12.6 ПЛАНУВАННЯ В WINDOWS 2000

12.6.1 Процеси і потоки

У різних ОС процеси реалізуються по-різному. Ці відмінності полягають у тому, якими структурами даних представлені процеси, як вони іменуються, якими способами захищені один від одного і які стосунки існують між ними. Процеси Windows 2000 (W2K) мають такі характерні властивості:

1. Процеси W2K реалізовані у формі об'єктів, і доступ до них здійснюється за допомогою служби об'єктів.

2. Процес W2K має багатопотокову організацію.

3. Як об'єкти-процеси, так і об'єкти-потоки мають вбудовані засоби синхронізації.

4 Менеджер процесів W2K не підтримує між процесами стосунків типу «батько-нащадок».

У будь-якій системі поняття «процес» включає:

- виконуваний код;
- власний адресний простір, який є сукупністю віртуальних адрес, які може використати процес;
- ресурси системи, такі як файли, семафори тощо, які призначені процесу операційною системою;
- хоч би один виконуваний потік.

Адресний простір кожного процесу захищений від втручання в нього будь-якого іншого процесу. Це забезпечується механізмами віртуальної пам'яті. Операційна система, звичайно, теж захищена від прикладних процесів. Щоб виконати яку-небудь процедуру ОС або прочитати що-небудь з її області пам'яті, потік повинен виконуватися в режимі ядра. Призначені для користувача процеси отримують доступ до функцій ядра за допомогою системних викликів. У призначеному для користувача режимі виконуються не лише прикладні програми, але й захищені підсистеми W2K.

У W2K процес – це просто об'єкт, що створюється і знищується менеджером об'єктів. Об'єкт-процес, як і інші об'єкти, містить заголовок, який створює і ініціалізує менеджер об'єктів. Менеджер процесів визначає атрибути, що зберігаються в тілі

об'єкта-процесу, а також забезпечує системний сервіс, який відновлює і змінює ці атрибути.

До числа атрибутів тіла об'єкта-процесу входять:

1. Ідентифікатор процесу – унікальне значення, яке ідентифікує процес у рамках операційної системи.
2. Ознака (token – токен) доступу – виконуваний об'єкт, що містить інформацію про безпеку.
3. Базовий пріоритет – основа для виконавчого пріоритету потоку процесу.
4. Процесорна сумісність – набір процесорів, на яких можуть виконуватися потоки процесу.
5. Граничні значення квот – максимальна кількість сторінкової і несторінкової системної пам'яті, дискового простору, призначеного для вивантаження сторінок, процесорного часу, які можуть бути використані процесами користувача.
6. Час виконання – загальна кількість часу, впродовж якого виконуються усі потоки процесу.

Нагадаємо, що потік є виконуваною одиницею, яка розташовується в адресному просторі процесу і використовує ресурси, виділені процесу. Подібно до процесу потік в W2K реалізований у формі об'єкта і управляється менеджером об'єктів.

Об'єкт-потік має такі атрибути:

1. Ідентифікатор клієнта – унікальне значення, яке ідентифікує потік при його зверненні до сервера.
2. Контекст потоку – інформація, яка потрібна ОС для того щоб продовжити виконання перерваного потоку. Контекст потоку містить поточний стан регістрів, стеків і індивідуальної області пам'яті, яка використовується підсистемами і бібліотеками.
3. Динамічний пріоритет – значення пріоритету потоку в даний момент.
4. Базовий пріоритет – нижня межа динамічного пріоритету потоку.
5. Процесорна сумісність потоків – перелік типів процесорів, на яких може виконуватися потік.

6. Час виконання потоку – сумарний час виконання потоку в призначеному для користувача режимі і в режимі ядра, накопичений за період існування потоку.

7. Стан попередження – прапор, який показує, що потік повинна виконувати виклик асинхронної процедури.

8. Лічильник призупинень – поточна кількість призупинень виконання потоку.

