

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>
	Грицай А. С.
	<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

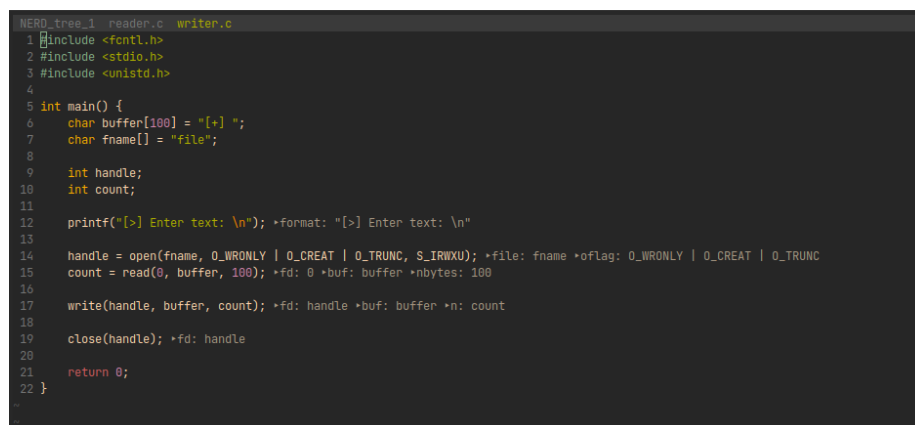
```
kurpenok@kurpenok-computer ~/0/0/3/0/Lab1 (main)> gcc main.c
kurpenok@kurpenok-computer ~/0/0/3/0/Lab1 (main)> ./a.out
[>] Enter text: random text
[+] random text
kurpenok@kurpenok-computer ~/0/0/3/0/Lab1 (main)> ./a.out > out
random text
kurpenok@kurpenok-computer ~/0/0/3/0/Lab1 (main)> cat out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

Разработать программу для Windows, которая должна запускаться в двух экземплярах - каждый в своем окне командной оболочки FAR или из ПРОВОДНИКА операционной системы. Программа использует заранее подготовленный текстовый файл. Она пытается открыть этот файл для чтения с указываемым при этом запрете для других использовать этот файл. По результатам выполнения системной функции открытия на экран выдается сообщение - удалось ли открыть файл и, если не удалось по причине отсутствия доступа к одновременно выполняемым программам, то сообщение именно об этой причине. При отсутствии указанного в программе файла после сообщения об этом отсутствии программа прекращает работу. При его наличии, но невозможности продолжения действий из-за блокировки, установленной другим экземпляром запущенной программы, выполняется ожидание освобождения файла от блокировки. При отсутствии указанной причины доступа программа должна ждать освобождения файла. В обоих случаях - ожидания освобождения или исходной доступности - программа читает из этого файла все находящиеся в нем данные и выводит их на экран. Сообщения должны выводиться цветные и в середине консольного окна.

```
1 #include <locale.h>
2 #include <stdio.h>
3 #include <windows.h>
4 #include <winerror.h>
5
6 int main() {
7     char buffer[100] = "";
8     DWORD len, actlen, dwfilesize;
9     HANDLE hstdout, fhandle;
10    char fname[] = "text.txt";
11    BOOL rc, open;
12    COORD cd_l;
13    cd_l.x = 50;
14    cd_l.y = 0;
15    const DWORD size = 100 + 1;
16    WCHAR lpzbuffer[size];
17    len = strlen(buffer);
18    hstdout = GetStdHandle(STD_OUTPUT_HANDLE);
19    if (hstdout == INVALID_HANDLE_VALUE)
20        return 0;
21
22    do {
23        fhandle = CreateFile(fname, GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
24        if (fhandle == INVALID_HANDLE_VALUE) {
25            setlocale(LC_ALL, "Rus");
26            system("cls");
27            DWORD Err = GetLastError();
28            SetConsoleCursorPosition(hstdout, cd_l);
29            SetConsoleTextAttribute(hstdout, FOREGROUND_INTENSITY | FOREGROUND_RED);
30            FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, NULL, Err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPTSTR)&lpzbuffer, size, NULL);
31            printf("error code: %d\nerror name: %s\n", Err, lpzbuffer);
32            Sleep(2000);
33            open = FALSE;
34        } else {
35            open = TRUE;
36        }
37    } while (open == FALSE);
38
39    system("cls");
40    SetConsoleCursorPosition(hstdout, cd_l);
41    SetConsoleTextAttribute(hstdout, FOREGROUND_INTENSITY | FOREGROUND_BLUE);
42    rc = ReadFile(fhandle, buffer + len, 80, &actlen, NULL);
43
44    if (!rc)
45        return 0;
46
47    WriteFile(hstdout, buffer, len + actlen, &actlen, NULL);
48    getch();
49    CloseHandle(fhandle);
50 }
```

Рис. 6: Код программы main.c

