

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:

Студент групи AI-202

Сідельніков М. В.

Перевірив:

Блажко О. А.

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:

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).

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.

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

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

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`

2.1 Програмування іменованих каналів

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

2.2 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.

2.3 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.

Хід роботи

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).

```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ mkfifo sidelnykov -m 600  
[sidelnikov_mikita@vpsj3IeQ ~]$ ls -l sidelnykov  
prw----- 1 sidelnikov_mikita sidelnikov_mikita 0 May 26 06:19 sidelnykov  
[sidelnikov_mikita@vpsj3IeQ ~]$
```

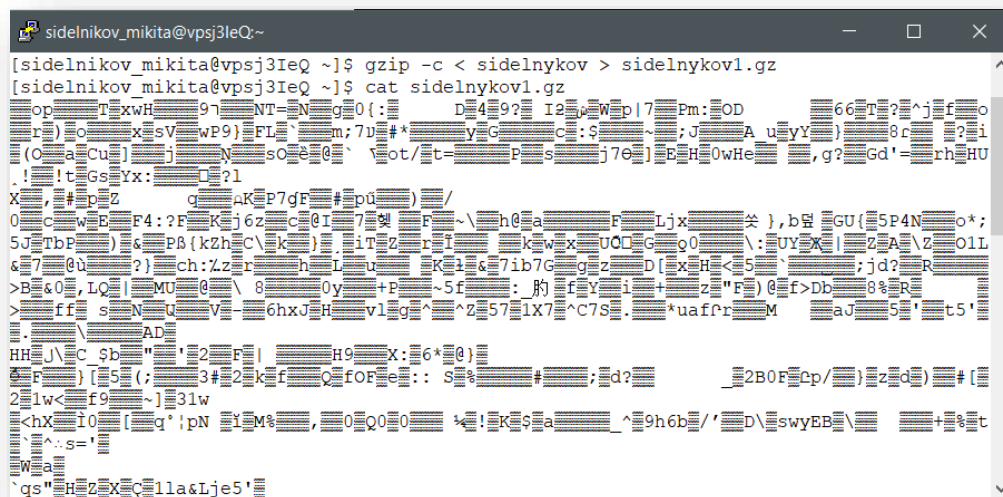
1.2 Підключіть до іменованого каналу процес, який буде в нього писати за такими командами:

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

```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ ls /etc > sidelnykov | find / -name "s*" > sidelnykov  
[sidelnikov_mikita@vpsj3IeQ ~]$
```

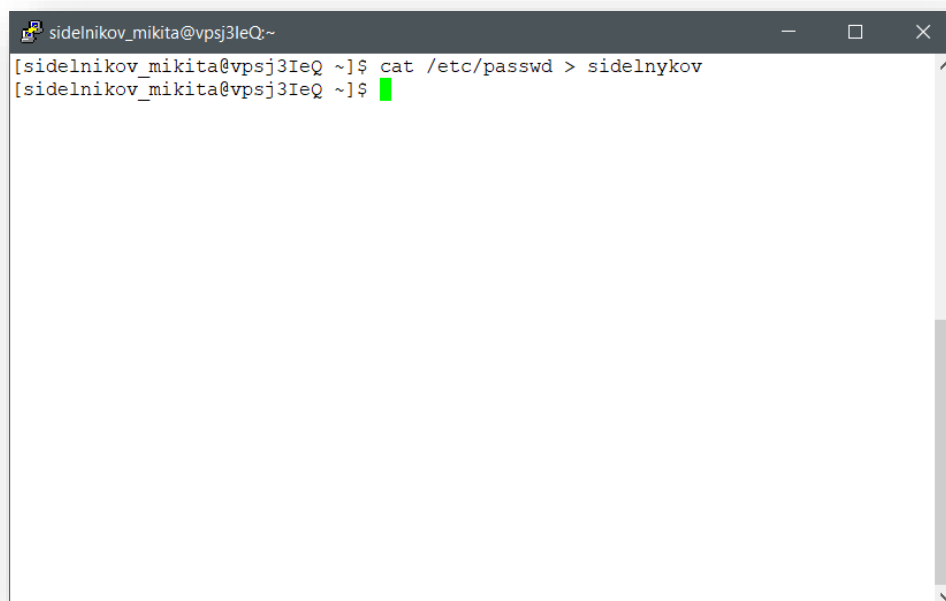
1.3 Перейдіть до нового терміналу роботи з ОС Linux та створіть процес, який буде читати зі створеного раніше каналу.

