

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем
Кафедра «Прикладная математика и фундаментальная информатика»

Лабораторная работа
по дисциплине «Операционные системы»

Студента	Курпенова Куата Ибраимовича
	<small>фамилия, имя, отчество полностью</small>
Курс	2, группа ФИТ-212
Направление	02.03.02 Прикладная математика и фундаментальная информатика
	<small>код, наименование</small>
Руководитель	ассистент
	<small>должность, ученая степень, звание</small>
	Горшенин А. Ю.
	<small>фамилия, инициалы</small>
Выполнил	
	<small>дата, подпись студента(ки)</small>

Омск 2022

Задание 1

Разработать в Linux программу, которая получает хэндлы стандартного ввода и вывода, выводит числовые с комментариями значения этих хэндлов, затем, используя стандартный ввод системными функциями небуферизованного ввода-вывода `ReadFile` или `WriteFile`, делает приглашение для ввода, вводит любой текст и выводит его с предупреждением, что он предварительно введен в программу, обеспечивая в этой системе вывод приглашения на ввод данных со стандартного ввода только в случае использовании ввода с консоли, при переадресации этого ввода на входной файл приглашение отображаться не должно. При переадресации стандартного вывода в файл отображение приглашения в случае ввода с консоли должно принудительно появляться на экране, а не в файле, на который переадресуется вывод. Числовые значения хэндлов стандартных ввода и вывода в этом варианте выводить не нужно.

Решение

```
NERD_tree_1 main.c
1 #include <stdio.h>
2
3 unsigned int read(int handle, void* buffer, unsigned int len);
4 unsigned int write(int handle, void* buffer, unsigned int len);
5
6 int main() {
7     int cb;
8     char buffer[100] = "[+] ";
9
10    write(1, "[>] Enter text: ", 16); *handle: 1 *buffer: "[>] Enter text: " *len: 16
11
12    cb = read(0, buffer+10, 80); *handle: 0 *buffer: buffer+10 *len: 80
13    cb += 10;
14
15    write(1, buffer, cb); *handle: 1 *len: cb
16
17    return 0;
18 }
```

Рис. 1: Код программы main.c

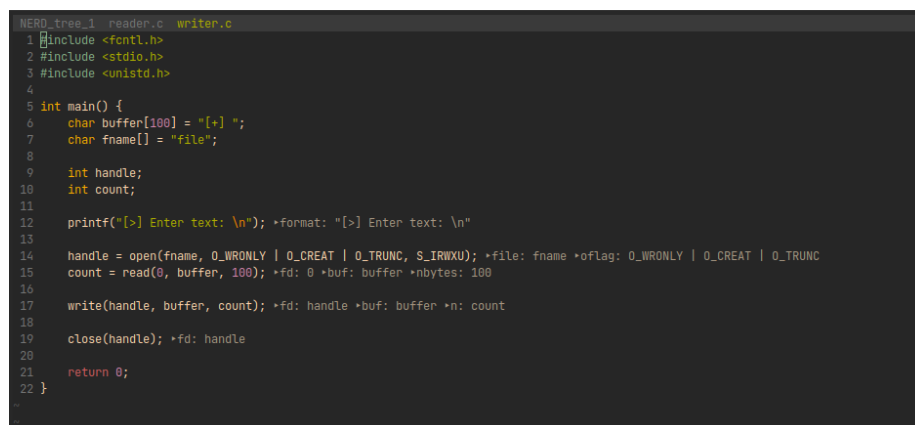
```
kurpenok@kurpenok-computer ~/0/0/3/0/Lab1 (main)> gcc main.c
kurpenok@kurpenok-computer ~/0/0/3/0/Lab1 (main)> ./a.out
[>] Enter text: random text
[+] random text
kurpenok@kurpenok-computer ~/0/0/3/0/Lab1 (main)> ./a.out > out
random text
kurpenok@kurpenok-computer ~/0/0/3/0/Lab1 (main)> cat out
[>] Enter text: [+] random text
kurpenok@kurpenok-computer ~/0/0/3/0/Lab1 (main)> 
```