Окрім перерахованих атрибутів, є і деякі інші атрибути. Як видно з переліку, багато атрибутів об'єкта-потоку аналогічні атрибутам об'єкта-процесу. Схожі і сервісні функції, які можуть бути виконані над об'єктами-процесами і об'єктами-потоками: створення, відкриття, завершення, призупинення, запит і установка інформації, запит і установка контексту та інші функції.

12.6.2 Планування процесів і потоків

Планування в Windows здійснюється на рівні потоків, а не процесів. Це здається зрозумілим, оскільки самі процеси не виконуються, а лише надають ресурси і контекст для виконання потоків. Тому при плануванні потоків система не звертає уваги на те, якому процесу вони належать. Наприклад, якщо процес А має 10 готових до виконання потоків, а процес В – два, і усі 12 потоків мають однаковий пріоритет, кожен з потоків отримає 1/12 процесорного часу.

У W2K реалізована витісняюча багатозадачність, при якій операційна система не чекає, коли потік сам захоче звільнити процесор, а примусово знімає його з виконання після того, як той витратив відведений йому час (квант), або якщо в черзі готових з'явився потік з вищим пріоритетом. При такій організації розподілу процесора жоден потік не займе процесор на дуже довгий час. В ОС W2K потік у ході свого існування може мати один з шести станів (рис. 12.19).

Життєвий цикл потоку починається в той момент, коли програма створює новий потік. Менеджер процесів виділяє пам'ять для об'єкта-потоку і звертається до ядра, щоб ініціалізувати об'єкт-потік ядра. Після ініціалізації потік проходить через наведені нижче стани:

Готовність. При пошуку потоку на виконання диспетчер переглядає тільки потоки, що знаходяться в стані готовності, в яких є все для виконання, але бракує тільки процесора.

Першочергова готовність (резервний). Для кожного процесора системи вибирається один потік, який виконуватиметься наступним (найперший потік в черзі). Коли умови дозволяють, відбувається перемикання на контекст цього потоку.

Виконання. Як тільки відбувається перемикання контекстів, потік переходить в стан виконання і знаходиться в ньому до тих пір, поки або ядро не витіснить його через те, що з'явився пріоритетніший потік або закінчився квант часу, виділений цьому потоку, або потік завершиться взагалі, або він за власною ініціативою перейде в стан очікування.

