Безпека операційних систем і комп'ютерних мереж

Лекція 12

Апаратні засоби (продовження)

Підтримка керування процесами (завданнями)

- У сучасних ОС до завдань керування процесами входять такі:
  - створення і знищення процесів;
  - забезпечення процесів необхідними ресурсами;
  - планування і диспетчеризація процесів (розподіл процесорного часу між процесами);
  - підтримка взаємодії між процесами.
- Створення, знищення і планування процесів реалізується програмно, а диспетчеризація процесів – з інтенсивним застосуванням апаратних засобів
- Функції захисту, які реалізуються безпосередньо засобами планування і диспетчеризації процесів:
  - контроль виділення процесам процесорного часу з метою запобігання перевищення квот або монополізації процесору;
  - контроль за викликами одними процесами інших з метою забезпечення встановлених правил доступу.

Контроль за викликами процесів: дискреційне і мандатне керування доступом

- Дискреційне керування:
  - У дескрипторі процесу повинна бути передбачена область даних, яка визначає права доступу інших процесів на запуск, зміну стану, зміну пріоритету, знищення цього процесу, або інші дії, які можуть бути виконані над дескриптором процесу в той час, коли сам процес не знаходиться у стані виконання
  - Права доступу можуть визначатись списком керування доступу
  - Список керування доступом є порівняно великою структурою, і роботу з ним важко організувати на апаратному рівні
- Мандатне керування:
  - Реалізація мандатного керування у цьому випадку є значно простішою і ефективнішою, і легко досягається апаратними засобами
  - Кожному процесу надається деякий рівень виконання, який визначається цілим числом. Таким чином утворюються так звані *кільця захисту*.
  - У найпростішому випадку процеси з нижчим рівнем виконання не мають доступу до виклику програмного коду, якому присвоєний вищий рівень виконання, а також до пов’язаних з таким кодом структур даних
  - Якщо кільця захисту реалізуються апаратно, то максимальна їх кількість визначається архітектурою центрального процесора
  - Більшість RISC процесорів підтримує лише два рівня виконання процесів, тобто два кільця захисту:
    - рівень ядра (*kernel mode*)
    - рівень користувача (*user mode*), або рівень прикладних програм
  - Процесори Intel x86 (починаючи з 80286) підтримують чотири кільця захисту, і існують деякі ОС, що повністю використовують цю можливість

Керування завданнями у процесорах Intel x86

- Процесори i386 надають засоби для підтримки *виклику процедур* і *виклику завдань*
- Виклик процедури – це виклик коду в межах одного процесу (потоку)
  - При цьому відбувається переключення на інший сегмент коду, але зберігаються структури, які керують адресним простором процесу (таблиця LDT, таблиця сторінок)
- Виклик завдання – це створення нового процесу з усіма відповідними структурами (що є здебільшого роботою ОС) і переключення на цей процес
- Виклик завдання також відбувається у багатозадачній ОС при переключенні між процесами
  - У цьому випадку необхідні структури вже створені, але необхідно здійснити переключення на них, забезпечивши збереження відповідних структур поточного процесу
- В усіх способах викликів використовується захист, що оснований на рівнях привілеїв

Виклик процедур

- Під час роботи будь-якого програмного коду у сучасних ОС постійно виникає необхідність виклику процедур, код яких знаходиться в інших сегментах, що можуть належати
  - іншому програмному модулю тієї ж програми,
  - бібліотеці,
  - іншій прикладній програмі,
  - ОС.
- Процесори i386 пропонують кілька способів виклику процедур з інших сегментів:
  - прямий виклик процедури з непідпорядкованого сегмента;
  - прямий виклик процедури з підпорядкованого сегмента;
  - непрямий виклик процедури через шлюз.
- Правила розмежування доступу на підставі привілеїв враховують:
  - Рівні привілеїв CPL, RPL і DPL
  - Підпорядкованість сегмента, що містить код процедури
  - Підпорядкованість сегмента задається прапорцем C в байті захисту дескриптора сегмента коду:
    - *C = 1 – підпорядкований сегмент,*
    - *C = 0 – непідпорядкований сегмент.*

