ЛЕКЦІЯ 3. ПРОЦЕСИ І ПОТОКИ В ОС

План

3.1. Визначення процесу та потоку
3.2. Реалізація та використання моделі процесів і багатопотоковості
3.3. Стани процесів і потоків
3.4. Опис процесів і потоків
3.5. Перемикання контексту й обробка переривань
3.6. Керування процесами в UNIX
3.7. Керування процесами та потоками у Windows XP

3.1. Визначення процесу та потоку

Під *процесом* розуміють абстракцію ОС, яка об'єднує все необхідне для виконання однієї програми в певний момент часу.

Для успішного виконання програми потрібні певні ресурси. До них належать:
-ресурси, необхідні для послідовного виконання програмного коду (передусім процесорний час);
-ресурси, що дають можливість зберігати інформацію, яка забезпечує виконання програмного коду (регістри процесора, оперативна пам'ять тощо).

Ці групи ресурсів визначають дві складові частини процесу:

- послідовність виконуваних команд процесора;

- набір адрес пам'яті (адресний простір), у якому розташовані ці команди і дані для них.

Виділення цих частин виправдане ще й тим, що в рамках одного адресного простору може бути кілька паралельно виконуваних послідовностей команд, що спільно використовують одні й ті самі дані. Необхідність розмежування послідовності команд і адресного простору підводить до поняття *потоку*.

Потоком (потік керування, нитка, thread) називають набір послідовно виконуваних команд процесора, які використовують загальний адресний простір процесу.

Оскільки в системі може одночасно бути багато потоків, завданням ОС є організація перемикання процесора між ними і планування їхнього виконання.

У багатопроцесорних системах код окремих потоків може виконуватися на окремих процесорах.

Тепер можна дати ще одне означення процесу.

Процесом називають сукупність одного або декількох потоків і захищеного адресного простору, у якому ці потоки виконуються.

3.2. Реалізація та використання моделі процесів і багатопотоковості

Максимально можлива кількість процесів (захищених адресних просторів) і потоків, які в них виконуються, може варіюватися в різних системах.

В *однозадачних* системах є тільки один адресний простір, у якому в кожен момент часу може виконуватися один потік.

У більшості сучасних ОС (таких, як лінія Windows XP, сучасні версії UNIX) може бути багато процесів, а в адресному просторі кожного процесу — багато потоків. Ці системи підтримують багатопотоковість, а процес у такій системі називають *багатопотоковим* процесом.

Складові елементи процесів і потоків

До елементів *процесу* належать:
- захищений адресний простір;
- дані, спільні для всього процесу (ці дані можуть спільно використовувати всі його потоки);
- інформація про використання ресурсів (відкриті файли, мережні з'єднання тощо);
- інформація про потоки процесу.

Потік містить такі елементи:

- стан процесора (набір поточних даних із його регістрів), зокрема лічильник поточної інструкції процесора;

- стек потоку (ділянка пам'яті, де перебувають локальні змінні потоку й адреси повернення функцій, що викликані у його коді).

Багатопотоковість

Використання декількох потоків у застосуванні означає внесення в нього паралелізму (concurrency).

Паралелізм — це одночасне (з погляду прикладного програміста) виконання дій різними фрагментами коду застосування.

Така одночасність може бути реалізована на одному процесорі шляхом перемикання задач (випадок: псевдопаралелізму), а може ґрунтуватися на паралельному виконанні коду на декількох процесорах (випадок справжнього паралелізму).

Можна виділити такі основні види паралелізму:

- паралелізм багатопроцесорних систем;
- паралелізм операцій введення-виведення;
- паралелізм взаємодії з користувачем;
- паралелізм розподілених систем.

Переваги і недоліки багатопотоковості

Назвемо проблеми, які можуть бути вирішені за допомогою потоків.

1. Використання потоків дає змогу реалізувати різні види паралелізму і дозволяє застосуванню масштабуватися із ростом кількості процесорів.

2. Для підтримки потоків потрібно менше ресурсів, ніж для підтримки процесів (наприклад, немає необхідності виділяти для потоків адресний простір).

3. Для обміну даними між потоками може бути використана спільна пам'ять (адресний простір їхнього процесу). Це ефективніше, ніж застосовувати засоби міжпроцесової взаємодії.

Незважаючи на перелічені переваги, використання потоків не є універсальним засобом розв'язання проблем паралелізму, і пов'язане з деякими труднощами.

1. Розробляти і налагоджувати багатопотокові програми складніше, ніж звичайні послідовні програми; досить часто впровадження багатопотоковості призводить до зниження надійності застосувань. Організація спільного використання адресного простору декількома потоками вимагає, щоб програміст мав високу кваліфікацію.

