**Робота №8. Засоби синхронізації потоків**

### Мета

**Оволодіння практичними навичками розроблення багатопотокових програм з підтримкою засобів синхронізації**

### Завдання для самостійної підготовки

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

 man pages;

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

 [11, с. 103-126];

 [12, розд. 7, 8];

 корисна стаття [13] (у цій роботі нас цікавлять лише семафори і м’ютекси, але ми до цієї статті ще повернемось);

 великі книги з програмування в Linux, що орієнтовані на кодерів, містять приклади коду, перекладені російською мовою, тому комусь можуть бути цікавими, зрозумілими, і взагалі дуже корисними [14, 15, 16] (для цієї роботи див. розділи про семафори, м’ютекси, умовні змінні, тощо).

2. 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

Оскільки потоки мають доступ до спільної пам’яті, проблема синхронізації стоїть дуже гостро. Можливі змагання (race condition) і, як наслідок, спотворення результатів (уявіть два потоки, що без синхронізаціїї друкують рядок за рядком деякий вірш), втрати даних (два потоки записують в одну й ту ж саму ділянку пам’яті), взаємні блокування. З іншого боку, оскільки потоки працюють хз спільною пам’яттю, принципово організувати синхронізацію дуже просто.

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

Для синхронізації можна застосовувати прості об’єкти, такі як семафори, м’ютекси, умовні змінні. Існують також засоби більш високого рівня — монітори. Семафори можуть застосовуватись як для блокування критичних секцій (бінарні семафори), так і для очікування події. М’ютекси спеціалізовані для блокування критичних секцій, Умовні змінні спеціалізовані для очікування подій.

Зверніть увагу на принципово різні правила застосування семафорів і умовних змінних, щоби уникнути взаємних блокувань.

# Завдання до виконання

1. Розминка. Стандартна задача виробник-споживач.

Задача була розглянута на лекції. Також детально розглянута в рекомендованих книжках [1, 5]. Розробіть програму, що демонструє рішення цієї задачі за допомогою семафорів. Для цього напишіть:

* + функції виробника і споживача (наприклад, як на лекції, або як у Шеховцові, але так, щоби працювало);
	+ функції створення і споживання об’єктів (рекомендується “створювати” рядки тексту шляхом зчитування їх з файлу, хоча можливі й інші варіанти за вибором викладача або за вашою фантазією, наприклад розрахунки геш-функцій sha2 з рядків рандомних символів, а “споживати” їх шляхом роздрукування на екрані з додатковою інформацією такою як ідентифікатор потоку і мітка часу, причому і там, і там для моделювання складного характеру реального життя виробників і споживачів можна додавати рандомні затримки);
	+ функцію main(), що створює потоки-виробники і потокиспоживачі, при цьому треба передбачити введення з клавіатури або як параметри командного рядка кількості записів у буфері, кількості виробників і кількості споживачів для досліджень їх роботи;
	+ обов’язково передбачити коректне завершення усього цього господарства.

Продемонструвати викладачеві як воно працює (не менше двох виробників і двох споживачів) і код, що ви написали.

1. Продовження розминки. Теж саме, але не на семафорах, а на м’ютексі і умовних змінних

Модифікуйте програму п. 1 так, щоби використовувати м’ютекс і умовну змінну.

1. Продовження розминки для тих, хто шукає пригод. Взаємне блокування

Модифікуйте програму п. 1 так, щоби викликати взаємне блокування. Для цього поміняйте місцями семафори. Переконайтесь у факті взаємного блокування і отримайте задоволення.

1. Індивідуальне завдання

А тепер напишіть програму згідно індивідуального завдання (варіант вказує викладач).

Усі отримані результати оформіть у вигляді протоколу. Під час здачі роботи продемонструйте викладачеві роботу розроблених вами програм і їх код.

###  Варіант 1 а. Обчислення числа π

Напишіть багатопотокову програму, що обчислює число π за допомогою ряда Ляйбниця:

.

Кількість потоків програми і кількість ітерацій мають визначатися параметрами командного рядка. Для передавання часткових сум ряду, що підраховані потоками, можна застосовувати pthread\_exit() і pthread\_join().

Слід врахувати, що на 32-розрядних платформах sizeof(double)>sizeof(void \*), тому часткову суму ряду не можна перетворювати у вказівник, а треба виділяти для неї власну память.

###  Варіант 1 б. Об числення π аж поки не набридне

Завдання аналогічне Варіанту 1 а. Відмінність полягає в тому, що програма не повинна сама по собі завершуватись. Завершення має відбуватись після натискання Ctrl+C (тобто, після одержання сигналу SIGINT), при чому програма повинна не перериватись, а саме якнайшвидше завершуватись, збираючи часткові суми ряду, після чого виводити отримане наближення числа.

Для розв’язвння поставленго завдання рекмендується встановите обробник SIGINT. Обробник має встановлювати глобальну змінну — прапор. Потоки, що виконують обчислення, мають переглядати значення прапору через певну (доволі велику) кількість ітерацій (проведіть дослідження, як ця кількість впливає на швидкість обчислень). Якщо прапор встановлений, потік має виходити за допомогою pthread\_exit.

