

Задание 1

```
C task1.c x C task2.c C task3.c C task3_p1.c C task3_p2.c C task4.c C task5.c
C task1.c > main()
10
11 #define FORK_FAILURE 1
12
13 int main()
14 {
15     int child[N];
16
17     printf("Parent process start! PID: %d, GROUP: %d\n", getpid(), getpgrp());
18
19     for (int i = 0; i < N; i++)
20     {
21         int child_pid = fork();
22
23         if(child_pid == ERR_FORK)
24         {
25             perror("Can't fork()\n");
26             return FORK_FAILURE;
27         }
28         else if (child_pid == 0)
29         {
30             printf("BEFORE SLEEP Child %d! PID: %d, PPID: %d, GROUP: %d\n", i + 1, getpid(), getppid(), getpgrp());
31
32             sleep(TIME_SLEEP);
33             printf("AFTER SLEEP Child %d! PID: %d, PPID: %d, GROUP: %d\n", i + 1, getpid(), getppid(), getpgrp());
34             exit(OK);
35         }
36         else
37         {
38             child[i] = child_pid;
39         }
40     }
41     printf("Parent process finished! Children: %d, %d\n\nParent: PID: %d, GROUP: %d\n", child[0], child[1], getpid(), getpgrp());
42
43     return OK;
44 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04$ gcc task1.c
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04$ ./a.out
Parent process start! PID: 8577, GROUP: 8577
BEFORE SLEEP Child 1! PID: 8578, PPID: 8577, GROUP: 8577
BEFORE SLEEP Child 2! PID: 8579, PPID: 8577, GROUP: 8577
Parent process finished! Children: 8578, 8579!
Parent: PID: 8577, GROUP: 8577
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04$ AFTER SLEEP Child 1! PID: 8578, PPID: 2838, GROUP: 8577
AFTER SLEEP Child 2! PID: 8579, PPID: 2838, GROUP: 8577
```

Задание 2

```
C task1.c C task2.c X C task3.c C task3_p1.c C task3_p2.c C task4.c C task5.c
C task2.c > main()
15 int main()
16 {
17     int child[N];
18     printf("Parent process start! PID: %d, GROUP: %d\n", getpid(), getppid());
19     for (int i = 0; i < N; i++)
20     {
21         int child_pid = fork();
22
23         if(child_pid == ERR_FORK)
24         {
25             perror("Can't fork()\n");
26             return FORK_FAILURE;
27         }
28         else if (!child_pid)
29         {
30             printf("Child %d! PID: %d, PPID: %d, GROUP: %d\n", i + 1, getpid(), getppid(), getpgrp());
31             exit(OK);
32         }
33         else
34         {
35             child[i] = child_pid;
36         }
37     }
38     for (int i = 0; i < N; i++)
39     {
40         int status;
41         int statval = 0;
42         pid_t child_pid = wait(&status);
43         printf("Child process %d finished. Status: %d\n", child_pid, status);
44         if (WIFEXITED(statval))
45         {
46             printf("Child process %d finished. Code: %d\n", i + 1, WEXITSTATUS(statval));
47         }
48         else if (WIFSIGNALED(statval))
49         {
50             printf("Child process %d finished from signal with code: %d\n", i + 1, WTERMSIG(statval));
51         }
52         else if (WIFSTOPPED(statval))
53         {
54             printf("Child process %d finished stopped. Number signal: %d\n", i + 1, WSTOPSIG(statval));
55         }
56     }
57     printf("Parent process finished! Children: %d, %d! \nParent: PID: %d, GROUP: %d\n ", child[0], child[1], getpid(), getppid());
58     return OK;
59 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04$ gcc task2.c
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04$ ./a.out
Parent process start! PID: 9195, GROUP: 9195
Child 1! PID: 9196, PPID: 9195, GROUP: 9195
Child 2! PID: 9197, PPID: 9195, GROUP: 9195
Child process 9196 finished. Status: 0
Child process 1 finished. Code: 0
Child process 9197 finished. Status: 0
Child process 2 finished. Code: 0
Parent process finished! Children: 9196, 9197!
Parent: PID: 9195, GROUP: 9195
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04$
```

