Міністерство освіти і науки України
Відокремлений структурний підрозділ
«Ковельський промислово-економічний фаховий коледж
Луцького національного технічного університету»

«Операційні системи»

Конспект лекцій

для здобувачів освітньо-професійного ступеня

фаховий молодший бакалавр IV курсу

спеціальності 122 Комп'ютерні науки

денної форми навчання

Ковель, 2021

ЗМІСТ

Лекція 1

Тема 1 Операційна система: призначення і основні функції

Причиною появи операційних систем була необхідність створення зручних у використанні комп'ютерних систем (під комп'ютерною системою розуміємо сукупність апаратного і програмного забезпечення комп'ютера). Комп'ютерні системи з самого початку розроблялися для вирішення практичних задач користувачів. Оскільки робити це за допомогою лише апаратного забезпечення опинилося складно, були створені прикладні програми. Для таких програм знадобилися загальні операції управління апаратним забезпеченням, розподіл апаратних ресурсів і ін. Ці операції згрупували в рамкахокремого рівня програмного забезпечення, який і почали називати операційною системою.

Операційна система (ОС) – це комплекс програм, який завантажується при включенні комп'ютера. ОС проводить діалог з користувачем, здійснює управління комп'ютером, його ресурсами (оперативною пам'яттю, місцем на дисках і так далі), запускає інші (прикладні) програми на виконання. Операційна система забезпечує користувачеві і прикладним програмам зручний спосіб спілкування (інтерфейс) з пристроями комп'ютера.

Елементарні операції для роботи з пристроями комп'ютера і управління ресурсами комп'ютера – це операції дуже низького рівня. Дії, необхідні користувачеві і прикладним програмам, складаються з декількох сотень або тисяч таких елементарних операцій. Саме їх і виконує ОС.

Операційна система приховує від користувача ці складні і непотрібні йому подробиці і надає йому зручний інтерфейс для роботи. Вона виконує також різні допоміжні дії, наприклад, копіювання або друк файлів. Крім того, операційна система здійснює завантаження в оперативну пам'ять всіх програм, передає їм управління на початку їх роботи, виконує різні допоміжні дії по запиту виконуваних програм і звільняє займану програмами оперативну пам'ять при їх завершенні.

Операційна система – це комплекс програм, організуючих управління роботою комп'ютера і його взаємодію з користувачем. Операційна система являється посередником між застосуваннями, утилітами і користувачем, з одного боку, і апаратним забезпеченням – з іншого. Вона контролює роботу прикладних програм і системних застосувань і виконує роль інтерфейсу між застосуваннями і апаратним забезпеченням комп'ютера. ОС – це програмне забезпечення, що реалізує зв'язок між прикладними програмами і апаратними засобами комп'ютера. Її призначення можна розділити на три основні складові.

- Зручність. Операційна система робить використання комп'ютера простим і зручним.
- Ефективність. Операційна система дозволяє ефективно використовувати ресурси комп'ютерної системи.
- Можливість розвитку. Операційна система повинна бути організована так, щоб вона допускала ефективну розробку, тестування і впровадження нових застосувань і системних функцій, причому це не повинно заважати нормальному функціонуванню обчислювальної системи.

Функція «Зручність» властива ОС як розширеній машині, а функція «Ефективність» – ОС як розподільника апаратних ресурсів.

Операційна система, з одного боку, спирається на базове програмне забезпечення комп'ютера, вхідне в його систему BIOS (базова система введення-виведення); з іншого боку, вона сама є опорою для програмного забезпечення вищих рівнів – прикладних і більшості службових застосувань. Застосуваннями операційної системи прийнято називати програми, призначені для роботи під управлінням даної системи.

За допомогою операційної системи у прикладного програміста повинне створюватися враження, що він працює з розширеною машиною.

Апаратне забезпечення комп'ютера недостатньо придатне для безпосереднього використання в програмах. Наприклад, якщо розглянути роботу пристроїв введення-виведення на рівні команд відповідних

контроллерів, то можна побачити, що набір таких команд обмежений, а для більшості пристроїв – примітивний. Операційна система приховує такий інтерфейс апаратного забезпечення, замість нього програмістові пропонує інтерфейс прикладного програмування(рис. 1.1), використовуючи поняття вищого рівня (їх називають абстракціями).

Наприклад, при роботі з диском типовою абстракцією є файл. Працювати з файлами простіше, ніж безпосередньо з контроллером диска, внаслідок цього програміст може зосередитися на суті прикладного завдання. За взаємодію з контроллером диска відповідає операційна система.

Виділення абстракцій дає можливість досягти того, що код ОС і прикладних програм не зажадає заміни при переході на нове апаратне забезпечення. Наприклад, якщо встановити на комп'ютері новий жорсткий диск (вінчестер), всі його особливості будуть враховані на рівні ОС, а прикладні програми використовуватимуть файли, як і раніше. Така характеристика системи називається апаратною незалежністю

Операційна система повинна ефективно розподіляти ресурси. Під ресурсами розуміють процесорний час, дисковий простір, пам'ять, засоби доступу до зовнішніх пристроїв. ОС виступає в ролі менеджера цих ресурсів і надає їх прикладним програмам.

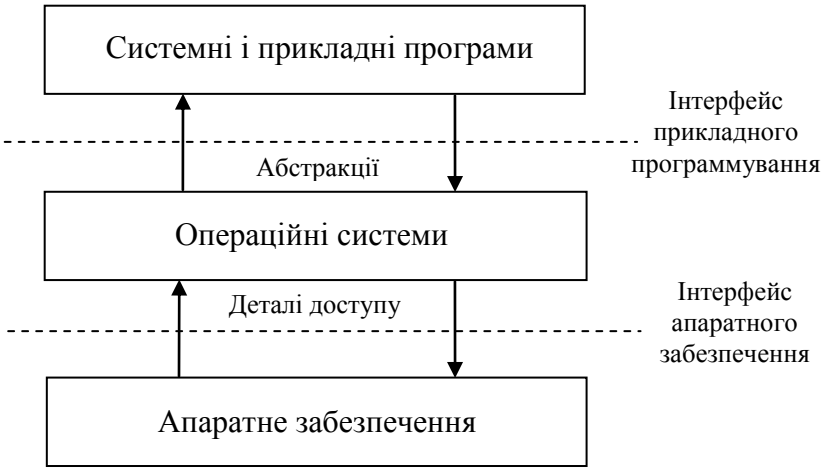


Рис.. 1.1 – Взаємодія ОС з апаратним забезпеченням і застосуваннями

Розрізняють два основні види розподілу ресурсів. В разі просторового розділення ресурс доступний декільком споживачам одночасно, при цьому кожен з них може користуватися частиною ресурсу (так розподіляється пам'ять). У разі часового розділення система ставить споживачів в чергу і згідно з нею надає їм спроможність користуватися всім ресурсом обмежений час (так розподіляється процесор в однопроцесорних системах).

При розділенні ресурсів ОС вирішує можливі конфлікти, запобігає несанкціонованому доступу програм до цих ресурсів, на які вони не мають прав, забезпечує ефективну роботу комп'ютерної системи.

Операційні системи для персонального комп'ютера можна розрізняти по декількох параметрах:

1) однозадачні і багатозадачні;
2) однокористувачеві і багатокористувачеві.

Однозадачні операційні системи дозволяють запустити програму в основному режимі. *Багатозадачні* дозволяють запустити одночасно декілька програм, які працюватимуть паралельно, які не заважають одна одній.

Однокористувачеві операційні системи дозволяють працювати на комп'ютері тільки одному користувачу. У *багатокористувачевій* системі роботу можна організувати так, що кожен користувач матиме доступ до інформації загального доступу, якщо задасть пароль до особистої інформації, доступної лише йому. Наприклад, UNIX дозволяє декільком користувачам одночасно працювати на одному комп'ютері за допомогою так званих терміналів, в ролі яких можуть виступати або спеціалізовані пристрої (відеомонітор з клавіатурою), або запущена на ПК спеціальна програма. Термінал може знаходитися в декількох метрах або в декількох тисячах кілометрів від комп'ютера. Термінал може бути пов'язаний з основним комп'ютером і через локальну мережу або Internet.

Тема 2 Історія розвитку однозадачних ОС

Перші обчислювачі – арифмометри (Блез Паскаль 1641г – тільки складання, Гельвеций Лейбніц – 1673 г – 4 дії). Етап механізації лічби.

Ідея програмованого обчислювача – Чарльз Беббідж (1792-1871) – Етап автоматизації лічби.

Представлення програми і оброблюваних даних як два потоки інформації, що зберігається в одному пристрої – оперативній пам'яті (ОП), запропонована на початку 40-х років 20 сторіччя – Джоном фон Нейманом (1903-1957). Етап розробки архітектури сучасною ЕОМ.

Перша ЕОМ (ENIAC) – розробка 1943-1945 років – в Електричній школі Мура Пенсільванського університету. Перше покоління ЕОМ, побудоване на електронних лампах, не мало зовнішніх пристроїв і операційних систем.

Вперше ОС розробляються для ЕОМ другого покоління, у зв'язку з появою в цих машин зовнішніх пристроїв. Це були, по суті, бібліотеки підпрограм по управлінню всіма пристроями ЕОМ, в першу чергу пристроями введення і виведення. Надалі перехід до програмування на алгоритмічних мовах змусив включити в систему сервісних програм транслятори, редактори зв'язків, бібліотекарі і інші допоміжні програми. Поява різних видів задач (трансляція, запис в бібліотеку, робота з текстом програми і так далі) призводить до включення до складу ОС спеціальної мови управління завданнями (JCL) і програм управління завданнями і задачами.

В цілях зниження простоїв найбільш дорогого пристрою – процесора, завдання складалися у вигляді пакету окремих задач, що послідовно вводилися набитими на перфокарти. Послідовне введення, запуск на виконання і завершення кожного завдання виконувала спеціальна програма, що управляє, яка отримала назву "монітора". Разом з програмами – драйверами зовнішніх пристроїв, трансляторами з мов програмування,

редакторами зв'язків, завантажувачами і іншими службовими програмами, монітор представляв першу операційну систему пакетної обробки. Взаємодія між окремими підпрограмами організовується за допомогою системи переривань, яка також включається до складу ОС.

Використання мікросхем замість напівпровідникових елементів в ЕОМ дозволило почати випускати нове покоління машин з великими можливостями. Перехід до ЕОМ третього покоління супроводжується інтенсивним розвитком операційних систем. У ОС ЕОМ 3-го покоління, що розробляються в період 1965-1980 рр., поступово були реалізовані всі основні моменти, властиві сучасним операційним системам.

До них можна віднести:

1) мультипрограмування (паралельне виконання декількох);
2) мультипроцесування (використання декількох процесорів);
3) багатокористувачевий режим;
4) віртуальну пам'ять;
5) файлову систему;
6) розмежування доступу;
7) мережну роботу.

Від ОС пакетного режиму роботи з однією чергою завдань (первинна – PCP –версия ОС IBM/360, ЄС/ЕОМ), відбувається перехід до системи багатозадачної з фіксованим числом задач (версія MFT), а потім і до багатозадачної із змінним числом задач (версія MVT). Досягається максимальне завантаження устаткування. Проте, що не витісняє багатозадачність цих версій, ОС не дозволяла працювати користувачам в інтерактивному режимі. Початкове виділення ресурсів в них виконується на основі пріоритетів задач але процес, що захопив процесор і що не має операцій введення-виведення тривалий час, міг надовго затримати завершення інших коротких задач.