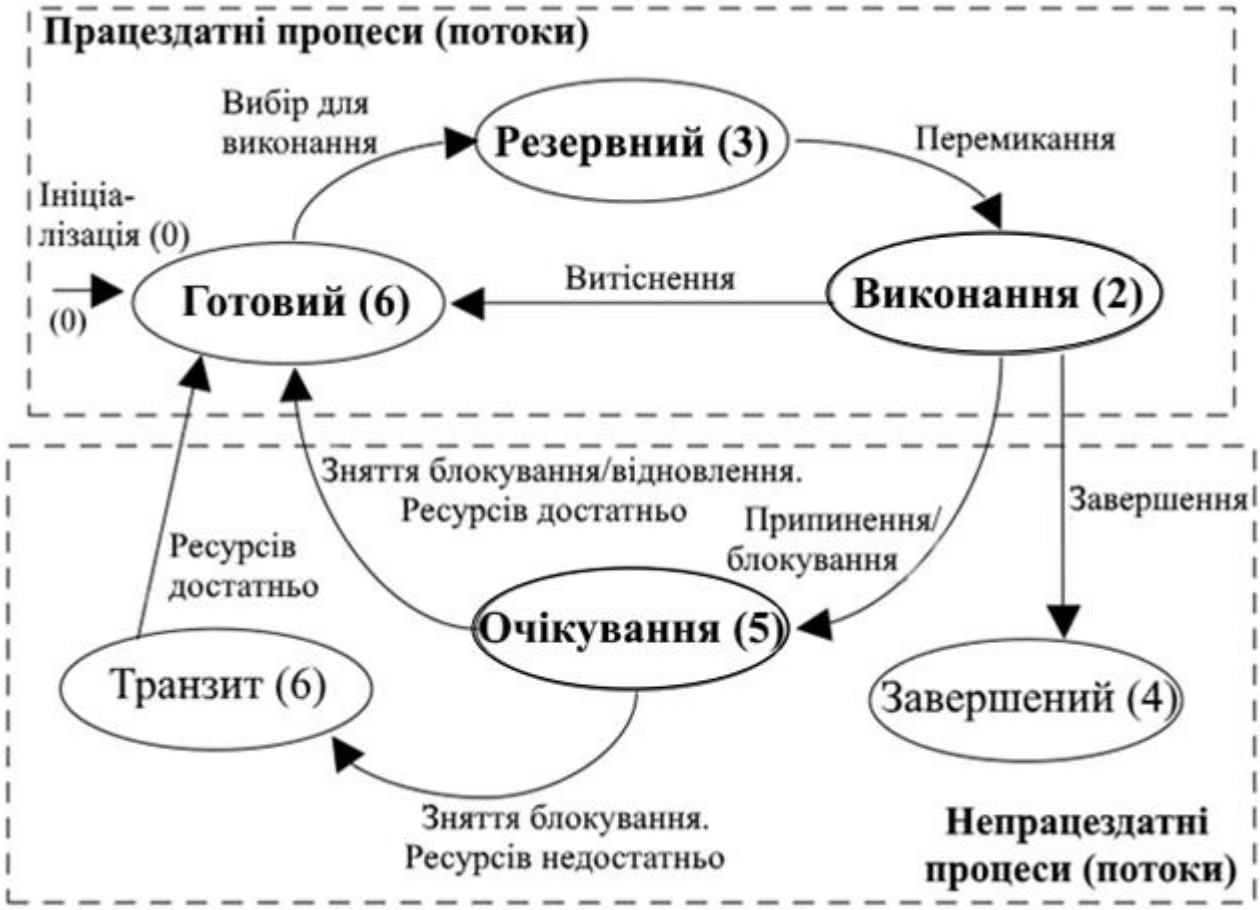

Рисунок 12.19 – Граф станів потоків в Windows

Очікування. Потік може входити в стан очікування декількома способами: потік за своєю ініціативою чекає деякий об'єкт для того, щоб синхронізувати своє виконання; операційна система (наприклад, підсистема введення-виведення) може чекати в інтересах потоку; підсистема оточення може безпосередньо змусити потік призупинити себе. Коли очікування потоку добіжить кінця, він повертається в стан готовності.

Перехідний стан (транзит). Потік входить в перехідний стан, якщо він готовий до виконання, але ресурси, які йому потрібні, зайняті. Наприклад, сторінка, що містить стек потоку, може бути вивантажена з оперативної пам'яті на диск. При звільненні ресурсів потік переходить в стан готовності.

Завершення. Коли виконання потоку закінчилося, він входить в стан завершення. Знаходячись в цьому стані, потік може бути або видалений, або не видалений. Це залежить від алгоритму роботи менеджера об'єктів, відповідно до якого він і вирішує, коли видаляти цей об'єкт.

Для визначення порядку виконання потоків диспетчер ядра використовує алгоритм, заснований на пріоритетах, відповідно до якого кожному потоку привласнюється число – пріоритет, і потоки з вищим пріоритетом виконуються раніше потоків з нижчим пріоритетом.

На початку потік отримує пріоритет від процесу, який його створює. У свою чергу, процес отримує пріоритет у той момент, коли його створює підсистема того або іншого прикладного середовища. Значення базового пріоритету надається процесу системою за умовчанням або системним адміністратором. Потік наслідує цей базовий пріоритет і може змінити його, трохи збільшивши або зменшивши. У ході виконання пріоритет планування може мінятися.

У Windows 2000 пріоритети організовані у вигляді двох груп, або класів: реального часу і динамічні. Кожна з груп складається з 16 рівнів пріоритетів (рис. 12.20). Потоки, що вимагають негайної уваги, знаходяться в класі реального часу, який включає функції здійснення комунікацій і задачі реального часу. Інші потоки потрапляють в клас потоків зі змінними (призначеними для користувача) пріоритетами.