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.



```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ gzip -c < sidelnikov > sidelnikov1.gz  
[sidelnikov_mikita@vpsj3IeQ ~]$ cat sidelnikov1.gz  
opT xwH 97 NT=N g0{ : D4 9? I2 w p|7 Pm: OD 66 T? ^j f o  
r) o x sV wP9) FL' m; 7u #* y G C: $ ~; J A u yY) 8 c ? i  
(O a Cu j N s O s @ ' \ ot/ t= P s j 7 e) E H OwHe , g? Gd' = rh HU  
.! t Gs Yx: O ? l  
X , # p Z g AK P7 dF # p u ) /  
0 c w E F4: ? F K j 6 z c @ I 7 % F ~ \ h @ a F L j x 笑 }, b 冪 GU{ 5 P4 N o * ;  
5 J Tb P ) & P B ( k Zh C \ k ) . i t Z r I k w x U O G q 0 : U Y X | Z A \ Z O 1 L  
& 7 @ u ? ) ch: Z z r h L u K i & 7 i b 7 G g z D [ x H < 5 ` ; j d ? R  
> B & 0 , L Q [ MU @ \ 8 0 y + P ~ 5 f : 的 f Y i + z " F ) @ f > D b 8 % R  
> f f s N U V 6 h x J H v l g ^ ^ z 5 7 1 X 7 ^ C 7 S . * u a f r M a J 5 ' t 5 '  
. . \ AD  
H H J \ C $ b " ' 2 F | H 9 X: 6 * 0 )  
F ) [ 5 ( ; 3 # 2 k f Q f O F e :: S % # ; d ? _ 2 B 0 F Cp / ) z d ) # [   
2 l w < f 9 ~ } 3 l w  
< h X i 0 [ q ' ! p N y M % , 0 Q 0 0 % ! K $ a _ ^ 9 h 6 b / ' D \ s w y E B \ + % t  
' ^ . s = '  
W a  
' q s " H 2 X C 1 l a & L j e 5 '
```

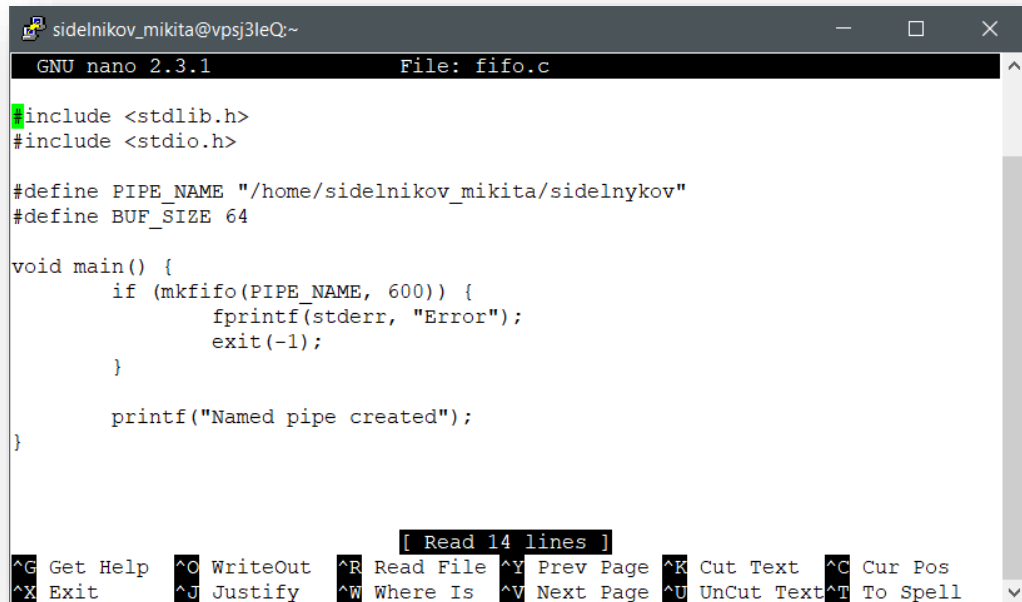
1.5 Перейдіть до 2-го терміналу роботи з ОС Linux та створіть процес, який буде читати зі створеного раніше каналу, архівуючи файл `/etc/passwd`



```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ cat /etc/passwd > sidelnikov  
[sidelnikov_mikita@vpsj3IeQ ~]$
```