Рис. 2: Вывод программы main.c

Задание 2

Результат выполнения лабораторной работы должен состоять из двух программ для Linux. Первая программа должна создавать текстовый файл, вводя данные со стандартного ввода. (Более детально: открывает файл для записи, читает текст со стандартного ввода и выводит этот прочитанный текст в файл.) Вторая программа открывает тот же файл (созданный перед этим другой программой) для чтения и хэндл, полученный при этом открытии, запоминает в 1-й переменной для хэндла. Используя этот хэндл, далее с помощью функции `dup()` получается новое значение хэндла для доступа к тому же файлу (2-й хэндл). Еще раз открывается тот же файл, запоминая 3-е значение хэндла. С помощью первого хэндла программа позиционирует чтение для 10-й позиции файла от начала этого файла. Далее программа должна выводить числовые значения всех трех хэндлов на экран. Используя по очереди все 3 хэндла, из файла читаются по 7 символов и тут же эти три прочитанных текста выводятся на экран, каждая в своей строке. Результаты вывода объяснить.

Решение



```
NERD_tree_1 reader.c writer.c
1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 int main() {
6     char buffer[100] = "[*] ";
7     char fname[] = "file";
8
9     int handle;
10    int count;
11
12    printf("[>] Enter text: \n"); *format: "[>] Enter text: \n"
13
14    handle = open(fname, O_WRONLY | O_CREAT | O_TRUNC, S_IRWXU); *file: fname *oflag: O_WRONLY | O_CREAT | O_TRUNC
15    count = read(0, buffer, 100); *fd: 0 *buf: buffer *nbytes: 100
16
17    write(handle, buffer, count); *fd: handle *buf: buffer *n: count
18
19    close(handle); *fd: handle
20
21    return 0;
22 }
```

Рис. 3: Код программы writer.c

```

NERD_tree_1 reader.c writer.c
1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 int main() {
6     char fname[] = "file";
7
8     int handle1;
9     int handle2;
10    int handle3;
11
12    char out[7];
13    int a;
14
15    handle1 = open(fname, O_RDONLY, 0); *file: fname *oflag: O_RDONLY
16    handle2 = dup(handle1); *fd: handle1
17    handle3 = open(fname, O_RDONLY, 0); *file: fname *oflag: O_RDONLY
18    lseek(handle1, 10, SEEK_SET); *fd: handle1 *offset: 10 *whence: SEEK_SET
19
20    printf("[+] First handle: %d\n", handle1); *format: "[+] First handle: %d\n"
21    printf("[+] Dup handle: %d\n", handle2); *format: "[+] Dup handle: %d\n"
22    printf("[+] Second handle: %d\n", handle3); *format: "[+] Second handle: %d\n"
23
24    fflush(stdout); *stream: stdout
25
26    a = read(handle1, out, 7); *fd: handle1 *buf: out *nbytes: 7
27    write(1, out, a); *fd: 1 *buf: out *n: a
28    printf("\n"); *format: "\n"
29
30    a = read(handle2, out, 7); *fd: handle2 *buf: out *nbytes: 7
31    write(1, out, a); *fd: 1 *buf: out *n: a
32    printf("\n"); *format: "\n"
33
34    a = read(handle3, out, 7); *fd: handle3 *buf: out *nbytes: 7
35    write(1, out, a); *fd: 1 *buf: out *n: a
36    printf("\n"); *format: "\n"
37
38    close(handle1); *fd: handle1
39    close(handle2); *fd: handle2
40    close(handle3); *fd: handle3
41
42    return 0;
43 }

