main.c

Share | Run

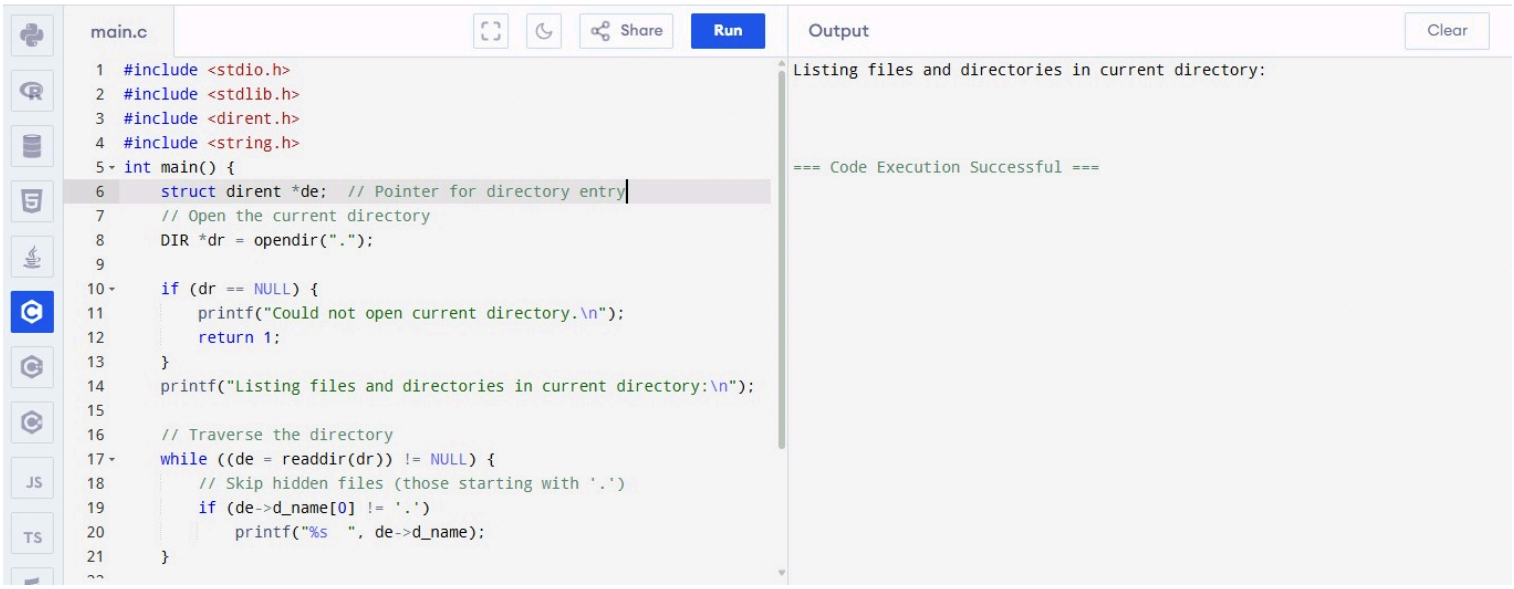
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void createFile(char *filename) {
    FILE *fp = fopen(filename, "w");
    if (fp == NULL) {
        printf("Error creating file.\n");
        return;
    }
    printf("File '%s' created successfully.\n", filename);
    fclose(fp);
}
void writeFile(char *filename) {
    FILE *fp = fopen(filename, "w");
    if (fp == NULL) {
        printf("Error opening file for writing.\n");
        return;
    }
    char data[1000];
    printf("Enter data to write (end with ~ on a new line):\n");
    getchar(); // clear newline
```

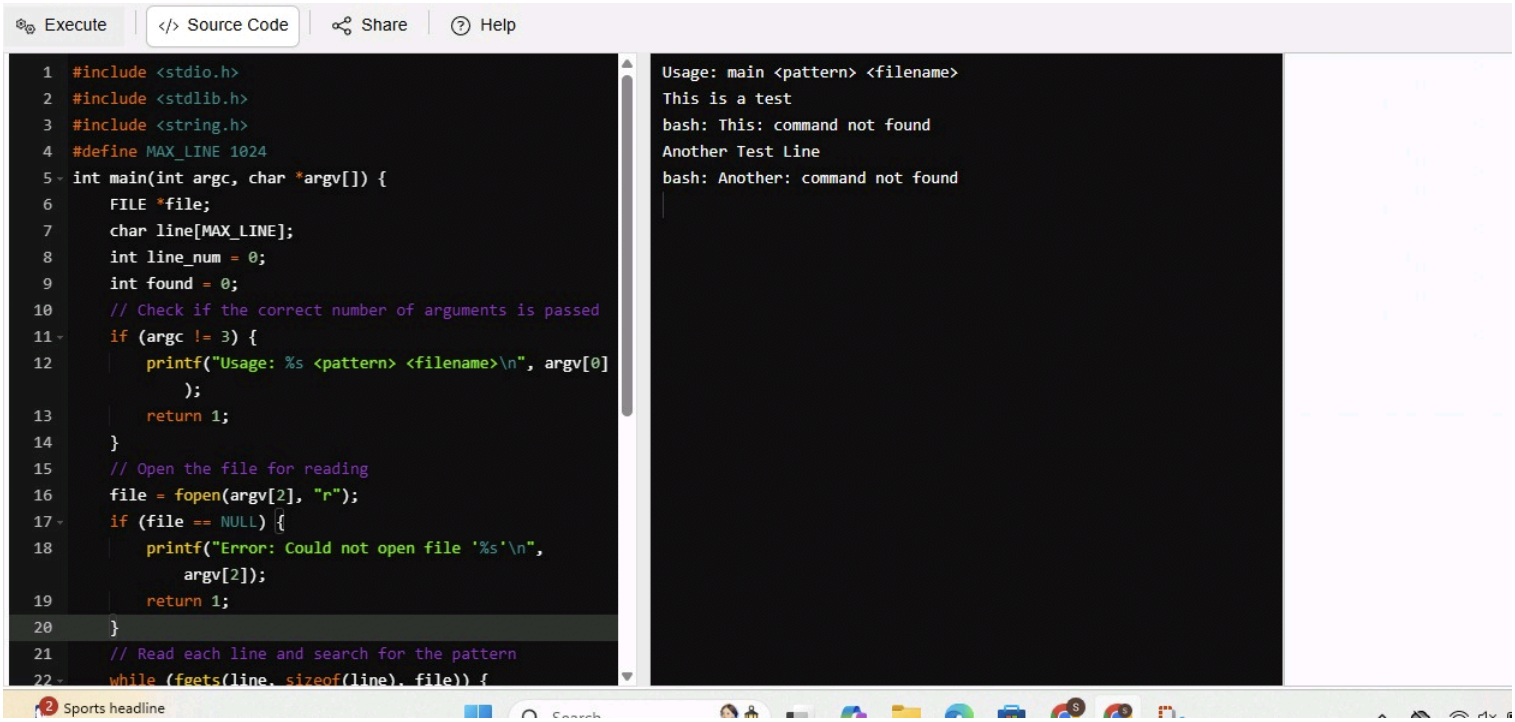
Output | Clear

