**Example 1:**

#include <iostream>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <csignal>

#include <cstring>

#include <cstdlib>

void handle\_sigusr1(int sig) {

std::cout << "Received SIGUSR1 signal in process " << getpid() << std::endl;

}

void handle\_sigusr2(int sig) {

std::cout << "Received SIGUSR2 signal in process " << getpid() << std::endl;

}

int main() {

signal(SIGUSR1, handle\_sigusr1);

signal(SIGUSR2, handle\_sigusr2);

int pipe\_fd[2];

if (pipe(pipe\_fd) == -1) {

std::cerr << "Pipe creation failed" << std::endl;

return 1;

}

pid\_t pid = fork();

if (pid < 0) {

std::cerr << "Fork failed" << std::endl;

return 1;

} else if (pid == 0) {

close(pipe\_fd[1]);

char buffer[128];

read(pipe\_fd[0], buffer, sizeof(buffer));

std::cout << "Child process received: " << buffer << std::endl;

sleep(2);

kill(getppid(), SIGUSR1);

close(pipe\_fd[0]);

} else {

close(pipe\_fd[0]);

const char\* message = "Hello from parent";

write(pipe\_fd[1], message, strlen(message) + 1);

pause();

std::cout << "Parent process received signal from child" << std::endl;

wait(NULL);

close(pipe\_fd[1]);

}

return 0;

}



**Example 2:**

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*file;

int num, sum = 0;

// Open the file for reading

file = fopen("input.txt", "r");

if (file == NULL) {

printf("Could not open file input.txt\n");

return 1;

}

// Read integers from the file and calculate the sum

while (fscanf(file, "%d", &num) != EOF) {

sum += num;

}

// Close the file

fclose(file);

// Print the result

printf("The sum of the numbers in input.txt is %d\n", sum);

return 0;

}

**Example 3:**

#!/bin/bash

# Function to list all files in the directory

list\_files() {

echo "Files in directory $1:"

ls -l "$1"

}

# Function to display the total number of files in the directory

count\_files() {

count=$(ls -l "$1" | grep -v ^d | wc -l)

echo "Total number of files in directory $1: $count"

}

# Function to copy a specified file to a new location

copy\_file() {

if [ ! -f "$1" ]; then

echo "Error: File $1 does not exist."

return

fi

cp "$1" "$2"

echo "File $1 copied to $2"

}

# Function to move a specified file to a new location

move\_file() {

if [ ! -f "$1" ]; then

echo "Error: File $1 does not exist."

return

fi

mv "$1" "$2"

echo "File $1 moved to $2"

}

# Function to delete a specified file

delete\_file() {

if [ ! -f "$1" ]; then

echo "Error: File $1 does not exist."

return

fi

rm "$1"

echo "File $1 deleted"

}

# Main script

if [ $# -lt 2 ]; then

echo "Usage: $0 <directory\_path> <command> [file\_path] [destination]"

echo "Commands:"

echo " list - List all files in the directory"

echo " count - Display the total number of files"

echo " copy <file> <dest>- Copy a file to a new location"

echo " move <file> <dest>- Move a file to a new location"

echo " delete <file> - Delete a specified file"

exit 1

fi

DIR\_PATH="$1"

COMMAND="$2"

FILE\_PATH="$3"

DEST\_PATH="$4"

# Check if the provided directory is valid

if [ ! -d "$DIR\_PATH" ]; then

echo "Error: $DIR\_PATH is not a valid directory."

exit 1

fi

# Execute the corresponding function based on the command

case $COMMAND in

list)

list\_files "$DIR\_PATH"

;;

count)

count\_files "$DIR\_PATH"

;;

copy)

copy\_file "$FILE\_PATH" "$DEST\_PATH"

;;

move)

move\_file "$FILE\_PATH" "$DEST\_PATH"

;;

delete)

delete\_file "$FILE\_PATH"

;;

\*)

echo "Invalid command. Use list, count, copy, move, or delete."

;;

Esac



**Example 4:**

#!/bin/bash

# Check if the user provided a directory path as an argument

if [ -z "$1" ]; then

echo "Usage: $0 <directory\_path>"

exit 1

fi

# Store the directory path

DIR\_PATH="$1"

# Check if the provided argument is a valid directory

if [ ! -d "$DIR\_PATH" ]; then

echo "Error: $DIR\_PATH is not a valid directory."

exit 1

fi

# Change permissions of all files within the directory

find "$DIR\_PATH" -type f -exec chmod 755 {} +

# Print a completion message

echo "Permissions changed to rwxr-xr-x for all files in $DIR\_PATH."

