#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// Define a struct to hold information about each file transfer

typedef struct {

char\* file\_name;

char\* status;

int progress;

time\_t start\_time;

time\_t end\_time;

} file\_transfer;

// Define a function to update the log file with the current status of a file transfer

void update\_log\_file(file\_transfer transfer) {

// Open the log file for writing

FILE\* log\_file = fopen("transfer\_log.txt", "a");

// Write the transfer information to the log file

fprintf(log\_file, "%s\t%s\t%d%%\t%ld\t%ld\n", transfer.file\_name, transfer.status, transfer.progress, transfer.start\_time, transfer.end\_time);

// Close the log file

fclose(log\_file);

}

int main() {

// Example usage

file\_transfer transfer;

transfer.file\_name = "example\_file.txt";

transfer.status = "transferring";

transfer.progress = 50;

transfer.start\_time = time(NULL);

// Update the log file with the current status of the file transfer

update\_log\_file(transfer);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#define LOG\_FILE "/var/log/scp.log"

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

if (argc != 3) {

fprintf(stderr, "Usage: %s <source\_file> <destination\_file>\n", argv[0]);

exit(EXIT\_FAILURE);

}

char \*src\_file = argv[1];

char \*dst\_file = argv[2];

// Open log file for writing

FILE \*log\_fp = fopen(LOG\_FILE, "a");

if (log\_fp == NULL) {

fprintf(stderr, "Failed to open log file %s\n", LOG\_FILE);

exit(EXIT\_FAILURE);

}

// Log start time and file information

time\_t start\_time = time(NULL);

fprintf(log\_fp, "Start time: %s", ctime(&start\_time));

fprintf(log\_fp, "File: %s -> %s\n", src\_file, dst\_file);

// Create child process to handle SCP file transfer

pid\_t pid = fork();

if (pid == -1) {

fprintf(stderr, "Failed to create child process\n");

exit(EXIT\_FAILURE);

} else if (pid == 0) {

// Child process: execute SCP command

execlp("scp", "scp", src\_file, dst\_file, NULL);

fprintf(stderr, "Failed to execute SCP command\n");

exit(EXIT\_FAILURE);

} else {

// Parent process: wait for child process to complete

int status;

waitpid(pid, &status, 0);

// Log end time and file status

time\_t end\_time = time(NULL);

double elapsed\_time = difftime(end\_time, start\_time);

if (WIFEXITED(status)) {

int exit\_status = WEXITSTATUS(status);

if (exit\_status == 0) {

fprintf(log\_fp, "Status: completed\n");

fprintf(log\_fp, "Elapsed time: %.2f seconds\n", elapsed\_time);

} else {

fprintf(log\_fp, "Status: failed (exit code %d)\n", exit\_status);

}

} else {

fprintf(log\_fp, "Status: failed (signal %d)\n", WTERMSIG(status));

}

fprintf(log\_fp, "End time: %s", ctime(&end\_time));

}

// Close log file

fclose(log\_fp);

return 0;

}