На противагу пакетним режимам, що відокремлюють програміста від ЕОМ, розробляється мультипрограмування з розділенням часу на основі

квантування процесорного часу. Крім того, управління і розподіл ресурсів виконується з урахуванням пріоритетів користувачів. Зростає число мов програмування і типів пристроїв, що підключаються до ЕОМ. Для зниження залежності одних користувачів від інших, що в цей же час працюють на ЕОМ, ОС починає моделювати для кожного користувача окрему ЕОМ зі своєю копією операційної системи. Така ОС отримує назву системи віртуальних машин (СВМ). Все це призводить до того, що комплекс програм, що входять до ОС ЕОМ стає надзвичайно складним. Операційні системи складалися з багатьох мільйонів асемблерних команд, написаних тисячами програмістів, і містили тисячі помилок. Вони були дуже дорогими, наприклад, розробка OS/360, що містить близько 8 Мбайт коду, обійшлася компанії IBM, що розробила її, в 80 мільйонів $. (10 $ за байт!).

Хоча такі ОС володіли великими можливостями і задовольняли запитам більшості користувачів, вони почали споживати значну частину ресурсів ЕОМ, і як всяка складна система, все більше і більше працювати "на себе".

На початку 70-х років з'являються перші мережі ЕОМ і мережні операційні системи. Останні, виконуючи всі функції локальних, володіли додатковими можливостями взаємодії з ОС інших ЕОМ. Централізовані розробки і фінансові вкладення дозволили поступово реалізувати всі основні технології роботи в мережах. Сюди можна віднести роботи із створення мережі суперкомп'ютерів під егідою міністерства оборони США – ARPANET (початок - 1969г), розробки IBM по створенню мережної архітектури своїх мейнфреймів – SNA (1974 г), розробки мереж Європи X.25 (міжнародний стандарт в 1974г). Всі перераховані мережі були віддаленими, зв'язувалися засобами телекомунікацій і об'єднували ЕОМ з різними ОС.

Паралельно з глобальними мережами, починають розвиватися і локальні мережі. Їх виникненню сприяє розповсюдження малих ЕОМ, що отримали назву "міні-комп'ютерів". Недорогі, з обмеженими можливостями, часто керівники лабораторними стендами, ці комп'ютери зустрічаються по

декілька штук в окремій організації (найчастіше – в університетах). Їх ОС почали робити також спеціалізованими. Наприклад, для популярного міні PDP-11 випускалася ОС RT-11 для управління в режимі реального часу і ОС RSX-11M для режиму розділення часу.

Важливим етапом розвитку ОС стала розробка ОС UNIX. Спочатку вона створювалася як ОС режиму розділення часу для міні-комп'ютера PDP-7. З середини 70-х років почалося масове поширення UNIX, чому сприяло написання ОС на мові С і постачання у вигляді початкового тексту. Надалі її версії були створені для всіх типів ЕОМ. З 1978 року почало розповсюджуватися перше мережне застосування для ОС UNIX – UUCP (UNIX-to-UNIX Copy Program).

Саме на основі міні-комп'ютерів починають будувати локальні мережі підприємств. Тому першою мережною операційною системою вважається UNIX в різних модифікаціях.

Наступними важливими моментами в розвитку ОС можна назвати розробку протоколу TCP/IP і поява Інтернету, а також поширення персональних ЕОМ. Ці події відбувалися в 80-і роки.

Робочий варіант протоколу TCP/IP з'явився в кінці 70-х років для зв'язку мережі ARPANET з іншими мережами ЕОМ. У 1983 році він прийнятий міністерством оборони США як стандарт. З того часу ОС UNIX переходить на цей стандарт, і всі версії UNIX стають мережними.

У 1983 році ARPANET була розділена на MILNET (військові відомства США) і нову ARPANET. Їх об'єднання отримало назва Internet.

На початку 80-х років починають активно продаватися персональні комп'ютери. (1981 г – IBM починає випуск ПК IBM PC на основі 8 розрядного процесора Intel8088).

Поява персональних ЕОМ супроводжується паралельним розвитком двох ліній операційних систем. Разом з подальшим розвитком ОС великих універсальних ЕОМ (SVM мейнфреймів) і мережних UNIX–систем починається розробка невеликих за розміром і простих по виконуваних

функціях операційних систем, керівників роботою персонального комп'ютера (ПК).

Одній з перших таких ОС стає операційна система з назвою Ср/m. Це однозадачна однокористувачева операційна система для 8-розрядних Мікро-ЕОМ і ПЕОМ, розроблена фірмою Digital Research. Вона забезпечувала роботу з файлами, редагування текстових файлів, підтримку асемблера і компіляторів з алгоритмічних мов. Була передбачена переносимість програм між машинами, що використовують цю операційну систему. Ср/м фактично була стандартом операційних систем 8-розрядних ПЕОМ.

Розвитком цієї ОС для 16–ти розрядних ПЕОМ на базі процесора Intel 8086 стала Ср/м86. Оскільки ці операційні системи розроблялися для ПЕОМ з невеликою оперативною пам'яттю і повинні були розміщуватися на дискеті, вони відрізняються економним використанням пам'яті, мінімальним набором функцій і простою командною мовою.

Надалі фірма Digital Research розробила однокористувачеву багатозадачну операційну систему Concurrent CP/M, яка дозволяла вести паралельно декілька завдань. У ній реалізовувалися як фонові процеси (незалежний роздрук файлів під час роботи), так і взаємодії між двома одночасно виконуваними задачами. Проте, не дивлячись на свою назву, гідну конкуренцію операційним системам фірми Microsoft для 16–ти розрядних ПЕОМ Ср/м не склали.

Операційна система DOS розроблена фірмою Microsoft в 1981 р. практично одночасно з появою IBM PC. Версії DOS, що випускаються самою фірмою, носять назву MS DOS, версії інших фірм – PC DOS (IBM), DR DOS (Digital Research) і так далі Первинна версія 1.0 була схожа на операційну систему 8-розрядних ПЕВМ (CP/M–86). Вона підтримувала роботу з гнучкими дисками 5.25" місткістю 160 Кбайт. Версія 1.1 (з травня 1982г.) працювала з дисками місткістю 320 Кбайт.

У березні 1983г. з'являється версія 2.0. У ній використовувалася деревовидна структура каталогів, забезпечувалася можливість доступу до

послідовних зовнішніх пристроїв, як до файлів, був розроблений стандартний інтерфейс для драйверів введення-виведення, перехід на 180 і 360 Кбайтні формати дискет. Крім того, був передбачений фоновий друк, як незалежний паралельний процес. Версія 2.1 дозволила використовувати жорсткий диск (вінчестер) місткістю до 32 Мбайт.

У серпні 1984г. з'являється версія 3.0, написана вже на мові "С". Вона використовує особливості PC AT, які в принципі дозволяють підтримувати багатозадачний режим. У ній можна використовувати таблицю розміщення файлів (FAT) із записами завдовжки 16 байт, обслуговуються дискети місткістю 1,2 Мбайт.

Версія 3.1 (листопад 1984г.) дозволяє сполучати окремі ПЕОМ в прості мережі, використовуючи спеціальні драйвери. Під впливом OS UNIX реалізована ієрархічна структура зберігання файлів в каталогах і підкаталогах, розширилася командна мова, і з'явилася можливість створювати командні файли, які виконуються операційною системою (так звані BAT–файли).

Версія 3.2 з'являється в кінці 1985г. Використовуються 3,5" дискети місткістю 720 Кбайт. Завантажувальні файли (програми) з диска можуть завантажуватися в стислому форматі.

Версія 3.3 (1987г.) дозволяє розбивати вінчестер на декілька логічних дисків, до 32 Мбайт кожен. Дискети 3,5" можна форматувати на 1,44 Мбайт. Вводиться поняття кодової таблиці для символів дисплея, клавіатури і принтера.

У 1988г. з'являється допрацьована версія 4.0, в якій можливе використання логічних дисків розміром понад 32 Мбайт, підтримується розширена пам'ять понад 1 Мбайт. Введений новий компонент – Shell, що забезпечує графічний командний інтерфейс користувача (паралельно ztрадиційним текстовим режимом роботи).

5-а версія (1991г) дозволяє використовувати багатовіконний інтерфейс графічного режиму, ядро системи може завантажуватися в старші адреси ОП,

за межами 1 Мбайта, залишаючи для задач користувача до 620 Кбайт доступної пам'яті. Дозволені логічні диски розміром до 2 Гбайт.

У квітні 1993г. Microsoft випускає версію 6.0, до якої увійшли всі зібрані до цього часу компанією програми. Використовується динамічне стиснення даних на диску програмою DoubleSpace.

Остання версія MS DOS була розроблена в 1994г. і відтоді на ПЕОМ вона поставляється безкоштовно під номером 6.22.

Лекція 2

Тема 3 Класифікація сучасних операційних систем

Розглянемо класифікацію сучасних операційних систем залежно від області їх застосування.

Насамперед відзначимо ОС великих ЕОМ (мейнфреймів). Основною характеристикою апаратного забезпечення, для якого їх розробляли, є продуктивність введення-виведення інформації: великі ЕОМ оснащують значною кількістю периферійних пристроїв (дисководів, терміналів, принтерів і так далі). Такі комп'ютерні системи використовують для надійної обробки значних об'ємів даних, при цьому ОС повинна ефективно підтримувати цю обробку (у пакетному режимі або в режимі розділення часу). Прикладом ОС такого класу може бути OS/390 фірми IBM.

До наступної категорії можна віднести *серверні ОС*. Головна характеристика таких ОС – здатність обслуговувати велику кількість запитів користувачів до спільно використовуваних ресурсів. Важливу роль для них грає мережна підтримка. Існують спеціалізовані серверні ОС, з яких виключені елементи, не пов'язані з виконанням їх основних функцій (наприклад, підтримка застосувань користувача). Зараз для реалізації серверів частіше застосовують універсальні ОС (UNIX або системи лінії Windows XP).

Наймасовіша категорія – *персональні ОС*. Деякі ОС цієї категорії розробляли з розрахунком на непрофесійного користувача (лінія Windows 95/98/ME фірми Microsoft), інші являються спрощеними версіями універсальних ОС. Особлива увага в персональних ОС приділяється підтримці графічного інтерфейсу користувача і мультимедіа-технології.

Виділяють також *ОС реального часу*. У такій системі кожна операція повинна бути гарантовано виконана в заданому часовому діапазоні. ОС реального часу можуть управляти польотом космічного корабля,

технологічним процесом або демонстрацією відеороликів. Існують спеціалізовані ОС реального часу, такі як QNX і VxWorks.

Ще однією категорією являються *вбудовані ОС*. До них відносяться керуючі програми для різних мікропроцесорних систем, які використовують у військовій техніці, системах побутової електроніки, смарт-картах і інших пристроях. До таких систем ставлять особливі умови: розміщення в малому об'ємі пам'яті, підтримка спеціалізованих засобів введення-виведення, можливість прошивки в постійному запам’ятовуючому пристрої (ПЗП). Часто вбудовані ОС розробляються під конкретний пристрій. До універсальних систем відносяться Embedded Linux і Windows CE.

Тема 4 Функціональні компоненти операційних систем

ОС можна розглядати як сукупність функціональних компонентів, кожен з яких відповідає за реалізацію певної функції системи.

