

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

Лабораторна робота №12 з
дисципліни
“ Операційні системи”

Тема
«Програмування міжпроцесної та багатопоточної взаємодії»

Виконав:
Голованчук М.Ю.

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.

Хід роботи:

1. Перелік завдань до лабораторної роботи
2. Результатами виконання пунктів завдань
3. Висновки

Перелік завдань до лабораторної роботи:

Завдання 1. Робота з іменованими каналами

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

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

Завдання 2 Програмування іменованих каналів

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

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

#define NAMEDPIPE_NAME "/tmp/my_named_pipe"
#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!");
        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 );
}
```

Рис. 1 - Приклад програми створення каналу

Завдання 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.

```
// компіляція з підключенням бібліотеки -lpthread
#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: %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: %d\n", i);
        sleep(1);
    }
    pthread_exit(0);
}
```

Рис. 2 – Приклад програми зі створення двох потоків

Завдання 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.

```

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

#define SEMAPHORE_NAME "/my_named_semaphore"

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, "sem_open error");
            return 1;
        }
        printf("sem_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;
    }
}

```

Рис. 3 – Приклад програми з семафором загального виду

Результати виконання завдань:

Завдання 1

В домашньому каталозі створимо іменованний канал з використанням команди *mkfifo*:

```

91.219.60.189 - PuTTY
login as: golovanchuk_mikola
golovanchuk_mikola@91.219.60.189's password:
Last login: Sun May 23 21:02:42 2021 from 188.163.103.236
-bash-4.2$ mkfifo demtsun
-bash-4.2$ chmod u=rw,g=,o= demtsun
-bash-4.2$ ls -l
total 804
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 1332 Mar 22 00:46 2.csv
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 5365 Mar 21 21:07 accounts.csvs
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 507 Apr 4 13:09 bash_csv.sh
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 794 Apr 4 16:06 BASH.sh
-rwxrwxr-x 1 golovanchuk_mikola golovanchuk_mikola 8680 Apr 19 15:14 create
-rw-r--r-- 1 golovanchuk_mikola golovanchuk_mikola 490 Apr 19 14:00 create.c
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 0 Mar 21 21:03 null
prw----- 1 golovanchuk_mikola golovanchuk_mikola 0 May 23 21:12 holovanchuk

```

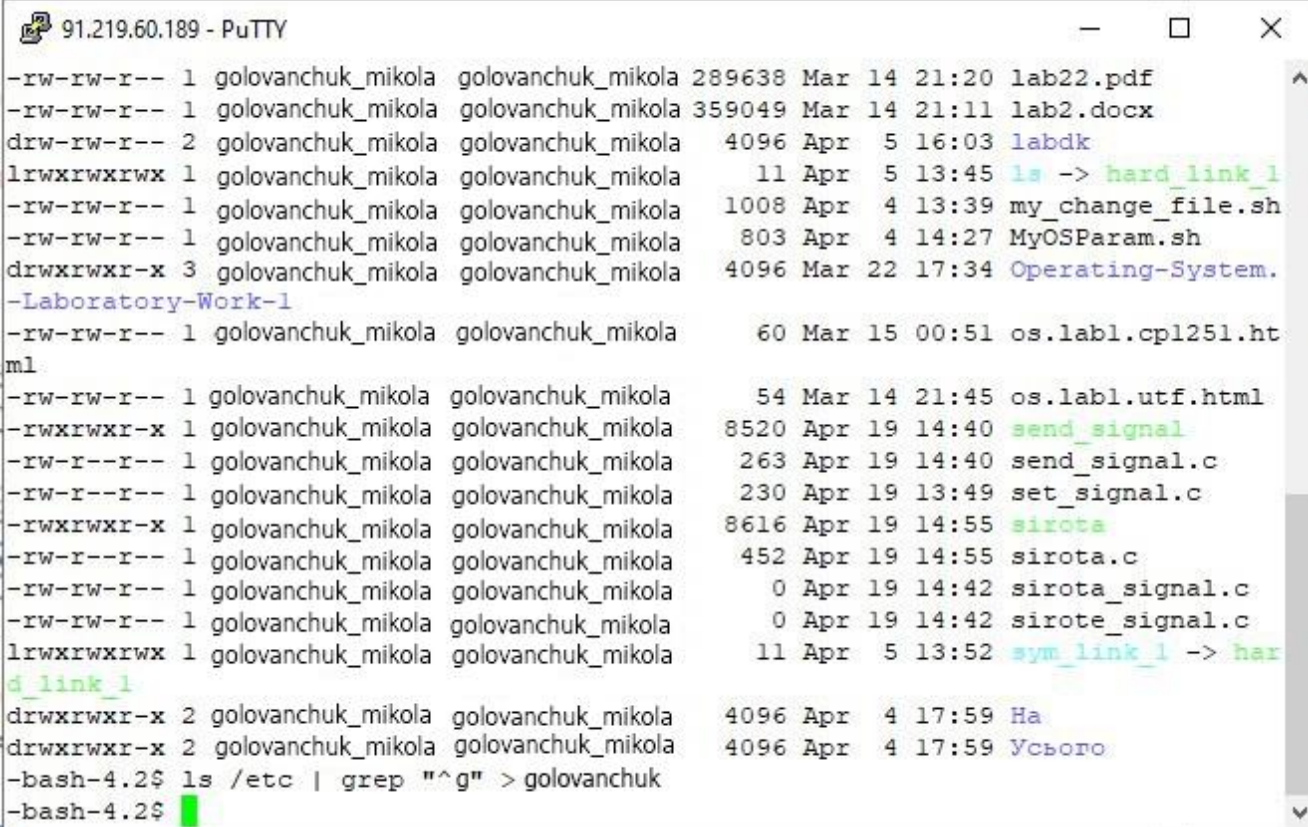

2 Підключимо до іменованого каналу процес, який буде в нього писати за такими командами:

- отримати зміст каталогу /etc

- отримати назви файлів, які починаються з букви вашого прізвища у транслітерації.

