**Operating Systems Lab**



**Submitted By:**

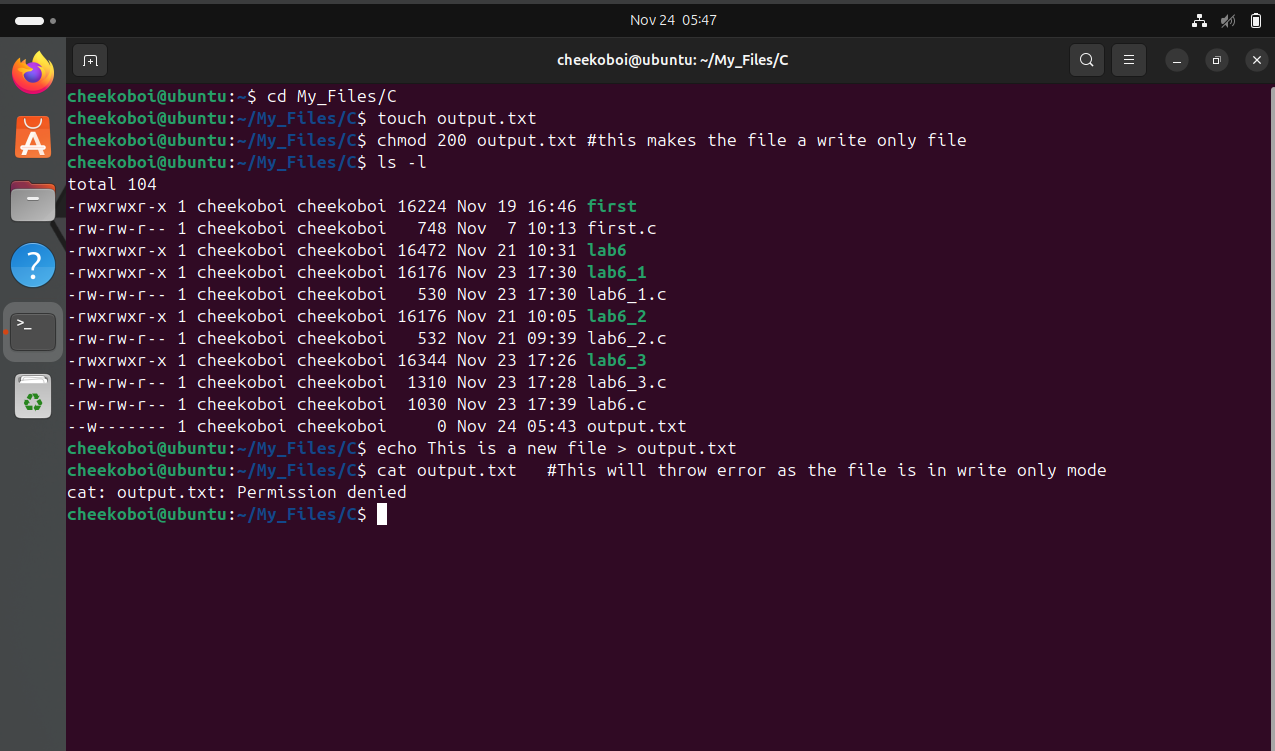
Muhammad Bin Nasir 220201012

**Department of Computer Science**

**Institute of Space Technology**

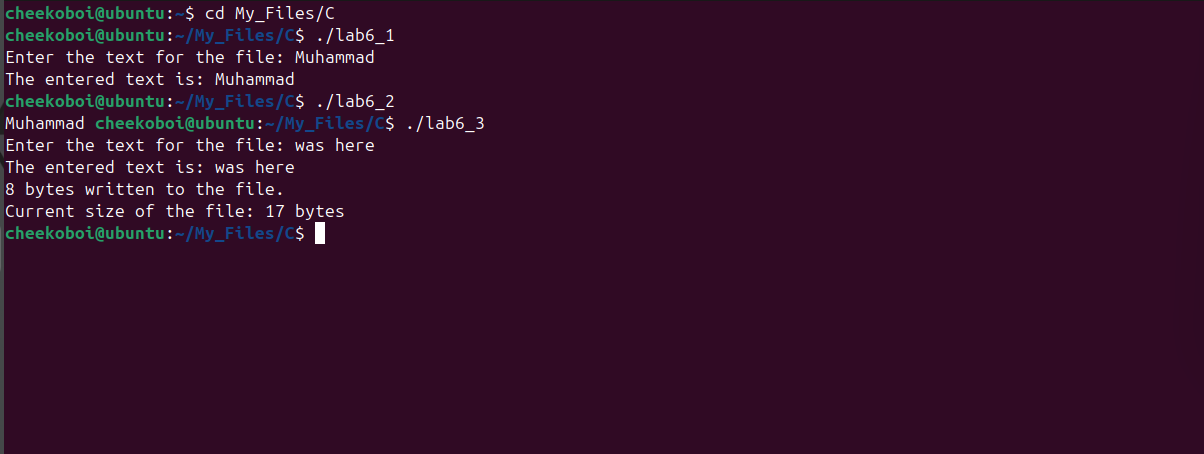
# Method 1:

In the first method, we have created a text file using the touch command. Then using the chmod command, we have given the file a write only permissions for the owner and all users. Then we first write to the file, which is successful. Then we read through it, which throws an error.



# Method 2:

We have written two code files in C for our tasks. The first file creates a file named output.txt, and puts some user given text in it. Second file when executed reads the data from the file. Third File when executed appends another user given string ahead of the existing data.



**Code for Task 1 (lab6\_1):**

#include <fcntl.h>

#include <unistd.h>

#include <sys/stat.h>

#include <stdio.h>

#include <string.h>

int main() {

int fd = open("output.txt", O\_CREAT | O\_WRONLY, S\_IRUSR | S\_IWUSR);

if (fd < 0) {

write(2, "Error creating file\n", 20);

return 1;

}

char \*text;

printf("Enter the text for the file: ");

scanf("%255[^\n]", text);

size\_t text\_length = strlen(text);

printf("The entered text is: %s\n", text);

write(fd, text, text\_length);

close(fd);

return 0;

}

**Code for Task 2 (lab6\_2):**

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

int main() {

int fd = open("output.txt", O\_RDONLY);