```
~/Documents/OS/Lab3
kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab3
$ ./a.exe
sh: line 1: cls: command not found
Error code: 0
Error name: The operation completed successfully.

sh: line 1: cls: command not found
Error code: 0
Error name: The operation completed successfully.

kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab3
$ touch text.txt

kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab3
$ ./a.exe
sh: line 1: cls: command not found

kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab3
$
```

Рис. 7: Результат работы программ main.c

Задание 4

Разработать программу для Linux, которая должна запускаться в двух экземплярах, каждый со своей виртуальной консоли. Программа использует заранее подготовленный текстовый файл. Она открывает этот файл, при невозможности этого действия выдается сообщение и прекращает выполнение. После успешного открытия файла делается попытка установить на весь файл многопользовательскую блокировку по записи. По результатам попытки выполнения блокировки – на экран выдается сообщение о его реализации или текущей невозможности это сделать. При невозможности установить блокировку сразу, программа задает блокировку с ожиданием ее выполнения. По установлении блокировки доступа программа читает из этого файла все находящиеся в нем данные и выводит их на экран. Затем программа делает задержку выполнения («засыпает») на 7 или 8 секунд, после чего снимает блокировку. Сообщения должны выводиться цветные и в середине экрана.

Решение

```
NERD_tree_1 main.c
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <sys/stat.h>
4 #include <fcntl.h>
5 #include <unistd.h>
6 #include <errno.h>
7 #include <string.h>
8
9 #define BUFFERSIZE 255
10
11 int main (int argc, char* argv[]) {
12     int file = open ("file.txt", O_RDWR); *file: "file.txt" *oflag: O_RDWR
13     char buffer[BUFFERSIZE];
14     if (file == -1) {
15         printf("file dont open with error %d\n", errno); *format: "file dont open with error %d\n"
16         return -1;
17     } else {
18         struct flock flptr;
19         flptr.l_start = 0;
20         flptr.l_whence = SEEK_SET;
21         flptr.l_len = 0;
22         flptr.l_type = F_WRLCK;
23
24         while (fcntl(file, F_SETLK, &flptr) == -1) { *fd: file *cmd: F_SETLK
25             printf("\033[1;31mcant change lock with error %d\033[0m\n", errno); *format: "\033[1;31mcant change lock with error %d\033[0m\n"
26             sleep(1); *seconds: 1
27         }
28         printf("\033[1;32mblock sucesfull!\n"); *format: "\033[1;32mblock sucesfull!\n"
29
30         if (read(file, buffer, BUFFERSIZE) == -1) { *fd: file *buf: buffer *nbytes: BUFFERSIZE
31             printf("\033[1;31mdont read with error %d\033[0m\n", errno); *format: "\033[1;31mdont read with error %d\033[0m\n"
32         }
33
34         printf("\033[1;32mfrom file was read %s\033[0m\n", buffer); *format: "\033[1;32mfrom file was read %s\033[0m\n"
35         sleep(20); *seconds: 20
36         flptr.l_type = F_ULOCK;
37
38         if (fcntl(file, F_SETLK, &flptr) == -1) { *fd: file *cmd: F_SETLK
39             printf("\033[1;31mcant unlock file with error %d\033[0m\n", errno); *format: "\033[1;31mcant unlock file with error %d\033[0m\n"
40         } else {
41             printf("\033[1;32mfile unlock\033[0m\n"); *format: "\033[1;32mfile unlock\033[0m\n"
42         }
43     }
44
45     return 0;
46 }
```

Рис. 8: Код программы main.c