Прямий виклик процедури з непідпорядкованого сегмента

- Такий спосіб полягає у розміщенні в полі команди **JMP** або **CALL** селектора, який вказує на дескриптор нового кодового сегмента
- Початкова адреса (точка входу) процедури визначається з базової адреси сегмента, яка знаходиться у дескрипторі, і зміщення, безпосередньо заданого у команді **JMP** або **CALL**
- Для такого виклику встановлені жорсткі правила захисту: якщо сегмент, який викликають, є непідпорядкованим, то виклик дозволено тоді і лише тоді, коли рівень привілеїв сегмента, з якого робиться виклик, дорівнює рівню привілеїв сегмента, який викликають, тобто CPL = DPL
  - Виклик процедури у випадку CPL > DPL (тобто, код, що виконується, має нижчий рівень привілеїв за код, який намагаються викликати) заборонений з очевидних міркувань захисту критичного коду з більшими привілеями від виклику з процедур, що мають менші привілеї
    - Наприклад, для захисту процедур ОС від безконтрольного виклику з процедур користувача (безконтрольного – тому що при такому виклику можна задати будь-яку точку входу і передати через стек будь-які параметри: контроль не передбачений)
  - Виклик у випадку CPL < DPL (тобто, код, що виконується, має вищий рівень привілеїв за код, який намагаються викликати), також забороненний
    - Це обмеження введено з міркувань того, що у загальному випадку привілейований код не може користуватись процедурами з низькими привілеями, оскільки останні вважаються ненадійними

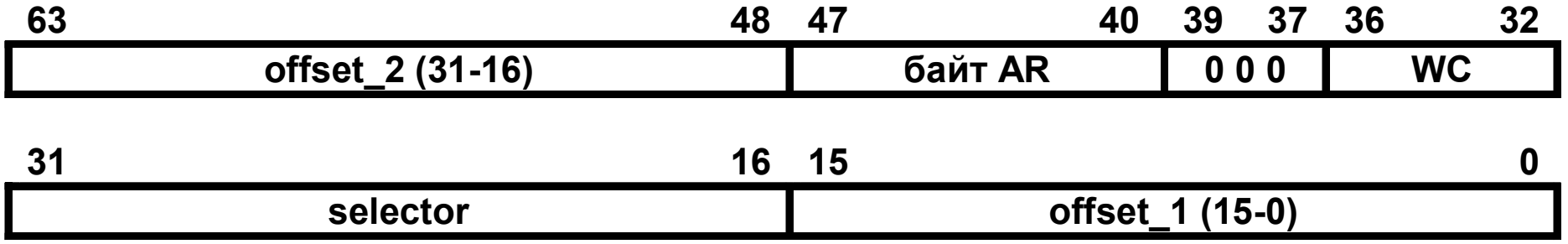
Прямий виклик процедури з підпорядкованого сегмента

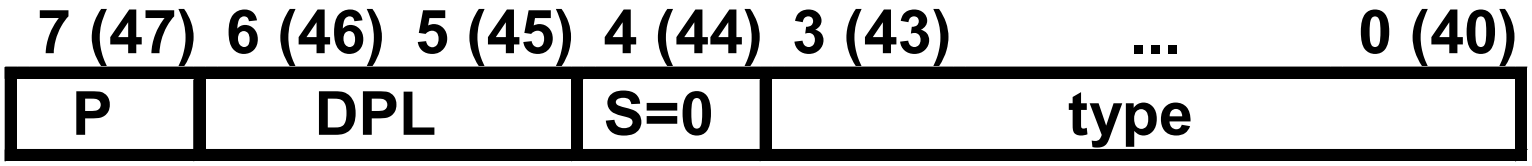
- Виклик підпорядкованих сегментів – це один із способів, який дає змогу програмам з низьким рівнем привілеїв використовувати код, що має високий рівень привілеїв
  - Наприклад, це може бути корисним при використанні програмами користувача системних бібліотек або виконанні системних викликів
- Код процедури розміщається у підпорядкованому сегменті
- Виклик здійснюється аналогічно виклику процедур з непідпорядкованих сегментів, і точки входу так само можна задавати довільно
- Дозволяється виклик при співвідношенні рівнів привілеїв **CPL ≥ DPL**, тобто код, що викликає, має привілеї не вищі за той код, який викликають
- Підпорядкований сегмент має особливу властивість: код з підпорядкованого сегмента наслідує рівень привілеїв коду, який його викликав
  - В процесі виклику CPL не зміниться, яким би не був DPL підпорядкованого сегмента
  - Наприклад, код системної бібліотеки, який міститься в сегменті з DPL = 0 буде виконуватись з CPL = 0, якщо його було викликано з ядра ОС, і з CPL = 3, якщо його було викликано з програми користувача
  - Можливості, які надаються при виклику процедур з підпорядкованого сегмента, недостатні для реалізації більшості системних викликів, оскільки при цьому часто необхідно не лише виконувати визначений код, але й звертатись до системних даних, що вимагає відповідного рівня CPL