if (fd == -1) {

perror("Error opening file");

return 1;

}

char buffer[50];

ssize\_t bytes\_read;

while ((bytes\_read = read(fd, buffer, sizeof(buffer) - 1)) > 0) {

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

printf("%s", buffer);

}

if (bytes\_read == -1) {

perror("Error reading from file");

close(fd);

return 1;

}

return 0;

}

**Code for Task 3 (lab6\_3):**

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main() {

int fd = open("output.txt", O\_APPEND | O\_WRONLY);

if (fd == -1) {

perror("Error opening file");

return 1;

}

// Use calloc to initialize memory to zero

char \*text = (char \*)calloc(256, sizeof(char));

if (text == NULL) {

perror("Memory allocation failed");

close(fd);

return 1;

}

printf("Enter the text for the file: ");

scanf(" %255[^\n]", text);

printf("The entered text is: %s\n", text);

size\_t text\_length = strlen(text);

ssize\_t bytes\_written = write(fd, text, text\_length);

if (bytes\_written == -1) {

perror("Error writing to file");

free(text);

close(fd);

return 1;

}

printf("%zd bytes written to the file.\n", bytes\_written);

off\_t file\_size = lseek(fd, 0, SEEK\_END); // Get the new file size

if (file\_size == -1) {

perror("Error determining file size");

free(text);

close(fd);

return 1;

}

printf("Current size of the file: %lld bytes\n", (long long)file\_size);

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

perror("Error closing file");

free(text);

