

CSSE2310/7231 Week 10 Tuesday Code Examples

To be commented in class.

Sample Address Handling Code 1

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    uint32_t hostRepresentation;
    struct in_addr networkRepresentation;

    if(argc != 2) {
        fprintf(stderr, "Usage: %s ip-address\n", argv[0]);
        exit(1);
    }

    if(!inet_aton(argv[1], &networkRepresentation)) {
        fprintf(stderr, "Invalid IP address: %s\n", argv[1]);
        exit(1);
    }

    hostRepresentation = ntohl(networkRepresentation.s_addr);

    printf("%s: printed as hex: %08x, network rep printed as hex: %08x\n",
        argv[1], hostRepresentation, networkRepresentation.s_addr);
    if(networkRepresentation.s_addr == hostRepresentation) {
        printf("Host uses big-endian representation\n");
    } else {
        printf("Host uses little-endian representation\n");
    }

    return 0;
}
```

Sample Address Handling Code 2

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>

int main(int argc, char* argv[])
{
    struct in_addr a;
    uint32_t hostAddress;

    hostAddress = (216<<24) + (254<<16) + (125<<8) + 70;

    a.s_addr = htonl(hostAddress);

    printf("IP address (host ordered, hex): %08x, String: %s\n",
        hostAddress, inet_ntoa(a));

    return 0;
}
```

Sample Client

```
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <netdb.h>
#include <string.h>

struct in_addr* name_to_IP_addr(char* hostname)
{
    int error;
    struct addrinfo* addressInfo;

    error = getaddrinfo(hostname, NULL, NULL, &addressInfo);
    if(error) {
        return NULL;
    }

    return &(((struct sockaddr_in*)(addressInfo->ai_addr))->sin_addr);
}

int connect_to(struct in_addr* ipAddress, int port)
{
    struct sockaddr_in socketAddr;
    int fd;

    fd = socket(AF_INET, SOCK_STREAM, 0);
    if(fd < 0) {
        perror("Error creating socket");
        exit(1);
    }

    socketAddr.sin_family = AF_INET;
    socketAddr.sin_port = htons(port);
    socketAddr.sin_addr.s_addr = ipAddress->s_addr;

    if(connect(fd, (struct sockaddr*)&socketAddr, sizeof(socketAddr)) < 0) {
        perror("Error connecting");
        exit(1);
    }
    return fd;
}

void send_HTTP_request(int fd, char* file, char* host)
{
    char* requestString;

    requestString = (char*)malloc(strlen(file) + 26);
    sprintf(requestString, "GET %s HTTP/1.0\r\nHost: %s\r\n\r\n", file, host);

    if(write(fd, requestString, strlen(requestString)) < 1) {
        perror("Write error");
        exit(1);
    }
}
```

```

void get_and_output_HTTP_response(int fd)
{
    char buffer[1024];
    int numBytesRead;
    int eof = 0;

    while(!eof) {
        numBytesRead = read(fd, buffer, 1024);
        if(numBytesRead < 0) {
            perror("Read error\n");
            exit(1);
        } else if(numBytesRead == 0) {
            eof = 1;
        } else {
            fwrite(buffer, sizeof(char), numBytesRead, stdout);
        }
    }
}

int main(int argc, char* argv[]) {
    int fd;
    struct in_addr* ipAddress;
    char* hostname;

    if(argc != 2) {
        fprintf(stderr, "Usage: %s hostname\n", argv[0]);
        exit(1);
    }
    hostname = argv[1];

    ipAddress = name_to_IP_addr(hostname);
    if(!ipAddress) {
        fprintf(stderr, "%s is not a valid hostname\n", hostname);
        exit(1);
    }

    fd = connect_to(ipAddress, 80);
    send_HTTP_request(fd, "/", hostname);
    get_and_output_HTTP_response(fd);
    close(fd);
    return 0;
}

```

Sample Server

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <unistd.h>
#include <netdb.h>

#define MAXHOSTNAMELEN 128

int open_listen(int port)
{
    int fd;
    struct sockaddr_in serverAddr;
    int optVal;

    fd = socket(AF_INET, SOCK_STREAM, 0);
    if(fd < 0) {
        perror("Error creating socket");
        exit(1);
    }

    optVal = 1;
    if(setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &optVal, sizeof(int)) < 0) {
        perror("Error setting socket option");
        exit(1);
    }

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(port);
    serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);

    if(bind(fd, (struct sockaddr*)&serverAddr, sizeof(struct sockaddr_in)) < 0){
        perror("Error binding socket to port");
        exit(1);
    }

    if(listen(fd, SOMAXCONN) < 0) {
        perror("Error listening");
        exit(1);
    }

    return fd;
}

char* capitalise(char* buffer, int len)
{
    int i;

    for(i=0; i<len; i++) {
        buffer[i] = (char)toupper((int)buffer[i]);
    }
    return buffer;
}
```

```

void process_connections(int fdServer)
{
    int fd;
    struct sockaddr_in fromAddr;
    socklen_t fromAddrSize;
    int error;
    char hostname[MAXHOSTNAMELEN];
    char buffer[1024];
    ssize_t numBytesRead;

    while(1) {
        fromAddrSize = sizeof(struct sockaddr_in);

        fd = accept(fdServer, (struct sockaddr*)&fromAddr, &fromAddrSize);
        if(fd < 0) {
            perror("Error accepting connection");
            exit(1);
        }

        error = getnameinfo((struct sockaddr*)&fromAddr, fromAddrSize, hostname,
            MAXHOSTNAMELEN, NULL, 0, 0);
        if(error) {
            fprintf(stderr, "Error getting hostname: %s\n", gai_strerror(error));
        } else {
            printf("Accepted connection from %s (%s), port %d\n",
                inet_ntoa(fromAddr.sin_addr),
                hostname,
                ntohs(fromAddr.sin_port));
        }

        while((numBytesRead = read(fd, buffer, 1024)) > 0) {
            capitalise(buffer, numBytesRead);
            write(fd, buffer, numBytesRead);
        }

        if(numBytesRead < 0) {
            perror("Error reading from socket");
            exit(1);
        }
        printf("Done\n");
        fflush(stdout);
        close(fd);
    }
}

int main(int argc, char* argv[])
{
    int portnum;
    int fdServer;

    if(argc != 2) {
        fprintf(stderr, "Usage: %s port-num\n", argv[0]);
        exit(1);
    }

    portnum = atoi(argv[1]);
    if(portnum < 1024 || portnum > 65535) {
        fprintf(stderr, "Invalid port number: %s\n", argv[1]);
        exit(1);
    }

    fdServer = open_listen(portnum);
    process_connections(fdServer);
    return 0;
}

```