Project #3

Due Date: 04/26/2023

Write a very small web server which has the following capabilities.

The goals of this assignment include the use of kernel threads, use of system calls like stat(), the use of signals, and the implementation of critical sections.

The basic function is that a main process (server) will create N worker processes. The main process will listen for network connections and hand off that connection to a worker thread. The worker thread will service the request until done and then make itself available for more work.

The worker thread will take requests like "info", "get", and "exit". There is no need to error check syntax -- if it is not correct, say unknown and continue servicing requests from the connection until either a timeout occurs or "exit."

The server will send back files and information to the client (you probably need to write a client to test your server). You should use a TCP/IP stack.

Must be written in C
Must be well documented -- must have clear areas described on what you are doing

1) The program shall take 4 parameters on startup
   a) The port to accept connections (PORT)
   b) The total number of connections allowed (#CON)
   c) The max time the server will wait for a new request (TIMEOUT).
   d) directory root where all served files will be pulled from (DIR)

2) The program shall open a listener on port PORT

3) The program shall create #CON threads which will suspend themselves until they receive a request from the parent process

4) Each thread will handle requests of the form
   a) info FILE
      The response to the request will be the modification date of the file and the size of the file (in bytes)
      -- please see http://linux.about.com/library/cmd/blcmdl2_stat.htm
      -- you should return st_mtime and st_size
      -- -1 -1 will be the response if the file does not exist or is not accessible
   b) get FILE
      -- returns st_size (-1 if the file does not exist or is not accessible)
      -- then the entire contents of the file

c) exit (closes connection)

d) all file accesses will be from DIR and below

5) each thread will close connections when TIMEOUT is reached and there has been no activity.

6) The main process will create #CON threads, establish a listener on port PORT. The main process will receive a connection request, set up the connection and then hand off the connection to waiting thread. The waiting service thread will then take care of interactions with the connection

You MUST pay careful attention to how a thread comes on and off the work queue. You should use semaphores or other techniques to ensure that there is never a race condition for a thread to place itself on the wait queue and for the main process to take something off the wait queue. You need to document how you do this activity.