**Робота №9. Засоби синхронізації і взаємодії процесів**

***Мета***

### Оволодіння практичними навичками використання

**засобів міжпроцесової взаємодії в Linux**

#### Завдання для самостійної підготовки

1. 1. Ознайомитись з документацією і прикладами використання засобів міжпроцесової взаємодії такими як семафори, м’ютекси, умовні змінні:

 man pages;

 книги з числа рекомендованих, зокрема [1, розд. 2.3], [5, розд. 6];

 стаття [13];

 книги з програмування в Linux [14, 15, 16].

1. 2. Якщо не робили попередні роботи, то перевірити, чи встановлений у вашій системі Linux компілятор С/С++ (g++). Якщо ні, встановіть за допомогою менеджера пакетів.

#### Довідковий матеріал

Тут наведено лише мінімальну інформацію, достатню хіба що для того, щоби зрозуміти, про що йде мова. Решту інформації необхідно здобути з джерел, названих вище (та інших).

Міжпроцесна взаємодія (Inter-process communication, IPC) — це набір методів і засобів для обміну даними між процесами. Можна говорити також про взаємодію потоків, і деякі засоби (такі, наприклад, як семафор) можуть застосовуватись для синхронізації потоків одного процесу. Але особливість засобів, що будуть розглянуті у цій роботі, полягає у тому, що операційна система надає їх потокам різних процесів, які мають різні адресні простори і взагалі можуть бути запущені на різних комп’ютерах, з’єднаних мережею. Прикладами механізмів IPC є сигнали, сокети, семафори, файли, повідомлення, канали, поділювана пам’ять. Для реалізації цих механізмів передбачені відповідні системні виклики.

Зверніть увагу на такі системні виклики, які вам ймовірно доведеться застосувати (деякі з них ви мабуть вже застосовували у попередніх роботах):

* Створення, завершення процесу, отримання інформації про процес: fork(), exit(), getpid(), getppid().
* Синхронізація процесів: signal(), kill(), sleep(), alarm(), wait(), pause().
* Створення інформаційного каналу і робота з ним: pipe(), read(), write().

Для роботи з семафорами підтримують три системних виклики:

* semget() для створення і одержання доступу до набору семафорів;
* semop() для маніпулювання значеннями семафорів (це той системний виклик, що дозволяє процесам

синхронізуватися на основі використання семафорів);

* semctl() для виконання різних операцій керуваннгя набором семафорів.

Прототипи цих системних викликів описані у файлах:

#include <sys/ipc.h>

#include <sys/sem.h>

Для обміну повідомленнями між процесами існує механізм черг, що підтримується такими системними викликами:

* msgget() для утворення нової черги повідомлень або одержання дескриптора черги, що існує;
* msgsnd() для постановки повідомлення у задану чергу повідомлень;
* msgrcv() для вибору повідомлення з черги повідомлень;  msgctl() для виконання низки дій керування.

Прототипи цих системних викликів описані у файлах:

#include <sys/ipc.h>

#include <sys/msg.h>

Для роботи з поділюванною пам’яттю застосовуються системні виклики:

* shmget() створює новий сегмент поділюваної пам’яті або знаходить сегмент, що існує, з тим самим ключем;
* shmat() підключає сегмент з указаним дескриптором до віртуального адресного простору процесу;
* shmdt() відключає від віртуального адресного простору сегмент зі вказаною віртуальною адресою початку, що був раніше до нього підключеним;
* shmctl() служить для керування різними параметрами, що пов’язані із сегментом, що існує.

Прототипи цих системних викликів описані у файлах:

#include <sys/ipc.h>

#include <sys/shm.h>

#### Завдання до виконання

1. Ознайомитись із завданням до лабораторної роботи (згідно варіанту, вказаного викладачем).

#### Варіант 1

Два дочірніх процеси виконують деякі цикли робіт, передаючи після закінчення чергового циклу через чергу повідомлень батьківському процесові чергові чотири рядка деякого віршу, при цьому перший процес передає непарні чотиривірші, а другий — парні. Цикли робіт процесів не збалансовані у часі. Батьківський процес компонує з фрагментів, що йому передають, закінчений вірш, і виводить його по завершенню роботи обох дочірніх процесів. Розв’язати задачу з використанням апарату семафорів.

#### Варіант 2

Два дочірніх процеси виконують деякі цикли робіт, передаючи після закінчення чергового циклу через один і той же сегмент поділюваної пам'яті батьківському процесові чергові чотири рядки деякого віршу, при цьому перший процес передає непарні чотиривірші, а другий — парні. Цикли робіт процесів не збалансовані за часом. Батьківський процес компонує з переданих фрагментів закінчений вірш і виводить його після закінчення роботи обох дочірніх процесів. Розв’язати задачу з використанням апарату семафорів.