2.1 Програмування іменованих каналів

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



```
sidelnikov_mikita@vpsj3IeQ:~
GNU nano 2.3.1 File: fifo.c

#include <stdlib.h>
#include <stdio.h>

#define PIPE_NAME "/home/sidelnikov_mikita/sidelnikov"
#define BUF_SIZE 64

void main() {
    if (mkfifo(PIPE_NAME, 600)) {
        fprintf(stderr, "Error");
        exit(-1);
    }

    printf("Named pipe created");
}

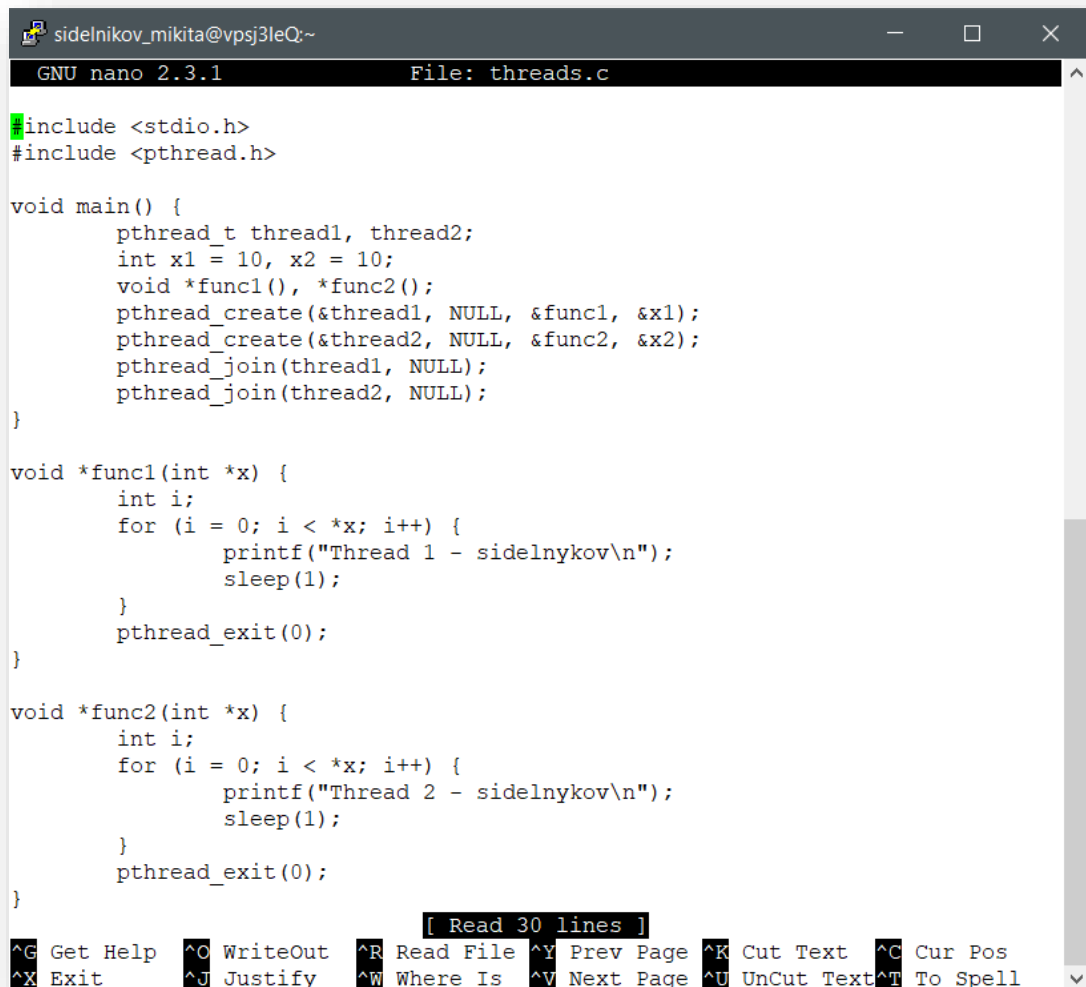
[ Read 14 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page  ^U UnCut Text ^T To Spell
```



```
sidelnikov_mikita@vpsj3IeQ:~
[sidelnikov_mikita@vpsj3IeQ ~]$ gcc fifo.c -o fifo
[sidelnikov_mikita@vpsj3IeQ ~]$ ./fifo
Named pipe created[sidelnikov_mikita@vpsj3IeQ ~]$
```