Задание 3

task1.c task2.c task3.c x task3_p1.c task3_p2.c task4.c task5.c

task3.c > main()

```
12 #define ERR_EXEC -1
13
14 #define FORK_FAILURE 1
15 #define EXEC_FAILURE 2
16
17 int main()
18 {
19     int child[N];
20     char *com[N] = {"/p1.exe", "/p2.exe"};
21     printf("Parent process start! PID: %d, GROUP: %d\n", getpid(), getppid());
22     for (int i = 0; i < N; i++)
23     {
24         int child_pid = fork();
25         if(child_pid == ERR_FORK)
26         {
27             perror("Can't fork()\n");
28             return FORK_FAILURE;
29         }
30         else if (!child_pid)
31         {
32             printf("Child %d! PID: %d, PPID: %d, GROUP: %d\n", i + 1, getpid(), getppid(), getppid());
33             int rc = execl(com[i], com[i], NULL);
34
35             if (rc == ERR_EXEC)
36             {
37                 perror("Can't exec");
38                 return EXEC_FAILURE;
39             }
40             exit(OK);
41         }
42         else
43         {
44             child[i] = child_pid;
45         }
46     }
47 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04\$ gcc task3.c

kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04\$./a.out

Parent process start! PID: 9460, GROUP: 9460

Child 1! PID: 9461, PPID: 9460, GROUP: 9460

Child 2! PID: 9462, PPID: 9460, GROUP: 9460

proc 1 (sort array) START
Arrat before: 4 9 2 -1 8 3 5
Array after: -1 2 3 4 5 8 9
proc 1 (sort array) END

proc 2 (reverse str) START
String before reverse: BMSTU IU7-52
String after reverse: 25-7UI UTSMB
proc 2 (reverse str) END

Child process 9461 finished. Status: 0
Child process 1 finished. Code: 0
Child process 9462 finished. Status: 0
Child process 2 finished. Code: 0
Parent process finished! Children: 9461, 9462!
Parent: PID: 9460, GROUP: 9460

Задание 4