#### Варіант 3

Чотири дочірніх процеси виконують деякі цикли робіт, передаючи після закінчення чергового циклу через один і той же сегмент поділюваної пам'яті батьківському процесові черговий рядок деякого віршу, при цьому перший процес передає 1-й, 5-й, 9-й і т.д. рядки, другий — 2-й, 6-й, 10-й і т.д. рядки, третій — 3-й, 7-й, 11-й і т.д. рядки, четвертий — 4-й, 8-й, 12-й і т.д. рядки. Цикли робіт процесів не збалансовані за часом. Батьківський процес компонує з переданих фрагментів закінчений вірш і виводить його після закінчення роботи всіх дочірніх процесів. Розв’язати задачу з використанням апарату семафорів.

#### 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.

#### Варіант 5

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 і вміст, на який потрібно замінити дані, що у ньому зберігаються. Батьківський процес виконує зазначену операцію і повертає попередній вміст рядка, що був змінений.

#### Варіант 6

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 вміст рядка, який потрібно додати до таблиці. Батьківський процес перевіряє, чи немає у таблиці такого рядка, і, якщо немає, додає рядок і повертає кількість рядків у таблиці.

#### Варіант 7

Чотири дочірніх процеси виконують деякі цикли робіт, передаючи після закінчення чергового циклу через чергу повідомлень батьківському процесові черговий рядок деякого віршу, при цьому перший процес передає 1-й, 5-й, 9-й і т.д. рядки, другий — 2-й, 6-й, 10-й і т.д. рядки, третій — 3-й, 7-й, 11й і т.д. рядки, четвертий — 4-й, 8-й, 12-й і т.д. рядки. Цикли робіт процесів не збалансовані за часом. Батьківський процес компонує з переданих фрагментів закінчений вірш і виводить його після закінчення роботи всіх дочірніх процесів. Розв’язати задачу з використанням апарату семафорів.

#### Варіант 8

Батьківський процес поміщає в сегмент поділюваної пам'яті імена програм з попередніх лабораторних робіт, які можуть бути запущені. Виконуючи деякі цикли робіт, породжені процеси випадковим чином обирають імена програм з таблиці сегмента поділюваної пам'яті, запускають ці програми, і продовжують свою роботу. За допомогою апарату семафорів має бути забезпечено, щоб не були одночасно запущені дві програми від одного процесу. В процесі роботи через чергу повідомлень батьківський процес інформується, які програми і від імені кого запущені.

#### Варіант 9

Батьківський процес поміщає в сегмент поділюваної пам'яті імена програм з попередніх лабораторних робіт, які можуть бути запущені. Виконуючи деякі цикли робіт, породжені процеси випадковим чином обирають імена програм з таблиці сегмента поділюваної пам'яті, запускають ці програми, і продовжують свою роботу. За допомогою апарату семафорів має бути забезпечено, щоб не були одночасно запущені дві однакові програми. В процесі роботи через чергу повідомлень батьківський процес інформується, які програми і від імені кого запущені.

#### Варіант 10

Програма моделює роботу монітора обробки повідомлень. Породжені процеси, що володіють різними пріоритетами і виконують деякі цикли робіт, за допомогою черги повідомлень передають батьківському процесові імена програм з попередніх лабораторних робіт, які їм повинні бути запущені. Батьківський процес, обробляючи повідомлення відповідно до їх пріоритетів, стежить, щоб одночасно було запущено не більше трьох програм.

1. Ознайомитись з основними поняттями механізму IPC (див. вище, див. літературу).
2. Розібратись з набором системних викликів, що забезпечують розв’язання завдання.
3. Налагодити і зневадити складену програму, використовуючи інструментарій ОС Linux.
4. Усі отримані результати оформити у вигляді протоколу.
5. Захистити роботу, продемонструвавши викладачеві роботу розробленої вами програми та її код, а також відповівши на контрольні запитання.

##### *Контрольні запитання*

1. У чому різниця між двійковим і звичайним семафорами?
2. Чим відрізняються операції P() і V() від звичайних операцій збільшення і зменшення на одиницю?
3. Для чого служить набір програмних засобів IPC?
4. Для чого введені масові операції над семафорами в ОС Linux?
5. Яке призначення механізму черги повідомлень?
6. Які операції над семафорами існують в ОС Linux?
7. Яке призначення системного виклику msgget()?
8. Які умови мають бути виконані для успішної постановки повідомлення в чергу?
9. Як отримати інформацію про власника і права доступу черги повідомлень?

10.Яке призначення системного виклику shmget()?