2. Використання потоків може спричинити зниження продуктивності застосувань. Переважно це трапляється в однопроцесорних системах.

3.3. Створення та завершення процесів і потоків

Для потоку дозволені такі стани (рис. 3.1):

- *створення* (new) — потік перебуває у процесі створення;
- *виконання* (running) — інструкції потоку виконує процесор (у конкретний момент часу на одному процесорі тільки один потік може бути в такому стані);
- *очікування (waiting)* — потік очікує деякої події (наприклад, завершення операції введення-виведення); такий стан називають також заблокованим, а потік — припиненим;
- *готовність (ready)* — потік очікує, що планувальник перемкне процесор на нього, при цьому він має всі необхідні йому ресурси, крім процесорного час;
- *завершення (terminated)* — потік завершив виконання (якщо при цьому його ресурси не були вилучені з системи, він переходить у додатковий стан).

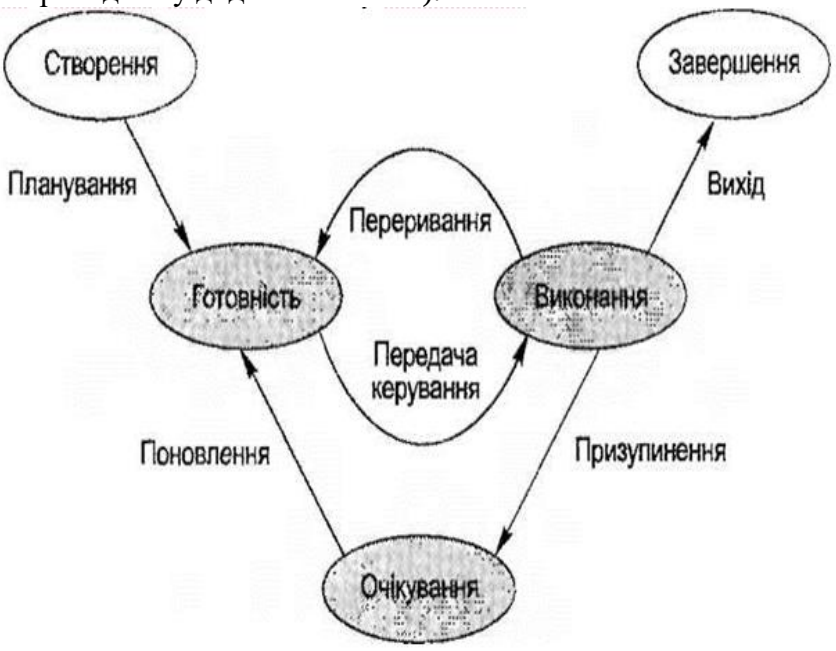

Рис. 3.1. Стани потоків

Перехід потоків між станами очікування і готовності реалізовано на основі планування задач, або планування потоків.

3.4. Опис процесів і потоків

Для керування розподілом ресурсів ОС повинна підтримувати структури даних, які містять інформацію, що описує процеси, потоки і ресурси. До таких структур даних належать:

- таблиці розподілу ресурсів: таблиці пам'яті, таблиці введення-виведення, таблиці файлів тощо;
- таблиці процесів і таблиці потоків, де міститься інформація про процеси і потоки, присутні у системі в конкретний момент.

Керуючі блоки процесів і потоків

Інформацію про процеси і потоки в системі зберігають у спеціальних структурах даних, які називають керуючими блоками процесів і керуючими блоками потоків. Ці структури дуже важливі для роботи ОС, оскільки на підставі їхньої інформації система здійснює керування процесами і потоками.

Керуючий блок потоку (Thread Control Block, TCB) відповідає активному потоку, тобто тому, який перебуває у стані готовності, очікування або виконання. Цей блок може містити таку інформацію:

- ідентифікаційні дані потоку (зазвичай його унікальний ідентифікатор);
- стан процесора потоку: користувацькі регістри процесора, лічильник інструкцій, покажчик на стек;
- інформацію для планування потоків.

Таблиця потоків — це зв'язний список або масив керуючих блоків потоку. Вона розташована в захищеній області пам'яті ОС.

Керуючий блок процесу (Process Control Block, PCB) відповідає процесу, що присутній у системі. Такий блок може містити:

- ідентифікаційні дані процесу (унікальний ідентифікатор, інформацію про інші процеси, пов'язані з даним);
- інформацію про потоки, які виконуються в адресному просторі процесу (наприклад, покажчики на їхні керуючі блоки);
- інформацію, на основі якої можна визначити права процесу на використання різних ресурсів (наприклад, ідентифікатор користувача, який створив процес);
- інформацію з розподілу адресного простору процесу;
- інформацію про ресурси введення-виведення та файли, які використовує процес.