**Example 5:**

#!/bin/bash

# Check if the user provided a directory path as an argument

if [ -z "$1" ]; then

echo "Usage: $0 <directory\_path>"

exit 1

fi

# Store the directory path

DIR\_PATH="$1"

# Check if the provided argument is a valid directory

if [ ! -d "$DIR\_PATH" ]; then

echo "Error: $DIR\_PATH is not a valid directory."

exit 1

fi

# Count the number of files in the directory

file\_count=$(find "$DIR\_PATH" -type f | wc -l)

# Count the number of directories in the directory

dir\_count=$(find "$DIR\_PATH" -type d | wc -l)

# Print the counts with appropriate labels

echo "Number of files in $DIR\_PATH: $file\_count"

echo "Number of directories in $DIR\_PATH: $dir\_count"

**Example 6:**

#!/bin/bash

# Check if the user provided the required arguments

if [ $# -ne 3 ]; then

echo "Usage: $0 <directory\_path> <search\_string> <replacement\_string>"

exit 1

fi

# Store the arguments in variables

DIR\_PATH="$1"

SEARCH\_STRING="$2"

REPLACEMENT\_STRING="$3"

# Check if the provided argument is a valid directory

if [ ! -d "$DIR\_PATH" ]; then

echo "Error: $DIR\_PATH is not a valid directory."

exit 1

fi

# Find and replace the search string with the replacement string in all files

find "$DIR\_PATH" -type f -exec sed -i "s/$SEARCH\_STRING/$REPLACEMENT\_STRING/g" {} +

# Print a completion message

echo "Find and replace operation completed in all files within $DIR\_PATH."

**Example 7:**

#!/bin/bash

# Check if the user provided a directory path as an argument

if [ -z "$1" ]; then

echo "Usage: $0 <directory\_path>"

exit 1

fi

# Store the directory path

DIR\_PATH="$1"

# Check if the provided argument is a valid directory

if [ ! -d "$DIR\_PATH" ]; then

echo "Error: $DIR\_PATH is not a valid directory."

exit 1

fi

# Generate a disk usage report for the directory

REPORT\_FILE="disk\_usage\_report.txt"

du -sh "$DIR\_PATH" > "$REPORT\_FILE"

du -ah "$DIR\_PATH" >> "$REPORT\_FILE"

# Print a message indicating where the report is saved

echo "Disk usage report saved to $REPORT\_FILE."

**Example 8:**

#include <iostream>

#include <fcntl.h>

#include <unistd.h>

#include <cstring>

int main() {

const char\* filename = "example.txt";

const char\* message = "Hello, World!";

char buffer[100];

// Create or truncate the file using open system call

int fd = open(filename, O\_CREAT | O\_WRONLY | O\_TRUNC, S\_IRUSR | S\_IWUSR);

if (fd == -1) {

std::cerr << "Error: Failed to create or open the file." << std::endl;

return 1;

}

// Write the string to the file using write system call

ssize\_t bytes\_written = write(fd, message, strlen(message));

if (bytes\_written == -1) {

std::cerr << "Error: Failed to write to the file." << std::endl;

close(fd);

return 1;

}

// Ensure that all bytes are written

if (bytes\_written != strlen(message)) {

std::cerr << "Error: Not all bytes were written to the file." << std::endl;

close(fd);

return 1;

}

// Reset the file pointer to the beginning using lseek system call

if (lseek(fd, 0, SEEK\_SET) == -1) {

std::cerr << "Error: Failed to reset the file pointer." << std::endl;

close(fd);

return 1;

}

// Read the contents of the file using read system call

ssize\_t bytes\_read = read(fd, buffer, sizeof(buffer) - 1);

if (bytes\_read == -1) {

std::cerr << "Error: Failed to read from the file." << std::endl;

close(fd);

return 1;

}

// Null-terminate the buffer and print the contents

buffer[bytes\_read] = '\0';

std::cout << "File contents: " << buffer << std::endl;

// Close the file descriptor using close system call

if (close(fd) == -1) {

std::cerr << "Error: Failed to close the file." << std::endl;

return 1;

}

// Delete the file using unlink system call

if (unlink(filename) == -1) {

std::cerr << "Error: Failed to delete the file." << std::endl;

return 1;

}

std::cout << "File operations completed successfully." << std::endl;

return 0;

}