```
kurpenok@kurpenok-computer ~/0/0/3/0/Lab4 (main)> touch file
kurpenok@kurpenok-computer ~/0/0/3/0/Lab4 (main)> gcc main.c
kurpenok@kurpenok-computer ~/0/0/3/0/Lab4 (main)> ./a.out
block sucesfull!
from file was read Some text

file unlock
kurpenok@kurpenok-computer ~/0/0/3/0/Lab4 (main)> █
```

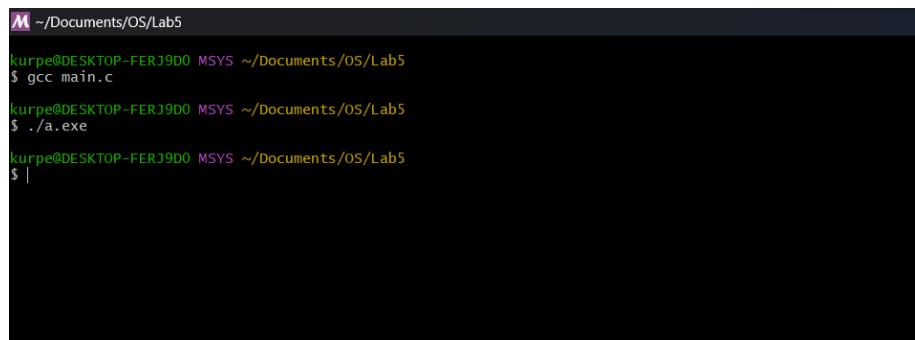
Рис. 9: Результат работы программ main.c

Задание 5

Программа для Windows должна открыть текстовый файл для чтения и все его содержимое вывести в текстовую консоль (текстовое окно на экране). Затем в программе запускается опрос мыши в консольном режиме. По щелчку мышью на любом слове, находящемся в текстовой консоли, в последней строке этого текстового окна выводится буква, на которой осуществлен щелчок, и координаты позиции, на которой он произошел, выраженные в виде номера строки и столбца текстового режима. Эта информация выводится только для не пустого позиции (для пробелов выводить ничего не нужно). Такие действия программ может осуществлять многократно. Завершение программы осуществляется по нажатию правой клавиши мыши.