4.1 Управління процесами і потоками

Найважливішою функцією ОС є виконання прикладних програм. Код і дані прикладних програм зберігаються в комп'ютерній системі на диску в спеціальних файлах. Після того, як користувач або ОС вирішують запустити на виконання такий файл, в системі створюється базова одиниця обчислювальної роботи, званої *процесом*.

Процес – це програма під час її виконання. Операційна система розподіляє ресурси між процесами. До процесів належать процесорний час, пам'ять, пристрої введення-виведення, дисковий простір у вигляді файлів. При розподілі пам'яті з кожним процесом зв'язується його адресний *простір* – набір адре в пам'яті, до яких йому дозволений доступ. У адресному просторі зберігаються код і дані процесу. При розподілі дискового простору для кожного процесу формується список відкритих файлів. Аналогічним чином розподіляються пристрої введення-виведення.

Процеси забезпечують захист ресурсів, якими володіють (до адресного простору неможливо безпосередньо звернутися іншим процесам – поточний процес являється захищеним, а при роботі з файлами може бути заданий режим, що забороняє доступ до файлів всім процесам, окрім поточного).

Розподіл процесорного часу між процесами необхідний для того, щоб процесор виконував інструкції одну за іншою, а для користувача процеси повинні виглядати як послідовності інструкцій, що виконуються паралельно. Щоб досягти цього ефекту, ОС надає процесор кожному процесу на деякий короткий час, після чого перемикає процесор на інший процес; при цьому виконання процесів поновлюється з того місця, де воно було перерване. У

багатопроцесорній системі процеси можуть виконуватися паралельно на різних процесорах.

Сучасні ОС окрім багатозадачності можуть підтримувати багатопоточність, яка передбачає в рамках процесу наявність декількох послідовностей інструкцій – *потоків*, які для користувача виконуються паралельно, подібно до самих процесів в ОС. На відміну від процесів потоки не забезпечують захист ресурсів (наприклад, вони разом використовують адресний простір свого процесу).

4.2 Управління пам'яттю

В ході виконання програмного коду процесор бере інструкції і дані з оперативної пам'яті комп'ютера. При цьому пам'ять відображається у вигляді масиву байтів, кожен з яких має адресу.

ОС відповідає за виділення пам'яті під захищений адресний простір процесу і за звільнення пам'яті після того, як виконання процесу буде завершено. Об'єм пам'яті, доступний процесу, може змінюватися в ході виконання програми або задачі, в цьому випадку говорять про динамічне розділення пам'яті.

ОС повинна забезпечувати можливість виконання програм, які окремо або разом перевищують за об'ємом доступну основну пам'ять. Для цього в ній повинна бути реалізована технологія віртуальної пам'яті. Така технологія дає можливість розміщувати в основній пам'яті лише ті інструкції і дані процесу, які потрібні у нинішній момент часу, при цьому вміст іншої частини адресного простору зберігається на диску.

4.3 Управління введенням-виведенням

ОС відповідає за управління пристроями введення-виведення, підключеними до комп'ютера. Підтримка таких пристроїв в ОС здійснюється на двох рівнях. До першого, нижнього, рівню належать *драйвери пристроїв* – програмні модулі, які управляють пристроями конкретного типу з

урахуванням усіх їх особливостей. До другого рівня відноситься універсальний *інтерфейс введення-виведення*, зручний для використання в прикладних програмах.

ОС повинна реалізовувати загальний інтерфейс драйверів введення-виведення, через який вони взаємодіють з іншими компонентами системи. Такий інтерфейс дає можливість спростити додавання в систему драйверів для нових пристроїв.

Сучасні ОС надають великий вибір готових драйверів для конкретних периферійних пристроїв. Чим більше пристроїв підтримує ОС, тим більше у неї шансів на практичне використання.

Лекція 3

4.4 Управління файлами і файлові системи

Для користувачів ОС і прикладних програмістів дисковий простір надається у вигляді сукупності файлів, організованих у файлову систему.

Файл – це набір даних у файловій системі, доступ до якої здійснюється по імені. Термін «файлова система» може уживатися для двох понять: принципу організації файлів і конкретного набору даних, організованих відповідно такого принципу. В рамкахОС може бути реалізована одночасна підтримка декількох файлових систем.

Файлові системи розглядають на логічному і фізичному рівнях. Логічний рівень визначає зовнішнє представлення системи як сукупність файлів, а також виконання операцій над файлами і каталогами. Фізичний рівень визначає принципи розміщення структур даних файлової системи на диску або іншому пристрої.

4.5 Мережна підтримка

Мережні системи

Сучасні ОС пристосовані до роботи в мережі, їх називають мережними *операційними системами*. Способи операційної підтримки дають ОС можливість:

- надавати локальні ресурси (дисковий простір, принтери і так далі) в спільне користування через мережу, тобто функціонувати як сервер;
- звертатися до ресурсів інших комп'ютерів через мережу, тобто функціонувати як клієнт.

Реалізація функціональності сервера і клієнта базується на *транспортних засобах*, що відповідають за передачу даних між комп'ютерами відповідно правив, обумовлених мережними протоколами.

Розподілені системи

Мережні ОС не приховують від користувача наявність мережі, мережна підтримка в них не визначає структуру системи, а збагачує її додатковими можливостями. Існує також *розподілені операційні системи*, які дають можливість об'єднати ресурси декількох комп'ютерів в *розподілену систему*. Вона виглядає для користувача як один комп'ютер з декількома процесорами, що працюють паралельно. Розподілені і багатопроцесорні системи являються двома основними категоріями ОС, які використовують декілька процесорів.

4.6 Безпека даних

Під безпекою даних в ОС розуміють забезпечення надійності системи (захист даних від втрати у разі збоїв) і захист даних від несанкціонованого доступу.

Для захисту від несанкціонованого доступу ОС повинна забезпечувати наявність засобів *аутентифікації* користувачів (система паролів) і їх *авторизації* (дозволяють перевірити права користувача, що пройшов аутентифікацію, на виконання певної операції).

4.7 Інтерфейс користувача

Розрізняють два типи засобів взаємодії користувача з ОС:

1) командний інтерпретатор;
2) графічний інтерфейс користувача.

Командний інтерпретатор дає можливість користувачам взаємодіяти з ОС, використовуючи спеціальну командну мову. Команди такої мови примушують ОС виконувати певні дії (запускати програми, працювати з файлами).

Графічний інтерфейс користувача надає можливість взаємодіяти з ОС, відкриваючи вікна і виконуючи команди за допомогою меню і кнопок. Підходи до реалізації графічного інтерфейсу різноманітні: у Windows-системах засоби його підтримки вбудовані в систему, а в UNIX вони являються зовнішніми для системи і спираються на стандартні засоби управління введення-виведення.

Лекція 4

Тема 5 Архітектура сучасної ОС

Під архітектурою звичайно розуміють структурну організацію ОС на основі складових її програмних модулів.

5.1 Ядро і допоміжні модулі

Проста структуризація ОС полягає в розділенні всіх компонентів ОС на модулі, що виконують основні функції (ядро) і модулі, що виконують допоміжні функції ОС. Основними (базовими) є функції управління процесами, пам'яттю, пристроями введення-виведення і т.п.

До складу ядра входять, по-перше, функції, що вирішують внутрішньосистемні задачі: перемикання контекстів, завантаження-вивантаження сторінок пам'яті, обробку переривань. Ці функції не доступні для застосувань. По-друге, в ядрі є засоби створення прикладного програмного середовища застосувань, які поводяться до нього з системними викликами при необхідності роботи з апаратними засобами ЕОМ.

Функції ядра, які можуть викликатися застосуваннями, утворюють інтерфейс прикладного програмування (API).

Для забезпечення високої швидкості роботи, велика частина ядра резидентно знаходиться в оперативній пам'яті.

Для підвищення надійності, ядро зазвичай працює в привілейованому режимі, а формат модуля ядра декілька відрізняється від формату застосувань користувачів, що будуть відтрансльовані.

Допоміжні модулі ОС зазвичай виконані у вигляді бібліотек процедур або окремих застосувань. Як застосування, вони часто декілька стирають межу між ОС і застосуваннями користувачів.

Допоміжні модулі діляться на наступні групи:

1. Утиліти – програми, що вирішують окремі задачі управління і обслуговування апаратних і програмних засобів ЕОМ.

2. Системні оброблювальні програми – текстові або графічні редактори, компілятори, компонувальники, відладчики.

3. Програми додаткових послуг – засоби налагоджування спеціального інтерфейсу, калькулятори, архіватори і так далі

4. Бібліотеки процедур, що спрощують розробку застосувань, наприклад, математичні бібліотеки, бібліотеки введення-виведення і т.д.

Допоміжні модулі звертаються до функцій ядра через системні виклики.

Допоміжні модулі завантажуються в оперативну пам'ять (ОП) тільки на час своєї роботи, тобто є транзитними.

Така побудова ОС спрощує розвиток і розширення можливостей ОС з мінімальними переробками.

5.2 Привілейований режим ядра

ОС повинна мати певні привілеї по відношенню до виконуваних в ній застосувань. Забезпечити привілеї можна, тільки якщо існують спеціальні засоби апаратури. Процесор повинен підтримувати як мінімум два режими – призначений (USER MODE) для користувача і привілейований (або режим ядра – KERNEL MODE). Звичайне ядро ОС працює в привілейованому режимі, решта модулів – в призначеному для користувача.

У призначеному для користувача режимі заборонені виконання деяких команд, пов'язаних з перемиканням процесора із задачі на задачу, управлінням пристроями введення-виведення, доступом до механізмів розподілу і захисту пам'яті – тобто до задач розподілу ресурсів ЕОМ.

Очевидно, що управління ресурсами, і навіть надання деяких привілеїв повинно відбуватися під управлінням ОС. Важливо, що механізми надання ресурсів, наприклад, розділення і захисту пам'яті, використовуються ОС не тільки для власного захисту, але і для захисту одних застосувань від некоректної роботи інших. Говорять, що кожне застосування виконується в своєму адресному просторі.

Між кількістю рівнів апаратних привілеїв і рівнів операційної системи немає прямої відповідності. Наприклад, на базі процесорів Intel з 4 рівнями, Windows NT і UNIX мають по два рівні, а OS/2 – три рівні привілеїв. Але навіть на основі 2-х рівнів, ОС може побудувати скільки завгодно складну багаторівневу систему допусків і індивідуального захисту ресурсів застосувань користувачів.

Підвищення стійкості ОС при використанні привілейованих режимів декілька уповільнює роботу, оскільки витрачається додатковий час на перемикання режимів при виклику допоміжних процедур. Більшість ОС йдуть на це, але OS NetWare (фірми Novell) виконує в привілейованому режимі всі модулі ОС, що підвищує швидкодію.

Модулі DOS, навпаки, всі працюють в призначеному для користувача реальному режимі (точніше в єдиному для Intel 8086). Цей режим було емульовано і на пізніших процесорах. Альтернативою є захищений режим, в якому можливе використання і привілейованих режимів роботи процесора.

Лекція 5

Тема 6 Реалізація архітектури операційних систем

6.1 Монолітні системи

ОС, в яких базові функції сконцентровані в ядрі, називають *монолітними системами*. У разі реалізації монолітного ядра ОС стає продуктивнішою (процесор не перемикається між режимами під час взаємодії між її компонентами), але менш надійною (весь її код виконується в привілейованому режимі, і помилка в кожному з компонентів є критичною).

