МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра Інформаційної Безпеки

Операційні системи

Комп’ютерний практикум

Робота №9. Засоби синхронізації і взаємодії процесів

Мета:

Оволодіння практичними навичками використання засобів міжпроцесової взаємодії в Linux

Перевірив:
_____

Виконав:
студент II курсу
групи ФБ-01
Сахній Н.Р.

Київ 2022

Завдання до виконання:

Варіант 4

Програма моделює роботу примітивної СКБД, що зберігає єдину таблицю в оперативній пам'яті. Виконуючи деякі цикли робіт, K породжених процесів за допомогою черги повідомлень передають батьківському процесові номер рядка, який потрібно вилучити з таблиці. Батьківський процес виконує зазначену операцію і повертає вміст вилученого рядка.

- Файл із таблицею даних (**Football / Soccer Club World Ranking**)

```
nazar@ubuntu:~$ cd OS; mkdir lab_9; cd lab_9
nazar@ubuntu:~/OS/lab_9$ cat database.txt
№       Football club           Country         Points
1       Bayern                  Germany         138.000
2       Liverpool               England         134.000
3       Manchester City         England         134.000
4       Chelsea                 England         123.000
5       Real Madrid             Spain           122.000
6       Barcelona               Spain           117.000
7       PSG                     France          112.000
8       Juventus                Italy           107.000
9       Manchester United       England         105.000
10      Atletico Madrid         Spain           105.000
nazar@ubuntu:~/OS/lab_9$
```

- Код програми:

```
nazar@ubuntu:~/OS/lab_9$ nano IPC_for_database.cpp
```

nazar@ubuntu: ~/OS/lab_9

GNU nano 4.8 IPC_for_database.cpp

```
#include <iostream>
#include <stdlib.h>

#include <sys/wait.h>
#include <sys/msg.h>

#include <unistd.h>

using namespace std;


typedef struct buffer {
    long mtype;
    int num;
}
mess_t;

int main() {

    pid_t pid_1, pid_2;
    mess_t buff_1;
    mess_t buff_2;

```

```
    int length;
    length = sizeof(mess_t) - sizeof(long);

    int msqid_1, msqid_2;
    msqid_1 = msgget(1, IPC_CREAT | 0660);
    msqid_2 = msgget(2, IPC_CREAT | 0660);

    if (!(pid_1 = fork())) {
        sleep(2);
        cout << "\n    Дочірній процес №1 виконує деякий цикл робіт..." << endl;
        buff_1.mtype = 1;

        cout << "Виберіть номер рядка, який необхідно видалити із таблиці→ ";
        cin >> buff_1.num; cout << "\n";

        msgsnd(msqid_1, & buff_1, length, 0);
        return 0;
    }
    else if (!(pid_2 = fork())) {
        sleep(5);
        cout << "\n    Дочірній процес №2 виконує деякий цикл робіт..." << endl;
        buff_2.mtype = 1;

        cout << "Виберіть номер рядка, який необхідно видалити із таблиці→ ";
        cin >> buff_2.num; cout << "\n";

        msgsnd(msqid_2, & buff_2, length, 0);
        return 0;
    }
    waitpid(pid_1, NULL, 0);
    waitpid(pid_2, NULL, 0);
    cout << "\n•••Батьківський процес видаляє зазначені рядки. \n \
    І повертає вміст вилучених записів:" << endl;

    FILE * dataframe;
    char str[128];
    dataframe = fopen("football_database.csv", "r");
    fgets(str, 126, dataframe);


    msgrcv(msqid_1, & buff_1, length, 1, 0);
    msgctl(msqid_1, IPC_RMID, 0);
    for (int i = 0; i < buff_1.num; ++i) {
        fgets(str, 126, dataframe);
    }
    system(("sed '" + to_string(buff_1.num + 1) + "d' football_database.csv > football_database.txt").c_str());
    cout << str << endl;


    msgrcv(msqid_2, & buff_2, length, 1, 0);
    msgctl(msqid_2, IPC_RMID, 0);
    for (int i = 0; i < buff_2.num - buff_1.num; ++i) {
        fgets(str, 126, dataframe);
    }
    system(("sed '" + to_string(buff_2.num + 1) + "d' football_database.txt > football_database.csv").c_str());
    cout << str << endl;

    exit(1);
    return 0;
}
```

```
nazar@ubuntu:~/OS/lab_9$ g++ -pthread IPC_for_database.cpp -o IPC_for_database
nazar@ubuntu:~/OS/lab_9$ ls -l
total 28
-rw-rw-r-- 1 nazar nazar   339 May 11 12:33 football_database.csv
-rwxrwxr-x 1 nazar nazar 17896 May 11 12:47 IPC_for_database
-rw-rw-r-- 1 nazar nazar  1859 May 10 23:57 IPC_for_database.cpp
nazar@ubuntu:~/OS/lab_9$
```

