```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include "file_operations.h"

int create_and_write_file(const char *filename, const char *content) {
    // Declare an integer 'fd' for the file descriptor.
    int fd;
    // Declare a variable 'bytes_written' of type ssize_t to store how many bytes are
written.
    ssize_t bytes_written;

    // Print a message showing which file is being created.
    printf("Creating file: %s\n", filename);
    // Print a message showing what content will be written.
    printf("Writing content: %s\n", content);

    // Open or create the file for writing using the open() system call.
    // Use flags O_CREAT I O_WRONLY I O_TRUNC and permissions 0644.
    fd = open(filename, O_CREAT I O_WRONLY I O_TRUNC, 0644);
    // Check if open() failed (fd == -1).
    if (fd == -1) {
        perror("Error opening file");
        return -1;
    }

    // Print the file descriptor value.
    printf("File descriptor: %d\n", fd);

    // Write the content to the file using the write() system call.
    bytes_written = write(fd, content, strlen(content));
    // Check if write() failed (bytes_written == -1).
    if (bytes_written == -1) {
        perror("Error writing to file");
        close(fd);
        return -1;
    }

    // Print a success message with the number of bytes written and the filename.
    printf("Successfully wrote %zd bytes to %s\n", bytes_written, filename);

    // Close the file using close(fd).
    if (close(fd) == -1) {
        perror("Error closing file");
        return -1;
    }

    // Print a message that the file was closed successfully.
    printf("File %s closed successfully.\n", filename);

    return 0;
}

int read_file_contents(const char *filename) {
    // Declare an integer 'fd' for the file descriptor.
    int fd;
    // Create a buffer array of size 1024 to store the file data.
    char buffer[1024];
    // Declare a variable 'bytes_read' of type ssize_t to store how many bytes are read.
    ssize_t bytes_read;

    // Print a message showing which file is being read.
    printf("Reading file: %s\n", filename);

    // Open the file for reading using the open() system call.
    fd = open(filename, O_RDONLY);
    // Check if open() failed (fd == -1).
    if (fd == -1) {
        perror("Error opening file");
        return -1;
    }

    // Print the file descriptor value.
    printf("File descriptor: %d\n", fd);
    // Print a header for the file contents.
    printf("----- File Contents Start -----\n");

    // Read the file contents using the read() system call in a loop.
    while ((bytes_read = read(fd, buffer, sizeof(buffer) - 1)) > 0) {
        // Null-terminate the buffer after each read.
        buffer[bytes_read] = '\0';
        // Print the contents of the buffer.
        printf("%s", buffer);
    }

    // Check if read() failed (bytes_read == -1).
    if (bytes_read == -1) {
        perror("Error reading file");
        close(fd);
        return -1;
    }
    // Print a footer for the end of the file.
    printf("\n----- File Contents End -----\n");

    // Close the file using close(fd).
    if (close(fd) == -1) {
        perror("Error closing file");
        return -1;
    }

    // Print a message that the file was closed successfully.
    printf("File %s closed successfully.\n", filename);

    return 0;
}
```