Кожного разу, коли необхідно вибрати потік для виконання, диспетчер переглядає чергу готових потоків реального часу і звертається до інших потоків тільки тоді, коли черга потоків реального часу порожня. Більшість потоків в системі потрапляють в клас потоків зі змінними пріоритетами, діапазон пріоритетів яких від 0 до 15. Цей клас має назву «Змінні пріоритети» (динамічні пріоритети потоку) тому, що диспетчер настроює систему, вибираючи (знижуючи або підвищуючи) пріоритети потоків цього класу.

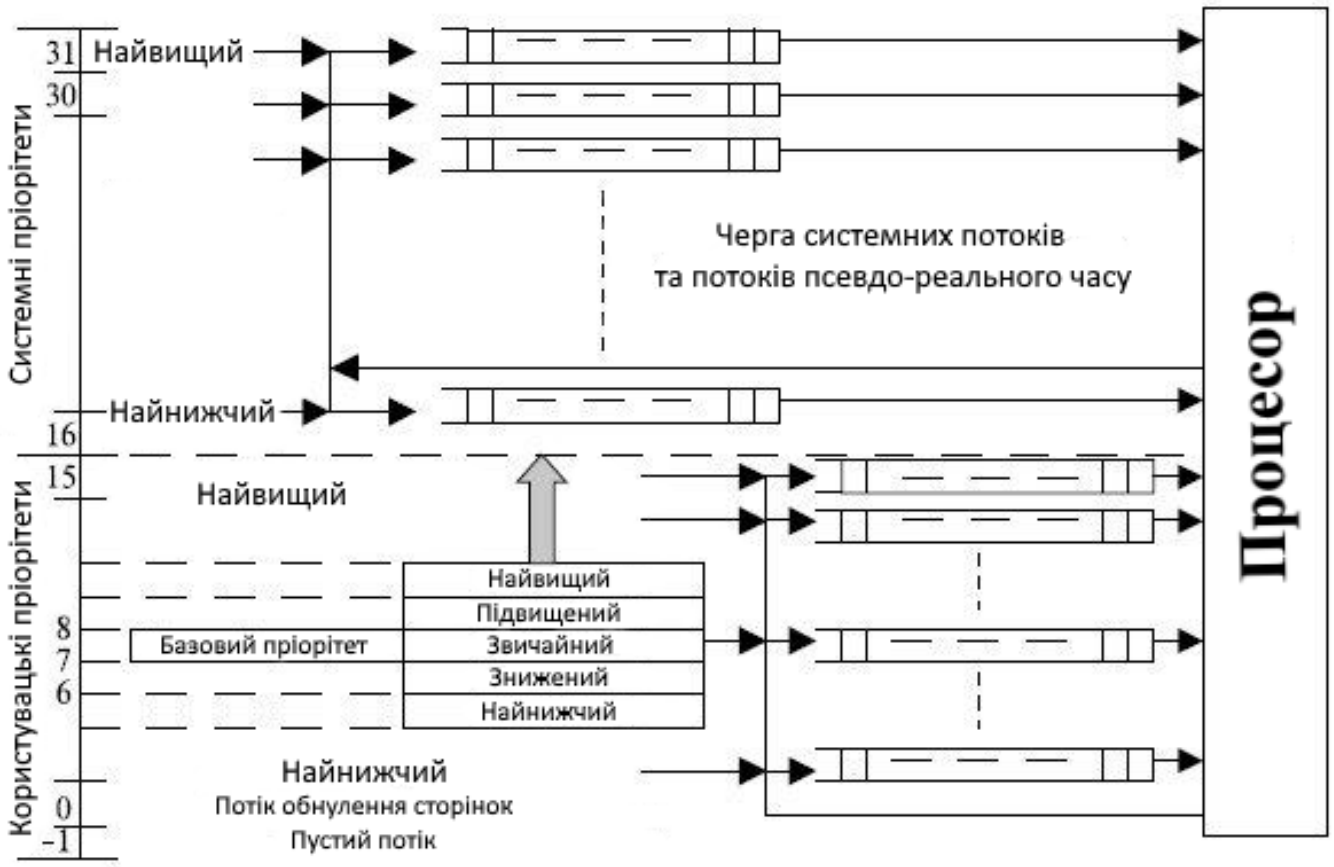

Рисунок 12.20 – Планування в Window

Алгоритм планування потоків в W2K об'єднує в собі обидві базові концепції – квантування і пріоритети. Як і в усіх інших алгоритмах, заснованих на квантуванні, кожному потоку надається квант часу, впродовж якого він може виконуватися.

Наприклад, для Win2000 Professional початкове значення кванта дорівнює 6, а для Win2000 Server – 36. Всякий раз, коли виникає переривання від таймера, із кванта потоку віднімається 3, і так до тих пір, поки він не досягне нуля. Частота спрацьовування таймера залежить від апаратної платформи. Наприклад, для більшості однопроцесорних x86 систем він складає 10 мс, а на більшості багатопроцесорних x86 систем – 15 мс.

Потік звільняє процесор, якщо:

- блокується, йдучи в стан очікування;
- завершується;
- вичерпаний квант;
- у черзі готових потоків з'являється пріоритетніший потік.

Використання динамічних пріоритетів, що змінюються в часі, дозволяє реалізувати адаптивне планування, при якому не дискримінуються інтерактивні задачі, що часто виконують операції введення-виведення, і які не використовують повністю виділені їм кванти часу. Якщо потік повністю вичерпав свій квант, то його пріоритет знижується на деяку величину. В той же час пріоритет потоків, які перейшли в стан очікування, не використавши повністю виділений їм квант часу, підвищується. Пріоритет не змінюється, якщо потік витіснений пріоритетнішим потоком.