Непрямий виклик процедури через шлюз

- За допомогою шлюзів реалізується механізм контрольованого виклику процедур, які будуть виконуватись із своїм рівнем привілеїв, можливо, вищим за той, що мав код, який здійснив виклик
- Шлюзом називається системний дескриптор спеціального виду
  - Шлюз може знаходитись і в таблиці GDT, і в таблиці LDT
  - Кожний шлюз призначений для виклику не сегмента коду, а окремої процедури із сегменту
    - Для виклику цієї процедури шлюз містить в собі адресу її точки входу – селектор сегмента і зміщення
  - Виклик здійснюється шляхом розміщення в полі команди **JMP** або **CALL** селектора, який вказує на шлюз, при цьому вказане в команді зміщення не враховується
  - Шлюз має свій власний DPL, який визначає можливість доступу коду до шлюзу: повинна виконуватись умова CPL ≤ DPL, тобто привілеї коду повинні бути не нижчими за привілеї шлюзу
    - Рівень DPL сегмента того коду, який викликається через шлюз, з рівнем CPL не порівнюється, а порівнюється із RPL селектора, що міститься у шлюзі
  - В разі успішного виклику код буде виконуватись із своїм CPL, який задається DPL сегмента

Формат шлюзу виклику процедури і байта захисту шлюзу





Призначення полів шлюзу виклику процедури

Номер байта в шлюзі	Ширина поля, розрядів	Символічне позначення	Призначення і вміст полів
0...1	16	**offset_1**	Молодші 16 розрядів зміщення точки входу
2...3	16	**selector**	Селектор сегмента, який містить код процедури
4			
біти 0...4	5	**WC**	Лічильник параметрів
біти 5...7	3		Заповнення нулями
5			Байт захисту
біти 0...3	4	**Type**	Тип: C (386) або 4 (286)
біт 4	1	**S**	=0 – “системний”
біти 5...6	2	**DPL**	Рівень привілеїв шлюзу
біт 7	1	**P**	=1 – шлюз відкритий =0 – шлюз закритий
6...7	16	**offset_1**	Старші 16 розрядів зміщення точки входу

Керування за допомогою шлюзів

- Шлюзом можна керувати: біт P в байті захисту шлюзу використовується для його “відкривання” або “закривання”
- Виклик через шлюз надає можливість контрольованої передачі параметрів через стек
  - У кожному процесі передбачено існування різних (до трьох) стеків, кожний з яких відповідає певному рівню привілеїв
  - Під час виклику через шлюз процедури, яка має рівень привілеїв, відмінний від CPL, процесор створює новий стек шляхом завантаження в регістр ss селектора, що відповідає потрібному рівню привілеїв
    - цей селектор міститься в контексті процесу – TSS
  - В новий стек з поточного стека копіюється стільки 32-розрядних слів (параметрів виклику процедури), скільки указано в полі WC шлюзу
    - Також в новий стек копіюється селектор старого стека, який дає змогу повернутись у процедуру, з якої здійснювався виклик

Виклик завдань

- Для переключення між завданнями (процесами) в команді **CALL** повинен бути заданий селектор, що вказує на дескриптор системного сегмента TSS – сегмента стану завдання
  - Такі дескриптори можуть знаходитись лише у таблиці GDT
  - Формат дескриптора аналогічний формату дескриптора сегмента даних, але значення біту S = 0
- Сегмент TSS зберігає контекст процесу, тобто всю інформацію, необхідну для поновлення виконання процесу після переривання
- Переключення контекстів здійснюється апаратно

