#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <sys/stat.h>

#include <libgen.h>

#include <libssh/libssh.h>

#include <libssh/sftp.h>

int scp\_send\_file(ssh\_session session, const char \*local\_path, const char \*remote\_path)

{

struct stat file\_info;

int access\_type, mode;

char \*filename, \*base;

sftp\_session sftp;

sftp\_file file;

ssh\_scp scp;

int rc;

/\* Get file info and open local file for reading \*/

if (stat(local\_path, &file\_info) != 0) {

fprintf(stderr, "Error getting file info: %s\n", strerror(errno));

return -1;

}

FILE \*local\_file = fopen(local\_path, "rb");

if (!local\_file) {

fprintf(stderr, "Error opening local file: %s\n", strerror(errno));

return -1;

}

/\* Connect to remote host and open SFTP session \*/

sftp = sftp\_new(session);

if (!sftp) {

fprintf(stderr, "Error creating SFTP session: %s\n", ssh\_get\_error(session));

fclose(local\_file);

return -1;

}

rc = sftp\_init(sftp);

if (rc != SSH\_OK) {

fprintf(stderr, "Error initializing SFTP session: %s\n", ssh\_get\_error(session));

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

/\* Create remote file and set permissions \*/

filename = strdup(remote\_path);

base = basename(filename);

file = sftp\_open(sftp, remote\_path, O\_CREAT | O\_TRUNC | O\_WRONLY, S\_IRWXU);

if (!file) {

fprintf(stderr, "Error creating remote file: %s\n", ssh\_get\_error(session));

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

access\_type = sftp\_stat(file, &file\_info);

mode = access\_type & 07777;

rc = sftp\_chmod(file, mode);

if (rc != SSH\_OK) {

fprintf(stderr, "Error setting permissions on remote file: %s\n", ssh\_get\_error(session));

sftp\_close(file);

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

/\* Copy local file to remote host \*/

scp = ssh\_scp\_new(session, SSH\_SCP\_WRITE | SSH\_SCP\_RECURSIVE, remote\_path);

if (!scp) {

fprintf(stderr, "Error creating SCP session: %s\n", ssh\_get\_error(session));

sftp\_close(file);

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

rc = ssh\_scp\_init(scp);

if (rc != SSH\_OK) {

fprintf(stderr, "Error initializing SCP session: %s\n", ssh\_get\_error(session));

ssh\_scp\_free(scp);

sftp\_close(file);

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

rc = ssh\_scp\_push\_file(scp, base, file\_info.st\_size, file\_info.st\_mode & 0777);

if (rc != SSH\_OK) {

fprintf(stderr, "Error pushing file to remote host: %s\n", ssh\_get\_error(session));

ssh\_scp\_free(scp);

sftp\_close(file);

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

ssh\_scp\_write(scp, local\_file, file\_info.st\_size);

ssh\_scp\_close(scp);

ssh\_scp\_free(scp);

sftp\_close(file);

sftp\_free(sftp);

fclose(local\_file);

return 0;

}

int main(int argc, char \*\*argv)

{

const char \*local\_path, \*remote\_path, \*hostname, \*username, \*password;

ssh\_session session;

int port = 22;

/\* Parse command line arguments \*/

if (argc < 5) {

fprintf(stderr, "Usage: %s local\_path remote\_path username hostname [port] [password]\n", argv[0]);

return 1;

}

local\_path = argv[1];

remote\_path = argv[2];

username = argv[3];

hostname = argv[4];

if (argc >= 6) {

port = atoi(argv[5]);

}

if (argc >= 7) {

password = argv[6];

}

/\* Initialize SSH session \*/

session = ssh\_new();

if (!session) {

fprintf(stderr, "Error creating SSH session\n");

return 1;

}

ssh\_options\_set(session, SSH\_OPTIONS\_HOST, hostname);

ssh\_options\_set(session, SSH\_OPTIONS\_PORT, &port);

ssh\_options\_set(session, SSH\_OPTIONS\_USER, username);

/\* Authenticate using password or key \*/

if (password) {

ssh\_options\_set(session, SSH\_OPTIONS\_PASSWORD, password);

} else {

// add code to authenticate with key

}

/\* Connect to remote host \*/

int rc = ssh\_connect(session);

if (rc != SSH\_OK) {

fprintf(stderr, "Error connecting to %s: %s\n", hostname, ssh\_get\_error(session));

ssh\_free(session);

return 1;

}

/\* Send file to remote host \*/

rc = scp\_send\_file(session, local\_path, remote\_path);

if (rc != 0) {

fprintf(stderr, "Error sending file to remote host\n");

ssh\_disconnect(session);

ssh\_free(session);

return 1;

}

/\* Cleanup and exit \*/

ssh\_disconnect(session);

ssh\_free(session);

return 0;

}

This code takes the path of a local file, the path where it should be copied to on a remote host, the username and hostname of the remote host, and optionally a port number and password. It initializes an SSH session, authenticates with the remote host, opens an SFTP session, creates the remote file with the appropriate permissions, and copies the local file to the remote host using SCP. The code returns 0 on success and -1 on failure.