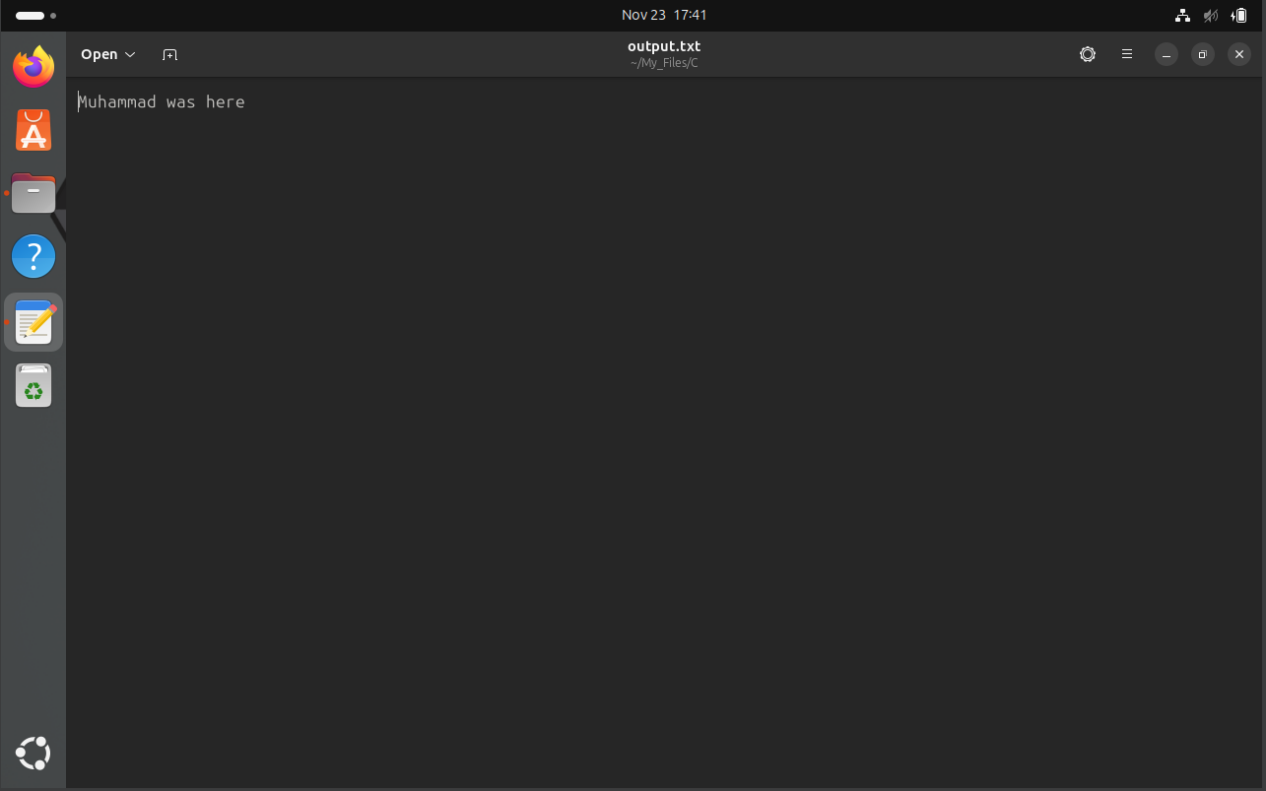
return 1;

}

free(text);