```
C task1.c C task2.c C task3.c C task3_p1.c C task4.c X C task3_p2.c ... C task4.c X
C task4.c > main()
14 #define ERR_PIPE -1
15
16 #define FORK_FAILURE 1
17 #define EXEC_FAILURE 2
18 #define PIPE_FAILURE 3
19
20 int main()
21 {
22     int child[N];
23     int fd[N];
24     char text[LEN] = { 0 };
25     char *mes[N] = {"BMSTU IU7-52 Kozlova\n", "ABCDEFGH\n"};
26     if (pipe(fd) == ERR_PIPE)
27     {
28         perror("Can't pipe!");
29         return PIPE_FAILURE;
30     }
31     printf("Parent process start! PID: %d, GROUP: %d\n", getpid(), getpgid(0));
32     for (int i = 0; i < N; i++)
33     {
34         int child_pid = fork();
35         if (child_pid == ERR_FORK)
36         {
37             perror("Can't fork()\n");
38             return ERR_FORK;
39         }
40         else if (!child_pid)
41         {
42             close(fd[0]);
43             write(fd[1], mes[i], strlen(mes[i]));
44             printf("Message %d sent to parent! %s", i + 1, mes[i]);
45             return OK;
46         }
47         else
48         {
49             child[i] = child_pid;
50         }
51     }
52     for (int i = 0; i < N; i++)
53     {
54         int status;
55         int statval = 0;
56
57         pid_t child_pid = wait(&status);
58
59         printf("Child process %d finished. Status: %d\n", child_pid, status);
60
61         if (WIFEXITED(statval))
62         {
63             printf("Child process %d finished. Code: %d\n", i + 1, WEXITSTATUS(statval));
64         }
65         else if (WIFSIGNALED(statval))
66         {
67             printf("Child process %d finished from signal with code: %d\n", i + 1, WTERMSIG(statval));
68         }
69         else if (WIFSTOPPED(statval))
70         {
71             printf("Child process %d finished stopped. Number signal: %d\n", i + 1, WSTOPSIG(statval));
72         }
73     }
74
75     printf("\nMessage receive :\n");
76     close(fd[1]);
77     read(fd[0], text, LEN);
78     printf("%s\n", text);
79
80     printf("Parent process finished! Children: %d, %d! \nParent: PID: %d, GROUP: %d\n", child[0], child[1], getpid(), getpgid(0));
81
82     return OK;
83 }
84
85
```

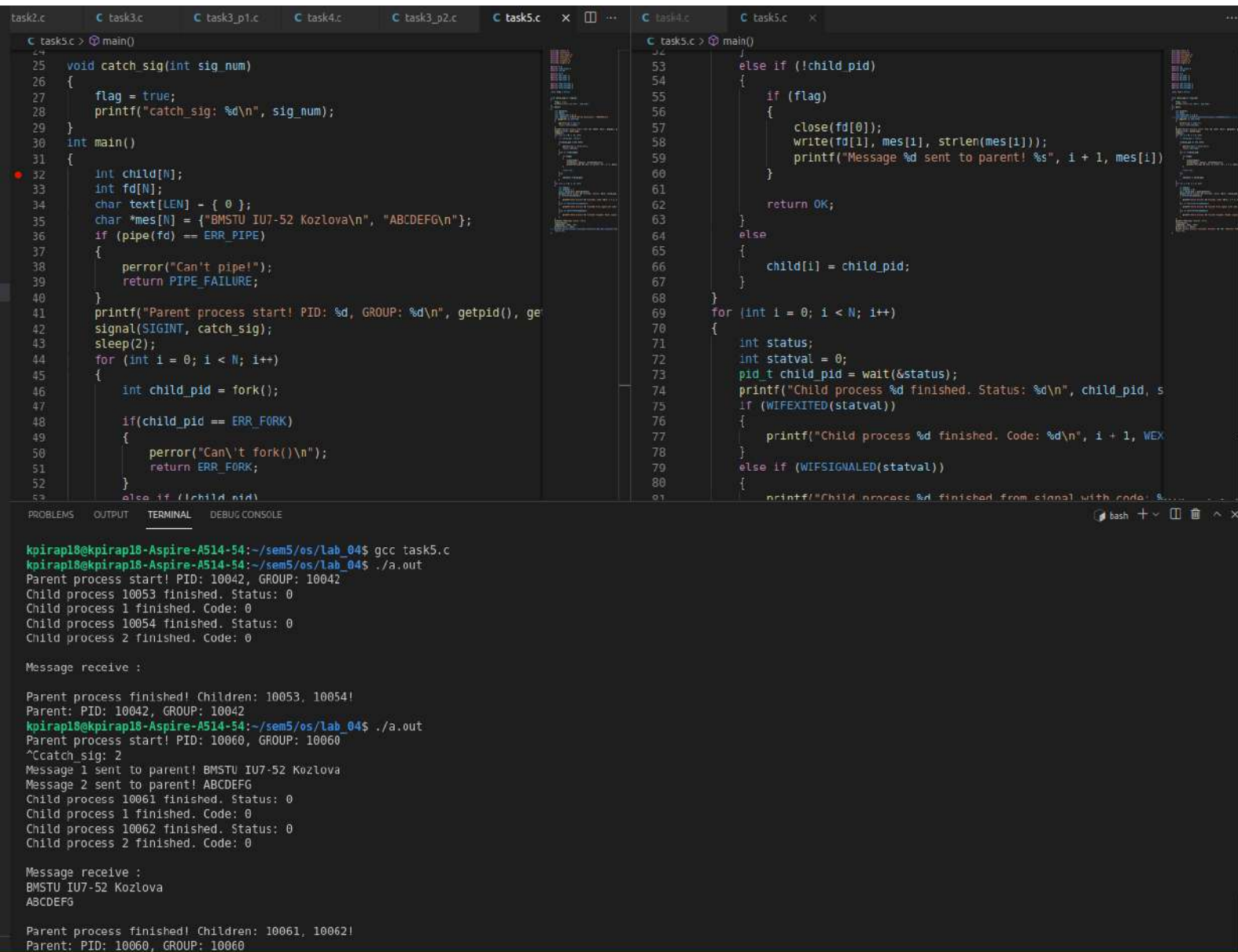
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/Lab_04$ gcc task4.c
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/Lab_04$ ./a.out
Parent process start! PID: 9749, GROUP: 9749
Message 1 sent to parent! BMSTU IU7-52 Kozlova
Message 2 sent to parent! ABCDEFGH
Child process 9750 finished. Status: 0
Child process 1 finished. Code: 0
Child process 9751 finished. Status: 0
Child process 2 finished. Code: 0

Message receive :
BMSTU IU7-52 Kozlova
ABCDEFGH

Parent process finished! Children: 9750, 9751!
Parent: PID: 9749, GROUP: 9749
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/Lab_04$
```