Для того щоб забезпечити хороший час реакції системи, алгоритм планування використовує разом з квантуванням концепцію абсолютних пріоритетів. Відповідно до цієї концепції при появі в черзі готових потоків, в яких пріоритет вищий, ніж у того, що виконується в даний момент, відбувається зміна активного потоку на потік з найвищим пріоритетом.

Прив'язка до процесорів. Якщо операційна система виконується на машині, де встановлено більше одного процесор, то за умовчанням потік виконується на будь-якому доступному процесорі. Проте в деяких випадках набір процесорів, на яких потік може виконуватись, може бути обмежений. Це явище називається прив'язкою до процесорів (processor affinity). Можна змінити прив'язку до процесорів програмно, через Win32-функції планування.

У багатопроцесорній системі з N процесорами завжди активні потоки з найвищими пріоритетами. З іншими потоками з нижчим пріоритетом N-1 працює єдиний процесор, що залишився.

У багатопроцесорних системах при диспетчеризації і плануванні потоків основною характеристикою є їх процесорна сумісність. Після того як ядро вибрало потік з найвищим пріоритетом, воно перевіряє, який процесор може виконати цей потік. Якщо атрибут потоку «процесорна сумісність» не дозволяє потоку виконуватися ні на одному з вільних процесорів, то вибирається наступний в порядку пріоритетів потік.

При роботі W2K в системі з одним процесором потік з найвищим пріоритетом завжди активний, якщо тільки не чекає настання якої-небудь події. Якщо є декілька

потоків з найвищим пріоритетом, то згідно з описаною схемою процесор працює з ними з використанням кругового планування.

Пам'ять. Кожному процесу в Win32 доступний лінійний 4-гігабайтний (232 = 4 294 967 296) віртуальний адресний простір. Зазвичай верхня половина цього простору резервується за операційною системою, а друга половина доступна процесу.

