<https://api.libssh.org/master/group__libssh__scp.html#ga2ca698c1e49612c083d9f8a72df52188>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char \*argv[]) {

if (argc != 2) {

fprintf(stderr, "Usage: %s filename\n", argv[0]);

exit(1);

}

FILE \*fp;

char buffer[4096];

int bytes\_read;

long file\_size;

fp = fopen(argv[1], "wb");

if (fp == NULL) {

perror("Error opening file");

exit(1);

}

while ((bytes\_read = fread(buffer, 1, sizeof(buffer), stdin)) > 0) {

if (fwrite(buffer, 1, bytes\_read, fp) != bytes\_read) {

perror("Error writing file");

exit(1);

}

}

if (ferror(stdin)) {

perror("Error reading input");

exit(1);

}

file\_size = ftell(fp);

fclose(fp);

printf("File transfer complete. File size: %ld bytes\n", file\_size);

// Send acknowledgment back to client

printf("ACK\n");

return 0;

}

On the client side, you can modify your SCP script to wait for the "ACK" message before sending the next file. Here's an example:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char \*argv[]) {

if (argc != 3) {

fprintf(stderr, "Usage: %s file server\n", argv[0]);

exit(1);

}

char command[1024];

char response[1024];

FILE \*fp;

fp = fopen(argv[1], "rb");

if (fp == NULL) {

perror("Error opening file");

exit(1);

}

sprintf(command, "scp -t %s", argv[2]);

FILE \*scp = popen(command, "w");

if (scp == NULL) {

perror("Error executing SCP command");

exit(1);

}

// Send file name and size

fprintf(scp, "C0644 %ld %s\n", (long)ftell(fp), argv[1]);

fflush(scp);

// Send file contents

int c;

while ((c = fgetc(fp)) != EOF) {

fputc(c, scp);

}

// Send end-of-file marker

fprintf(scp, "\x00");

fflush(scp);

// Wait for ACK message

fgets(response, sizeof(response), scp);

if (strcmp(response, "ACK\n") != 0) {

fprintf(stderr, "Error: invalid response from server");

exit(1);

}

pclose(scp);

fclose(fp);

return 0;

}

In this example, the client program sends a file to the server using SCP. After the file is sent, the client waits for an "ACK" message from the server using fgets(). If the message is not "ACK\n", the client program exits with an error.Note that this example assumes that the server program sends only one line of text (the "ACK" message) to the standard output.

**Client side**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char \*argv[]) {

if (argc != 3) {

fprintf(stderr, "Usage: %s file server\n", argv[0]);

exit(1);

}

char command[1024];

char response[1024];

FILE \*fp;

int c;

fp = fopen(argv[1], "rb");

if (fp == NULL) {

perror("Error opening file");

exit(1);

}

sprintf(command, "scp -t %s", argv[2]);

FILE \*scp = popen(command, "w");

if (scp == NULL) {

perror("Error executing SCP command");

exit(1);

}

// Send file name and size

fprintf(scp, "C0644 %ld %s\n", (long)ftell(fp), argv[1]);

fflush(scp);

// Send file contents

while ((c = fgetc(fp)) != EOF) {

fputc(c, scp);

}

// Send end-of-file marker

fprintf(scp, "\x00");

fflush(scp);

// Wait for ACK message

while (fgets(response, sizeof(response), stdin) != NULL) {

if (strcmp(response, "ACK\n") == 0) {

break;

}

}

pclose(scp);

fclose(fp);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char \*argv[]) {

if (argc != 3) {

fprintf(stderr, "Usage: %s file server\n", argv[0]);

exit(1);

}

char command[1024];

char response[1024];

FILE \*fp;

int c;

fp = fopen(argv[1], "rb");

if (fp == NULL) {

perror("Error opening file");

exit(1);

}

sprintf(command, "scp -t %s", argv[2]);

FILE \*scp = fopen("|-", "w");

if (scp == NULL) {

perror("Error executing SCP command");

exit(1);

}

// Connect to server using SSH and send SCP command

fprintf(scp, "%s\n", command);

fflush(scp);

// Wait for server's response

while (fgets(response, sizeof(response), scp) != NULL) {

if (response[0] == 'C' && response[1] == '0' && response[2] == '6' && response[3] == '4') {

break;

}

}

// Send file name and size

fprintf(scp, "C0644 %ld %s\n", (long)ftell(fp), argv[1]);

fflush(scp);

// Send file contents

while ((c = fgetc(fp)) != EOF) {

fputc(c, scp);

}

// Send end-of-file marker

fprintf(scp, "\x00");

fflush(scp);

// Wait for ACK message

while (fgets(response, sizeof(response), scp) != NULL) {

if (strcmp(response, "ACK\n") == 0) {

break;

}

}

fclose(scp);

fclose(fp);

return 0;

}

The program then waits for the server's response using fgets() and checks if the response starts with "C0644", which indicates that the server is ready to receive the file.

Once the server is ready, the program sends the file name and size using fprintf() and the file contents using fputc(). Finally, the program sends an end-of-file marker and waits for an "ACK" message from the server before closing the connection.

In SCP, the "C0644" response code indicates that the server is ready to receive a file with the specified file permissions. Specifically, "C" indicates that the following message is a control message, "0644" is the file permissions (in octal notation), and the rest of the message contains the file name and size.

When the client receives a "C0644" response from the server, it knows that it can send the file contents to the server.

In SCP, the "C0644" response code indicates that the server is ready to receive a file with the specified file permissions. Specifically, "C" indicates that the following message is a control message, "0644" is the file permissions (in octal notation), and the rest of the message contains the file name and size.

When the client receives a "C0644" response from the server, it knows that it can send the file contents to the server.

In SCP file transfer, "C0644" is a file mode that indicates the permissions of the file being transferred. The "C" indicates that the file is a regular file, and "0644" indicates the file permissions in octal notation. In this case, "0644" means that the file has read and write permissions for the owner and read-only permissions for everyone else.