Монолітність ядра не означає, що всі його компоненти повинні постійно знаходитися в пам'яті. Сучасні ОС дають можливість динамічно розміщувати в адресному просторі ядра фрагменти коду (*модулі ядра*). Реалізація модулів ядра дає можливість також досягти його розширеності (для додавання нових функцій досить розробити і завантажити в пам'ять відповідний модуль).

6.2 Багатошарова (багаторівнева) структура ОС

Обчислювальну систему можна розглядати як систему, що складається з декількох (як мінімум – три) шарів: апаратури, ядра ОС і застосувань. Застосування можуть спілкуватися з апаратурою тільки через шар ядра, але не безпосередньо. У більш загальному випадку, будь-яка складна програмна система може будуватися за багатошаровою схемою. На основі функцій нижчого шару, наступний шар будує свої функції – могутніші, які, у свою чергу, є примітивами для функцій наступного шару.

Для звернення до сусіднього нижчого рівня повинен використовуватися міжшаровий інтерфейс. Між модулями різних шарів зв'язки можуть бути довільними. До вищого шару, або вниз через шар звертатися не можна.

Такий підхід застосовується і для структури самого ядра. Воно може складатися з наступних шарів:

- Засоби апаратної підтримки ОС. Частина функцій ОС може виконувати апаратура: засоби підтримки привілейованого режиму, системи переривань, засоби перемикання контекстів процесів, засоби захисту областей пам'яті і так далі.
- Машинно-залежні компоненти ОС. Цей шар містить програмні модулі, що відображають специфіку використовуваної апаратної платформи комп'ютера. Теоретично, він повинен екранувати вищі шари від особливостей апаратури і полегшувати перехід на інші платформи.
- Базові механізми ядра. Шар виконує найбільш примітивні операції ядра. Сам рішень про ці операції не приймає, а лише виконує вказівки вищого шару.
- Менеджери ресурсів. Цей шар складається з функціональних модулів, що реалізовують стратегічні задачі по управлінню ресурсами обчислювальної системи. У цьому шарі зазвичай присутні менеджери (диспетчери) процесів, введення-виведення, файлової системи і оперативної пам'яті, Кожен з менеджерів веде облік свого ресурсу і управляє його розподілом через функції базових механізмів ядра.
- Інтерфейс системних викликів. Це самий верхній шар, він утворює прикладний програмний інтерфейс ОС (API).

Розбиття на шари може порушуватися для прискорення роботи (зменшення числа проміжних інтерфейсних витрат). Зазвичай "старі" ОС, наприклад, UNIX, мають невелике число шарів, а молоді (Windows NT) – більше число формалізованих шарів в ядрі.

6.3 Апаратна залежність і переносимість

Для того, щоб ОС могла працювати на різних апаратних платформах, необхідно в ядрі ОС виділити шар машинно-залежних компонент (який

доведеться змінювати при переході до іншої платформи), а решту шарів зробити загальними.

Для виділення машинно-залежних компонент, розглянемо, які функції ОС виконуються апаратно. Найчастіше, сучасні платформи апаратного підтримують:

- Привілейований режим процесора;
- Засоби трансляції (настройки) адресів;
- Перемикання процесів;
- Переривання;
- Системний таймер;
- Систему захисту областей пам'яті.

Система підтримки привілейованого режиму ґрунтується на стані одного або двох розрядів спеціального регістра процесора (який носить назву «Слово стану процесора»). Цей регістр містить характеристики поточного режиму роботи процесора, зокрема і рівень привілеїв. Якщо використовується один розряд – рівнів два (ядро-користувач), якщо два – рівнів чотири (наприклад, для Intel Pentium 0-1-2-3).

Зміна рівня може проводитися внаслідок переривання або виконання привілейованої команди. Система не допустить виконання команди з рівнем привілеїв вище, ніж в слові стану.

Трансляція адреси – це перетворення віртуальної адреси, записаної в програмі, у фізичну адресу ОП.

Засоби перемикання процесів служать для збереження стану контексту процесу, що переривається, і відновлення контексту завантажуваного процесу, який після цього активізується. *Контекст* – це зміст основних регістрів процесора, які відрізняються для різних задач. Контекст будь-якого процесу має фіксовану довжину і по команді переходу на нову задачу контекст задачі, що переривається, записується в певну область ОП, а в регістри процесора зчитується контекст задачі, що активізується.

Система переривань забезпечує реакцію процесора на подію, що відбувається несподівано, тобто не передбачене в ланцюжку виконуваних команд програми. Така подія зазвичай пов'язана з ситуацією в апаратних засобах ЕОМ, наприклад, в схемах контролю при виникненні помилки в програмі, або в контроллері зовнішнього пристрою, при завершенні передачі блоку даних, або в таймері після закінчення кванта часу і т.д. При цьому пристрій передає електричний сигнал в процесор, який по номеру переривання переходить на відповідну програму обробки переривання. Після закінчення обробки управління зазвичай повертається на перервану програму.

Окрім апаратних, існують і програмні переривання, що викликаються спеціальними командами (INT для процесора Intel, SVC – для OS/390). При цьому відбувається перехід в режим ядра ОС.

Системний таймер – це спеціальний регістр-лічильник, в який заноситься задане значення, і який автоматично зменшується по одиниці з частотою, визначуваною кварцовим генератором. Після досягнення нуля, таймер виробляє сигнал переривання, використовуваний для виконання потрібної процедури, наприклад, перемикання активного процесу. Системний таймер не слід плутати з вбудованим годинником, який працює постійно від незалежного джерела живлення (акумулятора).

Система захисту областей пам'яті забезпечує апаратну перевірку дозволу виконувати доступ даним в деякій області при виконанні будь-якої задачі. Проводиться порівняння сегменту пам'яті для даної задачі і адреси, що викликається. Якщо апаратура підтримує апаратну трансляцію адрес, система захисту входить в неї.

Таким чином, для добре спроектованої ОС, машинно-залежні компоненти утворюють тільки частку ядра, шар, що примикає до апаратної частини ЕОМ. Решта шарів ядра даної ОС буде однаковими для різних апаратних платформ. Обсяг машинно-залежної частки залежить від того, наскільки великі відмінності в платформах. Відмінності в наборах команд

процесорів усуваються шляхом написання ОС на мові машинно-незалежній, з подальшою компіляцією. Натомість перехід від 16-ти розрядної архітектури ЕОМ до 32-х розрядною вимагає повного переписування ОС. Відмінності в архітектурі обчислювальних машин, таких як наявність або відсутність механізмів віртуальної пам'яті, один або декілька процесорів, вимагають серйозної переробки ОС при переході до іншої апаратної платформи.

Щоб зменшити машинно-залежну частку, ОС звично розбивають на групу апаратних платформ, що не сильно розрізняються. Крім того, слід пам'ятати, що ідеальна шарувата система побудови ядра ОС часто порушується із-за драйверів зовнішніх пристроїв, які, з одного боку, входять до складу шару менеджера ресурсів (високий рівень), з іншої – залежать від апаратних особливостей пристроїв, тобто входять в шар низького рівня.

Для ЕОМ на основі Intel x86/Pentium ця проблема частково вирішується за рахунок BIOS, записаної в ПЗП. У BIOS зберігаються примітивні драйвери основних пристроїв ЕОМ. Ці драйвери можна використовувати як апаратно-залежний шар.

Переносимість ОС з однієї платформи на іншу визначається ступенем серйозності переробки при переході. Щоб забезпечити мобільність ОС, при її розробці треба прагнути виконати ряд правил:

- Велика частина коду повинна бути написана на мові, для якої є транслятори на всіх платформах, які передбачається охопити. Звичайно це мова С, транслятори з якої враховують деякі особливості апаратури. На асемблері пишуть ОС тільки якщо планують використовувати апаратні платформи з однаковою системою команд. Іноді на асемблері пишуть процедури апаратно-залежного шару і реалізацію обчислень з високою точністю.
- Об'єм машинно-залежних частин коду повинен бути мінімізований. Не слід користуватися стандартними характеристиками апаратури, що задаються за умовчанням. Як в об'єктному програмуванні, слід користуватися абстрактними поняттями, реалізованими через функції,

залежні від апаратури. Тоді при переході доведеться переписати лише ці функції.

- Апаратно-залежний код слід локалізувати в декількох модулях, а не розкидати за всім кодом ОС. Локалізацію треба провести для всіх частин ОС, які залежать від процесора і апаратної платформи.

6.4 Мікроядерна архітектура

Мікроядерна архітектура, у відмінності від звичайної, залишає в привілейованому режимі тільки невелику частину ядра – так зване мікроядро. Воно містить машинно-залежні модулі і основні базові функції, до яких зазвичай відносять функції по управлінню процесами, обробці переривань, управлінню віртуальною пам'яттю і управлінню пристроями введення-виведення (у частині запису і читання регістрів пристроїв). Решта частин ядра (менеджери ресурсів, інтерфейс API) виконується в режимі призначеному для користувача.

Хоча менеджери ресурсів оформлені у вигляді застосувань, їх побудова відрізняється від звичайних застосувань. Звичайні застосування звертаються до процедур і функцій ОС, але ніколи – до процедур і функцій інших застосувань, тому в операційних системах з класичною архітектурою немає таких механізмів, за допомогою яких викликаються функції іншого адресного простору.

Задачею менеджерів ресурсів ОС є обслуговування запитів застосувань. Але оскільки самі менеджери є застосуваннями в непривілейованому режимі, їх прямий обмін з іншими застосуваннями неможливий – вони виконуються в різних адресних просторах. На відміну від решти застосувань, вони носять назви «серверів».

Одним з основних завдань мікроядра ОС є забезпечення такого обміну, оскільки тільки в привілейованому режимі можливе звернення до «чужих» областей пам'яті. Всі обміни між серверами або між серверами і застосуваннями користувачів проводяться через мікроядро у формі

повідомлень-запитів і повідомлень-відповідей. Робота мікроядерної ОС відповідає моделі «клієнт-сервер», а роль транспортних засобів виконує мікроядро.

ОС, засновані на мікроядерній архітектурі, володіють більшістю властивостей, необхідних сучасним ОС, зокрема, переносимістю на інші платформи (всі машинно-залежні команди – в мікроядрі), легкою розширюваністю новими можливостями (додавання нової підсистеми – це розробка нового застосування, не чіпаючи решту частини ОС), надійністю (всі сервери в своїх областях пам'яті і не можуть пошкодити друг-друга) і придатністю підтримки розподілених застосувань.

Як недолік, доводиться відзначити меншу продуктивність, оскільки постійно проводиться зміна режиму роботи (привілейованого на призначений для користувача і назад).

Звичайна проблема розробників ОС – що включити в мікроядро. Зокрема, в Windows NT в мікроядро включено дуже багато що, і вона переноситься тільки на схожі архітектурні платформи.

