tldp.org/LDP/lpg/node11.html**
2. **http://www.unix.com/man-page/POSIX/1posix/export**
3. **http://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html**
4. **http://www.csc.villanova.edu/~mdamian/threads/posixsem.html**
5. **http://www.cs.fsu.edu/~baker/realtime/restricted/notes/pthreads.html**