

Міністерство освіти і науки України

НТУУ «Київський політехнічний інститут»

Фізико-технічний інститут

ОПЕРАЦІЙНІ СИСТЕМИ

**Комп’ютерний практикум №8**

Варіант №11

**Виконав:**

Студент 2 курсу ФТІ

Групи ФІ-92

Поночевний Назар Юрійович

**Перевірив:**

Ільїн Костянтин Іванович

Київ – 2021

**Робота №8. Засоби синхронізації потоків**

Варіант №11

**Мета:** Оволодіння практичними навичками розроблення багатопотокових програм з підтримкою засобів синхронізації.

**Завдання для самостійної підготовки**

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

**Завдання до виконання**

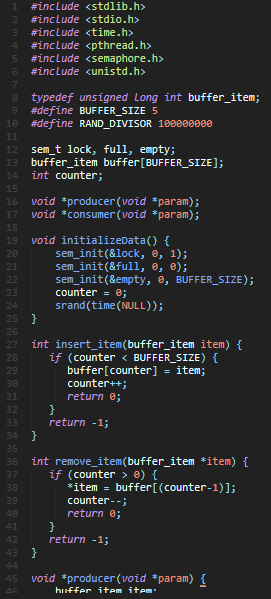
1. Розминка. Стандартна задача виробник-споживач. Задача була розглянута на лекції. Також детально розглянута в рекомендованих книжках [1, 5]. Розробіть програму, що демонструє рішення цієї задачі за допомогою семафорів. Для цього напишіть:

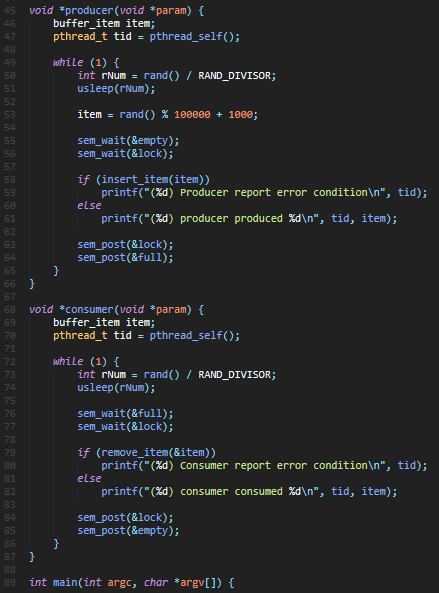
• функції виробника і споживача (наприклад, як на лекції, або як у Шеховцові, але так, щоби працювало);

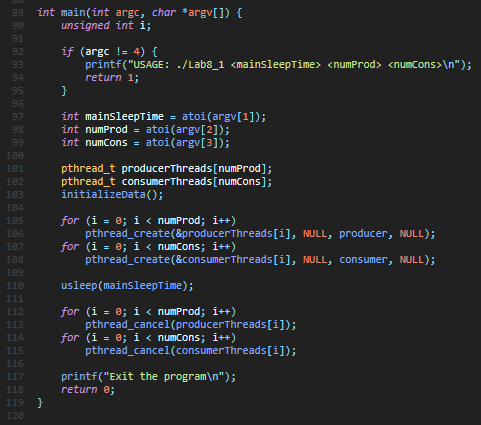
• функції створення і споживання об’єктів (рекомендується “UNIX-подібні”, є багатозадачнимистворювати” рядки тексту шляхом зчитування їх з файлу, хоча можливі й інші варіанти за вибором викладача або за вашою фантазією, наприклад розрахунки геш-функцій sha2 з рядків рандомних символів, а “UNIX-подібні”, є багатозадачнимиспоживати” 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 потоки-споживачі, 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.