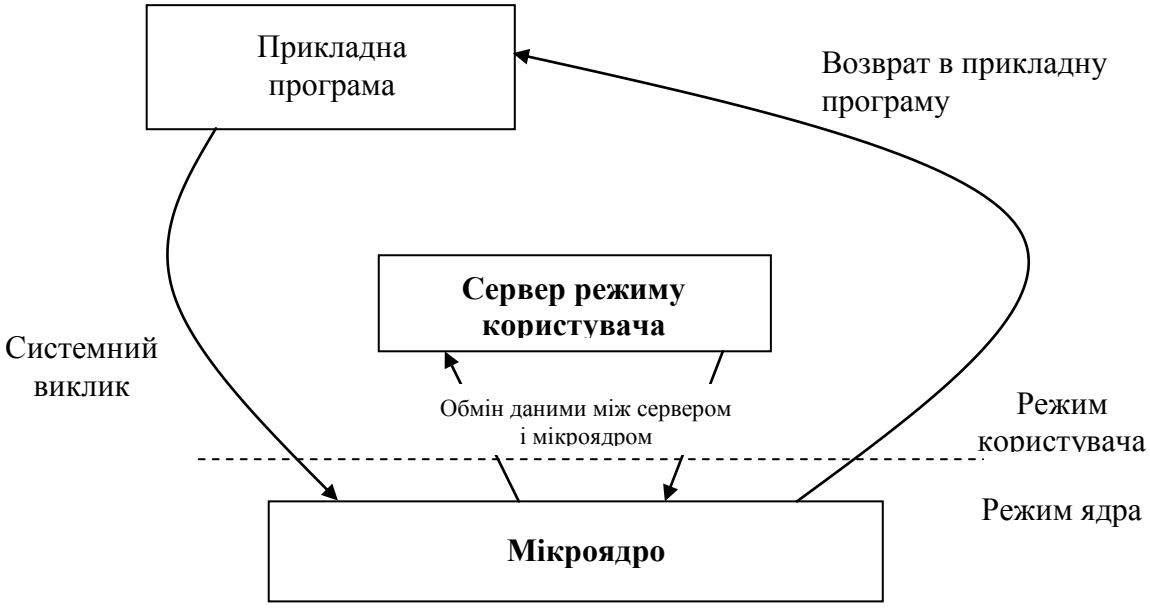


Рис.6.1 – Виконання системного виклику в архітектурі з мікроядром

Лекція 6

Тема 7 Особливості архітектури: UNIX і Linux

7.1 Базова архітектура UNIX

UNIX є прикладом достатньо простої архітектури ОС. Велика частина функціональності цієї системи розміщується в ядрі, ядро спілкується з прикладними програмами за допомогою системних викликів. Базова структура класичного ядра UNIX зображена на рис. 7.1.

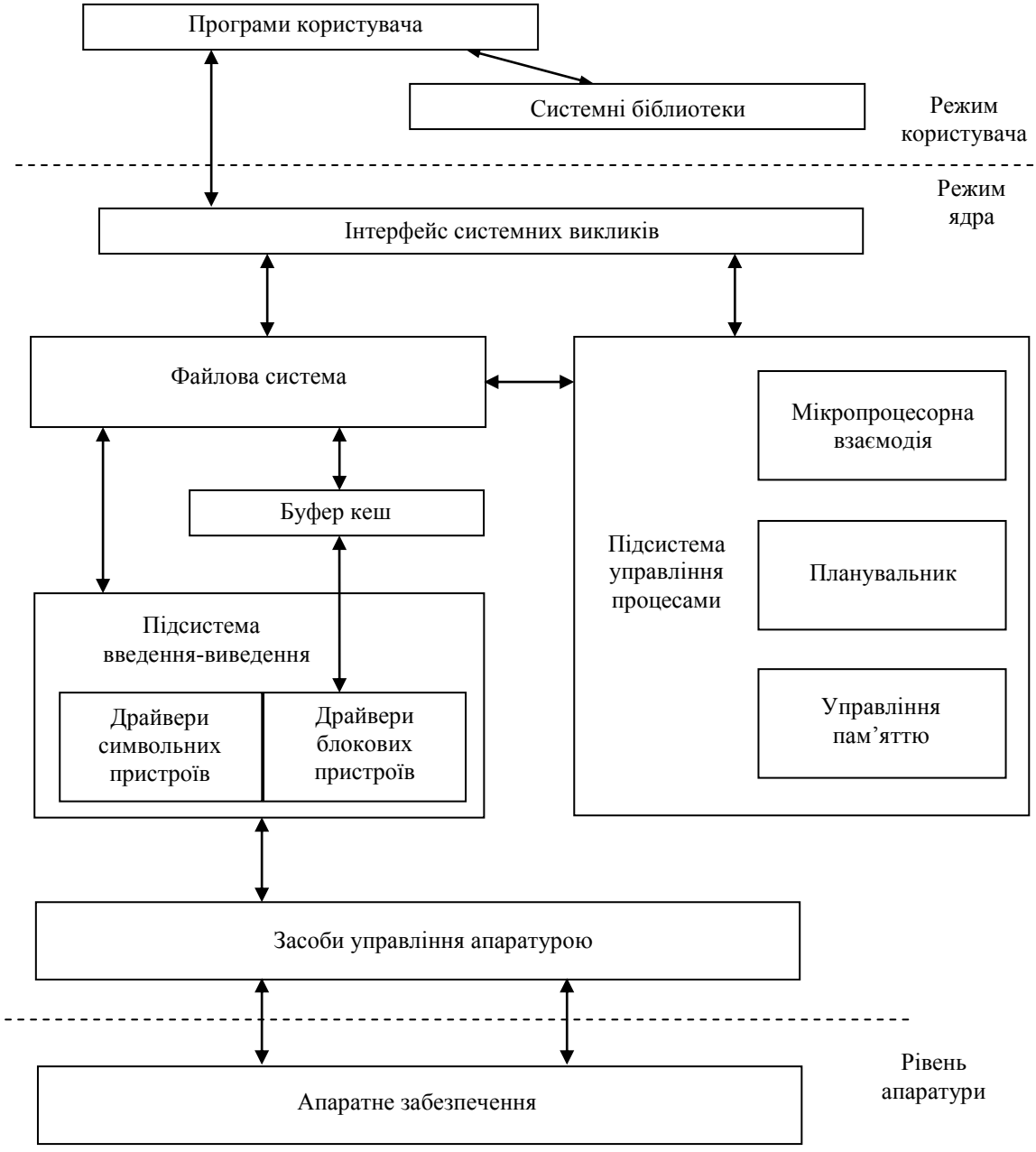


Рис. 7.1 – Архітектура UNIX

Система складається з трьох основних компонентів: підсистеми управління процесами, файлової підсистеми і підсистеми введення-виведення.

Підсистема управління процесами контролює утворення і видобування процесів, розподіл системних ресурсів між ними, міжпроцесорну взаємодію, управління пам'яттю.

Файлова підсистема забезпечує єдиний інтерфейс доступу до даним, розміщеним на дискових накопичувачах, і периферійним пристроям. Такий інтерфейс є однією з найважливіших особливостей UNIX. Одні і ті ж системні виклики використовують як для обміну даними з диском, так і для виведення на термінал або принтер (програма працює з принтером так само, як і з файлом). При цьому файлова система переадресовує запити відповідним модулям підсистеми введення-виведення, а ті – безпосередньо периферійним пристроям. Крім того, файлова підсистема контролює права доступу до файлів, яка визначає привілеї користувача в системі.

Підсистема введення-виведення виконує запити файлової підсистеми, взаємодіючи з драйверами пристроїв. У UNIX розрізняють два типи пристроїв: символьні (принтер) і блокові (жорсткий диск). Основна відмінність між ними полягає в тому, що блоковий пристрій допускає прямий доступ. Для підвищення продуктивності роботи з блоковими пристроями використовують буферний кеш – ділянку пам'яті, в якій зберігаються дані, що прочитані з диска останніми. Під час наступних звернень до цих даних вони можуть бути отримані з кеша.

Сучасні UNIX-системы декілька відрізняються по своїй архітектурі.

- У них виділений окремий менеджер пам'яті, відповідальний за підтримку віртуальної пам'яті.
- Стандартом для реалізації інтерфейсу файлової системи є *віртуальна файлова система*, яка абстрагують цей інтерфейс і дає можливість організувати підтримку різних типів файлових систем.

- У цих системах підтримується багатопроцесорна обробка, а також многопоточность.

Базові архітектурні рішення, такі як доступ до всіх пристроїв введення-виведення через інтерфейс файлової системи або організація системних викликів, залишається незмінною у всіх реалізаціях UNIX.

Лекція 7

7.2 Архітектура Linux

У ОС Linux можна виділити три основні частини:

1) *ядро*, яке реалізує основні функції ОС (управління процесами, пам'яттю, введенням-виведенням і так далі);
2) *системні бібліотеки*, які визначають стандартний набір функцій для використання в застосуваннях (виконання таких функцій не вимагає переходу в привілейований режим);
3) *системні утиліти* (прикладні програми, які виконують спеціалізовані задачі).

Призначення ядра Linux і його особливості

Linux реалізує технологію монолітного ядра. Весь код і структури даних ядра знаходяться в одному адресному просторі. У ядрі можна виділити декілька функціональних компонентів.

Планувальник процесів – відповідає за реалізацію багатозадачності в системі (обробка переривань, робота з таймером, створення і завершення процесів, перемикання контексту).

Менеджер пам'яті – виділяє окремий адресний простір для кожного процесу і реалізує підтримку віртуальної пам'яті.

Віртуальна файлова система – надає універсальний інтерфейс взаємодії з різними файловими системами і пристроями введення-виведення.

Драйвери пристроїв – забезпечують безпосередньо роботу з периферійними пристроями. Доступ до них здійснюється через інтерфейс віртуальної файлової системи.

Мережний інтерфейс – забезпечує доступ до реалізації мережних протоколів і драйверів мережних пристроїв.

Підсистема межпроцессового взаємодії – пропонує механізми, які дають можливість різним процесам в системі обмінюватися даними між собою.

Деякі з цих підсистем є логічними компонентами системи, вони завантажуються в пам'ять разом з ядром і залишаються там постійно. Компоненти інших підсистем (драйвери пристроїв) вигідно реалізовувати так, щоб їх код міг завантажуватися в пам'ять за запитом. Для вирішення цього завдання Linux підтримує концепцію модулів ядра.

Модулі ядра

Ядро Linux дає можливість за запитом завантажувати в пам'ять і вивантажувати з неї окремі секції коду. Такі секції називають *модулями ядра* і виконують в привілейованому режимі.

Модулі ядра дають низку переваг.

- Код модулів може завантажуватися в пам'ять в процесі роботи системи, що спрощує настройку компонентів ядра, насамперед драйверів.
- З'являється можливість змінювати набір компонентів ядра під час виконання: ті з них, які у цей момент не використовуються, можна не завантажувати в пам'ять.
- Модулі є виключенням з правила, по якому код, що розширює функції ядра, відповідно до ліцензії Linux повинен бути відкритим. Це дає можливість виготівникам апаратного забезпечення розробляти драйвери під Linux, навіть якщо не заплановано давати доступ до їх вихідного коду.

Підтримка модулів в Linux складається з трьох компонентів.

- Засоби управління модулями дають можливість завантажувати модулі в пам'ять і здійснювати обмін даними між модулями і іншою частиною ядра.
- Засоби реєстрації драйверів дозволяють модулям повідомляти іншу частину ядра про те, що новий драйвер став доступним.

- Засоби вирішення конфліктів дають можливість драйверам пристроїв резервувати апаратні ресурси і захищати їх від випадкового використання іншими драйверами.

Модулі можуть бути завантаженими заздалегідь – під час старту системи (завантажувальні модулі) або в процесі виконання програми, яка викличе їх функції. Після завантаження код модуля знаходиться в тому ж самому адресному просторі, що і інший код ядра. Помилка в модулі є критичною для системи.