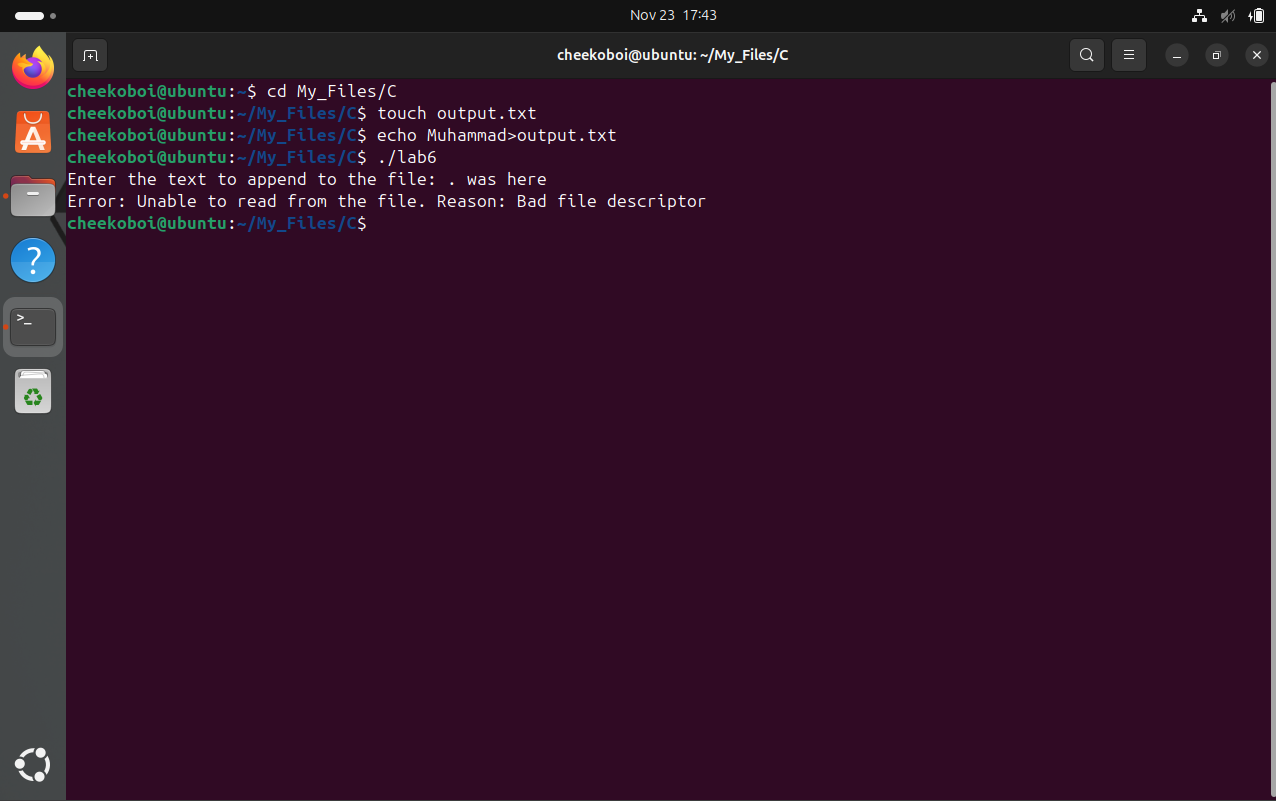
return 0;

}



# Method 2:

In this method, we have manually created the txt file, and added the initial string to it. Then we have created a C file, that opens the file in append mode, and first appends the user given string into the file. Then it tries to read the data fro mthe file, and hence throws an error due to the file being opened in append mode.



**Code for Task (lab6):**

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

int main() {

int fd = open("output.txt", O\_WRONLY | O\_APPEND | O\_CREAT, S\_IRUSR | S\_IWUSR);

if (fd == -1) {

perror("Error opening file in write-only mode");

return 1;

}

char text[256];

printf("Enter the text to append to the file: ");

scanf(" %[^\n]", text);

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

if (bytes\_written == -1) {

perror("Error writing to file");

close(fd);

return 1;

}

write(fd, "\n", 1);

char buffer[256];

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

if (bytes\_read == -1) {

fprintf(stderr, "Error: Unable to read from the file. Reason: %s\n", strerror(errno));

} else {

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

printf("Contents of the file: %s\n", buffer);

}

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

perror("Error closing file");

return 1;

}

return 0;

}