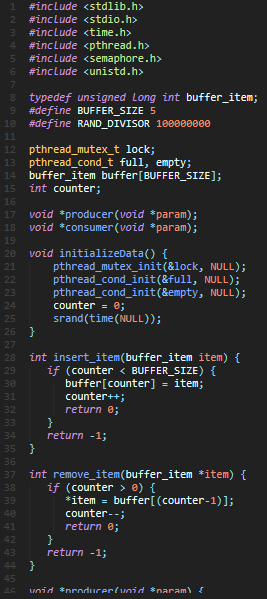


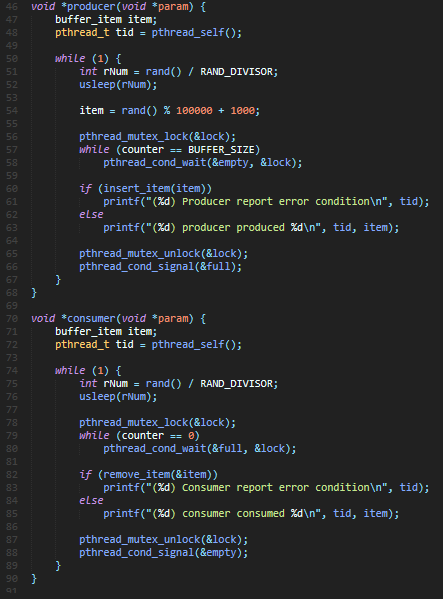




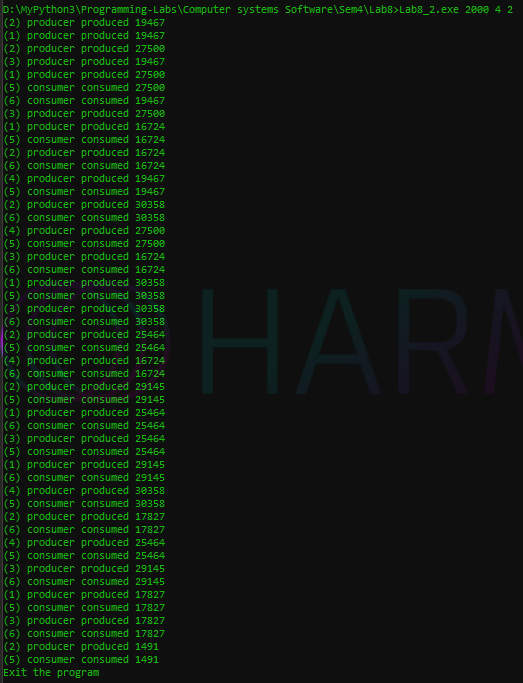


2. Продовження розминки. Теж саме, але не на семафорах, а на м’ютексі і умовних змінних. Модифікуйте програму п. 1 так, щоби використовувати м’ютекс і умовну змінну.

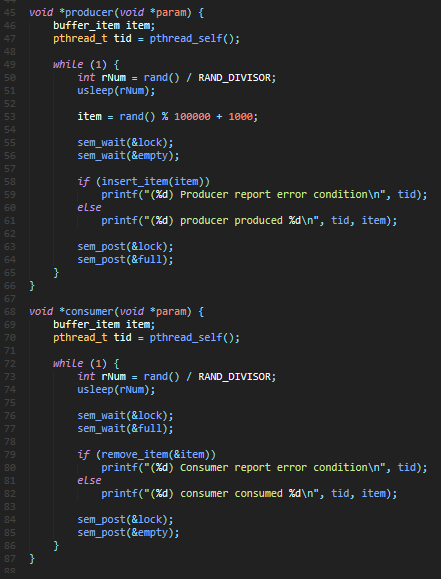


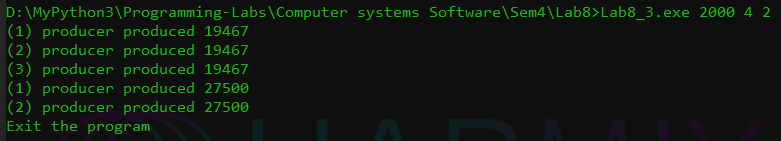






3. Продовження розминки для тих, хто шукає пригод. Взаємне блокування Модифікуйте програму п. 1 так, щоби викликати взаємне блокування. Для цього поміняйте місцями семафори. Переконайтесь у факті взаємного блокування і отримайте задоволення.





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

Варіант 11 (2 а). Філософи, 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