Особливості системних бібліотек

Системні бібліотеки Linux є *динамічними бібліотеками*, які завантажуються в пам'ять тільки тоді, коли у них виникає потреба. Вони виконують низку фунций:

- реалізацію пакувальників системних викликів;
- розширення функціональності системних викликів;
- реалізацію службових функцій режиму користувача (сортування, функції обробки рядків і так далі).

Застосування користувача

Застосування користувача в Linux використовують функції з системних бібліотек і через них взаємодіють з ядром за допомогою системних викликів.

Лекція 8

Тема 8 Особливості архітектури: Windows XP

Основні компоненти Windows XP зображені на рис. 8.1.

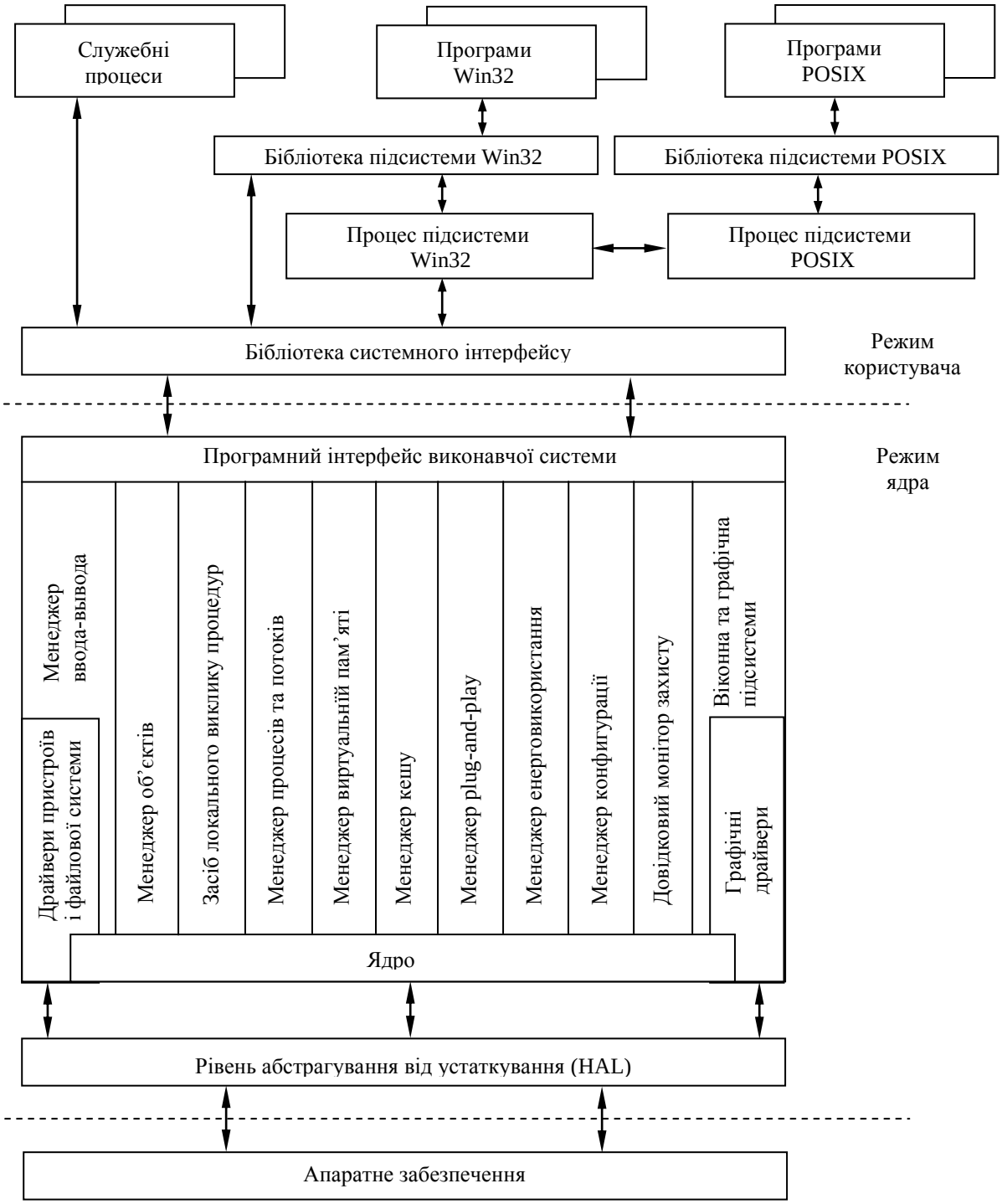


Рис. 8.1 – Базові компоненти Windows XP

Деякі компоненти Windows XP виконують в привілейованому режимі, інші компоненти – в режимі користувача.

8.1 Компоненти режиму ядра

У традиційному розумінні ядро ОС містить всі компоненти привілейованого режиму, проте в Windows XP поняття ядро закріплене тільки за одним з цих компонентів.

Рівень абстрагування від устаткування

У Windows XP реалізований рівень абстрагування від устаткування (у цій системі його називають HAL, hardware abstraction layer). Для різних апаратних конфігурацій фірма Microsoft або сторонні розробники можуть поставляти різні реалізації HAL. Хоч код HAL є дуже ефективним, його використання може знижувати продуктивність застосувань мультимедіа. У такому разі використовують спеціальний пакет DIRECTX, який дає можливість прикладним програмам звертатися безпосередньо до апаратного забезпечення, обходячи HAL і інші системи.

Ядро

Ядро Windows XP відповідає за базові операції системи. Його основними функціями являються:

- перемикання контексту, збереження і оновлення стану потоків;
- планування виконання потоків;
- реалізація засобів підтримки апаратного забезпечення, складніших чим засоби HAL (наприклад, передача управління обробникам переривань).

Ядро Windows XP відповідає базовим службам ОС і дає набір механізмів для реалізації політики управління ресурсами.

Основним завданням ядра є найбільш ефективне завантаження процесорів системи. Ядро постійно перебуває в пам'яті, послідовність виконання його інструкцій може порушити тільки переривання (під час

виконання коду ядра багатозадачність не підтримується). Для прискорення роботи ядро ніколи не перевіряє правильність параметрів, переданих під час виклику його функцій.

Windows XP не можна віднести до якогось певного класу ОС. Наприклад, хоча по функціональності ядро системи відповідає поняттю мікроядра, для самої ОС не характерна класична мікроядерна архітектура, оскільки в привілейованому режимі виконуються та інші її компоненти.

Виконавча система

Виконавча система (ІС) Windows XP (Windows XP Executive) – це набір компонентів, відповідальних за найважливіші служби ОС (управління пам'яттю, процесами і потоками, введенням-виведенням і так далі)

Компонентами ІС є, передусім, базові засоби підтримки. Ці засоби використовуються у всій системі.

Менеджер об'єктів – відповідає за розподіл ресурсів в системі, підтримуючи їх універсальну подачу через об'єкти.

Засіб локального виклику процедур (LPC) – забезпечує механізм зв'язку між процесами і підсистемами на одному комп'ютері.

Решта компонентів ІС реалізують найважливіші служби Windows XP. Зупинимося на деяких з них:

Менеджер процесів і потоків – створює і завершує процеси і потоки, а також розподіляє для них ресурси.

Менеджер віртуальної пам'яті – реалізовує управління пам'яттю в системі, в першу чергу підтримку віртуальної пам'яті.

Менеджер введення-виведення – управляє периферійними пристроями, надаючи іншим компонентам апаратно-незалежні засоби введення-виведення. Цей менеджер реалізує єдиний інтерфейс для драйверів пристроїв.

Менеджер кеша – управляє кешуванням для системи введення-виведення. Часто використані блоки диска тимчасово зберігаються в пам'яті,

наступні операції введення-виведення звертаються до цієї пам'яті, внаслідок чого підвищується продуктивність.

Менеджер конфігурації – відповідає за підтримку роботи з *системним реєстром* (registry) – ієрархічно організованим сховищем інформації про настройки системи і прикладних програм.

Довідковий монітор захисту – забезпечує політику безпеки на ізольованому комп'ютері, тобто захищає системні ресурси.

Драйвери пристроїв

У Windows XP драйвери не обов'язково пов'язані з апаратними пристроями. Застосування, якому потрібні засоби, доступні в режимі ядра, завжди необхідно оформляти як драйвер. Це пов'язано з тим, що для зовнішніх розробників режим ядра доступний тільки з коду драйвера.

Віконна і графічна підсистеми

Віконна і графічна підсистеми відповідають за інтерфейс користувача – роботу з вікнами, елементами управління і графічним виведенням.

Менеджер вікон – реалізує високорівневі функції. Він управляє віконним виведенням, обробляє введення з клавіатури або миші і передає застосуванням повідомлення користувача.

Інтерфейс графічних пристроїв (Graphical Device Interface, GDI) – складається з набору базових операцій графічного виведення, які не залежать від конкретного пристрою (креслення ліній, відображення тексту і так далі).

Драйвери графічних пристроїв (відеокарт, принтерів і так далі) – відповідають за взаємодію з контроллерами цих пристроїв.

Під час утворення вікон або елементів управління запит надходить менеджерові вікон, який для виконання базових графічних операцій звертається до GDI. Потім запит передається драйверу пристрою, потім – апаратному забезпеченню через HAL.

Лекція 9

8.2 Компоненти режиму користувача

Компоненти режиму користувача не мають прямого доступу до апаратного забезпечення, їх код виконується в ізольованому адресному просторі. Велика частка коду користувача перебуває в динамічних бібліотеках, які в Windows називають DLL (dynamic-link libraries).

Бібліотека системного інтерфейсу

Для доступу до засобів режиму ядра в режимі користувача необхідно звертатися до функцій бібліотеки системного інтерфейсу (ndll.dll). Ця бібліотека надає набір функцій-перехідників, кожною з яких відповідає функція режиму ядра (системний виклик). Застосування зазвичай не викликають такі функції безпосередньо, за це відповідають підсистеми середовища.

Підсистеми середовища

Підсистеми середовища надають застосуванням користувача доступ до служб ОС, реалізовуючи відповідний API. Розглянемо дві підсистеми середовища: Win32 і POSIX.

Підсистема Win32, яка реалізує Win32 API, є обов'язковим компонентом Windows XP. У неї входять такі компоненти:

- процес підсистеми Win32 (csrss.exe), що відповідає за реалізацію текстового (консольного) введення-виведення, створення і знищення процесів і потоків;
- бібліотеки підсистеми Win32, що надають прикладним програмам функції Win32 API. Найчастіше використовують бібліотеки gdi32.dll (низькорівневі графічні функції, незалежні від пристрою), user32.dll

(функції інтерфейсу користувача) і kernel32.dll (функції, реалізовані в ІС і ядрі).

Після того, як застосування звернеться до функції Win32 API, спочатку буде викликана відповідна функція з бібліотеки підсистеми Win32. Розглянемо варіанти виконання такого виклику.

1. Якщо функції потрібні тільки ресурси її бібліотеки, виклик повністю виконується в адресному просторі застосування без переходу в режим ядра.
2. Якщо необхідний перехід в режим ядра, з коду бібліотеки підсистеми виконується системний виклик. Так відбувається в більшості випадків, наприклад під час утворення вікон або елементів управління.
3. Функція бібліотеки підсистеми може звернутися до процесу підсистеми Win32, при цьому:
   - коли потрібна тільки функціональність, реалізована даним процесом, переходу в режим ядра не здійснюється;
   - коли потрібна функціональність режиму ядра, процес Win32 виконує системний виклик аналогічно варіанту 2.

