Monday, 11 February 2019

CSC320 — Assignment 6

Abdulellah Alnumay — 436102030

Write a small C program that opens a text file O_APPEND:

1- Check if you can read characters from position 10 to 30

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

// 1- Check if you can read characters from position 10 to 30
int main()
{
    int open_file;
    char *file_name = "./text";
    int num_bytes_to_read = 20;

    // checking and storing file descriptor
    if ((open_file = open(file_name, O_APPEND | O_RDONLY)) == -1)
    {
        perror("Open");
        exit(1);
    }

    // buffer for reading from open_file
    char *read_buffer_open_file = (char *)malloc(21 * sizeof(char));

    // findings: off_t doesn't allow for negative values.
    if (lseek(open_file, 9, SEEK_SET) > 0)
    {
        if (read(open_file, read_buffer_open_file, num_bytes_to_read) > 0)
        {
            printf("%s\n", read_buffer_open_file);
            return 0;
        }
        printf("%s\n", "there has been an error reading the file!");
        perror("read");
        return 0;
    }
    printf("%s\n", "there has been an error seeking the file!");
    perror("lseek");
    return 0;
}
```

Input:

```
abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
```

Output:

```
jklmnopqrstuvwxyzabc????
```

2- Check if you can write characters starting from position 15.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

// 2- Check if you can write characters starting from position 15.
int main()
{
    char *file_name = "./text2";
    // storing file descriptor
    int open_file = open(file_name, O_WRONLY);

    // buffer for writing to open_file
    char *write_buffer_open_file = "ABCDEFG";

    if (lseek(open_file, 14, SEEK_SET) > 0)
    {
        if (write(open_file, write_buffer_open_file, 7) >= 0)
        {
            printf("%s\n", "file has been written it.");
            return 0;
        }
        perror("Write");
        exit(1);
    }
    else
    {
        perror("lseek");
        exit(1);
    }
    return 0;
}
```

Input:

```
abcdefghijklmnopqrstuvwxyz
```

Output:

```
abcdefghijklmnABCDEFGvwxyz
```

3- What happens if I use lseek on StdIn

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

// 3- What happens if I use lseek on StdIn
int main()
{
    int stdin_fd = 0;

    // buffer for writing to open_file
    char *read_from_user = (char *)malloc(15 * sizeof(char));

    int stdin_seek = lseek(stdin_fd, 15, SEEK_CUR);
    if (stdin_seek > 0)
    {
        printf("%d\n", stdin_seek);
        printf("%s", "Please enter a string(word): ");
        scanf("%s", read_from_user);
    }
    else
    {
        perror("lseek");
        exit(1);
    }
    printf("%s\n", read_from_user);
    return 0;
}
```

Input:

```
Please enter a string(word): hello
```

Output:

```
hello
```

4- What happens if I use lseek on StdOut

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

// 4- What happens if I use lseek on StdOut
int main()
{
    int stdout_fd = 1;
    char *read_from_user = (char *)malloc(15 * sizeof(char));

    int stdout_seek = lseek(stdout_seek, 10, SEEK_CUR);
    if (stdout_seek > 0)
    {
        printf("%s\n", "this_is_a_long_string");
    }
    else
    {
        perror("lseek");
        exit(1);
    }
    printf("%s\n", read_from_user);
    return 0;
}
```

Input:

```
"this_is_a_long_string"
```

Output:

```
this_is_a_long_string
```

5- Close your StdIn and open a regular file. What happens after lseek.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

// 5- Close your StdIn and open a regular file. What happens after lseek.
int main()
{
    int open_file;
    char *file_name = "./text";
    if (close(0) == -1)
    {
        perror("Close");
        exit(0);
    }
    // checking and storing file descriptor
    if ((open_file = open(file_name, O_RDONLY)) == -1)
    {
        perror("Open");
        exit(1);
    }
    // fd[0] == file_name;
    char *read_from_scan = (char *)malloc(10*sizeof(char));
    // findings: off_t doesn't allow for negative values.
    int seek_file = lseek(open_file, 10, SEEK_CUR);
    if (seek_file > 0)
    {
        scanf("%s", read_from_scan);
    }
    else
    {
        perror("lseek");
        exit(1);
    }
    printf("%s\n", read_from_scan);
    return 0;

}
```

Input:

```
abcdefghijklmnopqrstuvwxyz
```

Output:

```
klmnopqrstuvwxyz
```

6- Do the same for StdOut.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

// 6- Do the same for StdOut.
int main()
{
    int open_file;
    char *file_name = "./text3";
    if (close(1) == -1)
    {
        perror("Close");
        exit(0);
    }
    // checking and storing file descriptor
    if ((open_file = open(file_name, O_APPEND | O_RDWR)) == -1)
    {
        perror("Open");
        exit(1);
    }
    char *read_from_scan = (char *)malloc(10*sizeof(char));
    // findings: off_t doesn't allow for negative values.
    int seek_file = lseek(open_file, 10, SEEK_CUR);
    if (seek_file > 0)
    {
        scanf("%s", read_from_scan);
    }
    else
    {
        perror("lseek");
        exit(1);
    }
    printf("%s\n", read_from_scan);
    return 0;
}
```

Input: `hello` Output: `hello`

Findings:

- Data type off_t in lseek doesn't seek for negative integers, passing a negative integer wouldn't seek backwards. But if header unistd.h is not included, passing negative integer would result in seeking backwards.
- Stdin and stdout are not seek-able files.
- Scanf would read from fd 0 until reading whitespace or eof.