```
~/Documents/OS/Lab5
1 #include <windows.h>
2 #include <stdio.h>
3
4 void main() {
5     char* buffer, symbol;
6     DWORD symlen, actlen = 0;
7     HANDLE fhandl, hstdin, hstdout;
8     fhandl = CreateFile("text.txt", GENERIC_READ, FILE_SHARE_READ, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
9     if (fhandl == INVALID_HANDLE_VALUE) {
10         return;
11     }
12     DWORD size = GetFileSize(fhandl, NULL);
13     buffer = (char*)malloc(size * sizeof(char));
14
15     hstdin = GetStdHandle(STD_INPUT_HANDLE);
16     if (hstdin == INVALID_HANDLE_VALUE)
17         return;
18     hstdout = GetStdHandle(STD_OUTPUT_HANDLE);
19     if (hstdout == INVALID_HANDLE_VALUE)
20         return;
21
22     INPUT_RECORD ibuf;
23     COORD symbolC;
24     ibuf.EventType = MOUSE_EVENT;
25
26     ReadFile(fhandl, buffer, size, &actlen, NULL);
27     WriteFile(hstdout, buffer, size, &actlen, NULL);
28
29     SetConsoleMode(hstdin, ENABLE_EXTENDED_FLAGS | ENABLE_MOUSE_INPUT);
30
31     printf("\n");
32     while (1) {
33         if (ReadConsoleInput(hstdin, &ibuf, 1, &symlen)) {
34
35             if (ibuf.Event.MouseEvent.dwButtonState == RIGHTMOST_BUTTON_PRESSED) {
36                 return;
37             }
38
39             if (ibuf.Event.MouseEvent.dwButtonState == FROM_LEFT_1ST_BUTTON_PRESSED) {
40                 symbolC.X = ibuf.Event.MouseEvent.dwMousePosition.X;
41                 symbolC.Y = ibuf.Event.MouseEvent.dwMousePosition.Y;
42                 ReadConsoleOutputCharacter(hstdout, &symbol, 1, symbolC, &symlen);
43
44                 if (symbol != ' ') {
45                     printf("%c (%d, %d)", symbol, symbolC.X, symbolC.Y);
46                 }
47             }
48         }
49     }
50 }
```

Рис. 10: Код программы main.c

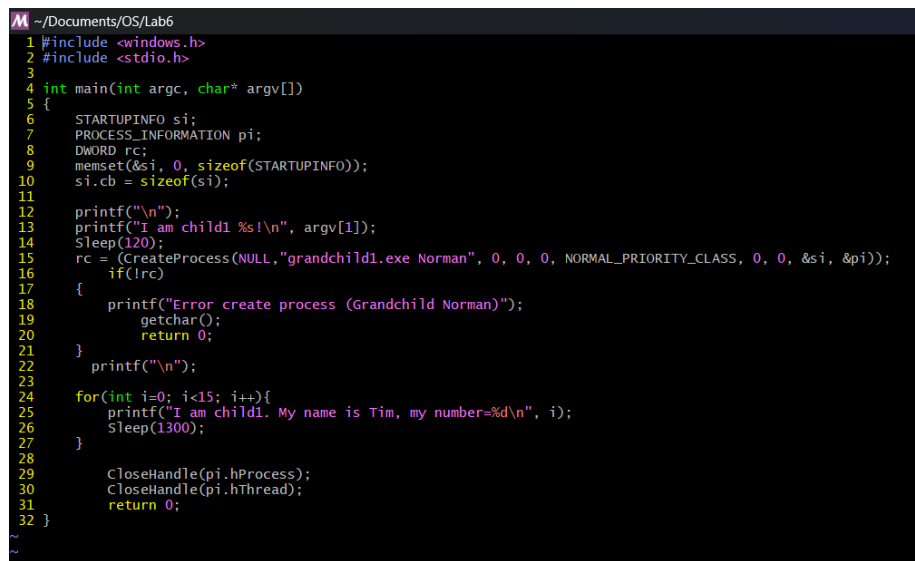


```
~/Documents/OS/Lab5
kurpe@DESKTOP-FER39D0 MSYS ~/Documents/OS/Lab5
$ gcc main.c
kurpe@DESKTOP-FER39D0 MSYS ~/Documents/OS/Lab5
$ ./a.exe
kurpe@DESKTOP-FER39D0 MSYS ~/Documents/OS/Lab5
$ |
```

Рис. 11: Результат работы программ main.c

Задание 6

В работе нужно разработать пять программ – одну для процесса-родителя, две – для дочерних процессов, еще две – для «внучатых» процессов. Все программы осуществляют вывод сообщений о своем «родственном» положении (например, «родитель», «дочерний», «внук» и номер шага в цикле-вывода таких сообщений, между которыми следует задавать задержки в пределах 1-2 секунд. Число повторений таких выводов сообщений должно составлять 10 – 20 для каждой программы, задержки для них следует выбирать различные, но незначительно – отличающиеся на 30–70