```
Enter the filename to manage: test.txt

==== File Management Menu ====
1. Create File
2. Write to File
3. Read File
4. Append to File
5. Delete File
6. Exit
Choose an option: 1
Error creating file.

==== File Management Menu ====
1. Create File
2. Write to File
3. Read File
4. Append to File
5. Delete File
6. Exit
Choose an option: 2
Error opening file for writing.
```


main.c
Share
Run
Output
Clear
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
int main() {
struct dirent *de; // Pointer for directory entry
// Open the current directory
DIR *dr = opendir(".");
if (dr == NULL) {
printf("Could not open current directory.\n");
return 1;
}
printf("Listing files and directories in current directory:\n");
// Traverse the directory
while ((de = readdir(dr)) != NULL) {
// Skip hidden files (those starting with '.')
if (de->d_name[0] != '.')
printf("%s ", de->d_name);
}
Listing files and directories in current directory:
=== Code Execution Successful ===


Execute
Source Code
Share
Help
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_LINE 1024
int main(int argc, char *argv[]) {
FILE *file;
char line[MAX_LINE];
int line_num = 0;
int found = 0;
// Check if the correct number of arguments is passed
if (argc != 3) {
printf("Usage: %s <pattern> <filename>\n", argv[0]
);
return 1;
}
// Open the file for reading
file = fopen(argv[2], "r");
if (file == NULL) {
printf("Error: Could not open file '%s'\n",
argv[2]);
return 1;
}
// Read each line and search for the pattern
while (fgets(line, sizeof(line), file)) {
Usage: main <pattern> <filename>
This is a test
bash: This: command not found
Another Test Line
bash: Another: command not found
Sports headline

Execute | Source Code | Share | Help

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

#define MAX_ITEMS 10  // Maximum items in the buffer
#define PRODUCER_SLEEP_TIME 1
#define CONSUMER_SLEEP_TIME 2

// Shared Buffer
int buffer[MAX_ITEMS];
int in = 0;  // Index where producer will insert the next item
int out = 0; // Index where consumer will consume the next item

// Semaphores and mutex
sem_t empty;   // Semaphore to track empty slots in the buffer
sem_t full;    // Semaphore to track full slots in the buffer
pthread_mutex_t mutex;  // Mutex to protect shared buffer access
```

```
Produced: 27 at index 1
Consumed: 86 from index 6
Produced: 90 at index 2
Produced: 59 at index 3
Consumed: 92 from index 7
Produced: 63 at index 4
Produced: 26 at index 5
Consumed: 49 from index 8
Produced: 40 at index 6
Produced: 26 at index 7
Consumed: 21 from index 9
Produced: 72 at index 8
Produced: 36 at index 9
Consumed: 62 from index 0
Produced: 11 at index 0
Consumed: 27 from index 1
Produced: 68 at index 1
Consumed: 90 from index 2
Produced: 67 at index 2
Consumed: 59 from index 3
Produced: 29 at index 3
Consumed: 63 from index 4
Produced: 82 at index 4
```

Execute | Source Code | Share | Help

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_THREADS 2

// Thread function
void* thread_func(void* arg) {
    int id = *(int*)arg;
    printf("Thread %d is starting...\n", id);
    sleep(2);  // Simulate some work being done
    printf("Thread %d is exiting...\n", id);
    pthread_exit(NULL);  // Exit the thread
}
int main() {
    pthread_t threads[NUM_THREADS];
    int thread_ids[NUM_THREADS] = {1, 2};
    // (i) Create threads
    for (int i = 0; i < NUM_THREADS; i++) {
        int result = pthread_create(&threads[i], NULL,
            thread_func, &thread_ids[i]);
        if (result != 0) {
            fprintf(stderr, "Error creating thread %d\n",
```

https://www.tutorialspoint.com/compilers/online-c-compiler.htm#code

```
Thread 1 is starting...
Thread 2 is starting...
Thread 1 is exiting...
Thread 2 is exiting...
Main thread joined with thread 1
Main thread joined with thread 2
Thread 0 and Thread 1 are different threads.
Main thread is exiting...
```