COMP3331 Lab03 report

Haibo Wang(z5135009)

Exercise 3

Q1:

```
IP address is 150.203.161.98.

DNS query type is A.
```

Q2:

```
CNAME: rproxy.cecs.anu.edu.au.

IP address is 150.203.161.98.

Faster final A record resolution speed,
and more mnemonic.
```

Q3:

```
Authority section indicates the server(s) that are the ultimate authority for
answering DNS queries about that domain. Additional section contains other help
records. For example, from my output it contains the A record(IPv4) and AAAA
record(IPv6) for authority servers
```

output:

```
$ dig www.cecs.anu.edu.au A
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.cecs.anu.edu.au A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4251
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 7

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.cecs.anu.edu.au.       IN  A

;; ANSWER SECTION:
www.cecs.anu.edu.au.    3600    IN  CNAME   rproxy.cecs.anu.edu.au.
rproxy.cecs.anu.edu.au. 3600    IN  A   150.203.161.98

;; AUTHORITY SECTION:
cecs.anu.edu.au.    2065    IN  NS  ns4.cecs.anu.edu.au.
```

```
cecs.anu.edu.au.    2065    IN  NS  ns2.cecs.anu.edu.au.
cecs.anu.edu.au.    2065    IN  NS  ns3.cecs.anu.edu.au.

;; ADDITIONAL SECTION:
ns2.cecs.anu.edu.au.    2065    IN  A   150.203.161.36
ns2.cecs.anu.edu.au.    2065    IN  AAAA    2001:388:1034:2905::24
ns3.cecs.anu.edu.au.    2065    IN  A   150.203.161.50
ns3.cecs.anu.edu.au.    2065    IN  AAAA    2001:388:1034:2905::32
ns4.cecs.anu.edu.au.    2065    IN  A   150.203.161.38
ns4.cecs.anu.edu.au.    2065    IN  AAAA    2001:388:1034:2905::26

;; Query time: 196 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Tue Aug 14 21:30:15 AEST 2018
;; MSG SIZE  rcvd: 271
```

Q4:

```
129.94.242.49
```

Q5:

```
NS:
    ns4.cecs.anu.edu.au.
    ns2.cecs.anu.edu.au.
    ns3.cecs.anu.edu.au.
ns2.cecs.anu.edu.au.    3490    IN  A       150.203.161.36
ns2.cecs.anu.edu.au.    3490    IN  AAAA    2001:388:1034:2905::24
ns3.cecs.anu.edu.au.    3490    IN  A       150.203.161.50
ns3.cecs.anu.edu.au.    3490    IN  AAAA    2001:388:1034:2905::32
ns4.cecs.anu.edu.au.    3490    IN  A       150.203.161.38
ns4.cecs.anu.edu.au.    3490    IN  AAAA    2001:388:1034:2905::26

Type=NS
```

Q6:

```
a.root-servers.net. nstld.verisign-grs.com

type=NS
```

Q7:

```
No. the server does not provide recursive query
```

Q8:

```
ns1.cecs.anu.edu.au. hostmaster.cecs.anu.edu.au
```

Q9:

```
MX
```

Q10:

```
6 times.
```

Q11:

```
Yes, but it is not recommended to assign multiple IP addresses on a physical
machine (ie:computer). For  avoiding the bottlenecks.
```

Exercise 4

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>

#define MAX_HEADER_LEN 8190

int main(int argc, char const *argv[]){
    // create socket
    int s = socket(AF_INET, SOCK_STREAM, 0);
    if (s < 0){
        perror(NULL);
        exit(1);
    }
    // bind the socket to the listing port
    struct sockaddr_in serverAddress;
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr = INADDR_ANY;
    int portNum = atoi(argv[1]);
    if(portNum < 0)
        portNum = 80;
    serverAddress.sin_port = htons(portNum);
    if(bind(s, (struct sockaddr *)&serverAddress, sizeof(serverAddress)) < 0){
        perror(NULL);
        exit(1);
    }
```

```
    //wait for a browser to request a connection
    int maxBacklog = 5;
    // listen for a connection
    if(listen(s, maxBacklog)){
        perror(NULL);
        exit(1);
    }

    char request[MAX_HEADER_LEN];
    while(1){
        struct sockaddr_in clientAddr;
        // open a new connection for this conversation
        socklen_t clientLen = sizeof(clientAddr);
        int ss = accept(s, (struct sockaddr *)&clientAddr, &clientLen);
        FILE *ss_stream = fdopen(ss, "r");
        fgets(request, MAX_HEADER_LEN - 1, ss_stream);

        char method[5];
        char query[100];
        if(sscanf(request, "%s %s ", method, query) != 2){
            fclose(ss_stream);
            continue;
        }
        printf("%s\n", request);

        // read until header finishes
        while(fgets(request, MAX_HEADER_LEN - 1, ss_stream)){
            if(strcmp(request, "\r\n") == 0)
                break;
        }

        int fd;
        if((fd = open(query + 1, O_RDONLY)) < 0){
            perror(NULL);
            char *msg = "HTTP/1.1 404 Not Found\r\nConnection: close
\r\nContent-Type: text/html\r\n\r\n<html><body>404 Not Found<body><html>";
            write(ss, msg, strlen(msg));
            fclose(ss_stream);
            continue;
        }

        // send HTTP header
        int len;
        "HTTP/1.1 200 OK\r\n";
        len = lseek(fd, 0, SEEK_END);
        lseek(fd, 0, SEEK_SET);
        char resHeader[100];
        char *token = strtok(query, ".");
        char *ext;
        char *mimeType;
        while(token != NULL){
            ext = token;
            token = strtok(NULL, ".");
        }
        if(strcmp(ext, "html") == 0){
            mimeType = "text/html";
        } else if(strcmp(ext, "jpg") == 0 ||
```

```
                strcmp(ext, "jpeg") == 0){
            mimeType = "image/jpeg";
        } else if(strcmp(ext, "png") == 0){
            mimeType = "image/png";
        } else{
            mimeType = "application/octet-stream";
        }


        sprintf(resHeader, "HTTP/1.1 200 OK\r\n"
            "Content-Length: %d\r\n"
            "Content-Type: %s\r\n"
            "Connection: close\r\n\r\n", len, mimeType);
        write(ss, resHeader, strlen(resHeader));
        char msg[1000];
        while((len = read(fd, msg, sizeof(msg))) > 0){
            write(ss, msg, len);
        }

        printf("Response sent.\n");

        close(fd);
        fclose(ss_stream);
    }

    close(s);
    return 0;
}
```