```
1 #include <windows.h>
2 #include <stdio.h>
3
4 int main(int argc, char* argv[])
5 {
6     STARTUPINFO si;
7     PROCESS_INFORMATION pi;
8     DWORD rc;
9     memset(&si, 0, sizeof(STARTUPINFO));
10    si.cb = sizeof(si);
11
12    printf("\n");
13    printf("I am child1 %s!\n", argv[1]);
14    Sleep(120);
15    rc = (CreateProcess(NULL, "grandchild1.exe Norman", 0, 0, 0, NORMAL_PRIORITY_CLASS, 0, 0, &si, &pi));
16    if(!rc)
17    {
18        printf("Error create process (Grandchild Norman)");
19        getchar();
20        return 0;
21    }
22    printf("\n");
23
24    for(int i=0; i<15; i++){
25        printf("I am child1. My name is Tim, my number=%d\n", i);
26        Sleep(1300);
27    }
28
29    CloseHandle(pi.hProcess);
30    CloseHandle(pi.hThread);
31    return 0;
32 }
```

Рис. 12: Код программы child1.c

```

~/Documents/OS/Lab6
1 #include <windows.h>
2 #include <stdio.h>
3
4 int main(int argc, char* argv[])
5 {
6     STARTUPINFO si;
7     PROCESS_INFORMATION pi;
8     DWORD rc;
9     memset(&si, 0, sizeof(STARTUPINFO));
10    si.cb = sizeof(si);
11
12    printf("\n");
13    printf("I am child2 %s!\n", argv[1]);
14    Sleep(140);
15    rc = (CreateProcess(NULL, "grandchild2.exe Sebastian", 0, 0, 0, NORMAL_PRIORITY_CLASS, 0, 0, &si, &pi));
16    if(!rc)
17    {
18        printf("Error create process (Grandchild Sebastian)");
19        getchar();
20        return 0;
21    }
22    printf("\n");
23
24    for(int i=0; i<15; i++){
25        printf("I am child2. My name is Jack, my number=%d\n", i);
26        Sleep(1500);
27    }
28
29    CloseHandle(pi.hProcess);
30    CloseHandle(pi.hThread);
31    return 0;
32 }
~

```

Рис. 13: Код программы child2.c

```

~/Documents/OS/Lab6
1 #include <stdio.h>
2 #include <windows.h>
3
4 int main(int argc, char* argv[])
5 {
6     printf("\n");
7     printf("Hi this is %s (grandchild1)\n", argv[1]);
8     Sleep(150);
9     for (int i = 0; i < 15; i++)
10    {
11        printf("Grandchild 1 %s: %i ---\n", argv[1], i);
12        Sleep(1700);
13    }
14    return 0;
15 }
~

```

Рис. 14: Код программы grandchild1.c

```

~/Documents/OS/Lab6
1 #include <stdio.h>
2 #include <windows.h>
3
4 int main(int argc, char* argv[])
5 {
6     printf("\n");
7     printf("Hi this is Sebastian (grandchild2)\n");
8     Sleep(160);
9     for (int i = 0; i < 15; i++)
10    {
11        printf("Grandchild 2: %i ---\n", i);
12        Sleep(2000);
13    }
14    return 0;
15 }
~

```

Рис. 15: Код программы grandchild2.c