```

Рис. 4: Код программы reader.c

```

kurpenok@kurpenok-computer ~/0/0/3/0/Lab2 (main)> gcc reader.c -o reader
kurpenok@kurpenok-computer ~/0/0/3/0/Lab2 (main)> gcc writer.c -o writer
kurpenok@kurpenok-computer ~/0/0/3/0/Lab2 (main)> ./writer
[+] Enter text:
I'm entering some text
kurpenok@kurpenok-computer ~/0/0/3/0/Lab2 (main)> ./reader
[+] First handle: 3
[+] Dup handle: 4
[+] Second handle: 5
ng some
text
I'm ent
kurpenok@kurpenok-computer ~/0/0/3/0/Lab2 (main)> 

```

Рис. 5: Результат работы программ writer.c и reader.c

Задание 7

Разработать программу с тремя дополнительными нитями (threads) относительно главной нити. Каждая из нитей должна использовать общие для всех нитей данные, представленные массивом символов, в которых записаны 20 первых букв латинского алфавита. Каждая из этих нитей на своем k-м шаге выводит со своей случайной задержкой на место «своего» столбца экрана k-ю букву из указанного массива латинских букв, причем с числом повторений, равному условному номеру нити, умноженному на два. Каждая из используемых нитей должен осуществлять вывод своим цветом, отличным от остальных нитей. На 6-м шаге главная нить делает попытку отмены первой из дополнительных нитей, а на 11-м делает попытку отмены третьей из дополнительных нитей. Первая и третья дополнительная нити в начале своей работы запрещают свою отмену. Третья нить на 13 шаге разрешает отмену, но в отложенном режиме. Точку отмены эта нить устанавливает между 16 и 17-м шагом своей работы. Все управляющие указания должны отображаться сообщениями без прокрутки экрана (в фиксированные позиции экрана).

Решение

```
NERD_tree_1 main.c
1 #include <pthread.h>
2 #include <stdio.h>
3 #include <signal.h>
4
5 char buffer[] = "abcdefghijklmnopqrstuvwxyz";
6 pthread_t th1, th2, th3;
7
8 void thread1(void* arg) {
9     pthread_setcancelstate(PTHREAD_CANCEL_DISABLE, NULL); *state: PTHREAD_CANCEL_DISABLE *oldstate: NULL
10
11     for (int i = 0; i < 20; i++) {
12
13         printf("\033[%d;20H\033[31m", i + 1); *format: "\033[%d;20H\033[31m"
14
15         for (int j = 0; j < (int)arg; j++) {
16             printf("%c", buffer[i]); *format: "%c"
17         }
18
19         printf("\n"); *format: "\n"
20         usleep(1011000);
21     }
22     pthread_exit(0); *retval: 0
23 }
24
```

Рис. 6: Код метода thread1

```
24
25 void thread2(void* arg) {
26     for (int i = 0; i < 20; i++) {
27
28         printf("\033[%d;40H\033[33m", i + 1); *format: "\033[%d;40H\033[33m"
29
30         for (int j = 0; j < (int)arg; j++) {
31             printf("%c", buffer[i]); *format: "%c"
32         }
33
34         printf("\n"); *format: "\n"
35         usleep(1022000);
36     }
37     pthread_exit(0); *retval: 0
38 }
39
```

Рис. 7: Код метода thread2

```

39
40 void thread3(void* arg) {
41     pthread_setcancelstate(PTHREAD_CANCEL_DISABLE, NULL); *state: PTHREAD_CANCEL_DISABLE *oldstate: NULL
42     pthread_setcanceltype(PTHREAD_CANCEL_DEFERRED, NULL); *type: PTHREAD_CANCEL_DEFERRED *oldtype: NULL
43
44     for (int i = 0; i < 20; i++) {
45         if (i == 12) {
46             pthread_setcancelstate(PTHREAD_CANCEL_ENABLE, NULL); *state: PTHREAD_CANCEL_ENABLE *oldstate: NULL
47         }
48
49         if (i == 16) {
50             pthread_testcancel();
51         }
52
53         printf("\033[%d;60H\033[32m", i + 1); *format: "\033[%d;60H\033[32m"
54
55         for (int j = 0; j < (int)arg; j++) {
56             printf("%c", buffer[i]); *format: "%c"
57         }
58
59         printf("\n"); *format: "\n"
60         usleep(1033000);
61     }
62     pthread_exit(0); *retval: 0
63 }
64