Таблицю процесів організовують аналогічно до таблиці потоків. Як елементи в ній зберігатимуться керуючі блоки процесів.

Образи процесу і потоку

Сукупність інформації, що відображає процес у пам'яті, називають образом процесу (process image), а всю інформацію про потік — образом потоку (thread image). До образу процесу належать:

- керуючий блок процесу;
- програмний код користувача;
- дані користувача (глобальні дані програми, загальні для всіх потоків);
- інформація образів потоків процесу.

Програмний код користувача, дані користувача та інформація про потоки завантажуються в адресний простір процесу. Образ процесу звичайно не є безперервною ділянкою пам'яті, його частини можуть вивантажуватися на диск.

Схема розташування в пам'яті образів процесу і його потоків зображена на рис. 3.3. Усі потоки конкретного процесу можуть користуватися загальною інформацією його образу.

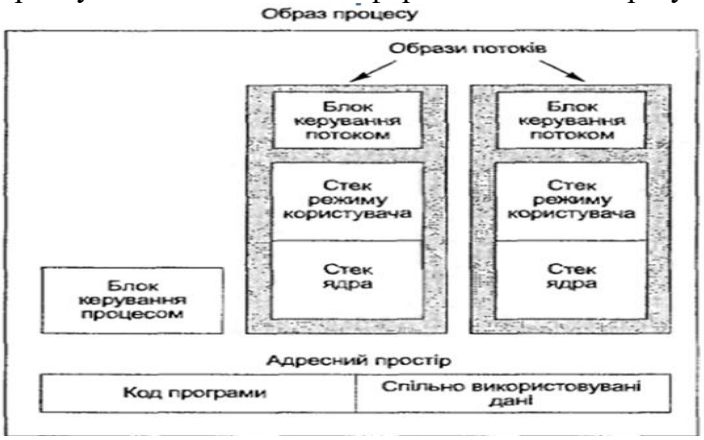

Рис. 3.3. Образи процесу і його потоків

3.5. Перемикання контексту й обробка переривань

3.5.1. Організація перемикання контексту

Найважливішим завданням операційної системи під час керування процесами і потоками є організація перемикання контексту — передачі керування від одного потоку до іншого зі збереженням стану процесора.

Загальних принципів перемикання контексту дотримуються у більшості систем, але їхня реалізація обумовлена конкретною архітектурою. Звичайно потрібно виконати такі операції:

- зберегти стан процесора потоку в деякій ділянці пам'яті (області зберігання стану процесора потоку);
- визначити, який потік слід виконувати наступним;
- завантажити стан процесора цього потоку із його області зберігання;
- продовжити виконання коду нового потоку.

Перемикання контексту звичайно здійснюється із залученням засобів апаратної підтримки. Можуть бути використані спеціальні регістри та ділянки пам'яті, які дають можливість зберігати інформацію про поточну задачу, а також спеціальні інструкції процесора для роботи з цими регістрами та ділянками пам'яті.

3.5.2. Обробка переривань

У процесі виконання потік може бути перерваний не лише для перемикання контексту на інший потік, але й у зв'язку із програмним або апаратним перериванням. Із кожним перериванням надходить додаткова інформація (наприклад, його номер). На підставі цієї інформації система визначає, де буде розміщена адреса процедури оброблювача переривання (список таких адрес зберігають у спеціальній ділянці пам'яті і називають вектором переривань).

Наведемо приклад послідовності дій під час обробки переривання:

- збереження стану процесора потоку;
- встановлення стека оброблювача переривання;
- початок виконання оброблювача переривання; для цього з вектора переривання завантажується нове значення лічильника команд;
- відновлення стану процесора потоку після закінчення виконання оброблювача і продовження виконання потоку.

3.6. Керування процесами в UNIX і Linux

Образ процесу

В UNIX-системах образ процесу містить такі компоненти:

- керуючий блок процесу;
- код програми, яку виконує процес;
- стек процесу, де зберігаються тимчасові дані (параметри процедур, повернені значення, локальні змінні тощо);
- глобальні дані, спільні для всього процесу.

Для кожного компонента образу процесу виділяють окрему ділянку пам'яті.

Ідентифікаційна інформація та атрибути безпеки процесу

Із кожним процесом у системі пов'язана ідентифікаційна інформація.