Віртуальний адресний простір процесу доступний усім потокам цього процесу. Іншими словами, усі потоки одного процесу виконуються в єдиному адресному просторі. З іншого боку, механізм віртуальної пам'яті дозволяє ізолювати процеси один від одного. Потоки одного процесу не можуть посилатися на адресний простір іншого процесу.

Оскільки далеко не кожен комп'ютер в змозі виділити по 4 Гб фізичної пам'яті на кожен процес, використовується механізм підкачування (swapping). Коли оперативної пам'яті бракує, операційна система переміщає частину вмісту пам'яті на диск, у файл (swap file або page file), звільняючи, таким чином, фізичну пам'ять для інших процесів. Коли потік звертається до сторінки віртуальної пам'яті, записаної на диск, диспетчер віртуальної пам'яті завантажує цю інформацію з диска назад в пам'ять.

Створення процесів. Створення Win32 процесу здійснюється викликом однієї з таких функцій, як CreateProcess, CreateProcessAsUser (для Win NT/2000) і CreateProcessWithLogonW (починаючи з Win2000) і відбувається в декілька етапів.

1. Відкривається файл образу (.exe), який виконуватиметься в процесі. Якщо виконуваний файл не є Win32 додатком, то шукається образ підтримки (support image) для запуску цієї програми. Наприклад, якщо виконується файл з розширенням .bat, запускається cmd.exe тощо.
2. Створюється об'єкт Win32 «процес».
3. Створюється первинний потік (стек, контекст і об'єкт «потік»).
4. Підсистема Win32 повідомляється про створення нового процесу і потоку.
5. Починається виконання первинного потоку.
6. У контексті нового процесу і потоку ініціалізувався адресний простір (наприклад, завантажуються необхідні dll-файли) і починається виконання програми.

Завершення процесів. Процес завершується якщо:

- вхідна функція первинного потоку повернула управління;
- один з потоків процесу викликав функцію ExitProcess;
- потік іншого процесу викликав функцію TerminateProcess.

Створення потоків. Первинний потік створюється автоматично при створенні процесу. Інші потоки створюються функціями CreateThread і CreateRemoteThread (тільки у Win NT/2000/XP).

Завершення потоків. Потік завершується якщо:

- функція потоку повертає управління;
- потік самознищується, викликавши ExitThread;
- інший потік цього або стороннього процесу викликає TerminateThread;
- завершується процес, що містить цей потік.

Об'єкти ядра. Ці об'єкти використовуються системою і додатками користувача для управління різними ресурсами: процесами, потоками, файлами тощо. Windows дозволяє створювати і оперувати з декількома типами таких об'єктів.

Об'єкт ядра – це, по суті, структура, створена ядром і доступна тільки йому. У додаток користувача передається тільки описувач (handle) об'єкта, а управляти об'єктом ядра можна за допомогою функцій Win32 API.

Wait–функції. Роботу потоку можна призупинити. Для цього існує багато способів. Ось деякі з них.

Функція Sleep() призупиняє роботу потоку на задане число мілісекунд. Якщо в якості аргументу вказати 0 ms, то станеться наступне. Потік відмовиться від свого кванта процесорного часу, проте тут же з'явиться в списку потоків готових до виконання. Іншими словами станеться навмисне перемикання потоків, вірніше сказати, спроба перемикання. Адже наступним для виконання потоком цілком може стати той же самий потік.

Функція WaitForSingleObject() призупиняє виконання потоку до тих пір, поки не станеться одна з двох подій:

- закінчиться таймаут очікування;
- очікуваний об'єкт перейде в сигнальний стан.

За поверненим значенням можна зрозуміти, яка з двох подій сталася. Чекати за допомогою wait-функцій може більшість об'єктів ядра, наприклад, об'єкт «процес» або «потік», щоб визначити, коли вони завершать свою роботу.

Функції WaitForMultipleObjects передається відразу масив об'єктів. Можна чекати спрацьовування відразу всіх об'єктів або якогось одного з них.