Як можна мінімізувати помилку, зумовлену тим, що різні потоки до завершення встигли пройти різну кількість ітерацій?

###  Варіант 2 а. Фі лософи, що обідають

Розробіть симулятор класичної задачі про філософів, що обідають. П’ять філософів сидять за круглим столом і їдять спагетті. Спагетті їдять за допомогою двох виделок. Всього виделок п’ять. Кожні двоє філософів, що сидять поруч, користуються однією спільною виделкою.

Кожний філософ незалежно від інших може знаходитись в одному з двох станів — їсть або думає. Філософ думає деякий час (передбачте можливість рандомізувати цей час у певному інтервалі, а також можливість задавати цей інтервал для дослідження), потім він намагається взяти виделки.

У цьому варіанті завдання усі філософи спочатку намагаються взяти ліву виделку, а потім праву. Якщо йому вдалося захопити обидві виделки, він починає їсти. Ість він також деякий час (як і думає — але співвідношення часів варто змінювати для дослідження), після чого він звільняє обидві виделки і знову починає думати. І так далі, поки у нього не закінчаться спагетті. Якщо одну з виделок взяти неможливо, філософ чекає, поки вона звільниться. Якщо йому протягом певного часу (помітно більшого, ніж час їжи і час роздумів) так і не вдається ухопити дві виделки, він падає в обморок (потік завершується).

Природно моделювати філософів за допомогою потоків, а виделки — за допомогою м’ютексів. Програма повинна синхронно (тобто, у тому ж порядку, як воно і відбувалося) друкувати усі події з мітками часу. Наприклад:

10:31:11.253 Філософ 1 узяв виделку 5 (ліву). Стан виделок

ХОООХ

10:31:11.255 Філософ 2 узяв виделку 2 (праву). Стан виделок ХХООХ

10:31:11.541 Філософ 1 не зміг узяти виделку 1 (праву). Стан виделок ХХООХ

10:31:11.883 Філософ 4 узяв виделку 3 (ліву). Стан виделок ХХХОХ

10:31:11.253 Філософ 2 почав їсти.

10:31:12.117 Філософ 3 не зміг узяти виделку 2 (ліву). Стан виделок ХХХОХ

10:31:12.733 Філософ 4 узяв виделку 4 (праву). Стан виделок ХХХХХ

...

10:31:14.125 Філософ 2 закінчив їсти.

...

10:31:11.255 Філософ 2 поклав виделку 1 (ліву). Стан виделок ОХХХХ

...

Чи спостерігалося взаємне блокування? За яких умов? Як його уникнути?

###  Варіант 2 б. Фі лософи, що обідають 2

Все тотожнє варіанту 2 а, за винятком протоколу захоплення виделок. Коли філософ може взяти одну виделку, але не може взяти другу, він має покласти виделку на стіл і повторити спробу через деякий час. Реалізуйте активне очікування.

Чи часто філософи падають в обморок?

###  Варіант 2 в. Фі лософи, що обідають 3

Все тотожнє варіанту 2 б, за винятком типу очікування. Якщо філософ не може взяти виделку (першу або другу), він має покласти на стіл першу виделку, якщо він її вже захопив, і заснути. Реалізувати можна за допомогою додаткових м’ютексів і умовних змінних. Відповідно, коли філософ звільнює виделки, він має сигналізувати про це.

Чи часто філософи падають в обморок? А якщо час його сну додавати до часу “голодування”?

###  Варіант 2 г. Фі лософи, що обідають 4

Все тотожнє варіанту 2 а, за винятком протоколу захоплення виделок. Усе рандомно — кожний філософ у рандомні моменти бере (або намагається узяти) і кладе виделки, причому майже незалежно одну від одної (все ж задайте тенденцію захопити дві виделки, бо інакше зголодніє). Коли у нього дві виделки, він їсть.

Чи часто філософи падають в обморок?

### Варіант 3. Пором у Парку культури та відпочинку

У Парку культури та відпочинку десантники та морські пехотинці одночасно відмічають свята відповідних родів військ. Повертатись з парку вони мають на поромі, що вміщає 2N вояків. Пором можна відправляти лише тоді, коли він повністю заповнений і на ньому або представники лише одного роду військ, або рівна кількість десантників і морських піхотинців. Якщо сили будуть нерівними, неминуча жорстока бійка. Переправа займає рандомний час. На березі вояки поводять себе культурно, бо там чергує озброєний контингент “Беркуту”, що контролює посадку на пором. Вояки підходять з парку поодинці у випадкові моменти часу і стають у чергу.

Змоделювати роботу такої переправи, створюючи для “обслуговування” кожного вояка окремий потік. Програма має синхронно (тобто, у тому ж порядку, як воно і відбувалося) писати у журнал події приходу вояків, посадки їх на пором, відправлення і повернення порому, з мітками часу і звітом, скільки яких вояків у черзі і скільки їх на поромі.