Lab exp 3 answers

Ans1

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#define BUF\_SIZE 4096

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

if (argc != 3) {

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

exit(EXIT\_FAILURE);

}

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

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

int source\_fd = open(source\_file, O\_RDONLY);

if (source\_fd == -1) {

perror("Failed to open source file");

exit(EXIT\_FAILURE);

}

off\_t file\_size = lseek(source\_fd, 0, SEEK\_END);

if (file\_size == -1) {

perror("Failed to get file size");

close(source\_fd);

exit(EXIT\_FAILURE);

}

off\_t half\_size = file\_size / 2;

off\_t offset = 0;

int choice;

printf("Enter 1 to copy the first half or 2 to copy the second half: ");

scanf("%d", &choice);

if (choice == 1) {

// Copy the first half

offset = 0;

} else if (choice == 2) {

// Copy the second half

offset = half\_size;

} else {

fprintf(stderr, "Invalid choice\n");

close(source\_fd);

exit(EXIT\_FAILURE);

}

// Create destination file

int dest\_fd = open(dest\_file, O\_WRONLY | O\_CREAT | O\_TRUNC, 0666);

if (dest\_fd == -1) {

perror("Failed to create destination file");

close(source\_fd);

exit(EXIT\_FAILURE);

}

// Set the offset in source file

if (lseek(source\_fd, offset, SEEK\_SET) == -1) {

perror("Failed to set offset in source file");

close(source\_fd);

close(dest\_fd);

exit(EXIT\_FAILURE);

}

// Copy data from source to destination

char buffer[BUF\_SIZE];

ssize\_t bytes\_read, bytes\_written;

while ((bytes\_read = read(source\_fd, buffer, sizeof(buffer))) > 0) {

bytes\_written = write(dest\_fd, buffer, bytes\_read);

if (bytes\_written != bytes\_read) {

perror("Write error");

close(source\_fd);

close(dest\_fd);

exit(EXIT\_FAILURE);

}

}

if (bytes\_read == -1) {

perror("Read error");

close(source\_fd);

close(dest\_fd);

exit(EXIT\_FAILURE);

}

printf("File copied successfully.\n");

// Close file descriptors

close(source\_fd);

close(dest\_fd);

return 0;

}

Ans2

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#define BUF\_SIZE 4096

int main() {

char buffer[BUF\_SIZE];

int input\_fd;

ssize\_t bytes\_read;

// Open or create input.txt for writing

input\_fd = open("input.txt", O\_WRONLY | O\_CREAT | O\_TRUNC, 0666);

if (input\_fd == -1) {

perror("Failed to open input.txt for writing");

exit(EXIT\_FAILURE);

}

printf("Type your input. Type '$' to end.\n");

// Read from console until '$' is entered

while ((bytes\_read = read(STDIN\_FILENO, buffer, sizeof(buffer))) > 0) {

// Write to input.txt

if (write(input\_fd, buffer, bytes\_read) != bytes\_read) {

perror("Write error");

close(input\_fd);

exit(EXIT\_FAILURE);

}

// Check if '$' is entered

if (buffer[0] == '$') {

break;

}

}

if (bytes\_read == -1) {

perror("Read error");

close(input\_fd);

exit(EXIT\_FAILURE);

}

printf("Input saved to input.txt.\n");

// Close input file

close(input\_fd);

return 0;

}

Ans. 3

#include <stdio.h>

#define SHIFT\_AMOUNT 3 // You can change this to any desired value

int main() {

FILE \*input\_file, \*output\_file;

char input\_filename[] = "input.txt";

char output\_filename[] = "encrypted.txt";

char ch;

// Open input file for reading

input\_file = fopen(input\_filename, "r");

if (input\_file == NULL) {

perror("Error opening input file");

return 1;

}

// Open output file for writing

output\_file = fopen(output\_filename, "w");

if (output\_file == NULL) {

perror("Error opening output file");

fclose(input\_file);

return 1;

}

// Encrypt and write content to the output file

while ((ch = fgetc(input\_file)) != EOF) {

// Check if character is a letter

if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')) {

// Encrypt uppercase letters

if (ch >= 'A' && ch <= 'Z') {

ch = 'A' + (ch - 'A' + SHIFT\_AMOUNT) % 26;

}

// Encrypt lowercase letters

else if (ch >= 'a' && ch <= 'z') {

ch = 'a' + (ch - 'a' + SHIFT\_AMOUNT) % 26;

}

}

// Write encrypted character to the output file

fputc(ch, output\_file);

}

// Close files

fclose(input\_file);

fclose(output\_file);

printf("File encrypted successfully. Encrypted content saved to encrypted.txt.\n");

return 0;

}

Lab exp 4 answers

Ans.1

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <sys/types.h>

int main() {

char dirname[100];

// Prompt user to enter directory name

printf("Enter directory name: ");

scanf("%s", dirname);

// Create directory using mkdir system call

if (mkdir(dirname, 0777) == -1) {

perror("Error creating directory");

exit(EXIT\_FAILURE);

}

printf("Directory '%s' created successfully.\n", dirname);

return 0;

}

Ans.2

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

int main() {

DIR \*dir;

struct dirent \*entry;

// Open current directory

dir = opendir(".");

if (dir == NULL) {

perror("Error opening directory");

exit(EXIT\_FAILURE);

}

// Read directory entries and display them

printf("Contents of current directory:\n");

while ((entry = readdir(dir)) != NULL) {

printf("%s\n", entry->d\_name);

}

// Close directory

closedir(dir);

return 0;

}

Ans.3

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

char dirname[100];

// Prompt user to enter directory name

printf("Enter directory name to delete: ");

scanf("%s", dirname);

// Delete directory using rmdir system call

if (rmdir(dirname) == -1) {

perror("Error deleting directory");

exit(EXIT\_FAILURE);

}

printf("Directory '%s' deleted successfully.\n", dirname);

return 0;

}

Ans. 4

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define MAX\_PATH\_LEN 4096

int main() {

char cwd[MAX\_PATH\_LEN];

// Get current working directory using getcwd system call

if (getcwd(cwd, sizeof(cwd)) == NULL) {

perror("Error getting current working directory");

exit(EXIT\_FAILURE);

}

// Print current working directory

printf("Current working directory: %s\n", cwd);

return 0;

}