```

~/Documents/OS/Lab6
1 #include <windows.h>
2 #include <stdio.h>
3 int main()
4 {
5     DWORD rc;
6     STARTUPINFO si1, si2;
7     PROCESS_INFORMATION pi1, pi2;
8
9     memset(&si1, 0, sizeof(STARTUPINFO));
10    si1.cb = sizeof(si1);
11
12    memset(&si2, 0, sizeof(STARTUPINFO));
13    si2.cb = sizeof(si2);
14
15    rc = CreateProcess(NULL, "child1.exe Tim", NULL, NULL, FALSE,
16                      NORMAL_PRIORITY_CLASS, NULL, NULL, &si1, &pi1);
17    if (!rc)
18    {
19        printf("Error create Process, codeError = %ld\n", GetLastError());
20        getchar();
21        return 0;
22    }
23    printf("Child1 process:\n");
24    printf("hProcess=%ld hThread=%ld ProcessId=%ld ThreadId=%ld\n", pi1.hProcess, pi1.hThread, pi1.dwProcessId, pi1.dwThreadId);
25    Sleep(100);
26    rc = CreateProcess(NULL, "child2.exe Jack", NULL, NULL, FALSE,
27                      NORMAL_PRIORITY_CLASS, NULL, NULL, &si2, &pi2);
28    if (!rc)
29    {
30        printf("Error create Process, codeError = %ld\n", GetLastError());
31        getchar();
32        return 0;
33    }
34    printf("Child2 process:\n");
35    printf("hProcess=%ld hThread=%ld ProcessId=%ld ThreadId=%ld\n", pi2.hProcess, pi2.hThread, pi2.dwProcessId, pi2.dwThreadId);
36    Sleep(110);
37    HANDLE job = CreateJobObject(NULL, NULL);
38    AssignProcessToJobObject(job, pi2.hProcess);
39
40    for (int i=0; i<12; i++) {
41        if(i == 7) {
42            TerminateProcess(pi1.hProcess, 0);
43            printf("Child1 - Tim was killed\n");
44        }
45        if(i == 11) {
46            TerminateJobObject(job, 0);
47            printf("Child2 - Jack was killed\n");
48        }
49        printf("I am Parent... (my K=%d)\n", i);
50        Sleep(1000);
51    }
52
53    CloseHandle(pi1.hProcess);
54    CloseHandle(pi1.hThread);
55    CloseHandle(pi2.hProcess);
56    CloseHandle(pi2.hThread);
57    getchar();

```

Рис. 16: Код программы parent.c

```

kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab6
$ ./parent.exe
Child1 process:
hProcess=348 hThread=200 ProcessId=5352 ThreadId=14748
Child2 process:
hProcess=396 hThread=372 ProcessId=17092 ThreadId=6320
I am Parent... (my K=0)
I am Parent... (my K=1)
I am Parent... (my K=2)
I am Parent... (my K=3)
I am Parent... (my K=4)
I am Parent... (my K=5)
I am Parent... (my K=6)
Child1 - Tim was killed
I am Parent... (my K=7)
I am Parent... (my K=8)
I am Parent... (my K=9)
I am Parent... (my K=10)
Child2 - Jack was killed
I am Parent... (my K=11)

```

Рис. 17: Результат работы программ parent.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. 18: 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. 19: 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. 20: 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. 21: 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. 22: 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. 23: Результат работы программы в момент работы третьего потока

Задание 8

Разработать многопоточную программу, отображающие на экране взаимодействие трех нитей «читателей» из общей области данных и трех «писателей», записывающих в этот буфер данные. Буфер предназначен для хранения 12 символов. Первая нить-писатель выводит латинскими буквами название города "Novosibirsk" вторая нить-писатель выводит латинскими буквами название города "Semipalatinsk" а третья — название города "Ekaterinburg". Такой вывод эти три нити осуществляют в два приема, первый из которых записывает половину своего текста без завершающего этот промежуточный текст нуля. Между такими половинами вывода нити производят задержку на случайную величину миллисекунд, но не более 1 сек. После вывода своего текста в буфер каждая нить-писатель переходит в ожидание порядка 2-3 сек до следующей попытки записи в буфер. Нити-читатели - через случайный интервал порядка 300 мсек - читают данные из буфера, если это позволяют средства синхронизации доступа между нитями, и вывод прочитанный текст на экран, каждая в свой столбец. Каждый вывод нити-читателя осуществляется в новую строку своего столбца, поэтому суммарные действия вывода в таких нитях предусмотреть только для 20 - 24 строк. Синхронизацию осуществить с помощью семафоров.

