Comprehensive Guide to I/O Operations in Linux

[Previous content remains unchanged...]

7. Practice Exercises

Exercise 1: Write to and Read from a File

Create a program that opens a file, writes some text to it, closes the file, then opens it again and reads the content.

```c
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int main() {
    const char *filename = "example.txt";
    const char *text = "Hello, this is a test file.\n";
    char buffer[100];
    int fd, bytes_read;

    // Open file for writing
    fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd == -1) {
        perror("Error opening file for writing");
        return 1;
    }

    // Write to file
    if (write(fd, text, strlen(text)) == -1) {
        perror("Error writing to file");
        return 1;
    }

    // Close file
    close(fd);

    // Open file for reading
    fd = open(filename, O_RDONLY);
    if (fd == -1) {
        perror("Error opening file for reading");
        return 1;
    }

    // Read from file
    bytes_read = read(fd, buffer, sizeof(buffer) - 1);
```

```
    if (bytes_read == -1) {
        perror("Error reading from file");
        return 1;
    }

    // Null-terminate the string and print
    buffer[bytes_read] = '\0';
    printf("File contents: %s", buffer);

    // Close file
    close(fd);

    return 0;
}
```

To compile and run:

```
gcc exercise1.c -o exercise1
./exercise1
```

Exercise 2: Timestamp and User Input

Modify the timestamp writing program to allow the user to input a message, then write both the message and the timestamp to the file.

```
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <time.h>
#include <unistd.h>

#define MAX_INPUT 1000

char* get_timeStamp() {
    time_t now = time(NULL);
    return asctime(localtime(&now));
}

int main() {
    char *filename = "user_message.txt";
    char user_input[MAX_INPUT];
    char *timeStamp = get_timeStamp();

    // Get user input
    printf("Enter your message: ");
```

```
    fgets(user_input, MAX_INPUT, stdin);

    // Open file
    int fd = open(filename, O_WRONLY | O_APPEND | O_CREAT, 0666);
    if (fd == -1) {
        perror("Error opening file");
        return 1;
    }

    // Write user input and timestamp
    if (write(fd, user_input, strlen(user_input)) == -1 ||
        write(fd, timeStamp, strlen(timeStamp)) == -1) {
        perror("Error writing to file");
        close(fd);
        return 1;
    }

    printf("Message and timestamp written to %s\n", filename);

    close(fd);
    return 0;
}
```

To compile and run:

```
gcc exercise2.c -o exercise2
./exercise2
```

Exercise 3: File Copy Program

Create a program that copies the contents of one file to another using `read()` and `write()` system calls.

```
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>

#define BUFFER_SIZE 4096

int main(int argc, char *argv[]) {
    int fd_source, fd_dest, bytes_read, bytes_written;
    char buffer[BUFFER_SIZE];

    if (argc != 3) {
        fprintf(stderr, "Usage: %s <source_file> <destination_file>\n", argv[0]);
        return 1;
    }
```

```c
    // Open source file for reading
    fd_source = open(argv[1], O_RDONLY);
    if (fd_source == -1) {
        perror("Error opening source file");
        return 1;
    }

    // Open destination file for writing (create if not exists, truncate if exists)
    fd_dest = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd_dest == -1) {
        perror("Error opening destination file");
        close(fd_source);
        return 1;
    }

    // Copy content
    while ((bytes_read = read(fd_source, buffer, BUFFER_SIZE)) > 0) {
        bytes_written = write(fd_dest, buffer, bytes_read);
        if (bytes_written != bytes_read) {
            perror("Write error");
            close(fd_source);
            close(fd_dest);
            return 1;
        }
    }

    if (bytes_read == -1) {
        perror("Read error");
        close(fd_source);
        close(fd_dest);
        return 1;
    }

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

    close(fd_source);
    close(fd_dest);
    return 0;
}
```

To compile and run:

```
gcc exercise3.c -o exercise3
./exercise3 source_file.txt destination_file.txt
```

These exercises provide practical experience with file I/O operations in C, in-

cluding opening, reading, writing, and closing files, as well as working with user input and command-line arguments.

Remember to compile your C programs using gcc and run them with any necessary command-line arguments. Always check for errors and handle them appropriately in your programs.