2.2 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.

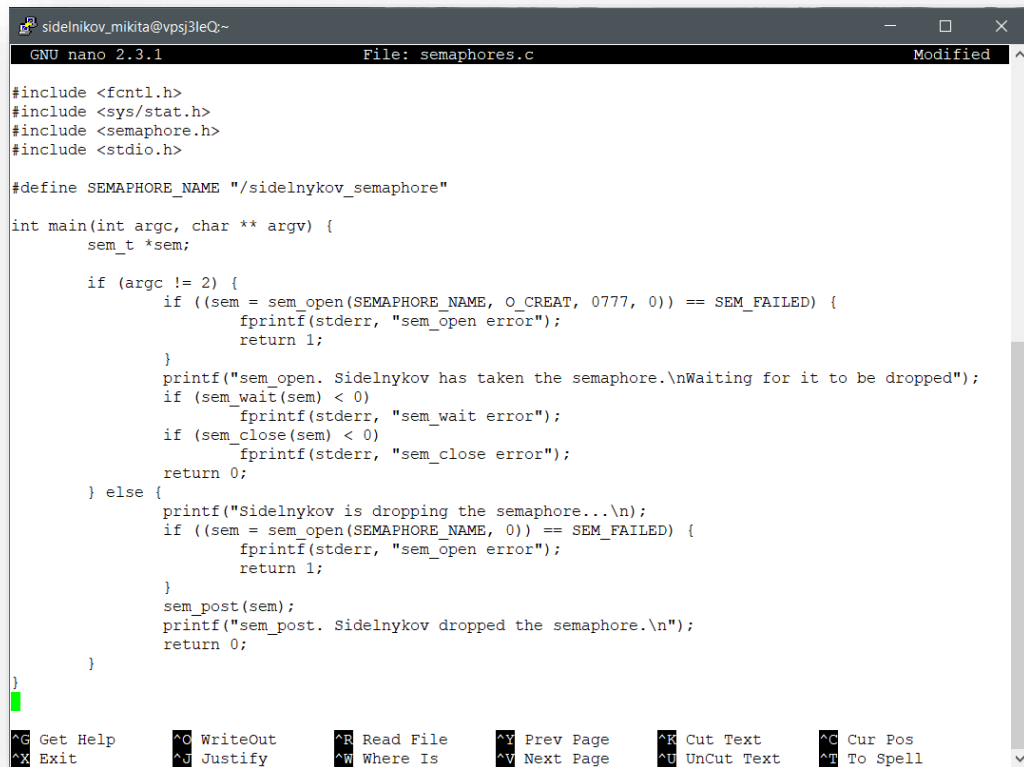


```
sidelnikov_mikita@vpsj3leQ:~  
GNU nano 2.3.1 File: threads.c  
#include <stdio.h>  
#include <pthread.h>  
  
void main() {  
    pthread_t thread1, thread2;  
    int x1 = 10, x2 = 10;  
    void *func1(), *func2();  
    pthread_create(&thread1, NULL, &func1, &x1);  
    pthread_create(&thread2, NULL, &func2, &x2);  
    pthread_join(thread1, NULL);  
    pthread_join(thread2, NULL);  
}  
  
void *func1(int *x) {  
    int i;  
    for (i = 0; i < *x; i++) {  
        printf("Thread 1 - sidelnykov\n");  
        sleep(1);  
    }  
    pthread_exit(0);  
}  
  
void *func2(int *x) {  
    int i;  
    for (i = 0; i < *x; i++) {  
        printf("Thread 2 - sidelnykov\n");  
        sleep(1);  
    }  
    pthread_exit(0);  
}  
[ Read 30 lines ]  
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ nano threads.c  
[sidelnikov_mikita@vpsj3IeQ ~]$ gcc threads.c -o threads -lpthread  
[sidelnikov_mikita@vpsj3IeQ ~]$ ./threads  
Thread 1 - sidelnikov  
Thread 2 - sidelnikov  
Thread 1 - sidelnikov  
Thread 2 - sidelnikov  
Thread 1 - sidelnikov  
Thread 2 - sidelnikov  
Thread 1 - sidelnikov  
Thread 2 - sidelnikov  
Thread 2 - sidelnikov  
Thread 1 - sidelnikov  
Thread 2 - sidelnikov  
Thread 1 - sidelnikov  
Thread 1 - sidelnikov  
Thread 2 - sidelnikov  
Thread 1 - sidelnikov  
Thread 2 - sidelnikov  
Thread 1 - sidelnikov  
Thread 2 - sidelnikov  
Thread 1 - sidelnikov  
Thread 2 - sidelnikov  
[sidelnikov_mikita@vpsj3IeQ ~]$
```