Підсистема POSIX працює в режимі користувача і реалізує набір функцій, визначених стандартом POSIX 1003.1. Оскільки застосування або прикладні програми, написані для однієї підсистеми, не можуть використовувати функції інших, в POSIX-програмах не можна користуватися засобами Win32 API, що знижує важливість цієї підсистеми. Підсистема POSIX не є обов'язковим компонентом Windows XP.

Зумовлені системні процеси

Ряд важливих процесів користувача система запускає автоматично до завершення завантаження. Розглянемо деякі з них.

- Менеджер сесій (smss.exe) утворюється в системі першим. Він запускає важливі процеси (процес підсистеми Win32, процес реєстрації в системі і

тому подібне), а також відповідає за їх повторне виконання під час аварійного завершення.

- Процес реєстрації в системі (winlogon.exe) відповідає за допуск користувача в систему. Він відображає діалогове вікно для введення пароля, після введення передає пароль в підсистему безпеки і, у разі успішної його верифікації, запускає засоби створення сесії користувача.
- Менеджер управління службами (services.exe) відповідає за автоматичне виконання певних застосувань під час завантаження системи. Застосування, які будуть виконані при цьому, називають службами (services). Такі служби, як журнал подій, планувальник задач, менеджер друку, забезпечують разом з системою. Крім того, є багато служб зовнішніх розробників; так зазвичай реалізовують серверні застосування (сервери баз даних, веб-сервери і так далі).

Застосування користувача

Застосування користувача можуть бути створені для різних підсистем середовища. Такі застосування використовують тільки функції відповідного API. Виклики цих функцій перетворюються в системні виклики за допомогою динамічних бібліотек підсистем середовища.

Лекція 10

Тема 9 Управління процесами в ОС

Найважливіша функція ОС – розподіл ресурсів. У багатозадачних (мультипрограмних) ОС застосування конкурують між собою за ресурси. Від того, як будуть вони розподілятися, залежить продуктивність всієї обчислювальної системи. Розглянемо основні поняття багатозадачних ОС.

9.1 Мультипрограмування

Мультипрограмування (Multitasking) – спосіб організації обчислювального процесу, при якому на одному процесорі поперемінно виконуються декілька програм. Воно призначене для підвищення ефективності використання ВС. Найчастіше, під ефективністю розуміють:

- *Пропускну спроможність* – кількість задач, виконаних ЕОМ в одиницю часу;
- *Зручність роботи користувача* – можливість одночасно виконувати і управляти роботою декількох програм на одній ЕОМ;
- *Реактивність системи* – здатність системи витримувати заздалегідь задані інтервали часу між запуском програми і отриманням результатів.

Залежно від вибраного (і реалізованого) критерію ефективності, ОС діляться на *системи пакетної обробки*, *системи розділення часу* і *системи реального часу*.

У *системах пакетної обробки* мінімізується час простоїв всіх систем, і, насамперед, процесора. На період очікування реакції користувача або завершення операції обміну із зовнішніми пристроями, процесор перемикається на іншу задачу. Системи пакетної обробки призначені для вирішення обчислювальних задач. Пакет формується на початку і поповнюється залежно від запрошуваних кожною задачею ресурсів. Вигідно, щоб в одному пакеті були задачі, що вимагають різні ресурси. Найчастіше,

виграш досягається при поєднанні процесів з обмінами і процесів з обчисленнями.

У мейнфреймах для цього використовуються спеціальні процесори – канали, що працюють по команді ЦП, паралельно з ним. У ПЕОМ операції введення-виведення виконують контроллери.

У системах пакетної обробки перемикання на іншу задачу відбувається за рішенням виконуваної задачі. Сумарний час виконання декількох задач менше суми часів послідовного виконання цих задач, хоча час виконання кожної задачі більш ніж при монопольному режимі.

Системи пакетної обробки підвищують продуктивність роботи апаратури, але знижують ефективність роботи користувачів.

У *системах розділення часу* кожне відкрите застосування отримує управління гарантовано часто. При цьому сама ОС примусово перемикає виконувані програми. Квантування часу призводить до зниження пропускної спроможності ЕОМ.

У *системах реального часу* мультипрограмним пакетом є фіксований набір програм. Критерієм ефективності є забезпечення виконання задачі за заданий проміжок часу. Час реакції системи може бути різним для різних задач.

Здатність швидко відреагувати залежить насамперед від швидкості обробки переривань. Це накладає певні вимоги на процесор (навіть при невеликому завантаженні). Замість максимального завантаження устаткування, в системі реального часу прагнуть мати деякий запас незайнятих ресурсів на випадок пікових навантажень.

Не слід плутати мультипрограмну і мультипроцесорну обробки. Остання – це виконання однієї або декількох задач одночасне на декількох процесорах. Мультипроцесування значно ускладнює управління ресурсами.

9.2 Мультипроцесування

Мультипроцесорна обробка – це спосіб організації обчислювального процесу в системах з декількома процесорами, при якому декілька задач (процесів, потоків) можуть одночасно виконуватися на різних процесорах системи.

Мультипроцесорні системи часто характеризують або як симетричні, або як несиметричні. При цьому слід чітко визначати, до якого аспекту мультипроцесорної системи відноситься ця характеристика – до типу архітектури або до способу організації обчислювального процесу.

Симетрична архітектура мультипроцесорної системи передбачає однорідність всіх процесорів і одноманітність включення процесорів в спільну схему мультипроцесорної системи. Традиційні симетричні мультипроцесорні конфігурації розділяють одну велику пам'ять між всіма процесорами.

Масштабованість, або можливість нарощування числа процесорів, в симетричних системах обмежена внаслідок того, що всі вони користуються однією і тією ж оперативною пам'яттю і повинні розташовуватися в одному корпусі. Така конструкція, звана *масштабуємою по вертикалі*, практично обмежує число процесорів до 4-х або 8-и.

У *асиметричній архітектурі* різні процесори можуть відрізнятися як своїми характеристиками (продуктивністю, надійністю, системою команд і так далі, аж до моделі мікропроцесора), так і функціональною роллю, яка доручається їм в системі. Наприклад, одні процесори можуть призначатися для роботи як основні обчислювачі, інші – для управління підсистемою введення-виведення, треті – ще для якихось особливих цілей.

Масштабування в асиметричной архітектурі реалізується інакше, ніж в симетричній. Оскільки вимога єдиного корпусу відсутня, система може складатися з декількох пристроїв, кожний з яких містить один або декілька процесорів. Це *масштабування по горизонталі.* Кожний такий пристрій називається *кластером*, а вся мультипроцесорна система – кластерною.

Іншим аспектом мультипроцесорних систем, який може характеризуватися симетрією або її відсутністю, є спосіб організації обчислювального процесу. Останній визначається і реалізується ОС.

Асиметричне мультипроцесування є найбільш простим способом організації обчислювального процесу в системах з декількома процесорами. Цей спосіб називають також «відучий-ведений».

Функціонування системи за принципом «відучий-ведений» передбачає виділення одне з процесорів як «ведучого», на якому працює ОС і який управляє рештою всіх «ведених» процесорів. Тобто провідний процесор бере на себе функції розподілу задач і ресурсів, а ведені процесори працюють тільки як оброблювальні пристрої і жодних дій з організації роботи обчислювальної системи не виконують.

Асиметрична організація обчислювального процесу може бути реалізована як для симетричної мультипроцесорної архітектури, в якій всі процесори апаратного невиразні, так і для несиметричної, для якої характерна неоднорідність процесорів, їх спеціалізація на апаратному рівні.

Симетричне мультипроцесування як спосіб організації обчислювального процесу може бути реалізоване в системах тільки з симетричною мультипроцесорною архітектурою.

Симетричне мультипроцесування реалізується загальною для всіх процесорів операційною системою. При симетричній організації всі процесори рівноправний беруть участь і в управлінні обчислювальним процесом, і у виконанні прикладних задач. Різні процесори можуть в якийсь момент одночасно обслуговувати як різні, так і однакові модулі загальної операційної системи. Для цього програми операційної системи повинні володіти властивістю повторного входу (*реєнтерабельністю*).

Симетрична і асиметрична організація обчислювального процесу в мультипроцесорній системі не пов'язана безпосередньо з симетричною або асиметричною архітектурою, вона визначається типом. Так, в симетричній архітектурі обчислювальний процес може бути організований як симетричним чином, так і асиметричним. Проте асиметрична архітектура неодмінно спричиняє за собою і асиметричний спосіб організації обчислень.

Лекція 11

9.3 Процеси і потоки

У мультипрограмних системах розподілом ресурсів між програмами займається підсистема управління процесами і потоками. Вона займається їх створенням, підтримує взаємодію між ними, знищує їх і розподіляє процесорний час між паралельно виконуваними процесами.

Процес – це програма або задача, потік – дрібніша одиниця роботи. Зазвичай всі ресурси виділяються процесам, за винятком квантів процесорного часу, які виділяються потокам. Кожен процес існує в своєму віртуальному просторі, потоки одного процесу – загалом віртуальному просторі.

Віртуальний адресний простір – безліч адресів, якими може користуватися програмний модуль процесу. ОС відображає віртуальний адресний простір процесу на виділену йому фізичну пам'ять.

Для взаємодії між собою, процеси звертаються до ОС, яка надає їм засоби зв'язку, – конвеєри, поштові скриньки, загальні сегменти пам'яті і так далі Потоки звертаються до ОС як засобу розпаралелювання задач для прискорення їх виконання. Поняттю «*потік» відповідає послідовний перехід процесора від виконання однієї команди до іншої*.

Паралельне виконання декількох робіт в рамках одного застосування підвищує ефективність роботи користувача. Наприклад, при наборі тексту в Word, паралельно може виконуватися його форматування і зберігання. У системах, в яких відсутнє поняття потоку, виникають проблеми при організації розпаралелювання роботи програми. Тому в сучасних ОС є механізм багатопотокової обробки (multithreading).

Потоки одного процесу можуть обмінюватися між собою через загальну область пам'яті, мають доступи до стеків один одного і не вимагають звернення до функцій ядра при взаємодії. Від потоків інших процесів вони захищені, як і процеси.

Запустити на виконання застосування – це означає створити новий процес. При цьому створюється описувач процесу, що містить відомості про ресурси, що виділяються йому, – пам'ять, пріоритет і так далі Виконуваний модуль і дані в повному об'ємі або частково (у системах з віртуальною пам'яттю – окремими сегментами) завантажуються в пам'ять з диска. Для кожного процесу створюється хоча би один потік, якому також створюється інформаційна структура в ОС. У перший момент, потік знаходиться в припиненому стані, надалі він активізується відповідно до прийнятої системи надання задачам (потокам) процесорного часу.

Надалі, процесу може потрібно створення паралельного потоку, який часто створюється як потік-нащадок по відношенню до першого – «батьківського» потоку, і успадковує більшість властивостей першого потоку (доступи до ресурсів процесу).

При управлінні процесами, ОС створює два типи інформаційних структур для кожного процесу: *дескриптор процесу* і *контекст процесу*.

Дескриптор процесу – містить інформацію про процес, необхідну ядру ОС весь час до завершення процесу. Зберігається в області пам'яті ядра (утворюючи таблицю всіх процесів), і включає оперативну інформацію про поточний стан і знаходження процесу (у ОП або на диску), породжені потоки, пріоритетах і так далі

Контекст процесу містить об'ємнішу інформацію: вміст регістрів перерваного процесу, коди помилок виконуваних системних викликів, інформацію про всі відкриті процесом файли і незавершені операції введення-виведення і так далі Контекст, як і дескриптор, доступний тільки програмам ядра, але зберігається не в області ядра, а безпосередньо примикає до області пам'яті з образом процесу, і переміщається з ним з пам'яті на диск і назад, при свопінгу (див. управління пам'яттю).