```
~/Documents/OS/Lab8
1 #include <windows.h>
2 #include <process.h>
3 #include <stdio.h>
4 #include <conio.h>
5 #include <time.h>
6
7 HANDLE hthread[3],hsm,hsm1;
8 char buffer[12],buff[12];
9 void gotoxy(const int x,const int y){//аналог встроенной функции gotoxy. Взята данная функция т.к gotoxy не поддерживалась
10 //моим компилятором/код функции взят с интернета
11 HANDLE OutputHandle;
12 CONSOLE_SCREEN_BUFFER_INFO ScreenBufInfo;
13 OutputHandle=GetStdHandle(STD_OUTPUT_HANDLE);
14 GetConsoleScreenBufferInfo(OutputHandle,&ScreenBufInfo);
15 ScreenBufInfo.dwCursorPosition.X=x;
16 ScreenBufInfo.dwCursorPosition.Y=y;
17 SetConsoleCursorPosition(OutputHandle,ScreenBufInfo.dwCursorPosition);
18 }
19 DWORD WINAPI Writer(void*n){//функция писатель, используемая как прототип для создания нитей-писателей
20 short i;
21 while(1) {
22 WaitForSingleObject(hsm,3000);//ожидание освобождения семафора, максимальное время ожидания 3 сек.
23 if((int)n==3){//определение того, что нужно записать в буфер
24 strcpy(buff,"Novosibirsk", 12);//заполнение массива, откуда будут браться данные в буфер
25 }else if((int)n==4){
26 strcpy(buff,"Ekaterinburg", 12);
27 }else if((int)n==5){
28 strcpy(buff,"Semipalatinsk", 12);
29 }
30 for(i=0;i<5;i++){//запись первой половины в буфер
31 buffer[i] = buff[i];
32 }
33 Sleep(1000);//пауза 1 сек.
34 for(i=6;i<11;i++){//запись второй половины в буфер
35 buffer[i]=buff[i];
36 }
37 ReleaseSemaphore(hsm1,1,NULL);//освобождение семафора
38 }
39 return 0;
40 }
```

Рис. 24: Код программы main1.c

```

41 DWORD WINAPI Reader(void*n){//функция читатель, используемая как прототип для создания нитей-читателей
42     short i=0,j;
43     while(1){
44         i++;//для смещения позиции вниз на 1 строку
45         WaitForSingleObject(hsm1,INFINITE);//ожидание освобождения семафора
46         gotoxy((i-1)*20,i);//задается позиция вывода
47         printf("%s",buffer);//непосредственно вывод
48         ReleaseSemaphore(hsm,1,NULL);//освобождение семафора
49         Sleep(290);//пауза
50     }
51     return(0);
52 }
53
54 void main(){
55     int i;
56     system("cls");//очистка экрана
57     hsm = CreateSemaphore(NULL,0,1,NULL);//создание семафора
58     hsm1 = CreateSemaphore(NULL,0,1,NULL);
59     for (i=3;i<=5;i++){ //создание писателей
60         hthread[i]=CreateThread(NULL,4096,writer,(void*)i,0,NULL);
61     }
62     for (i=0;i<=2;i++){//создание читателей
63         hthread[i]=CreateThread(NULL,4096,Reader,(void*)i,0,NULL);
64     }
65     ReleaseSemaphore(hsm,1,NULL);//освобождение семафора
66     getchar();//ожидаем ввод для закрытия программы
67     for (i=0;i<=4;i++)CloseHandle(hthread[i]);//закрываем все потоки
68     CloseHandle(hsm);
69 }

```

Рис. 25: Код программы main2.c

```

kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab8
$ ./a.exe
sh: line 1: cls: command not found
sdf
SemipalatinkSemipalatinkSemipalatinkSemipalatinkSemipalatinkSemipalatinkNovosibirskNovosibirskNovos
birk
kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab8
$ |

```

Рис. 26: Результат работы программ main.c

Задание 9

Разработать программы в ОС Windows для двух отдельных процессов, использующих общую память. В первой программе должна создаваться разделяемая память и семафор для взаимодействия между процессами. Во второй открывается доступ к этой же разделяемой памяти и семафору. Первая программа должна после задержки в 10 - 15 секунд записать какой-то текст в разделяемую память и указать его готовность с помощью семафора, а вторая программа должна вначале прочитать текстовую информацию из того места разделяемой памяти, где она должна появиться и выдать полученную информацию на экран с примечанием о существе действия, затем перейти к ожиданию разрешающего значения семафора и только после его завершения выдать на экран содержимое из разделяемой памяти с соответствующим примечанием. Вместо семафора в Windows можно использовать событие или мьютекс. Обе программы должны после получения действующего указателя на область общей памяти вывести адрес этой области в консольное окно с соответствующими пояснениями.

Первая программа, кроме того, после формирования данных для второго процесса, должна запросить 1000 байтов дополнительной памяти с помощью универсальной функции VirtualAlloc и записать в эту область памяти, начиная с ее начала, сколько возможно символов латинского алфавита, записывая эти символы, пропуская по 399 свободных байтов (т.е. записывая 'a' в байт с нулевым смещением этой области памяти, затем 'b' в байт со смещением 400, далее 'c' в байт со смещением 800 и т.д., выполняя такое продвижение на 400 байтов по крайней мере 15 раз). Каждую такую запись байта сопровождать детальным сообщением о действии и числовым значением адреса, по которому размещается этот байт в памяти. Наблюдаемые результаты объяснить.