2.3 Програмування семафорів

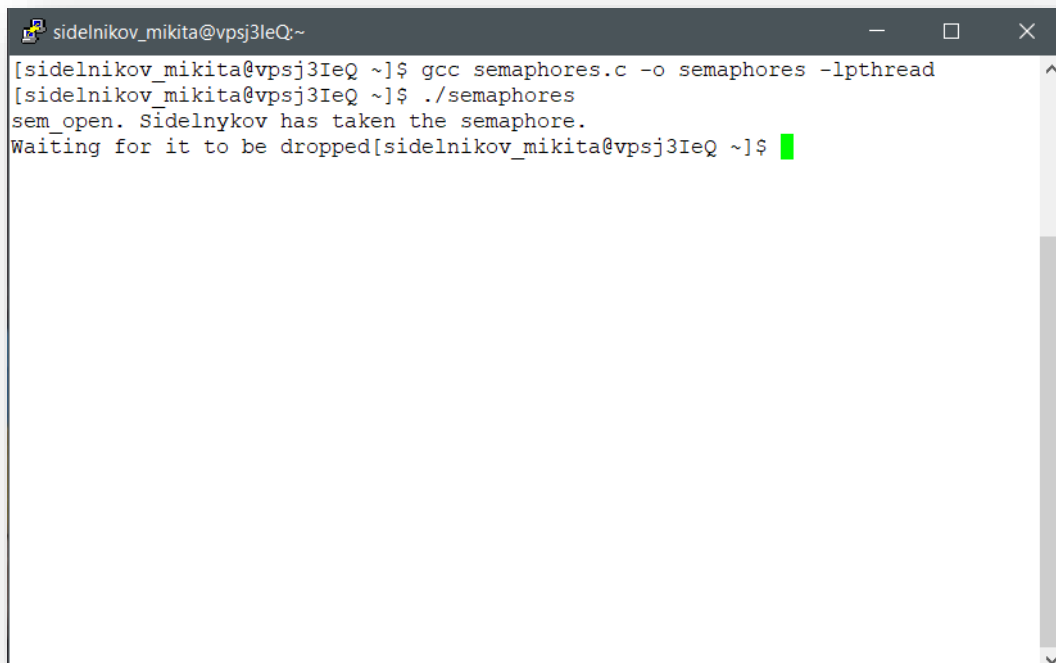
За прикладом з рисунку 3 розробіть програму керування семафором, в якій в повідомленнях буде вказано ваше прізвище латиницею. Виконайте програму в двох терміналах за вказаним прикладом.



```
sidelnikov_mikita@vpsj3IeQ:~  
GNU nano 2.3.1 File: semaphores.c Modified  
  
#include <fcntl.h>  
#include <sys/stat.h>  
#include <semaphore.h>  
#include <stdio.h>  
  
#define SEMAPHORE_NAME "/sidelnikov_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. Sidelnikov has taken the semaphore.\nWaiting for it to be dropped");  
        if (sem_wait(sem) < 0) {  
            fprintf(stderr, "sem_wait error");  
        }  
        if (sem_close(sem) < 0) {  
            fprintf(stderr, "sem_close error");  
        }  
        return 0;  
    } else {  
        printf("Sidelnikov is dropping the semaphore...\n");  
        if ((sem = sem_open(SEMAPHORE_NAME, 0)) == SEM_FAILED) {  
            fprintf(stderr, "sem_open error");  
            return 1;  
        }  
        sem_post(sem);  
        printf("sem_post. Sidelnikov dropped the semaphore.\n");  
        return 0;  
    }  
}
```