- Результат виконання програми:

```
nazar@ubuntu:~/OS/lab_9$ ./IPC_for_database

    Дочірній процес №1 виконує деякий цикл робіт...
Виберіть номер рядка, який необхідно видалити із таблиці→ 2


    Дочірній процес №2 виконує деякий цикл робіт...
Виберіть номер рядка, який необхідно видалити із таблиці→ 5


•••Батьківський процес видаляє зазначені рядки.
     І повертає вміст вилучених записів:
2       Liverpool               England         134.000

5       Real Madrid             Spain           122.000

nazar@ubuntu:~/OS/lab_9$ cat football_database.csv
№       Football club           Country         Points
1       Bayern                  Germany         138.000
3       Manchester City         England         134.000
4       Chelsea                 England         123.000
5       Real Madrid             Spain           122.000
7       PSG                     France          112.000
8       Juventus                Italy           107.000
9       Manchester United       England         105.000
10      Atletico Madrid         Spain           105.000
nazar@ubuntu:~/OS/lab_9$
```

Висновки:

Під час виконання даної роботи я зрозумів, як правильно використовувати засоби міжпроцесової взаємодії в Linux. А також я дізнався про такі механізми IPC, як сигнали, сокети, семафори, повідомлення, канали, поділювана пам’ять.
Я навчився синхронізувати процеси, створювати чергу повідомлень та передавати повідомлення від дочірніх процесів до батьківського. Мною було створено програму, що імітує роботу СКБД і за введеним номером рядка вилучає його та повертає вміст на екран.

Контрольні запитання

1. У чому різниця між двійковим і звичайним семафорами?

За допомогою двійкового семафора можна організувати взаємне виключення, тобто захищати код критичних секцій від виконання більш як одним потоком.

А от для організації очікування виконання деякої умови, наприклад поки у буфері не звільниться місце або не добавиться новий об'єкт (як в задачі виробників-споживачів), краще використовувати звичайний семафор

2. Чим відрізняються операції P() і V() від звичайних операцій збільшення і зменшення на одиницю?

Семафор — це спільно використовуваний невід'ємний цілочисловий лічильник, для якого задано початкове значення і визначено такі атомарні операції.

- *Зменшення семафора* (down): якщо значення семафора більше від нуля, його зменшують на одиницю, якщо ж значення дорівнює нулю, цей потік переходить у стан очікування доти, поки воно не стане більше від нуля (кажуть, що потік «очікує на семафорі» або «заблокований на семафорі»). Цю операцію називають також *очікуванням* — wait. Ось її псевдокод:

```
void down (semaphore_t sem) {
    if (sem > 0) sem--;
    else sleep();
}
```

- *Збільшення семафора* (up): значення семафора збільшують на одиницю; коли при цьому є потоки, які очікують на семафорі, один із них виходить із очікування і виконує свою операцію down. Якщо на семафорі очікують кілька потоків, то внаслідок виконання операції up його значення залишається нульовим, але один із потоків продовжує виконання (у більшості реалізацій вибір цього потоку буде випадковим). Цю операцію також називають *сигналізацією* — post. Ось її псевдокод:

```
void up (semaphore_t sem) {
    sem++;
    if (waiting_threads()) wakeup (some_thread);
}
```

3. Для чого служить набір програмних засобів IPC?

Набір програмних засобів IPC Служить для обміну даними між процесами

4. Для чого введені масові операції над семафорами в ОС Linux?

Оскільки в багатьох додатках потрібно більше семафора, ОС UNIX надає можливість створювати безліч семафорів.

5. Яке призначення механізму черги повідомлень?

Черга повідомлень – це механізм операційної системи, що використовується для організації взаємодії між процесами програми за допомогою черги, що займається впорядкуванням повідомлень.

.

6. Які операції над семафорами існують в ОС Linux?

Над кожним семафором, за допомогою системного виклику semop можна виконати будь-яку з трьох операцій:

- Збільшити значення.
- Зменшити значення.
- Дочекатися обнулення.

7. Яке призначення системного виклику msgget()?

Системний виклик msgget() призначений для створення нової черги повідомлень або одержання дескриптора черги, що існує.

8. Які умови мають бути виконані для успішної постановки повідомлення в чергу?

Черга повідомлень не повинна бути заповненою, так як повідомлення не запишеться в чергу, а керування повернеться процесу виклику.
Якщо не вказано, то процес виклику буде призупинено (заблоковано), доки повідомлення не буде записано.

Повинен бути дозвіл на запис повідомлення до черги;

Має існувати черга повідомлень;

Необхідно достатньо пам’яті для копіювання буфера повідомлень.

9. Як отримати інформацію про власника і права доступу черги повідомлень?

Інформацію про власника і права доступу черги повідомлень можна отримати за допомогою системного виклику msgctl() із структури даних ipc_perm.

10. Яке призначення системного виклику shmget()?

Системний виклик shmget() призначений для створення нового сегмента поділюваної пам’яті або знаходження сегмента, що існує, з тим самим ключем.