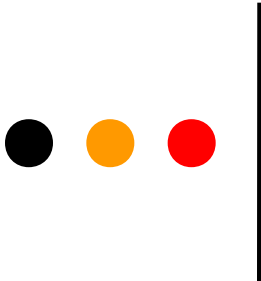




Операційні системи

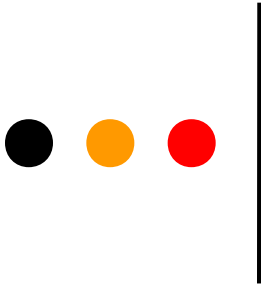
Лекція 8

Керування оперативною пам'яттю
в ОС Linux і Windows



План лекції

- Керування пам'яттю в ОС Linux
- Керування пам'яттю в ОС Windows

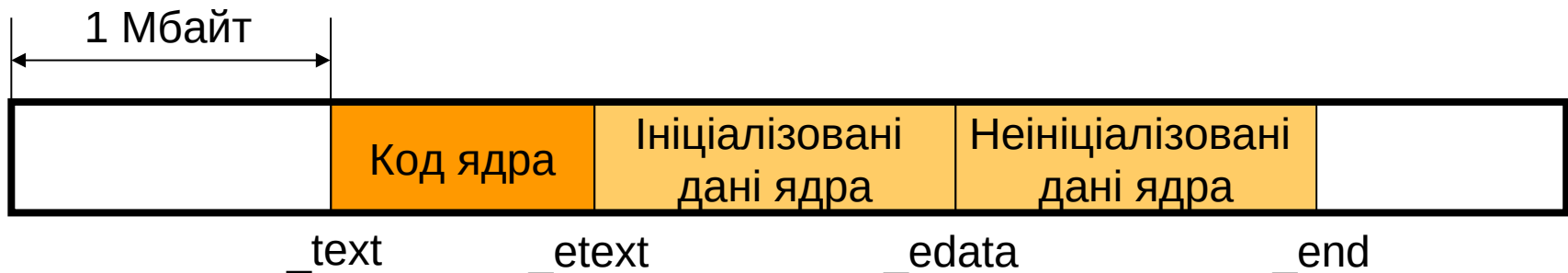


Керування пам'яттю в Linux (x86)

- Ядро системи практично не застосовує засобів підтримки сегментації, які надає процесор x86
- Підтримують мінімальну кількість сегментів — 4
 - Сегменти коду і даних ядра
 - Сегменти коду і даних режиму користувача
- Код ядра і режиму користувача спільно використовує ці сегменти
 - Сегменти коду доступні і для виконання, і для зчитування
 - Сегменти даних доступні і для зчитування, і для записування
 - Сегменти даних ядра доступні лише з режиму ядра
- Для усіх сегментів визначають межу у 4 ГБ
 - Таким чином, керування пам'яттю фактично передають на рівень лінійних адрес (які є зміщенням у сегментах)

Розміщення ядра у фізичній пам'яті

- Ядро розміщують у фізичній пам'яті, починаючи з другого мегабайта
- Фрейми пам'яті, у яких розміщено ядро, заборонено вивантажувати на диск і передавати процесам користувача
- З ядра завжди можна визначити фізичні адреси початку та кінця його коду і даних





Особливості адресації процесів і ядра

- У віртуальному адресному просторі процесу **перші 3 ГБ** (молодші адреси) відображають захищений адресний простір процесу
 - Використовують у режимі ядра та користувача
 - Елементи глобального каталогу, що визначають ці адреси, можуть задаватися самим процесом
- **Останній 1 ГБ** (старші адреси) віртуального адресного простору процесу відображає адресний простір ядра
 - Використовують лише у режимі ядра
 - Елементи глобального каталогу, що визначають ці адреси, однакові для усіх процесів, і можуть задаватися лише ядром
 - Коли передають керування потоку ядра, глобальний каталог (значення реєстру **cr3**) не змінюють, оскільки ядро використовує лише ту частину каталогу, яка є спільною для усіх процесів користувача



Особливості використання пам'яті в x86-64 (amd64)