Ідентифікатор процесу (pid) є унікальним у межах усієї системи, і його використовують для доступу до цього процесу. Ідентифікатор процесу init завжди дорівнює одиниці.

Ідентифікатор процесу-предка (ppid) задають під час його створення. Будь-який процес має мати доступ до цього ідентифікатора. Так в UNIX-системах обов'язково підтримується зв'язок «предок-нащадок». Якщо предок процесу P завершує виконання, предком цього процесу автоматично стає init, тому ppid для P дорівнюватиме одиниці.

Із процесом також пов'язаний набір атрибутів безпеки.

Реальні ідентифікатори користувача і групи процесу (uid, gid) відповідають користувачеві, що запустив програму, внаслідок чого в системі з'явився відповідний процес.

Ефективні ідентифікатори користувача і групи процесу (euid, egid) використовують у спеціальному режимі виконання процесу — виконанні з правами власника.

3.7. Керування процесами та потоками у Windows XP

Поняття *процесу й потоку* у Windows XP чітко розмежовані. Процеси в даній системі визначають «поле діяльності» для потоків, які виконуються в їхньому адресному просторі. Серед ресурсів, з якими процес може працювати прямо, відсутній процесор — він доступний тільки потокам цього процесу.

Складові елементи процесу

Розглянемо базові складові елементи процесу.

Адресний простір процесу складається з набору адрес віртуальної пам'яті, які він може використати. Ці адреси можуть бути пов'язані з оперативною пам'яттю, а можуть — з відображеними у пам'ять ресурсами. Адресний простір процесу недоступний іншим процесам.

Процес володіє *системними ресурсами*, такими як файли, мережні з'єднання, пристрої введення-виведення, об'єкти синхронізації тощо.

Процес містить деяку *стартову інформацію для потоків*, які в ньому створюватимуться. Наприклад, це інформація про базовий пріоритет і прив'язання до процесора.

Процес має містити хоча б *один потік,* який система скеровує на виконання. Без потоків у Windows XP наявність процесів неможлива.

Структури даних процесу

Для виконавчої системи Windows XP кожний процес зображається об'єктом-процесом виконавчої системи (executive process object); його також називають керуючим блоком процесу (executive process block, EPROCESS). Для ядра системи процес зображається об'єктом-процесом ядра (kernel process object), його також називають блоком процесу ядра (process kernel block, KPROCESS).

У режимі користувача доступним є блок оточення процесу (process environment block, PEB), що перебуває в адресному просторі цього процесу.

Розглянемо структури даних процесу докладніше. Зазначимо, що EPROCESS і KPROCESS, на відміну від PEB, доступні тільки із привілейованого режиму.

Керуючий блок процесу містить такі основні елементи:

- *блок процесу ядра (KPROCESS);*
- *ідентифікаційну інформацію;*
- *інформацію про адресний простір процесу;*
- *інформацію про ресурси, доступні процесу, та обмеження на використання цих ресурсів;*
- *блок оточення процесу (PEB);*
- *інформацію для підсистеми безпеки.*

Створення процесів

У Win32 API прийнято модель запуску застосування за допомогою одного виклику, який створює адресний простір процесу і завантажує в нього виконуваний файл.

Такий виклик реалізує функція CreateProcess. Вона не є системним викликом ОС — це бібліотечна функція Win32 API.

Завершення процесів

У разі завершення процесу відповідний об'єкт-процес стає кандидатом на вилучення із системи. При цьому диспетчер об'єктів викликає метод delete для об'єктів-процесів, який закриває всі дескриптори в таблиці об'єктів цього процесу.

Процеси і ресурси. Таблиця об'єктів процесу

Кожен процес може користуватися ресурсами через дескриптори відповідних об'єктів. Відкриті дескриптори об'єктів є індексами в таблиці об'єктів (object table), що зберігається в керуючому блоці процесу. Ця таблиця містить покажчики на всі об'єкти, дескриптори яких відкриті процесом. Процес може отримати дескриптор об'єкта кількома способами:

- створивши новий об'єкт;
- відкривши дескриптор наявного об'єкта;
- успадкувавши дескриптор від іншого процесу;
- отримавши дублікат дескриптора з іншого процесу.

Кожен елемент таблиці об'єктів містить права доступу відповідного дескриптора і його режим спадкування, який визначає, чи отримають процеси, створені розглядуваним процесом, копію дескриптора відповідного об'єкта. Режим спадкування задають під час створення об'єкта.

Об'єкт може одночасно бути використаний декількома процесами, при цьому кожен з них отримує унікальний дескриптор, що відповідає цьому об'єкту.