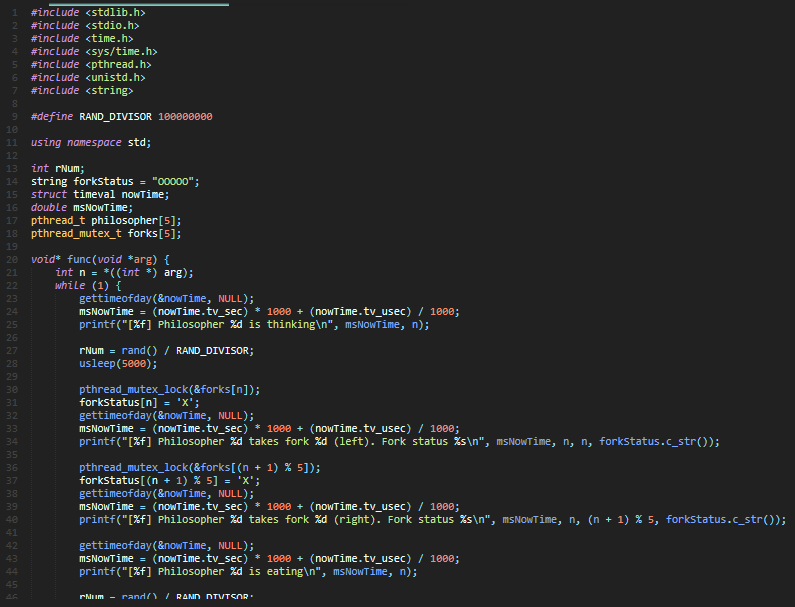
10:31:12.733 Філософ 4 узяв виделку 4 (праву). Стан виделок ХХХХХ ...

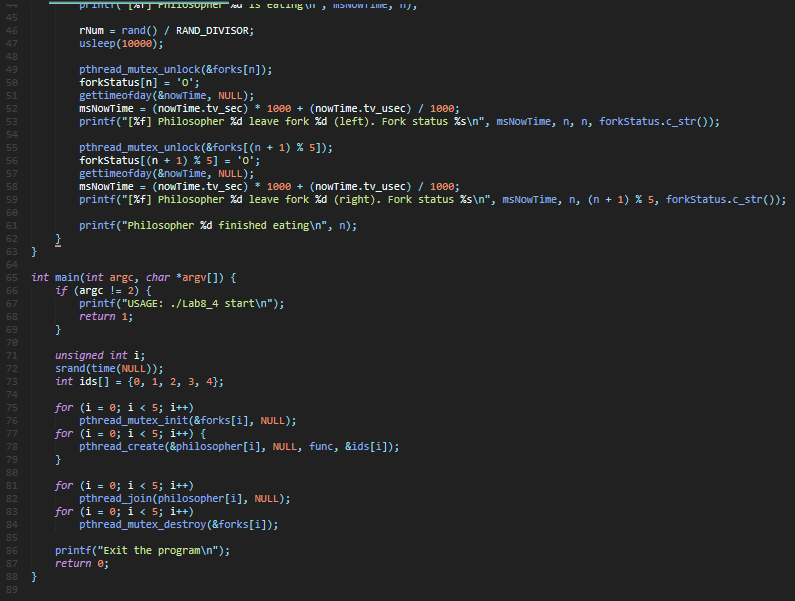
10:31:14.125 Філософ 2 закінчив їсти.

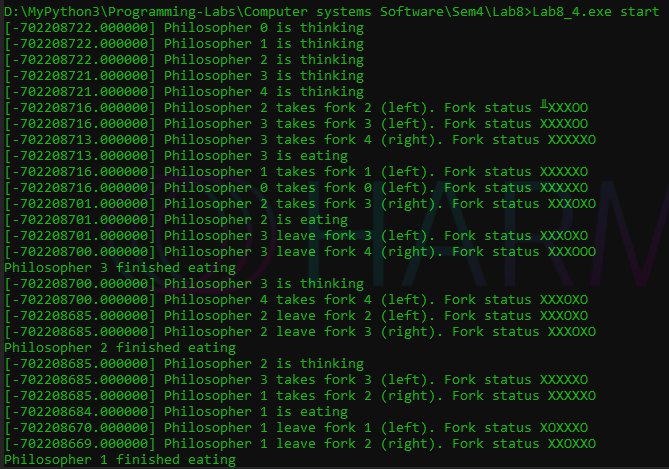
...

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

...







**Висновок**

ОС Linux має багато можливостей для управління потоками, але без методів синхронізації все це немає сенсу. На щастя, в ОС Linux також є всі відомі методи синхронізації, такі як умовна змінна, мютекси та семафори. На Windows я ще знаю є Event, Critical Section та Timer, але вони як правило додаткові. Загалом мені сподобався механізм роботи з синхронізацією. Весь процес доволі інтуїтивний та ефективно синхронізує потоки, щоб вони не заважали один одному при використанні загальних ресурсів.