./scp\_example /path/to/local/file.txt /remote/path/file.txt username remote.host.com 22 mypassword

argc: 7

argv[0]: ./scp\_example

argv[1]: /path/to/local/file

argv[2]: /remote/path

argv[3]: username

argv[4]: example.com

argv[5]: 22

argv[6]: password

````````````````````````````````````````````````````````````````````````````````````````````

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <errno.h>

#include <libssh/libssh.h>

#include <libssh/sftp.h>

/\* Send a local file to a remote host using SCP \*/

int scp\_send\_file(ssh\_session session, const char \*local\_path, const char \*remote\_path) {

/\* Open local file \*/

FILE \*local\_file = fopen(local\_path, "r");

if (!local\_file) {

fprintf(stderr, "Error opening local file '%s': %s\n", local\_path, strerror(errno));

return -1;

}

/\* Get file info \*/

struct stat file\_info;

if (fstat(fileno(local\_file), &file\_info) != 0) {

fprintf(stderr, "Error getting file info for '%s': %s\n", local\_path, strerror(errno));

fclose(local\_file);

return -1;

}

/\* Open SFTP session \*/

sftp\_session sftp = sftp\_new(session);

if (!sftp) {

fprintf(stderr, "Error creating SFTP session: %s\n", ssh\_get\_error(session));

fclose(local\_file);

return -1;

}

if (sftp\_init(sftp) != SSH\_OK) {

fprintf(stderr, "Error initializing SFTP session: %s\n", ssh\_get\_error(session));

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

/\* Create remote file \*/

sftp\_file file = sftp\_open(sftp, remote\_path, O\_WRONLY | O\_CREAT | O\_TRUNC, S\_IRUSR | S\_IWUSR);

if (!file) {

fprintf(stderr, "Error creating remote file '%s': %s\n", remote\_path, ssh\_get\_error(session));

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

/\* Send file \*/

ssh\_scp scp = ssh\_scp\_new(session, SSH\_SCP\_WRITE | SSH\_SCP\_RECURSIVE, remote\_path);

if (!scp) {

fprintf(stderr, "Error creating SCP session: %s\n", ssh\_get\_error(session));

sftp\_close(file);

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

if (ssh\_scp\_init(scp) != SSH\_OK) {

fprintf(stderr, "Error initializing SCP session: %s\n", ssh\_get\_error(session));

ssh\_scp\_free(scp);

sftp\_close(file);

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

if (ssh\_scp\_push\_file(scp, local\_path, file\_info.st\_size, S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH) != SSH\_OK) {

fprintf(stderr, "Error sending file '%s' to remote host: %s\n", local\_path, ssh\_get\_error(session));

ssh\_scp\_close(scp);

ssh\_scp\_free(scp);

sftp\_close(file);

sftp\_free(sftp);

fclose(local\_file);

return -1;

}

/\* Cleanup \*/

ssh\_scp\_close(scp);

ssh\_scp\_free(scp);

sftp\_close(file);

sftp\_free(sftp);

fclose(local\_file);

return 0;

}

int main(int argc, char \*\*argv)

{

const char \*local\_path, \*remote\_path, \*hostname, \*username, \*password;

ssh\_session session;

int port = 22;

/\* Parse command line arguments \*/

if(argc < 6 || argc > 7) {

fprintf(stderr, "Usage: %s local\_file remote\_path username hostname [port] [password]\n", argv[0]);

return 1;

}

local\_path = argv[1];

remote\_path = argv[2];

username = argv[3];

hostname = argv[4];

if (argc == 6) {

port = atoi(argv[5]);

} else {

password = argv[6];

}

/\* Initialize SSH library \*/

ssh\_init();

/\* Create SSH session \*/

session = ssh\_new();

if (!session) {

fprintf(stderr, "Error creating SSH session: %s\n", strerror(errno));

return 1;

}

/\* Set SSH options \*/

ssh\_options\_set(session, SSH\_OPTIONS\_HOST, hostname);

ssh\_options\_set(session, SSH\_OPTIONS\_USER, username);

ssh\_options\_set(session, SSH\_OPTIONS\_PORT, &port);

/\* Connect to remote host \*/

if (ssh\_connect(session) != SSH\_OK) {

fprintf(stderr, "Error connecting to remote host: %s\n", ssh\_get\_error(session));

ssh\_free(session);

return 1;

}

/\* Authenticate \*/

if (argc == 6) {

if (ssh\_userauth\_publickey\_auto(session, NULL, NULL) != SSH\_AUTH\_SUCCESS) {

fprintf(stderr, "Error authenticating with public key: %s\n", ssh\_get\_error(session));

ssh\_disconnect(session);

ssh\_free(session);

return 1;

}

} else {

if (ssh\_userauth\_password(session, NULL, password) != SSH\_AUTH\_SUCCESS) {

fprintf(stderr, "Error authenticating with password: %s\n", ssh\_get\_error(session));

ssh\_disconnect(session);

ssh\_free(session);

return 1;

}

}

/\* Send file \*/

if (scp\_send\_file(session, local\_path, remote\_path) != 0) {

fprintf(stderr, "Error sending file '%s' to remote host: %s\n", local\_path, ssh\_get\_error(session));

ssh\_disconnect(session);

ssh\_free(session);

return 1;

}

/\* Disconnect from remote host \*/

ssh\_disconnect(session);

/\* Free SSH session \*/

ssh\_free(session);

/\* Cleanup \*/

ssh\_finalize();

return 0;

}

This code can be compiled and executed in a terminal with the sample arguments as shown above.

argc: 7

argv[0]: ./scp\_example

argv[1]: /path/to/local/file

argv[2]: /remote/path

argv[3]: username

argv[4]: example.com

argv[5]: 22

argv[6]: password

`````````````````````````1````````````````````````````````````````````````````````````````````````````````````````````````````````````````