Керування потоками у Windows XP

Для того щоб виконувати код, у рамках процесу обов'язково необхідно створити потік. У системі Windows XP реалізована модель потоків «у чистому вигляді». Процеси і потоки є різними сутностями в системі, що перебувають у чітко визначеному взаємозв'язку один з одним; для роботи з ними використовують різні системні виклики. У Windows XP ніколи не використовували модель процесів, подібну до традиційної моделі UNIX.

Багатопотоковість Windows XP базується на схемі 1:1. Кожному потоку користувача відповідає сутність у ядрі, при цьому ядро відповідає за планування потоків.

Складові елементи потоку

Потік у Windows XP складається з таких елементів:

- вмісту набору регістрів, який визначає стан процесора;
- двох стеків — один використовують для роботи в режимі користувача, інший — у режимі ядра; ці стеки розміщені в адресному просторі процесу, що створив цей потік;
- локальної пам'яті потоку (TLS);
- унікального ідентифікатора потоку (thread id, tid), який вибирають із того самого простору імен, що й ідентифікатори процесів.

Сукупність стану процесора, стеків і локальної пам'яті потоку становить контекст потоку. Кожний потік має власний контекст. Усі інші ресурси процесу (його адресний простір, відкриті файли тощо) спільно використовуються потоками.

Розрізняють два види потоків: потоки користувача і потоки ядра, які у Windows XP називають системними робочими потоками — system worker threads. Перші з них створюють у режимі користувача й тільки за необхідності перемикають у режим ядра. Інші створюють в ядрі під час його ініціалізації і виконують у режимі ядра протягом усього часу їхнього існування.

Висновки

♦ Процеси і потоки є активними ресурсами обчислювальних систем, які реалізують виконання програмного коду. Потоком називають *набір послідовно виконуваних команд процесора*. Процес є сукупністю одного або декількох *потоків* і *захищеного адресного простору*, в якому вони виконуються. Потоки одного процесу можуть разом використовувати спільні дані, для потоків різних процесів без використання спеціальних засобів це неможливо. У традиційних системах кожний процес міг виконувати тільки один потік, виконання програмного коду пов'язували із процесами. Сучасні ОС підтримують концепцію багатопотоковості.

♦ Використання потоків у застосуванні означає внесення в нього паралелізму — можливості одночасного виконання дій різними фрагментами коду. Паралелізм у програмах відображає асинхронний характер навколишнього світу, його джерелами є виконання коду на декількох процесорах, операції введення-виведення, взаємодія з користувачем, запити застосувань-клієнтів. Багато-потоковість у застосуваннях дає змогу природно реалізувати цей паралелізм і домогтися високої ефективності. З іншого боку, використання багатопотоковості досить складне і вимагає високої кваліфікації розробника.

♦ Розрізняють потоки користувача, які виконуються в режимі користувача в адресному просторі процесу, і потоки ядра, з якими працює ядро ОС. Взаємовідношення між ними визначають схему реалізації моделі потоків. На практиці найчастіше використовують схему 1:1, коли кожному потоку користувача відповідає один потік ядра, і саме ядро відповідає за керування потоками користувача.

♦ Потік може перебувати в різних станах (виконання, очікування, готовності тощо). Принципи переходу з одного стану в інший залежать від принципів планування потоків і планування процесорного часу. Перехід процесу зі стану виконання у будь-який інший стан зводиться до перемикання контексту — передачі керування від одного потоку до іншого зі збереженням стану процесора.

♦ Кожному процесу і потоку в системі відповідає його керуючий блок — структура даних, що містить усю необхідну інформацію. Під час створення процесу (зазвичай за допомогою системного виклику fork() створюють його керуючий блок, виділяють пам'ять і запускають основний потік. Створення потоку простіше і виконується швидше, оскільки не потрібно виділяти пам'ять під новий адресний простір.

Контрольні запитання і завдання

1. Назвіть основні складові *процесу*
3. Назвіть основні складові *потоку*
4. Що означає образ процесу? Які компоненти образу процесу в UNIX?
5. Можливі стани процесів і потоків
6. Що означає перемикання контексту при виконанні процесу?

7. Складові елементи процесу у Windows XP
8. Основні переваги і недоліки багатопотоковості
9. Основні види паралелізму та їх характеристика
10. Керуючі блоки процесів і потоків. Структура та призначення.

Література до лекції 3.

1. Шеховцов В. А. Операційні системи. Підручник для ВНЗ. – К.: ВНV, 2008. –576 с.
2. Таненбаум Э. Операционные системы– СПб: Питер. – 2002 г. – 1040 с.