

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ
по лабораторной работе №6
«Средства межпроцессного взаимодействия»

Выполнил:

Студент 2 курса
группы ПО-9
Мисиюк Алексей Сергеевич
(№ зач. книги 210664)

Проверила:

Давидюк Ю. И.

Брест 2022

Цель работы: познакомиться с организацией межпроцессного взаимодействия в ОС Linux.

Ход работы

Вариант индивидуального задания № 19

Код программы

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <string.h>
#include <signal.h>

// common space

int fd;
char *array;

void send_to_child(pid_t pid, char *string1, char *string2);
void send_to_parent(char *string1, char *string2);
void get_from_child(char *str);
void get_from_parent(char *string1, char *string2);

void *child_handler(int nsig);

int child_waiting = 1;
void child();

//pid - pid to another process
void parent(pid_t pid);

int main() {
    // create file; get fd
    if ((fd = open("commonfile.tmp", O_CREAT | O_RDWR, 0666)) < 0) {
        perror("Error opening file!");
        exit(-1);
    }

    // get file adress
    array = mmap(NULL, 2048, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

    //division zone:
```

```

pid_t child_pid = fork();
switch (child_pid) {
case -1:
    printf("Error doing fork()!\n");
    exit(-1);
    break;
case 0:

    // child space

    printf( "C\n");
    child();

    printf("Child exit 0\n");
    break;
    // ^ child space end

default:

    // parent space

    sleep(1);
    printf("P\n");
    parent(child_pid);

    close(fd);
    printf("Parent exit 0\n");
    break;
    // ^ parent space end
}

//common space

return 0;
}

void *child_handler(int nsig) {
    char string1[255]; char string2[255];

    get_from_parent(string1, string2);
    send_to_parent(string1, string2);

    printf("handler\n");

```

```

        child_waiting = 0; //stop child
    }

void child() {

    //setting handler
    struct sigaction act;
    memset(&act, 0, sizeof(act));
    act.sa_handler = child_handler;
    sigset_t set;
    sigemptyset(&set);
    sigaddset(&set, SIGUSR1);
    act.sa_mask = set;
    sigaction(SIGUSR1, &act, 0);

    signal(SIGUSR1, child_handler);

    while (child_waiting) {
        printf("Waiting for signal from parent...\n");
        sleep(1);
    }
}

void parent(pid_t pid) {
    char string1[] = "world!\n";
    char string2[] = "Hello \n";

    send_to_child(pid, string1, string2);
    sleep(1); //wait();

    char str[255];
    get_from_child(str);
    printf("Received:\n");
    write(1, str, strlen(str)-1);
}

void send_to_child(pid_t pid, char *string1, char *string2) {
    printf("Parent sending to child...\n");

    //strcpy(array, strcat(string1, string2));
    int i = 0;
    for (; string1[i] != '\n'; i++) {
        array[i] = string1[i];
    }
}

```

```

    }
    array[i++] = '\n';
    int j = 0;
    for (; string2[j] != '\n'; i++, j++) {
        array[i] = string2[j];
    }
    array[i++] = '\n';

    printf("Parent sent to child:\n%s%s", string1, string2);
    kill(pid, SIGUSR1);
    printf("Parent sent signal to child\n");
}

void send_to_parent(char *string1, char *string2) {
    strcpy(array, strcat(string2, string1));
    array[strlen(array)] = '\n';
}

void get_from_parent(char *string1, char *string2) {
    //strcpy(string1, array);
    //strcpy(string2, array);

    int i = 0;
    for (; array[i] != '\n'; i++) {
        string1[i] = array[i];
    }
    i++;

    int j = 0;
    for (; array[i] != '\n'; i++, j++) {
        string2[j] = array[i];
    }
}

void get_from_child(char *str) {
    strcpy(str, array);
}

```

Пример работы

```

C
Waiting for signal from parent...
P
Parent sending to child...
Parent sent to child:

```

```
world!  
Hello  
Parent sent signal to child  
handler  
Child exit 0  
Received:  
Hello world!  
Parent exit 0
```

Вывод: в данной программе был опробован метод общих файлов, а также сигналов для передачи информации между процессами.