```

Рис. 8: Код метода thread3

```

64
65 int main() {
66     void* status = NULL;
67
68     printf("\033[H\033[J"); *format: "\033[H\033[J"
69
70     if (pthread_create(&th1, NULL, (void*)thread1, (void*)2)) { *newthread: &th1 *attr: NULL *start_routine: (void*)thread1 *ar
71         printf("\033[21;1H"); *format: "\033[21;1H"
72         printf("\033[21;1H\033[34m[-] Error create thread 1\n"); *format: "\033[21;1H\033[34m[-] Error create thread 1\n"
73     }
74
75     if (pthread_create(&th2, NULL, (void*)thread2, (void*)4)) { *newthread: &th2 *attr: NULL *start_routine: (void*)thread2 *ar
76         printf("\033[21;1H"); *format: "\033[21;1H"
77         printf("\033[21;1H\033[34m[-] Error create thread 2\n"); *format: "\033[21;1H\033[34m[-] Error create thread 2\n"
78     }
79
80     if (pthread_create(&th3, NULL, (void*)thread3, (void*)6)) { *newthread: &th3 *attr: NULL *start_routine: (void*)thread3 *ar
81         printf("\033[21;1H"); *format: "\033[21;1H"
82         printf("\033[21;1H\033[34m[-] Error create thread 3\n"); *format: "\033[21;1H\033[34m[-] Error create thread 3\n"
83     }
84
85     fflush(stdout); *stream: stdout
86
87     for (int i = 0; i < 20; i++) {
88         printf("\033[%d;1H\033[37m", i + 1); *format: "\033[%d;1H\033[37m"
89         printf("%d %c\n", i + 1, buffer[i]); *format: "%d %c\n"
90
91         if (i == 5) {
92             pthread_cancel(th1); *th: th1
93             printf("\033[21;1H"); *format: "\033[21;1H"
94             printf("\033[21;1H\033[34m[+] Attempt to cancel thread 1\n"); *format: "\033[21;1H\033[34m[+] Attempt to cancel thr
95         }
96
97         if (i == 10) {
98             pthread_cancel(th1); *th: th1
99             printf("\033[21;1H"); *format: "\033[21;1H"
100             printf("\033[21;1H\033[34m[+] Attempt to cancel thread 3\n"); *format: "\033[21;1H\033[34m[+] Attempt to cancel thr
101         }
102
103         usleep(1000000);
104     }
105
106     printf("\033[21;1H"); *format: "\033[21;1H"
107     getchar();
108     printf("\033[37m"); *format: "\033[37m"
109
110     return 0;
111 }

```

Рис. 9: Код метода main

```

1 a      aa      aaaa      aaaaaa
2 b      bb      bbbb      bbbbbb
3 c      cc      cccc      cccccc
4 d      dd      dddd      dddddd
5 e      ee      eeee      eeeeee
6 f      ff

```

Рис. 10: Результат работы программы в момент работы первого потока

```

1 a      aa      aaaa      aaaaaa
2 b      bb      bbbb      bbbbbb
3 c      cc      cccc      cccccc
4 d      dd      dddd      dddddd
5 e      ee      eeee      eeeeee
6 f      ff      ffff      ffffff
7 g      gg      gggg      gggggg
8 h      hh      hhhh      hhhhhh
9 i      ii      iiii      iiiiii
10 j     jj      jjjj      jjjjjj
11 k     kk      kkkk      kkkkkk
12 l     ll      llll      llllll
13 m     mm      mmmm      mmmmmm
14 n     nn      nnnn      nnnnnn
15 o     oo      oooo      oooooo
16 p     pp      pppp      pppppp
17 q     qq      qqqq      qqqqqq
18 r     rr      rrrr      rrrrrr
19 s     ss      ssss      ssssss
20 t     tt      tttt      tttttt
[+] Attempt to cancel thread 3

```

Рис. 11: Результат работы программы в момент работы третьего потока