Brief Design(Q2)

- concurrent_web_client.c is a program to download a file from a server over the internet using the HTTP GET request along with the range field.

- The basic design is as follows :
  (1) The user executes the program from the terminal providing the domain name, file path(on the server), N-the level of concurrency, i.e., the number of child processes that would separately fetch the data in the respective ranges and the name that the downloaded file would be stored as.
  (2) The program resolves the domain name and creates a socket to bind to the server using a TCP connection
  (3) Once connected the client sends a HEAD request specifying the path of the file which gets the reponse without the body. This is basically done to retrieve length of the file to be downloaded by extracting the information stored in the Content-Length field of the response packet.
  (4) The content length is then split into N parts. This would be the amount of data that each forked() child would be fetching individually
  Eg: if the file to be downloaded is 1048576 bytes(1MB) and the level of concurrency is 3 (N=3) then the split would be as follows
  Child[1] = 349525 bytes
  Child[2] = 349525 bytes
  Child[3] = Rest of the bytes = 349526
  (5) Next the parent runs a loop N times. Each time, it forks() a child process and calls a function fetch_and_push(range , i) where range is the amount of data to be fetched by the i'th child
  (6) When the child is executing fetch_and_push(), it sends the GET request with the respective number of bytes as recieved in range and stores the received data in a temporary file on the disk with the respective child number.
  Eg: temp3.txt
  (7) The child then pushes the response status, the content length received and the message type is set to the child process number(i)
  (8) After each child is done with the above the parent picks the messages one by one from the message queue, if the status is 206(partial content) or 200(ok) it marks it and moves ahead but if the status is anything else, it forks() and calls fetch_and_push(range , i) where i is the child for which the status was not ok, hence the created process again tries to fetch the data in that range and pushes it to the message queue.
  (9) Once all the messages are marked ok the file assembly starts
  (10) The parent process reads the files in which the responses were stored one by one sequentially and copies the data to the final download file destroying the temporary files on the way.
  (11) Once assembly is done, the parent prints "success" and exits.
  (12) The maximum number of retries to fetch a particular segement is set to 20 after which if the content is still not received correctly, the download is aborted. This can be changed in the code.

Authors:
Burhan Boxwalla – 2017A7PS0097P.
Harpinder Jot Singh – 2017A7PS0057P.