Структура сегмента TSS

Бітова карта введення/виведення (БКВВ)		8 кБ
Додаткова інформація ОС		
Відносна адреса БКВВ	0 . . . 0	64
0 . . . 0	Селектор LDT	60
0 . . . 0	gs	5C
0 . . . 0	fs	58
0 . . . 0	ds	54
0 . . . 0	ss	50
0 . . . 0	cs	4C
0 . . . 0	es	48
edi		44
esi		40
ebp		3C
esp		38
ebx		34
edx		30
ecx		2C
eax		28
eflags		24
eip		20
cr3		1C
0 . . . 0	ss рівня 2	18
esp2		14
0 . . . 0	ss рівня 1	10
esp1		0C
0 . . . 0	ss рівня 0	08
esp0		04
0 . . . 0	селектор TSS повернення	00

Послідовність дій під час переключення контекстів

1. Виконується команда **CALL**, в якій задано селектор, що указує на дескриптор сегмента типу **TSS**
2. Здійснюється перевірка прав доступу
   - Якщо **CPL > DPL**, доступ заборонено
3. Селектор поточного сегмента **TSS** береться з регістру **tr**
   До **TSS** заносяться значення регістрів процесора
4. В регістр tr завантажується селектор **TSS** завдання, на яке переключається процесор
5. З нового **TSS** в регістр **ldtr** завантажується селектор **LDT**
6. З відповідних полів нового **TSS** оновлюються всі регістри процесора
7. Селектор сегмента **TSS** попереднього завдання заноситься в поле селектора повернення нового сегмента **TSS**

Вплив рівнів привілеїв процесів на виклик завдань

- Для виклику нового завдання поточний процес повинен мати рівень привілеїв, не нижчий за нове завдання (CPL ≤ DPL)
  - Таким чином, більш привілейований код має змогу викликати менш привілейовані завдання
    - Наприклад, ОС має змогу запускати і переключати програми користувача
- Передбачена також можливість виклику і більш привілейованого коду
  - Для цього використовуються шлюзи, аналогічні тим, що використовуються для виклику процедур
    - Для виклику завдання шлюз повинен указувати на дескриптор TSS
    - При цьому шлюзи можуть знаходитись як у таблиці GDT, так і в LDT

Привілейовані команди

- Процесори i386 надають права виконання деяких команд лише програмам, що мають визначений рівень привілеїв CPL
  - Такі команди називаються привілейованими
  - Цей механізм дозволяє реалізувати захист структур ОС від дій процесів користувача
- До числа команд, які можуть виконуватись лише при CPL = 0, тобто з найвищими привілеями, відносяться:
  - команди роботи з регістрами керування cr0…cr4;
  - команди завантаження регістрів системних адрес gdtr, ldtr, idtr, tr;
  - команда зупинки процесора **HALT**.
- Існують також команди, які стосуються операцій введення/виведення і вимагають, аби рівень привілеїв поточного процесу CPL був вищий, ніж рівень привілеїв вводу/виводу IOPL (тобто, CPL ≤ IOPL)
  - Такі команди іноді називають чутливими
  - До них належать:
    - команди заборони/дозволу маскованих переривань **CLI/SLI**;
    - команди вводу/виводу **IN**, **INS**, **OUT**, **OUTS**.
- Рівень привілеїв вводу/виводу IOPL зберігається у спеціальному полі регістру eflags

Бітова карта введення-виведення

- Якщо при спробі виконання команди введення-виведення не виконується умова **CPL ≤ IOPL**, то здійснюється звернення до бітової карти введення-виведення (**БКВВ**)
- БКВВ має розмір 8 кбайт і описує доступ до 65536 портів
- Можливість доступу до порту з конкретною адресою визначається відповідним бітом у БКВВ
- Коли біт, що відповідає певному порту, дорівнює нулю, операція введення-виведення дозволяється
- Це дає можливість відкрити прямий доступ до окремих портів програмам з низьким рівнем привілеїв

Висновок

- Процесори i386 створюють умови для реалізації багатозадачних захищених ОС шляхом забезпечення базових функцій захисту коду та окремих команд від несанкціонованого виконання з використанням мандатного принципу розмежування доступу