- У сучасних реалізаціях x86-64 насправді може використовуватись лише невелика частина адрес
 - Для перетворення адреси (пошуку по таблиці сторінок) використовуються лише **найменші значимі 48 біт** адреси віртуальної пам'яті
 - Біти від 48 до 63 для будь-якої віртуальної адреси повинні бути копією біта 47 — інакше процесор згенерує виняткову ситуацію
 - Формат адрес що відповідають цьому правилу називають "канонічною формою"
- Реально використовуються 2 діапазони адрес:
 - від 0 до 00007FFFFFFFFFFFFF,
 - від FFFF800000000000 до FFFFFFFFFFFFFFFFFF
 - Кожний з цих діапазонів — у 128 TiB, разом — 256 TiB віртуального адресного простору
 - Це приблизно в 64000 разів більше від віртуального адресного простору в 32-розрядній архітектурі, але все ж є мізерною частинкою повного 64-розрядного адресного простору
- Linux використовує діапазон з вищими значеннями адрес для адресного простору ядра, а діапазон з меншими значеннями адрес — для користувацького простору



Керування адресним простором процесу в Linux

- Адресний простір процесу складається з усіх лінійних адрес, які йому дозволено використовувати
- Ядро може динамічно змінювати адресний простір процесу шляхом додавання або вилучення інтервалів лінійних адрес
- Інтервали лінійних адрес зображуються спеціальними структурами даних – **регіонами пам'яті** (*memory regions*)
 - Розмір регіону кратний 4 кБ
 - Регіони пам'яті процесу ніколи не перекриваються
 - Ядро намагається з'єднувати сусідні регіони у більший регіон



Опис адресного простору процесу в Linux

- Кожний регіон описують **дескриптором регіону** (**vm_area_struct**). Дескриптор регіону містить:
 - Початкову лінійну адресу регіону
 - Першу адресу після кінцевої адреси регіону
 - Прапорці прав доступу
 - зчитування, записування, виконання, заборона вивантаження на диск тощо
- Усю інформацію про адресний простір процесу описують **дескриптором пам'яті** (**memory descriptor**, **mm_struct**). Дескриптор пам'яті містить:
 - Кількість регіонів пам'яті
 - Показчик на глобальний каталог сторінок
 - Адреси різних ділянок пам'яті
 - коду, даних, динамічної ділянки, стека
 - Показчик на однозв'язний список усіх регіонів процесу
 - Цей список використовують для прискорення сканування всього адресного простору
 - Показчик на бінарне дерево пошуку, що об'єднує усі регіони процесу
 - Це дерево застосовують для прискорення пошуку конкретної адреси пам'яті



Сторінкова організація пам'яті в Linux

- Три рівня
 - *Page Global Directory, PGD*
 - *Page Middle Directory, PMD*
 - *Page Table*
- Елементи таблиць сторінок **PTE** вказують на фрейми фізичної пам'яті
- Кожний процес має свій **PGD** і набір таблиць сторінок
- На архітектурі Intel x86 **PMD** пустий
 - **PGD** відповідає каталогу сторінок x86
 - Між таблицями сторінок Linux і таблицями сторінок x86 завжди дотримується однозначна відповідність
- Під час переключення контексту **cr3** зберігають у блоці керування процесу



Сторінкові переривання

- Виникають під час звернення до логічної адреси пам'яті, якій у конкретний момент не відповідає фізична адреса
- Якщо переривання відбулось у режимі ядра, поточний процес негайно завершують
- Якщо переривання відбулось у режимі користувача:
 - Перевіряють регіон пам'яті, якому належить адреса. Якщо регіон відсутній, процес завершують
 - Якщо переривання викликане спробою записування у регіон, відкритий лише для зчитування, процес завершують
 - Перевіряють таблицю сторінок процесу
 - Якщо сторінка відсутня, ядро створює новий фрейм і завантажує у нього сторінку
 - Так реалізують **завантаження сторінок на вимогу**
 - Якщо сторінка є, але позначена “тільки для зчитування”, переривання могло виникнути лише під час спроби записування. Тоді ядро створює новий фрейм і копіює у нього дані зі сторінки
 - Так реалізують технологію **копіювання під час записування**

Списки сторінок менеджера віртуальної пам'яті Linux





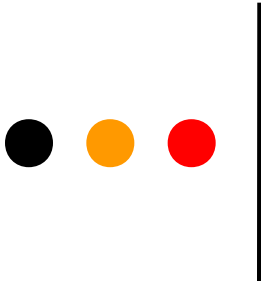
Керування пам'яттю в ОС Windows

- Сегментна модель – як і в Linux
 - Для усіх сегментів в програмі задають однакове значення бази й межі
 - Тобто, також передають керування оперативною пам'яттю на рівень лінійних адрес
- Розподіл віртуального адресного простору в x86
 - Перші 2 GiB – доступні для процесу в режимі користувача
 - Інші 2 GiB – доступні лише в режимі ядра і відображають системний адресний простір
 - Можна виділити процесу 3 GiB, а системному адресному простору — 1 GiB
- Розподіл віртуального адресного простору в x86-64
 - Ті ж діапазони адрес, що і для Linux (це визначається процесором)
 - Але!
8 TiB — для режиму користувача, і 248 TiB — для режиму ядра