```
-bash-4.2$ ls /etc | grep "^d" > demtsun  
-bash-4.2$
```

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



```
91.219.60.189 - PuTTY  
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 289638 Mar 14 21:20 lab22.pdf  
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 359049 Mar 14 21:11 lab2.docx  
drwxrwxr-- 2 golovanchuk_mikola golovanchuk_mikola 4096 Apr 5 16:03 labdk  
lrwxrwxrwx 1 golovanchuk_mikola golovanchuk_mikola 11 Apr 5 13:45 ls -> hard_link_1  
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 1008 Apr 4 13:39 my_change_file.sh  
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 803 Apr 4 14:27 MyOSParam.sh  
drwxrwxr-x 3 golovanchuk_mikola golovanchuk_mikola 4096 Mar 22 17:34 Operating-System.  
-Laboratory-Work-1  
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 60 Mar 15 00:51 os.lab1.cpl251.ht  
ml  
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 54 Mar 14 21:45 os.lab1.utf.html  
-rwxrwxr-x 1 golovanchuk_mikola golovanchuk_mikola 8520 Apr 19 14:40 send_signal  
-rw-r--r-- 1 golovanchuk_mikola golovanchuk_mikola 263 Apr 19 14:40 send_signal.c  
-rw-r--r-- 1 golovanchuk_mikola golovanchuk_mikola 230 Apr 19 13:49 set_signal.c  
-rwxrwxr-x 1 golovanchuk_mikola golovanchuk_mikola 8616 Apr 19 14:55 sirota  
-rw-r--r-- 1 golovanchuk_mikola golovanchuk_mikola 452 Apr 19 14:55 sirota.c  
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 0 Apr 19 14:42 sirota_signal.c  
-rw-rw-r-- 1 golovanchuk_mikola golovanchuk_mikola 0 Apr 19 14:42 sirote_signal.c  
lrwxrwxrwx 1 golovanchuk_mikola golovanchuk_mikola 11 Apr 5 13:52 sym_link_1 -> har  
d_link_1  
drwxrwxr-x 2 golovanchuk_mikola golovanchuk_mikola 4096 Apr 4 17:59 Ha  
drwxrwxr-x 2 golovanchuk_mikola golovanchuk_mikola 4096 Apr 4 17:59 Усього  
-bash-4.2$ ls /etc | grep "^g" > golovanchuk  
-bash-4.2$
```

Повернемося до 1-го терміналу та підключимо 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

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