```

~/Documents/OS/Lab9
1 #include<stdio.h>
2 #include <sys/types.h>
3 #include <unistd.h>
4 #include <string.h>
5 #include <pthread.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>
8 #include <semaphore.h>
9 #include <stdlib.h>
10 #include <errno.h>
11 #include <sys/mman.h>
12 #include <sys/shm.h>
13 #define BUFFERSIE 1000
14
15 char* sem_name = "sem192";
16 char* shm_name = "shm192";
17
18 int main(){
19     printf("fork ok\n");
20     sem_t *sem = sem_open(sem_name, 0, 0);
21     int fd = shm_open(shm_name, O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
22     if (fd == -1) {
23         perror("shm_open");
24         exit(EXIT_FAILURE);
25     }
26     ftruncate(fd, BUFFERSIE);
27     char *str = mmap(NULL, sizeof(char) * 1000, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
28     printf("i get first message: %s\n", str);
29     sleep(1);
30     int sem_val;
31     sem_getvalue(sem, &sem_val);
32     while (sem_val == 0) sem_getvalue(sem, &sem_val);
33     printf("i get second message: %s\n", str);
34 }
~
~

```

Рис. 27: Код программы app.c

```

~/Documents/OS/Lab9
1 #include<stdio.h>
2 #include <sys/types.h>
3 #include <unistd.h>
4 #include <string.h>
5 #include <pthread.h>
6 #include <sys/stat.h>
7 #include <sys/shm.h>
8 #include <fcntl.h>
9 #include <semaphore.h>
10 #include <stdlib.h>
11 #include <errno.h>
12 #include <sys/mman.h>
13 #define BUFFERSIE 1000
14
15 char* sem_name = "sem92";
16 char* shm_name = "shm192";
17 char* text = "Hello from main 1";
18 char* text2 = "Hello from main 2";
19 int main() {
20     sleep(5);
21     sem_t* sem = sem_open(sem_name, O_CREAT, 0777);
22     sem_post(sem);
23     int fd = shm_open(shm_name, O_RDWR, 0);
24     if (fd == -1) {
25         perror("shm_open");
26         char* str = mmap(NULL, BUFFERSIE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
27         sem_wait(sem);
28         printf("i send first message \n");
29         int a = strlen(text);
30         memcpy(str, text, strlen(text));
31         /* if (0 == fork()) {
32             printf("fork ok");
33             if (execle("app", NULL) == -1) printf("Error to creat process %d\n", errno);
34             perror("execle");
35         } else {*/
36             sleep(10);
37             printf("i send second message\n");
38             memcpy(str, text2, strlen(text2));
39             sem_post(sem);
40
41             // free sem and shm
42             sem_close(sem);
43             shm_unlink(shm_name);
44         }
45     }
~

```

Рис. 28: Код программы main.c

```

~/Documents/OS/Lab9
kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab9
$ ./app.exe shm
fork ok
shm_open: Invalid argument

kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab9
$ ./main.exe shm
shm_open: Invalid argument
i send first message
i send second message

kurpe@DESKTOP-FERJ9D0 MSYS ~/Documents/OS/Lab9
$

```

Рис. 29: Результат работы программ app.c и main.c