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

#include "media_transfer.h"

#define LEN 1024

int send_media(int sockfd, const char *media_path, size_t length) {
    int n;
    char *data = malloc(length);

    FILE *fp = fopen(media_path, "rb");
    if(fp == NULL){
        printf("File: %s, not Found", media_path);
        return -1;
    }

    size_t sent = 0;
    fread(data, length, 1, fp);
    while(sent < length) {
        size_t t = send(sockfd, data, length, 0);
        if (t != -1) {
            sent += t;
        } else {
            perror("send_media");
            exit(1);
        }
    }

    fclose(fp);
    free(data);
    return 1;
}

int receive_media(int sockfd, const char *filename, size_t length) {
    unsigned int n = 0;
    size_t pos = 0;
    FILE *fp;
    char buffer[LEN];
    char *media = malloc(length);

    while (1) {
        n = read(sockfd, buffer, LEN);
        if (n < 0) continue;
        memcpy(media + pos, buffer, n);
        pos += n;
        if (pos >= length) break;
    }

    fp = fopen(filename, "w");
    fwrite(media, length, 1, fp);
    fclose(fp);
    free(media);

    return 1;
}

int get_media_list(const char *path, char *buffer, size_t buffer_size) {
    DIR *dh = opendir(path);
    struct dirent *d;
    struct stat fstat;

```

```
70      int n = 0;
71      n += sprintf(buffer, "\tSize\t\tName\n");
72      while((d = readdir(dh)) != NULL) {
73          stat(d->d_name, &fstat);
74          n += sprintf(buffer + n, "\t%ld\t\t%s\n", fstat.st_size, d->d_name);
75      }
76      closedir(dh);
77      return 1;
78  }
79
80  int send_header(int client_socket, int port, size_t media_size, const char *media_type,
    int status) {
81      char host[256];
82      char *IP;
83      struct hostent *host_entry;
84      int hostname;
85
86      //find the host name
87      hostname = gethostname(host, sizeof(host));
88      if(hostname == -1) {
89          printf("Cannot find host information");
90      }
91
92      //find host information
93      host_entry = gethostbyname(host);
94      if(host_entry == NULL) {
95          printf("Cannot find the host from id\n");
96      }
97
98      //Convert into IP string
99      IP = inet_ntoa(*((struct in_addr*) host_entry->h_addr_list[0]));
100
101     // create the header
102     char header[LEN];
103     int n = 0;
104     n += sprintf(header, "Status: %d\r\n", status);                  // req is valid
105     n += sprintf(header + n, "Host: %s:%d\r\n", IP, port);           // append host
    information
106     n += sprintf(header + n, "Type: %s\r\n", media_type);            // append file type
107     n += sprintf(header + n, "Length: %ld\r\n\r\n", media_size);         // append file
    length
108
109     // finally send the header packet
110     if(send(client_socket, header, n, 0) == -1) {
111         return -1;
112     }
113     else{
114         return 0;
115     }
116 }
117
```