Структура віртуального адресного простору процесу

- Перші 64 кБ – спеціальна ділянка, доступ до якої спричиняє помилки
- Ділянка, яку процес може використовувати під час виконання
- Блок оточення потоку ТЕВ (4 кБ)
- Блок оточення процесу РЕВ (4 кБ)
- Ділянка пам'яті, в яку відображають системні дані (системний час, номер версії системи тощо)
 - для доступу до цих даних процесу не треба перемикатись у режим ядра
- Останні 64 кБ – ділянка, спроба доступу до якої завжди спричиняє помилки



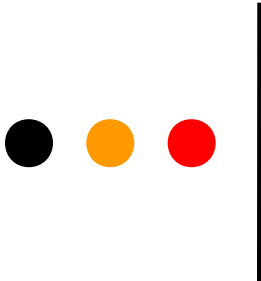
Структура системного адресного простору (спрощена)

- Перші 512 МБ – для завантаження **ядра системи**
- 4 МБ – **каталог сторінок** і **таблиці сторінок** процесу
- 4МБ – **гіперпростір** (*hyperspace*) – використовують для відображення різних структур даних, специфічних для процесу, на системний адресний простір
- 512 МБ – **системний кеш**
- **Вивантажуваний** (*paged*) пул
- **Невивантажуваний** (*nonpaged*) пул
- Приблизно 4 МБ – структури даних для створення аварійного дампу пам'яті



Сторінкова адресація в ОС Windows

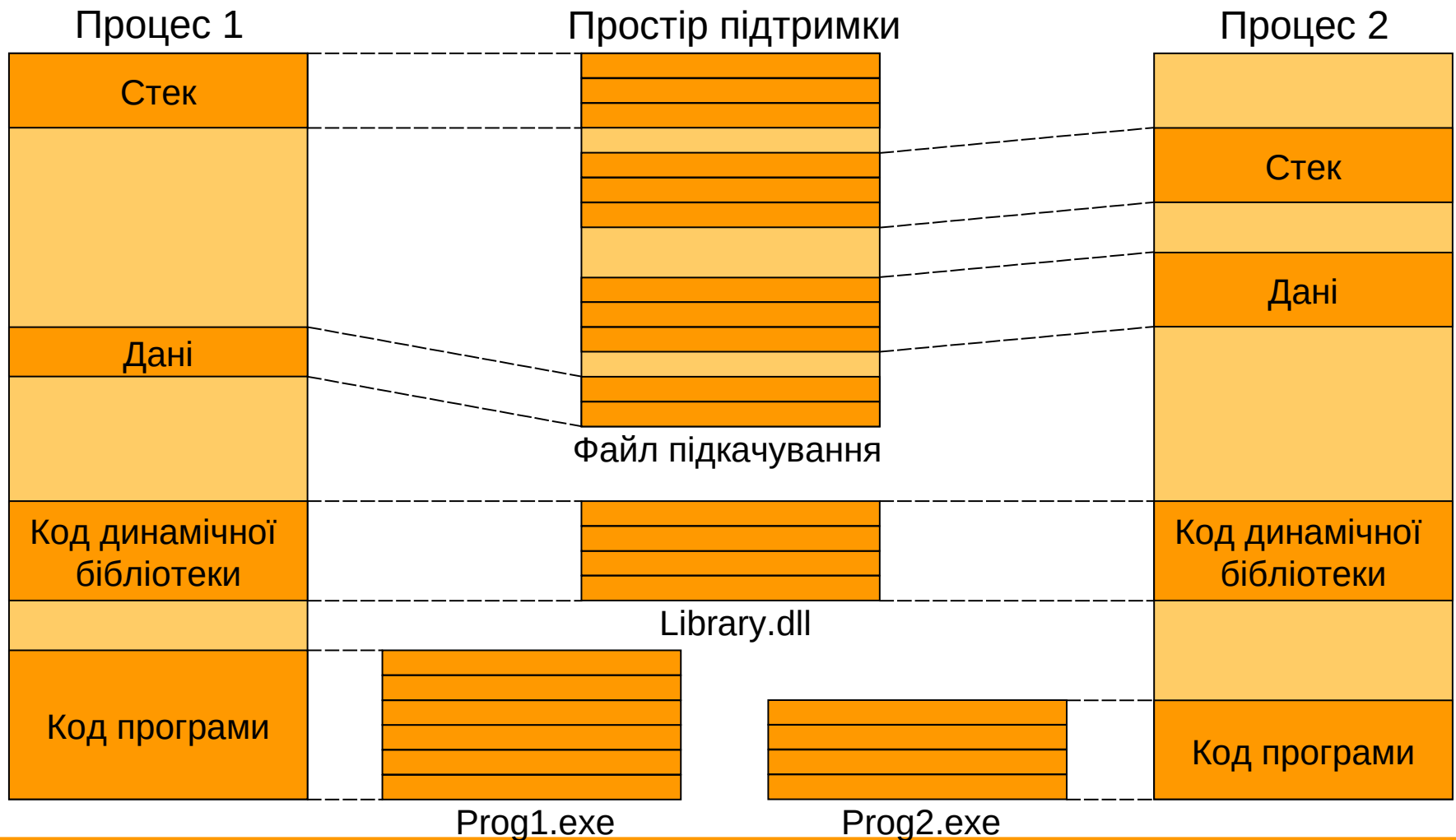
- Здійснюється у повній відповідності до архітектури Intel x86
 - У кожного процесу є свій **каталог сторінок**, кожний елемент якого вказує на **таблицю сторінок**
 - Таблиці сторінок містять по 1024 елементи, кожний з яких вказує на **фрейм фізичної пам'яті**
 - Адресу каталогу сторінок зберігають у **KPROCESS**
- Лінійна адреса – 32 біти
 - 10 – індекс у каталозі сторінок,
 - 10 – індекс у таблиці сторінок,
 - 12 – зміщення
- **Елемент таблиці сторінок (дескриптор сторінки)** – також 32 біти
 - 20 – адресують конкретний фрейм, якщо сторінка є у фізичній пам'яті, або зміщення у файлі підкачування, якщо сторінка не перебуває у фізичній пам'яті
 - 12 – атрибути сторінки

