```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <fcntl.h>
#include "utils.h"

/* Return the timestamp at the time of invocation*/
char *current_datetime_str()
{
    time_t tim = time(NULL);
    return duplicate_until_newline(ctime(&tim));
}

/* Open a log file with the given filename and return its file descriptor. */
int open_file(char *filename)
{
    int fptr;
    if ((fptr = open(filename, O_CREAT | O_APPEND | O_WRONLY, 0755)) == -1)
    {
        fprintf(stderr, "Error: failed to open \"%s\"\n", filename);
        perror("open");
        exit(EXIT_FAILURE);
    }
    return fptr;
}

/*Checks if the argument is any of the whitespace characters*/
int is_space(char c)
{
    return (c == ' ' ||
            c == '\t' ||
            c == '\n' ||
            c == '\r' ||
            c == '\x0b' ||
            c == '\x0c');
}

/*Removes the whitespace characters to the left of the argument string*/
char *left_strip(char *s)
{
    int i;

    i = 0;
    while (is_space(s[i]))
        ++i;

    return s + i;
}

/*Return a copy of the string*/
char *duplicate(char *s)
{
    char *copy;

    copy = malloc(sizeof(char) * strlen(s));
    strcpy(copy, s);

    return copy;
}

/*Returns a copy of the string until a newline character is encountered in the process of l
eft to right traversal*/
char *duplicate_until_newline(char *s)
{
    int i, c;
    char *copy;
```

```
    i = -1;
    copy = malloc(sizeof(char) * strlen(s));
    while ((c = s[++i]) != '\0' && c != '\n')
        copy[i] = c;
    copy[i] = '\0';

    return copy;
}

/*Creates an array of strings with NULL as the last element by parsing the user input from
the submit the submit command for the purpose of passing it as an argument to the execvp ca
ll*/
char **create_exec_args(char *line)
{
    char *copy = malloc(sizeof(char) * (strlen(line) + 1));
    strcpy(copy, line);

    char *arg;
    char **args = malloc(sizeof(char *));
    int i = 0;
    while ((arg = strtok(copy, " \t")) != NULL)
    {
        args[i] = malloc(sizeof(char) * (strlen(arg) + 1));
        strcpy(args[i], arg);
        args = realloc(args, sizeof(char *) * (++i + 1));
        copy = NULL;
    }
    args[i] = NULL;
    return args;
}

/*Reads a single line character by character until a newline character is encountered  from
 the standard input*/
int read_line(char *s, int n)
{
    int i, c;
    for (i = 0; i < n - 1 && (c = getchar()) != '\n'; ++i)
    {
        if (c == EOF)
            return -1;
        s[i] = c;
    }
    s[i] = '\0';
    return i;
}
```