```
-bash-4.2$ gzip -c < /dev/pts/2 > file1.gz  
-bash-4.2$
```

```
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  
dbus:x:81:81:System message bus:/:/sbin/nologin  
polkitd:x:999:997:User for polkitd:/:/sbin/nologin  
postfix:x:89:89:/:/var/spool/postfix:/sbin/nologin  
chrony:x:998:996:/:/var/lib/chrony:/sbin/nologin  
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin  
soft:x:1000:1000:/:/home/soft:/sbin/nologin
```

Завдання 2

Програмування іменованих каналів

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

91.219.60.189 - PuTTY

GNU nano 2.3.1

File: fifo.c

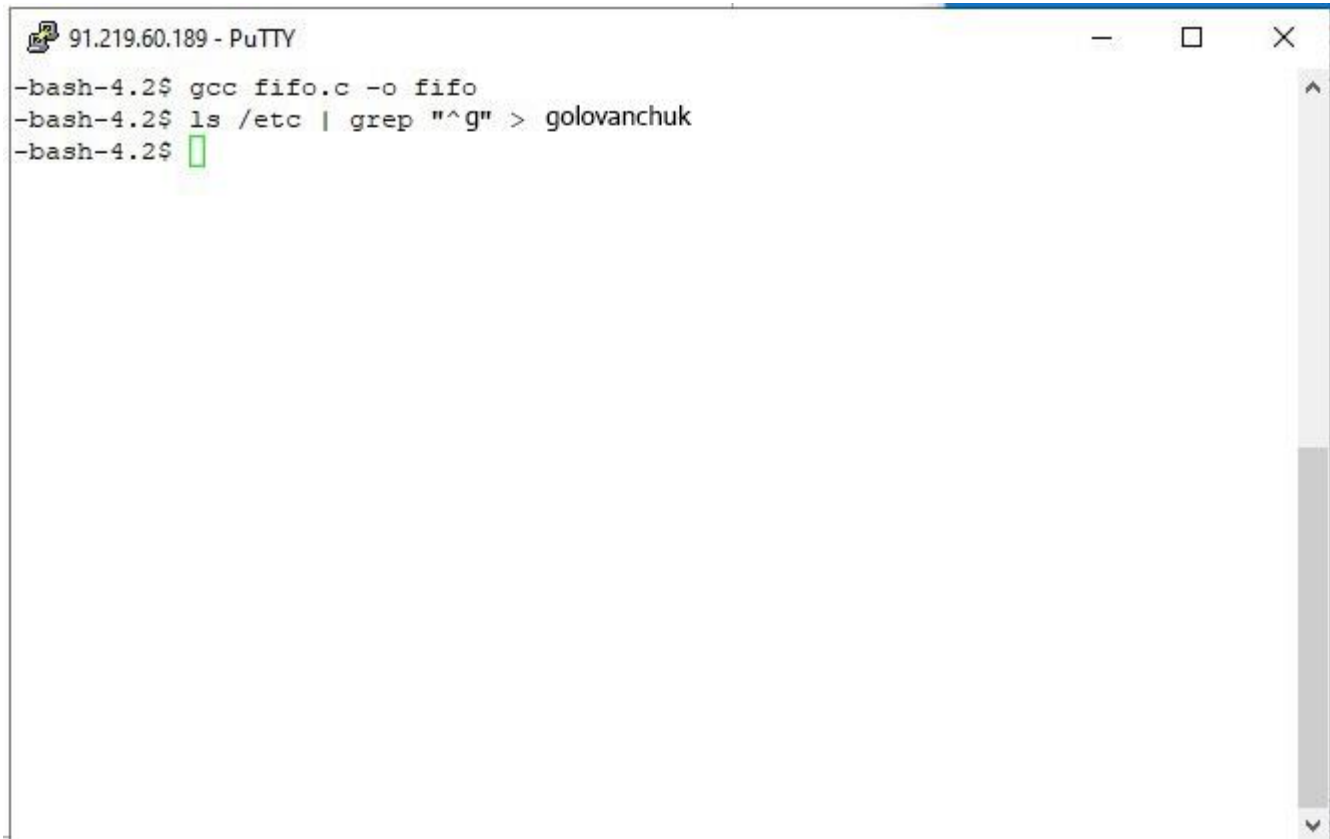
```
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <stdio.h>

