

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ  
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ  
КАФЕДРА «ІНФОРМАЦІЙНІ СИСТЕМИ»

Лабораторна робота №12

З дисципліни

«Операційні системи»

Тема

**«Програмування міжпроцесної та багатопотокової взаємодії»**

Виконав:

Студент групи АІ-203

Вояковський Д. П.

Перевірили:

Дрозд М.О. Блажко О.А.

Одеса 2021

## **Завдання для виконання**

### **2.1 Робота з іменованими каналами**

2.1.1 В домашньому каталозі вашого користувача створіть іменований канал з використанням команди `mkfifo`:

- назва каналу співпадає з вашим прізвищем у транслітерації
- права доступу до каналу ( можна лише читати та писати власнику).

2.1.2 Підключіть до іменованого каналу процес, який буде в нього писати за такими командами:

- отримати зміст каталогу `/etc`
- отримати назви файлів, які починаються з букви вашого прізвища у транслітерації.

2.1.3 Перейдіть до нового терміналу роботи з ОС Linux та створіть процес, який буде читати зі створеного раніше каналу.

2.1.4 Поверніться до 1-го терміналу та підключіть до іменованого каналу процес, який буде в нього писати, архівуючи файл командою `gzip -c < pipe > file1.gz` де `pipe` – назва вашого каналу, `file1.gz` – назва файлу, який буде створено в результаті архівації

2.1.5 Перейдіть до 2-го терміналу роботи з ОС Linux та створіть процес, який буде читати зі створеного раніше каналу, архівуючи файл `/etc/passwd`

### **2.2 Програмування іменованих каналів**

Повторіть попереднє завдання, але пункт 2.1.1 виконайте через програмування іменованого каналу за прикладом з рисунку 1.

### **2.3 Програмування потоків**

За прикладом з рисунку 2 розробіть програму керування потоками, в якій в повідомленнях буде вказано ваше прізвище латиницею.

Виконайте програму за вказаним прикладом.

### **2.4 Програмування семафорів**

За прикладом з рисунку 3 розробіть програму керування семафором, в якій в повідомленнях буде вказано ваше прізвище латиницею.

Виконайте програму в двох терміналах за вказаним прикладом.

## 2.1

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ mkfifo voyakovskij
[voyakovskij_dmitro@vpsj3IeQ ~]$ ls -l voyakovskij
prw-rw-r-- 1 voyakovskij_dmitro voyakovskij_dmitro 0 May 22 19:48 voyakovskij
[voyakovskij_dmitro@vpsj3IeQ ~]$ chmod 600 voyakovskij
[voyakovskij_dmitro@vpsj3IeQ ~]$ ls -l voyakovskij
prw----- 1 voyakovskij_dmitro voyakovskij_dmitro 0 May 22 19:48 voyakovskij
```

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ find . -maxdepth 1 -name "v*" > voyakovskij
[voyakovskij_dmitro@vpsj3IeQ ~]$ ls /etc > voyakovskij
```