12.6.3 Синхронізація потоків

Працюючи паралельно, потоки спільно використовують адресний простір процесу. Також усі вони мають доступ до описувачів відкритих у процесі об'єктів. Якщо декілька потоків одночасно звертаються до одного ресурсу або необхідно якось упорядкувати роботу потоків, то для цього використовують об'єкти синхронізації і відповідні механізми.

М’ютекси (Mutex) – це об'єкти ядра, які створюються функцією CreateMutex(). М’ютекс буває в двох станах – зайнятому і вільному. М’ютексом добре захищати одиничний ресурс від одночасного звернення до нього різних потоків.

Семафори (Semaphore) створюється функцією CreateSemaphore(). Він дуже схожий на м’ютекс, тільки на відміну від нього у семафора є лічильник. Семафор відкритий якщо лічильник більше 0 і закритий, якщо лічильник дорівнює 0.

Події (Event) як і м’ютекси мають два стани – встановлений і скинутий. Події бувають із скиданням вручну і з автоскиданням. Коли потік дочекався (wait-функція повернула управління) події з автоскиданням, така подія автоматично скидається. Інакше подію треба скидати вручну, викликавши функцію ResetEvent(). Припустимо, що відразу декілька потоків чекають однієї і тієї ж події, і подія спрацювала. Якщо це була подія з автоскиданням, то вона дозволить працювати тільки одному потоку (адже відразу ж після повернення з його wait-функції подія скинеться автоматично), а інші потоки залишаться чекати. Якщо ж це була подія із скиданням вручну, то усі потоки отримають управління, а подія так і залишиться у встановленому стані, поки який-небудь потік не викличе ResetEvent().

Критичні секції. Синхронізація в режимі користувача. Критична секція гарантує вам, що спеціальні ділянки коду програми не виконуватимуться одночасно. Строго кажучи, критична секція не є об'єктом ядра. Вона є структурою, що містить декілька прапорів і якийсь (не важливо) об'єкт ядра. При вході в критичну секцію

спочатку перевіряються прапори, і якщо з'ясовується, що вона вже зайнята іншим потоком, то виконується звичайна wait-функція. Критична секція примітна тим, що для перевірки, чи зайнята вона чи ні, програма не переходить в режим ядра (не виконується wait-функція), а лише перевіряються прапори. Через це вважається, що синхронізація за допомогою критичних секцій найшвидш а. Таку синхронізацію називають «синхронізація в режимі користувача».

Синхронізація процесів. Описувачі об'єктів ядра залежні від конкретного процесу. Існують способи роботи з одними і тими ж об'єктами ядра різними процесами.

По-перше, це спадок описувача. При створенні об'єкта можна вказати чи буде його описувач наслідувати дочірні (породжені цим процесом) процеси.

По-друге, дублювання описувача. Функція DuplicateHandle дублює описувач об'єкту одного процесу в іншій. Тобто, по суті, бере запис в таблиці описувачів одного процесу і створює його копію в таблиці іншого.

І, нарешті, іменування об'єкту ядра. При створенні об'єкта ядра для синхронізації (м’ютекса, семафора, очікуваного таймера або події) можна задати його ім'я. Воно має бути унікальним в системі. Тоді інший процес може відкрити цей об'єкт ядра, вказавши у функції Open(OpenMutex, OpenSemaphore, OpenWaitableTimer, OpenEvent) це ім'я. Насправді, при виклику функції Create...() система спочатку перевіряє, чи не існує вже об'єкт ядра з таким іменем. Якщо ні, то створюється новий об'єкт. Якщо так, ядро перевіряє тип цього об'єкту і права доступу. Якщо типи не співпадають або ж процес не має повних прав на доступ до об'єкта, виклик Create…() функції закінчується невдало і повертається NULL. Якщо все нормально, то просто створюється новий описувач (handle) існуючого вже об'єкту ядра. За кодом повернення функції GetLastError() можна зрозуміти, що сталося: створився новий об'єкт або Create() повернула вже існуючий.