#define NAMEDPIPE_NAME "golovanchuk_2"
#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!");
        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 );
}
```


Завдання 3



The screenshot shows a PuTTY terminal window titled "91.219.60.189 - PuTTY". The terminal displays three lines of commands in a bash shell:

```
-bash-4.2$ gcc fifo.c -o fifo  
-bash-4.2$ ls /etc | grep "^g" > golovanchuk  
-bash-4.2$
```

The prompt after the third line is highlighted with a green box.

Програмування потоків

За прикладом з рисунку 2 розробимо програму керування потоками, в якій в повідомленнях буде вказано ваще прізвище латиницею

Програмування семафорів

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

Виконаймо програму в двох терміналах за вказаним прикладом.

Завдання 4

91.219.60.189 - PuTTY

GNU nano 2.3.1

File: thread.c

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

int 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("golovanchuk(f1): %d\n", i);
        sleep(1);
    }
    pthread_exit(0);
}

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

```
-bash-4.2$ nano thread.c
-bash-4.2$ gcc thread.c -o thread -lpthread
-bash-4.2$ ./thread
golovanchuk (f2): 1
golovanchuk (f1): 1
golovanchuk (f2): 2
golovanchuk (f1): 2
golovanchuk (f2): 3
golovanchuk (f1): 3
golovanchuk (f2): 4
golovanchuk (f1): 4
golovanchuk (f2): 5
golovanchuk (f1): 5
golovanchuk (f2): 6
golovanchuk (f1): 6
golovanchuk (f2): 7
golovanchuk (f1): 7
golovanchuk (f2): 8
golovanchuk (f1): 8
golovanchuk (f2): 9
golovanchuk (f1): 9
-bash-4.2$
```

Завдання 5



The image shows two terminal windows from a PuTTY session. The left window displays the source code of a C program named 'semaphore.c'. The code defines a semaphore with a name '1' and a value of 1. It includes functions for opening, waiting, and dropping the semaphore, with error handling using fprintf to stderr. The right window shows the execution of the program. It starts with a login prompt, followed by the execution of './semaphore 1', which outputs 'sem_open. Semaphore is taken.' and 'Waiting for it to be dropped.'.

```
91.219.60.189 - PuTTY
ch function it appears in
n ' [enabled by default]
printf("sem_open. Semaphore is taken.\nWaiting for it to be dropped.");
;
warning: incompatible implicit declaration of macro 'sem_wait' [enabled by default]
on ' [enabled by default]
fprintf(stderr, "sem_wait error");
warning: incompatible implicit declaration of macro 'sem_post' [enabled by default]
n ' [enabled by default]
printf("Dropping semaphore...\n");
warning: incompatible implicit declaration of macro 'sem_wait' [enabled by default]
on ' [enabled by default]
fprintf(stderr, "sem_open error");
-bash-4.2$ nano semaphore.c
-bash-4.2$ gcc semaphore.c -o semaphore -lpthread
-bash-4.2$ ./semaphore
sem_open. Semaphore is taken.
Waiting for it to be dropped.
-bash-4.2$
```

```
91.219.60.189 - PuTTY
login as: golovanchuk_mikola
golovanchuk_mikola@91.219.60.189's password:
Last login: Sun May 23 21:33:32 2021 from 188.163.103.236
-bash-4.2$ ./semaphore 1
sem_open. Semaphore is taken.
Waiting for it to be dropped.
-bash-4.2$
```

Висновок: Під час виконання лабораторної роботи ми ознайомились із особливостями обміну інформацією між процесами за допомогою

іменованих каналів, керування потоками, а також синхронізацію процесів через семафори та м'ютекси. Усі завдання були однаково складні