7.2 ВЗАЄМНЕ ВИКЛЮЧЕННЯ ЧЕРЕЗ ЗАГАЛЬНІ ЗМІННІ

Для синхронізації потоків одного процесу прикладний програміст може використати глобальні блокуючі змінні. З цими змінними, до яких усі потоки процесу мають прямий доступ, програміст працює не звертаючись до системних викликів ОС. Кожному набору критичних даних ставиться у відповідність двійкова змінна, якій потік привласнює значення 0, коли він входить в критичну секцію, і значення 1, коли він її покидає. На рис. 7.4 показаний фрагмент алгоритму потоку, що використовує для реалізації взаємного виключення доступу до критичних даних D блокуючу змінну F(D).

Рисунок 7.4 – Реалізація критичних секцій з використанням блокуючих змінних

Перед входом в критичну секцію потік перевіряє, чи не працює вже який-небудь потік з даними D. Якщо змінна F(D) встановлена в 0, то дані зайняті, і перевірка циклічно повторюється. Якщо ж дані вільні (F = 1), то значення змінної F(D) встановлюється в 0 і потік входить в критичну секцію. Після того як потік виконає усі дії з даними D, значення змінної F(D) знову встановлюється рівним 1. Наведемо приклад лістингу програми, що використовує загальні змінні для взаємного виключення (лістинг 7.1) [13]. Введемо булеву змінну mutEx, яка повинна набувати значення true(1), якщо входження в критичну секцію заборонене, або false(0), якщо входження дозволене.

Лістинг 7.1 – Реалізація критичних секцій з використанням блокуючих змінних

```
1    static char mutEx = 0;
2    void csBegin ( void ) {
3    while ( mutEx );
4    mutEx = 1; 5 }
6    void csEnd ( void ) {
7    mutEx = 0; 8 }
```

При входженні у функцію csBegin процес потрапляє в цикл очікування (рядок 3), в якому знаходиться до тих пір, поки стан змінної виключення не дозволить йому увійти до критичної секції. Вийшовши з цього циклу, процес встановлює цю змінну в 1, забороняючи тим самим іншим процесам входити в їх критичні секції. Процес, який виконувався в критичній секції, при виході з останньої скидає змінну виключення в 0, дозволяючи цим іншим процесам входити в їх критичні секції.

Це рішення базується на неперервності доступу до пам'яті – до змінної mutEx, але воно є неправильним. Розглянемо такий випадок. Нехай процес А увійшов до своєї критичної секції і встановив mutEx=1. Поки процес А виконується всередині своєї критичної секції, два інші процеси – В і С також підійшли до своїх критичних секцій і звернулися до функції csBegin. Оскільки змінна mutEx встановлена в 1, процеси В і С зациклюються в рядку 3 коду функції csBegin. Коли процес А вийде зі своєї критичної секції і встановить mutEx=0, інший процес, наприклад В, вийде з циклу рядка 3. Але є ймовірність того, що перш, ніж процес В встигне виконати рядок

4 коду і цим заборонити вхід в критичну секцію іншим процесам, вийде з циклу рядка 3 і процес С. Таким чином, два процеси – В і С входять в критичну секцію, задача взаємного виключення не виконується.

Цей алгоритм можна удосконалити. Введемо для кожного процесу свою змінну, що відображає його знаходження в критичній секції. Ці змінні зведені в масив inside. Елемент масиву inside[i] має значення 1, якщо і-й процес знаходиться в критичній секції, і 0 – інакше (лістинг 7.2).

Лістинг 7.2 – Покращений алгоритм реалізації критичних секцій з використанням блокуючих змінних

```
1    static char inside[2] = { 0,0 };
2    void csBegin ( int proc ) {
3    int competitor;
4    competitor = other ( proc );
5    do {
6    inside[proc] = 1;
7    if ( inside[competitor] ) inside[proc] = 0;
8    } while ( ! inside[proc] ); 9 }
10   void csEnd (int proc ) {
11   inside[proc] = 0; 12 }
```

Процес встановлює свою ознаку входження (рядок 6). Але якщо він виявляє, що ознака входження конкурента теж встановлена (рядок 7), то він свою ознаку скидає. Ці дії повторюватимуться до тих пір, поки наш процес не збереже свою ознаку зведеного (рядок 8), а це можливо тільки в тому випадку, якщо ознака конкурента скинута. Це рішення не може бути прийняте ось з якої причини. Можливе таке співвідношення швидкостей процесів, при якому вони одночасно виконуватимуть рядок 7 – і одночасно скидати свої ознаки. Така «надмірна поступливість» процесів призведе до нескінченного відкладання рішення про вхід у критичну секцію.