 voyakovskij\_dmitro@vpsj3IeQ:~

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ cat voyakovskij
./voyakovskiy
./voyakovskiy.sh
./v_a.csv
./voyakovskiy2.csv
./voyakovskiy_lab_3
./v_22.csv
./va.csv
./voyakovskij
./voyakovskiy_2.csv
[voyakovskij_dmitro@vpsj3IeQ ~]$ cat voyakovskij
adjtime
aliases
aliases.db
alternatives
anacrontab
asound.conf
audisp
audit
bash_completion.d
bashrc
binfmt.d
```

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ gzip -c < voyakovskij > f.gz
[voyakovskij_dmitro@vpsj3IeQ ~]$ gunzip -c f.gz
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

[voyakovskij_dmitro@vpsj3IeQ ~]$ cat /etc/passwd > voyakovskij
[voyakovskij_dmitro@vpsj3IeQ ~]$
[voyakovskij_dmitro@vpsj3IeQ ~]$
```


```

#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <stdio.h>

#define NAMEDPIPE_NAME "/home/voyakovskij_dmitro/voyakovskij"
#define BUFSIZE 50

int main (int argc, char ** argv) {
    int fd, len;
    char buf[BUFSIZE];
    if ( mkfifo(NAMEDPIPE_NAME, 0600) ) {
        fprintf(stderr, "Error in mkfifo!");
        return 1;
    }
    printf("%s is created\n", NAMEDPIPE_NAME);
    if ( (fd = open(NAMEDPIPE_NAME, O_RDONLY)) <= 0 ) {
        fprintf(stderr, "Error in open!");
        return 1;
    }
    printf("%s is opened\n", NAMEDPIPE_NAME);
    do {
        memset(buf, '\0', BUFSIZE);
        if ( (len = read(fd, buf, BUFSIZE-1)) <= 0 ) {
            printf("END!");
            close(fd);
            remove(NAMEDPIPE_NAME);
            return 0;
        }
        printf("Incomming message (%d): %s\n", len, buf);
    } while ( 1 );
}

```

 voyakovskij\_dmitro@vpsj3IeQ:~

[voyakovskij\_dmitro@vpsj3IeQ ~]\$ gcc -o 12 12.c

[voyakovskij\_dmitro@vpsj3IeQ ~]\$ ./12

/home/voyakovskij\_dmitro/voyakovskij is created

/home/voyakovskij\_dmitro/voyakovskij is opened

Incomming message (49): adjtime

aliases

aliases.db

alternatives

anacronta

Incomming message (49): b

asound.conf

audisp

audit

bash\_completion.d

bash

Incomming message (49): rc

binfmt.d

centos-release

[voyakovskij\_dmitro@vpsj3IeQ ~]\$ ls /etc > voyakovskij

```
#include <stdio.h>
#include <pthread.h>

main() {
    pthread_t f2_thread, f1_thread;
    void *f2(), *f1();
    int i1 = 10, i2 = 10;
    pthread_create(&f1_thread, NULL, f1, &i1);
    pthread_create(&f2_thread, NULL, f2, &i2);
    pthread_join(f1_thread, NULL);
    pthread_join(f2_thread, NULL);
}

void *f1(int *x) {
    int i,n;
    n = *x;
    for (i=1;i<n;i++) {
<----->printf("f1: %s\n", "Voyakovskiy_1");
<----->sleep(1);
    }
    pthread_exit(0);
}

void *f2(int *x) {
    int i,n;
    n = *x;
    for (i=1;i<n;i++) {
<----->printf("f2: %s\n", "Voyakovskiy_2");
<----->sleep(1);
    }
    pthread_exit(0);
}
```

[illegible]

```
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <stdio.h>

#define SEMAPHORE_NAME "/semaphore"

int main(int argc, char ** argv) {
    sem_t *sem;
    if ( argc != 2 ) {
        if ((sem = sem_open(SEMAPHORE_NAME, O_CREAT, 0600, 0)) == SEM_FAILED ) {
            fprintf(stderr, "sem_open error");
            return 1;
        }
        printf("Voyakovskiy\nsem_open. Semaphore is taken.\nWaiting for it to be dropped.\n");
        if (sem_wait(sem) < 0 )
            fprintf(stderr, "sem_wait error");
        if ( sem_close(sem) < 0 )
            fprintf(stderr, "sem_close error");
        return 0;
    }
    else {
        printf("Dropping semaphore...\n");
        if ( (sem = sem_open(SEMAPHORE_NAME, 0)) == SEM_FAILED ) {
            fprintf(stderr, "sem_open error");
            return 1;
        }
        sem_post(sem);
        printf("sem_post. Semaphore dropped.\n");
        return 0;
    }
}
```

**Висновки:** під час виконання лабораторної роботи усі завдання були помірно складні.