Державний університет «Одеська Політехніка»
Інститут комп’ютерних систем
Кафедра «Інформаційних систем»

Лабораторна робота №12

З дисципліни

«Операційні системи»

Тема: «Програмування міжпроцесної та багатопоточної взаємодії»

Виконала: Студентка групи АІ-204
Озарчук А.С.
Перевірили: Блажко О. А
Дрозд М. О.

Одеса-2021

Мета роботи: вивчити особливості обміну інформацією між процесами за допомогою іменованих каналів, керування потоками, а також синхронізацію процесів через семафори та м`ютекси.

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.

Виконані завдання

2.1 Робота з іменованими каналами

2.1.1 В домашньому каталозі користувача я створила іменований канал з використанням команди mkfifo:

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

```
[ozarchuk_anna@vpsj3IeQ ~]$ mkfifo ozarchuk
[ozarchuk_anna@vpsj3IeQ ~]$ chmod 600 ozarchuk
[ozarchuk_anna@vpsj3IeQ ~]$ ls -l
total 256
drwxrwxr-x 2 ozarchuk_anna ozarchuk_anna  4096 Mar 28 11:11 1234567Anya
drwxrwxr-x 2 ozarchuk_anna ozarchuk_anna  4096 Mar 28 11:18 123Ozarchuk
-rw-rw-r-- 1 ozarchuk anna ozarchuk anna  1291 Mar 20 21:44 1.csv
```

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

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

```
[ozarchuk anna@vpsj3IeQ ~]$ ls /etc |grep "^o" >ozarchuk
```

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

```
[ozarchuk_anna@vpsj3IeQ ~]$ cat ozarchuk
openldap
opt
oraInst.loc
oratab
os-release
```

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

```
[ozarchuk_anna@vpsj3IeQ ~]$ gzip -c <ozarchuk> file1.gz
[ozarchuk_anna@vpsj3IeQ ~]$
```

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

```
[ozarchuk_anna@vpsj3IeQ ~]$ cat /etc/passwd >ozarchuk | gunzip -c file1.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
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
```

2.2 Програмування іменованих каналів

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

```
[ozarchuk_anna@vpsj3IeQ ~]$ ./chanel
./ozarchuk is created
./ozarchuk is opened
Incoming message (43): openldap
opt
oraInst.loc
oratab
os-release

END![ozarchuk_anna@vpsj3IeQ ~]$ nano chanel.c
[ozarchuk_anna@vpsj3IeQ ~]$
```

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

#define NAMEDPIPE_NAME "./ozarchuk"
#define BUFSIZE   50

int main (int argc, char ** argv){
    int fd, len;
    char buf[BUFSIZE];

    if ( mkfifo(NAMEDPIPE_NAME, 0777) ) {
        fprintf(stderr,"Error in mkfifo!\n");
        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("Incoming message (%d): %s\n", len, buf);
    } while ( 1 );
}
```

2.3 Програмування потоків

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

Виконала програму за вказаним прикладом.

```
[ozarchuk_anna@vpsj3IeQ ~]$ gcc task3.c -o task3 -lpthread
[ozarchuk_anna@vpsj3IeQ ~]$ ./task3
f2: Ozarchuk 1
f1: Ozarchuk 1
f2: Ozarchuk 2
f1: Ozarchuk 2
f2: Ozarchuk 3
f1: Ozarchuk 3
f2: Ozarchuk 4
f1: Ozarchuk 4
f2: Ozarchuk 5
f1: Ozarchuk 5
f2: Ozarchuk 6
f1: Ozarchuk 6
f2: Ozarchuk 7
f1: Ozarchuk 7
f2: Ozarchuk 8
f1: Ozarchuk 8
f2: Ozarchuk 9
f1: Ozarchuk 9
[ozarchuk_anna@vpsj3IeQ ~]$
```

```c
#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: Ozarchuk %d\n", i);
     sleep(1);
     }
     pthread_exit(0);
}

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

2.4 Програмування семафорів

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

Виконала програму в двох терміналах за вказаним прикладом.

```
[ozarchuk_anna@vpsj3IeQ ~]$ gcc semaphore.c -o semaphore -lpthread
[ozarchuk_anna@vpsj3IeQ ~]$ ./semaphore
Ozarchuk : sem_open. Semaphore is taken.
Waiting for it to be dropped.
[ozarchuk_anna@vpsj3IeQ ~]$
```

```
[ozarchuk_anna@vpsj3IeQ ~]$ ./semaphore ozarchuk
Ozarchuk : Dropping semaphore...
Ozarchuk : sem_post. Semaphore dropped.
[ozarchuk_anna@vpsj3IeQ ~]$
```

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

#define SEMAPHORE_NAME "Ozarchuk"

int main(int argc, char ** argv) {

    sem_t *sem;

    if ( argc != 2 ) {
       if ((sem = sem_open(SEMAPHORE_NAME, O_CREAT, 0777, 0)) == SEM_FAILED ) {
           fprintf(stderr,"Ozarchuk : sem_open error\n");
           return 1;
       }
       printf("Ozarchuk : sem_open. Semaphore is taken.\nWaiting for it to be dropped.\n");
       if (sem_wait(sem) < 0 )
          fprintf(stderr,"Ozarchuk : sem_wait error\n");
       if ( sem_close(sem) < 0 )
          fprintf(stderr,"Ozarchuk : sem_close error\n");
       return 0;
    }
    else {
       printf("Ozarchuk : Dropping semaphore...\n");
       if ( (sem = sem_open(SEMAPHORE_NAME, 0)) == SEM_FAILED ) {
            fprintf(stderr,"Ozarchuk : sem_open error\n");
            return 1;
       }
       sem_post(sem);
       printf("Ozarchuk : sem_post. Semaphore dropped.\n");
       return 0;

    }
}
```

Висновок: Я вивчила особливості обміну інформацією між процесами за допомогою іменованих каналів, керування потоками, а також синхронізацію процесів через семафори та м`ютекси.