Задание 5:

В программу с программным каналом включить собственный обработчик сигнала. Использовать сигнал для изменения хода выполнения программы.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>
#include <time.h>
#include <sys/wait.h>

#define LEN 100
#define DELAY 2

int p_flag = 0;

void catch_sig(int signum) {
    printf("Proccess %d: catched signal %d\n", getpid(), signum);
    p_flag = 1;
}

int main() {
    //Pipe
    int fd[2];
    if (pipe(fd) == -1)
    {
        perror("Couldn't pipe.");
        exit(1);
    }
    // Ctrl + Z
    void (*old_handler)(int) = signal(SIGTSTP, catch_sig);

    // Child 1
    pid_t child1 = fork();
    if (child1 == -1)
    {
        perror("Couldn't fork.");
        exit(1);
    }
    else if (child1 == 0)
    {
        while (!p_flag) ;
        close(fd[0]);
        char msg1[] = "Hello From Child 1 ";
        write(fd[1], msg1, LEN);
        printf("Child 1: writen [%s] to Pipe\n", msg1);
    }

    // Child 2
    pid_t child2 = fork();
    if (child2 == -1)
    {
        perror("Couldn't fork.");
        exit(1);
    }
    else if (child2 == 0)
    {
        while (!p_flag) ;
```

```
56.         close(fd[0]);
57.         char msg2[] = "Hello From Child 2 ";
58.         write(fd[1], msg2, LEN);
59.         printf("Child 2: writen [%s] to Pipe\n", msg2);
60.     }
61.     // Parent
62.     if (child1 != 0 && child2 != 0)
63.     {
64.         printf("Parent: pid = %d\n", getpid());
65.         printf("Child 1: pid = %d\n", child1);
66.         printf("Child 2: pid = %d\n\n", child2);
67.         printf("Parent: waiting for CTRL+Z signal\n");
68.         while (!p_flag);
69.
70.         int status1, status2;
71.         pid_t ret1 = wait(&status1);
72.         pid_t ret2 = wait(&status2);
73.
74.         printf("------------\n");
75.         close(fd[1]);
76.         char msg1[LEN], msg2[LEN];
77.         read(fd[0], msg1, LEN);
78.         read(fd[0], msg2, LEN);
79.         printf("Parent: read from Pipe [%s%s]\n", msg1, msg2);
80.
81.         if (WIFEXITED(status1))
82.             printf("Parent: child %d finished with %d code.\n", ret1, WEXITSTATUS(status1));
83.         else if (WIFSIGNALED(status1))
84.             printf("Parent: child %d finished from signal with %d code.\n", ret1, WTERMSIG(status1));
85.         else if (WIFSTOPPED(status1))
86.             printf("Parent: child %d finished from signal with %d code.\n", ret1, WSTOPSIG(status1));
87.
88.         if (WIFEXITED(status2))
89.             printf("Parent: child %d finished with %d code.\n", ret2, WEXITSTATUS(status2));
90.         else if (WIFSIGNALED(status2))
91.             printf("Parent: child %d finished from signal with %d code.\n", ret2, WTERMSIG(status2));
92.         else if (WIFSTOPPED(status2))
93.             printf("Parent: child %d finished from signal with %d code.\n", ret2, WSTOPSIG(status2));
94.     }
95.     signal(SIGTSTP, old_handler);
96.     return 0;
97. }
```

```
nguyensang@K-virtual-machine:~/Desktop/OS2020/2$ ./5.exe
Parent: pid = 25678
Child 1: pid = 25679
Child 2: pid = 25680

Parent: waiting for CTRL+Z signal
^ZProccess 25678: catched signal 20
Proccess 25680: catched signal 20
Child 2: writen [Hello From Child 2 ] to Pipe
Proccess 25679: catched signal 20
Child 1: writen [Hello From Child 1 ] to Pipe
------------
Parent: read from Pipe [Hello From Child 2 Hello From Child 1 ]
Parent: child 25680 finished with 0 code.
Parent: child 25679 finished with 0 code.
nguyensang@K-virtual-machine:~/Desktop/OS2020/2$
```