Design Document: HTTP client and server

Yujia Li
CruzID: yli302

1 Goals

This is an simple single-threaded HTTP client and server system, which implements PUT and GET features.

2 Design

Functions:

```
     // use isdigit to check if char *port is valid(all
numbers).
     bool isValidPort(char *port)

     // use atoi to convert the number after Content-Length to a
     // int32_t
     Int32_t parseContentLength(char *cLength)

     /* check if the request is a valid Header:
      *   valid header include:
      *    PUT httpname HTTP/1.1\r\n\r\n
      *     or
      *    PUT httpname HTTP/1.1\r\nContent-Length: num\r\nDATA
      *     or
     */    GET httpname HTTP/1.1\r\n\r\n
     bool isValidHeader(char **header)

     /* check if the arguemnt is a valid request:
      *   valid request include:
      *    s:filename:httpname
      *     or
     */    r:httpname:filename
     bool isValidRequest(char **request)
```

Client part:

Input: Argument count: `arg_count`

Input: First argument: `address:port`

Input: rest arguments: send instructions or receive instructions.

```
s:filename:httpname
r:httpname:filename
```

Output:

```
------------------------
if argc < 2
     error(invalid argument)
else
     if (token = strtok(argv[1], split) then
          if !isValidPort(token) then
               error(invalid port number)
               exit(1)
          end
     end
end
create a socket
for(i=2 to argc)
     if sock == -1 || connect() == -1 then
          error(fail create socket)
          continue
     end
     for(j=0 to curr_argv[j] != NULL)
          error(request argument is wrong)
     End

     // PUT request
     if argv[0] == 's' then
          if open(filename, O_RDONY) == -1 ||
             read(fd, fileBuf, 4096) == -1 then
               error(fail to read)
               continue
          end
          send("PUT httpname HTTP/1.1\r\n
               Content-Length: file_size\r\n\r\n")
          while count != 0 do
               send(sock, fileBuf, count, 0)
               count = read(fd, fileBuf, 4096)
          End
          fprintf(stdout, headerBuf)
```

```
          continue
     end

     // GET request
     if argv[0] == 'r' then
          if open(filename, O_CREAT|O_WRONLY|O_TRUNC) == -1 then
               error(fail to open)
               continue
          end
          if recv(sock, headerBuf, 4096, 0) == 0 then
               error(no response header)
          end
          obtain status code by computing address
          if Content-Length exists then
               cLength = atoi(Content-Length)
          end
          if \r\n\r\n doesn't exists then
               error(invalid response header)
               continue
          end
          store data in fileBuf
          while count != 0 do
               if write(fd, fileBuf, strlen(fileBuf)) == -1 then
                    error(fail to write)
                    Continue
               end
               if (bufferRead += count) == cLength then
                    break
               end
               count = recv(sck, fileBuf, 4096, 0)
          end
          continue
     end
end
```

Server part:

Input: Argument count: `arg_count`

Input: First argument: `address:port`

```
if  argc != 2 then
     error(invalid argument)
else
     if (token = strtok(argv[1],split)) != NULL then
          if port number is invalid then
               error(invalid port number)
          end
     PORT_NUMBER = atoi(token)
end
create a socket
fprintf(stdout, "HTTPServer address: SERVER_NAME PORT_NUMBER")
if sock == -1 || setsockopt() == -1 || bind() == -1 then
     error(fail to create socket)
while true do
     if listen() == -1 || accept() == -1 then
          error(faile to listen / accept)
     end
     obtain header by computing address and print stdout
     if !isValidHeader(request_header) then
          send(400 Bad Request\r\n\r\n)
          fprintf(stderr, "400 Bad Request")
          continue
     end

     // PUT request
     if request_header[0] == PUT then
          if open(fd, O_WRONLY|O_TRUNC) == -1 then
             if open(fd, O_CREAT|O_WRONLY|O_TRUNC,S_IRWXU) == -1
               send(403 Forbidden\r\n\r\n)
               continue
             end
          end
          if write(fd, 0, 0) == -1 then
               send(403 Forbidden\r\n\r\n)
               fprintf(stderr, "403 Forbidden")
               continue
          end
```

```
          send(200 OK\r\n\r\n or 201 Created\r\n\r\n)
          While recv() != 0 do
               if write() == -1 then
                    send(403 Forbidden\r\n\r\n)
                    fprintf(stderr, 403 Forbidden)
                    Continue
               End
               If bufferRead == content_Length break
               Count = recv()
          End
     end

     // GET request
     if request_header[0] == GET then
          if open(filename, O_RDONY) == -1 ||
             read(fd, fileBuf, 4096) == -1 then
               error(fail to read)
               continue
          end
          get content length
          send(HTTP/1.1 200 OK\r\nContent-Length: file_size\r\n)
          while(local file is not end) then
               sned(fileData)
          end
     end
end
```