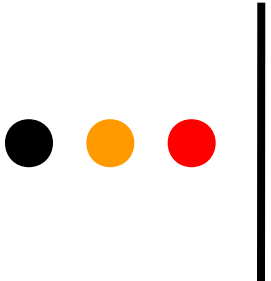


Сторінки адресного простору

- Сторінки адресного простору можуть бути
 - **вільні** (*free*)
 - **зарезервовані** (*reserved*)
 - **підтверджені** (*committed*)
- Вільні сторінки використовувати не можна. Спочатку вони мають бути зарезервовані
 - Будь-який процес може зарезервувати сторінки
 - Після цього інші процеси не можуть резервувати ті самі сторінки
 - Для використання сторінок процесом вони мають бути підтверджені
- Підтверджені сторінки пов'язані з **простором підтримки** на диску. Вони можуть бути двох типів:
 - Сторінки, що пов'язані з файлами на диску
 - Простір підтримки – той самий файл
 - Сторінки, що не пов'язані з файлами на диску
 - Простір підтримки – файл підкачування (у файлі підкачування резервуються так звані **тіньові сторінки** – *shadow pages*)

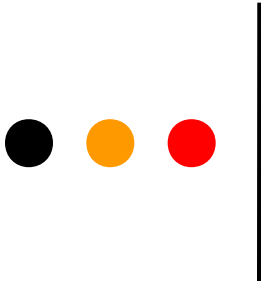
Процеси і простір підтримки у Windows





Регіони пам'яті у Windows

- Регіон відображає неперервний блок логічного адресного простору процесу
- Регіон характеризується початковою адресою і довжиною
 - Початкова адреса регіону повинна бути кратною 64 кБ
 - Розмір регіону повинен бути кратним розміру сторінки – 4 кБ
- Регіони необхідно резервувати і підтверджувати
 - Після резервування регіону інші запити не можуть його повторно резервувати
 - У процесі підтвердження регіону для нього створюються тіньові сторінки – операція підтвердження вимагає доступу до диску і є значно повільнішою, ніж резервування
 - Типовою стратегією є резервування великого регіону, а далі поступове підтвердження його малих частин
 - Для резервування і підтвердження регіону використовують функцію **VirtualAlloc()** (з різними параметрами)



Причини виникнення сторінкових переривань

- Звернення до сторінки, що не була підтверджена
 - фатальна помилка
- Звернення до сторінки із недостатніми правами
 - фатальна помилка
- Звернення для записування до сторінки, що спільно використовується процесами
 - технологія копіювання під час записування
- Необхідність розширення стека процесу
 - оброблювач переривання має виділити новий фрейм і заповнити його нулями
- Звернення до сторінки, що була підтверджена, але не завантажена у фізичну пам'ять
 - застосовують випереджаюче зчитування