Задание 5



```
task2.c  task3.c  task3_p1.c  task4.c  task3_p2.c  task5.c  task4.c  task5.c
C task5.c > main()
25 void catch_sig(int sig_num)
26 {
27     flag = true;
28     printf("catch_sig: %d\n", sig_num);
29 }
30 int main()
31 {
32     int child[N];
33     int fd[N];
34     char text[LEN] = { 0 };
35     char *mes[N] = {"BMSTU IU7-52 Kozlova\n", "ABCDEFG\n"};
36     if (pipe(fd) == ERR_PIPE)
37     {
38         perror("Can't pipe!");
39         return PIPE_FAILURE;
40     }
41     printf("Parent process start! PID: %d, GROUP: %d\n", getpid(), getpgid(0));
42     signal(SIGINT, catch_sig);
43     sleep(2);
44     for (int i = 0; i < N; i++)
45     {
46         int child_pid = fork();
47         if (child_pid == ERR_FORK)
48         {
49             perror("Can't fork()\n");
50             return ERR_FORK;
51         }
52         else if (!child_pid)
53             continue;
54         else if (!child_pid)
55         {
56             if (flag)
57             {
58                 close(fd[0]);
59                 write(fd[1], mes[i], strlen(mes[i]));
60                 printf("Message %d sent to parent! %s", i + 1, mes[i]);
61             }
62             return OK;
63         }
64         else
65         {
66             child[i] = child_pid;
67         }
68     }
69     for (int i = 0; i < N; i++)
70     {
71         int status;
72         int statval = 0;
73         pid_t child_pid = wait(&status);
74         printf("Child process %d finished. Status: %d\n", child_pid, status);
75         if (WIFEXITED(status))
76         {
77             printf("Child process %d finished. Code: %d\n", i + 1, WEXITSTATUS(status));
78         }
79         else if (WIFSIGNALED(status))
80         {
81             printf("Child process %d finished from signal with code: %d\n", i + 1, WTERMSIG(status));
82         }
83     }
84     printf("Parent process finished! Children: %d, %d\n", child[0], child[1]);
85     printf("Parent: PID: %d, GROUP: %d\n", getpid(), getpgid(0));
86 }
```

```
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04$ gcc task5.c
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04$ ./a.out
Parent process start! PID: 10042, GROUP: 10042
Child process 10053 finished. Status: 0
Child process 1 finished. Code: 0
Child process 10054 finished. Status: 0
Child process 2 finished. Code: 0

Message receive :

Parent process finished! Children: 10053, 10054!
Parent: PID: 10042, GROUP: 10042
kpirap18@kpirap18-Aspire-A514-54:~/sem5/os/lab_04$ ./a.out
Parent process start! PID: 10060, GROUP: 10060
^Ccatch_sig: 2
Message 1 sent to parent! BMSTU IU7-52 Kozlova
Message 2 sent to parent! ABCDEFG
Child process 10061 finished. Status: 0
Child process 1 finished. Code: 0
Child process 10062 finished. Status: 0
Child process 2 finished. Code: 0

Message receive :
BMSTU IU7-52 Kozlova
ABCDEFG

Parent process finished! Children: 10061, 10062!
Parent: PID: 10060, GROUP: 10060
```