Тому синхронізувати потоки всередині різних процесів можна так як і в межах одного. Треба тільки правильно передати описувач синхронізуючого об'єкта від одного процесу до іншого будь-яким з перелічених вище способів.

12.6.4 Взаємодія між процесами

Потоки одного процесу не мають доступу до адресного простору іншого процесу. Однак існують механізми для передачі даних між процесами.

Колективна пам'ять. Як уже говорилося, система віртуальної пам'яті в Win32 використовує файл підкачки – swap file (або файл розміщення – page file), маючи можливість перетворення сторінок оперативної пам'яті в сторінки файлу на диску і навпаки. Система може проектувати на оперативну пам'ять не тільки файл розміщення, а й будь-який інший файл. Додатки можуть використовувати цю можливість. Це може використовуватися для забезпечення швидшого доступу до файлів, а також для спільного використання пам'яті.

Такі об'єкти називаються проекціями файлів на оперативну пам'ять (file-mapping object). Для створення проекції файлу спочатку викликається функція CreateFileMapping(). Їй передається дескриптор вже відкритого файлу або вказується, що потрібно використовувати page file операційної системи. Крім цього, в параметрах їй передається прапор захисту, максимальний розмір проекції і ім'я об'єкта. Потім викликається функція MapViewOfFile(). Вона відображає уявлення файлу (view of a file) в адресний простір процесу. Після закінчення роботи викликається функція UnmapViewOfFile(). Вона звільняє пам'ять і записує дані у файл (якщо це не файл підкачки). Щоб записати дані на диск негайно, використовується функція FlushViewOfFile(). Проекція файлу, як і інші об'єкти ядра, може використовуватися іншими процесами через успадкування, дублювання дескриптора або за іменем.

Інші механізми (сокети, англ. pipe). Крім колективної пам'яті, в Windows є й інші способи передачі інформації між процесами, наприклад, канали, іменовані канали та сокети. Усі вони мають подібний принцип і являють собою своєрідний канал або з'єднання, «трубу», що сполучає процеси. Програма, маючи один кінець такого з'єднання, може читати і/або писати в нього дані, обмінюючись інформацією з програмою на іншому кінці.

Канали використовуються для пересилання даних в одному напрямку між дочірнім та батьківським процесами або між двома дочірніми процесами. Операції читання/запису в канал схожі на подібні операції при роботі з файлами.

Іменовані канали використовуються для двостороннього обміну даними між процесом-сервером і одним або декількома процесами-клієнтами. Як і анонімні канали, вони використовують файлоподібний інтерфейс, але, на відміну від перших, придатні також для обміну даними мережею.

Сокет – це абстрактний об'єкт для позначення одного з кінців мережевого з'єднання, в тому числі і через Internet. Сокети Windows бувають двох типів: сокети дейтаграм і сокети потоків. Інтерфейс Windows Sockets (WinSock) заснований на BSD-версії сокетів, але в ньому є також розширення, специфічні для Windows.

Повідомлення в Windows (віконні повідомлення). Говорячи про Windows не можна не згадати про такі поняття як windows (вікна), messages (повідомлення), message queue (черга повідомлень) тощо.

Window – це прямокутна область екрану, в якій додаток відображає інформацію і отримує інформацію від користувача. Вікна належать потокам. Потік, який створив вікно вважається власником цього вікна. Потік може бути власником кількох вікон.

Вікна управляються повідомленнями. Всі події, що відбуваються з вікном, супроводжуються посилкою йому повідомлень: створення та знищення вікна, введення з клавіатури, переміщення миші, перемалювання і переміщення вікна тощо. Повідомлення вікну можуть надсилатися як самою системою, так і додатками користувача. Кожному вікну приписана функція, яка називається віконною процедурою (window procedure), і яка викликається при обробці повідомлення.

Повідомлення можна надсилати не тільки вікнам, а й самому потоку. Кожен потік, який володіє вікном, має чергу повідомлень. Як правило, потік, що володіє вікнами, тільки тим і займається, що обробляє повідомлення, що посилаються його вікнам.