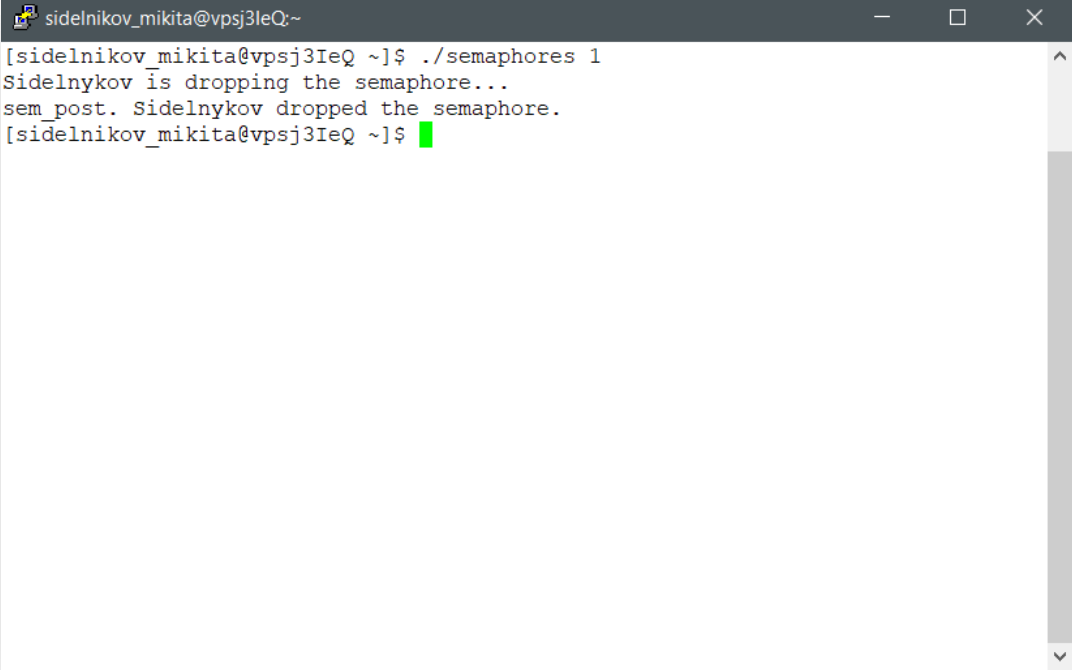
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

1-й термінал:



```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ gcc semaphores.c -o semaphores -lpthread  
[sidelnikov_mikita@vpsj3IeQ ~]$ ./semaphores  
sem_open. Sidelnikov has taken the semaphore.  
Waiting for it to be dropped[sidelnikov_mikita@vpsj3IeQ ~]$
```

2-й термінал:

A terminal window with a dark title bar containing the text 'sidelnikov_mikita@vpsj3IeQ:~' and standard window controls. The terminal output shows a user running a command to create a semaphore, followed by a message indicating its removal and a confirmation message. The prompt is a green cursor.

```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ ./semaphores 1  
Sidelnikov is dropping the semaphore...  
sem_post. Sidelnikov dropped the semaphore.  
[sidelnikov_mikita@vpsj3IeQ ~]$
```

Висновки: ми вивчили особливості обміну інформацією між процесами за допомогою іменованих каналів, керування потоками, а також синхронізацію процесів через семафори та м'ютекси.