Під час виконання застосування (процесу) його потоки можуть багато разів уриватися і знов продовжуватися. Перемикання з потоку на потік проводиться на основі планування і диспетчеризації.

Планування потоків включає вирішення двох задач:

- Визначення моменту часу для зміни активного потоку;
- Вибір потоку для продовження роботи з черги готових потоків.

Зазвичай алгоритм планування і визначає тип ОС. Звичайне планування здійснюється динамічно – рішення ухвалюється на основі поточного стану готових потоків. Недоліком є помітна витрата ресурсів на рішення цієї складної задачі.

Альтернативою служить статичне планування, яке використовується в ОС реального часу, коли набір одночасно вирішуваних задач відомий заздалегідь, і перемикання треба проводити за розкладом. Планувальник називається статичним або попереднім, і накладні витрати ОС при перемиканні потоків за розкладом виявляються істотно менше.

Диспетчеризація потоків включає вирішення трьох задач:

- Збереження контексту поточного потоку;
- Завантаження контексту нового потоку, вибраного планувальником;
- Запуск потоку на виконання.

Оскільки операція перемикання контекстів істотно впливає на продуктивність обчислювальної системи, дії з перемикання частково виконуються апаратними методами. Крім того, якщо потоки, що перемикаються, відносяться до одного процесу, частина контексту залишається незмінною, і час на її переписування не витрачається.

У мультипрограмній ОС потік може знаходитися в одному з трьох станів:

- *Виконання* – активний стан, коли він працює;
- *Очікування* – пасивний стан, коли він стоїть по внутрішніх причинах (чекає закінчення введення-виведення, надання якого-небудь ресурсу (окрім процесора));
- *Готовність* – пасивний стан, коли він готовий, але виконується інший потік (тобто він чекає процесорного часу).

Для вибору потоків на виконання, їх описувачі утворюють в пам'яті однозв’язний кільцевий список (тобто в кожному описувачі – посилання на описувач наступного потоку). Така організація дозволяє легко пересортувати чергу або виключити описувач при завершенні потоку.

Алгоритми планування діляться на два типи:

Алгоритми, що витісняють – рішення про перемикання потоків ухвалює ОС. Майже всі сучасні ОС (UNIX, Windows NT/XP, OS/2, VAX/VMS) використовують витісняючі алгоритми планування.

Алгоритми, що не витісняють – коли рішення про тимчасове припинення приймає сам виконуваний потік – у зв'язку з очікуванням якого-небудь ресурсу. При цьому програмісти повинні складати програми так, щоб вони часто уривалися (передавати управління ОС), що утрудняє розробку програм. Натомість переривання відбуваються в зручний для потоку момент, а накладні витрати ОС знижуються. Це реалізовано в ОС NetWare для файл-серверов 3.x і 4.x, завдяки чому забезпечується висока швидкість файлових операцій.

Лекція 12

9.4 Переривання

Переривання – це той механізм, за допомогою якого реально проводиться перемикання виконуваних потоків команд. На відміну від виклику підпрограми, передбаченої у визначеному місці програми, переривання відбуваються несподівано. Залежно від джерела, вони діляться на три класи:

1) зовнішні;
2) внутрішні;
3) програмні.

Зовнішні відбуваються при діях користувача або від сигналів зовнішніх пристроїв при операціях введення-виведення. Вони називаються апаратними. Відбуваються асинхронно виконуваному потоку.

Внутрішні переривання (часто звані винятками) відбуваються синхронно виконуваній програмі при появі аварійної ситуації: по захисту пам'яті, по помилках виконання операцій і так далі.

Програмні переривання, по суті є цілеспрямованими викликами процедур обробки деяких спеціальних ситуацій, виникають при виконанні команди, що імітує переривання, і ОС, що передає управління.

У кожного переривання є пріоритет, відповідно до якого всі переривання діляться на групи. У кожній групі переривання мають однаковий пріоритет.

Переривання обробляються модулями ОС, оскільки зазвичай вимагають взаємодії з апаратними засобами. Програми, що викликаються перериваннями, називаються обробниками переривань (Interrupt Service Routine – ISR). Апаратні переривання обробляються драйверами відповідних пристроїв, винятки – спеціальними модулями ядра, а програмні переривання – процедурами ОС, обслуговуючими системні виклики.

Для управління послідовністю обробки переривань, що виникають одночасно, використовується спеціальний модуль – обробник переривань. Механізм переривань спирається на апаратні засоби і програми ОС.

Існують два способи виконання переривань: векторний і опитуваний. У векторному, пристрій видає сигнал, виставляє пріоритет і видає адресу програми-обробника переривання. У опитуваному способі, процесор отримує тільки пріоритет переривання, наприклад, номер IRQ на шині ISA. Оскільки з кожним IRQ може бути зв'язане декілька пристроїв, процесор по черзі викликає програми-обробники переривань даного IRQ, поки не знайдеться потрібна. Якщо з цим IRQ пов’язаний тільки один пристрій, обробка почнеться відразу, інакше це процес довший. Цей спосіб підтримують шини ISA, Extended ISA, MCA, PCI і Sbus.

Пріоритети переривань бувають абсолютні і відносні. Вибір при одночасному запиті виконується по пріоритету, але при відносному, обробка переривання не може бути перервана з появою пріоритетнішого сигналу, а при абсолютному – уривається. Щоб не заважати обробці переривань менш пріоритетними сигналами, виставляється «маска» заборонених переривань.

Програмні переривання використовуються для переходу на підпрограму за допомогою спеціальної команди процесору (наприклад, INT для Intel). При цьому відбуваються ті ж дії, що і при апаратних перериваннях. У ОС передбачено 256 векторів переривань, частина з них може використовуватися для системних викликів (звернень до функцій ОС). Для цього з командою передається операнд, рівний номеру програми-обробника переривання. По-перше, так виявляється коротшим і швидшим, по-друге, при цьому відбувається зміна режиму роботи процесора на привілейований.

Диспетчер переривань працює як планувальник і диспетчер потоків, організовуючи черги процедур – обробників переривань, але слід враховувати, що будь-який потік має пріоритет нижче, ніж пріоритет будь-якого переривання і черги тут інші. Диспетчер запитів на переривання приймає сигнали завжди, але залежно від рівня пріоритету запиту (Interrupt

Request Level – IRQL) або поміщає його в чергу, або витісняє оброблюваний потік, запускаючи програму обробник переривання.

Вищі пріоритети встановлюються для помилок шини або інших важких апаратних збоїв, нижчий пріоритет – у виконуваних потоків процесів.

9.5 Системні виклики

Системний виклик дозволяє застосуванню звернутися до операційної системи для виконання дій, що входять в прерогативу ядра ОС, тобто заборонених в призначеному для користувача режимі. Реалізація системних викликів повинна володіти наступними властивостями:

- Забезпечувати перемикання в привілейований режим (виконує механізм переривань)
- Виконуватися швидко
- Мати однаковий вигляд в різних апаратних платформах
- Допускати легке розширення набору системних викликів
- Виконуватися під контролем ОС.

Вимоги частково суперечливі.

Високу швидкість забезпечує векторний спосіб переривань, але він прив'язаний до особливостей апаратури, і вільні вектора можуть зникати в нових версіях ОС. Зазвичай використовується централізована схема виклику за допомогою диспетчера системних викликів, що має фіксовану адресу (вектор переривання), а номер програмного переривання передається окремо.

Диспетчер системних викликів контролює запит і передає виклик процедурі ОС, адреса якої визначається номером запиту. Процедура, якщо треба, вибирає із стека дані для своєї роботи, можливо, звертається до деяких підсистем ядра ОС, а після завершення роботи повертає управління диспетчерові системних викликів.

Для застосування, системний виклик зовні нічим не відрізняється від виклику функції з бібліотеки функцій мови С.

Операційна система може виконувати системні виклики в синхронному і асинхронному режимах. При синхронному, процес, що зробив системний виклик, блокується до закінчення обробки, при асинхронному – може бути продовжений до моменту, коли необхідний результат. При цьому він може бути перерваний іншим потоком. Більшість системних викликів є синхронними, при роботі мікроядерної ОС – асинхронними.

Лекція 13

Тема 10 Управління пам'яттю в сучасних ЕОМ

Розглядається внутрішня (оперативна) пам'ять (memory) для мультипрограмної ОС.

10.1 Функції ОС по управлінню пам'яттю

Оперативна пам'ять – найважливіший ресурс ВС, і управління цим ресурсом включає ряд функцій:

- Відстежування вільної і зайнятої пам'яті;
- Виділення пам'яті процесам (первинне і додаткове) і повернення її в список вільної пам'яті (по директивах процесу і після його завершення);
- Настроювання адрес програми на конкретну область фізичної пам'яті;
- Повне або часткове вивантаження процесу-застосування з оперативної пам'яті на диск при браку фізичної пам'яті і відновлення його при появі фізичної пам'яті, що звільнилася;
- Захист пам'яті у формі контролю і заборони одному процесу звертатися до області пам'яті, виділеної іншому процесу. Виконується спільно апаратними і програмними засобами;
- Фіксоване або динамічне виділення пам'яті під системні потреби ядра ОС;
- Управління кешуванням оперативної пам'яті.

10.2 Типи адрес

Адреси використовуються для посилань в командах на дані (змінні) і інші команди. В процесі створення і запуску програми, адреси існують у формісимвольних імен і міток, віртуальних адрес і, нарешті, фізичних адрес ОП. (рис. 10.1).

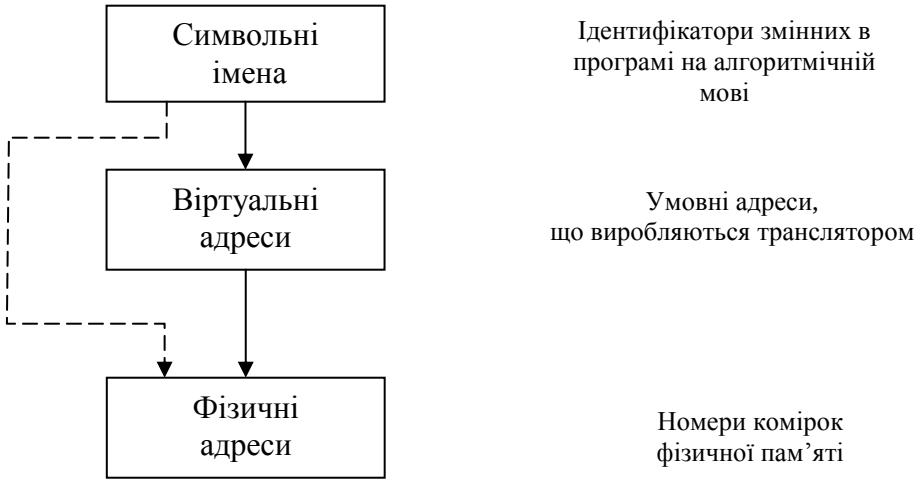


Рис. 10.1 – Типи адрес

Символьні імена по правилах використовуваної мови створюються програмістом в процесі написання програми. Вони існують тільки в початкових текстах.

Віртуальні адреси виробляє транслятор в процесі компіляції початкового тексту у виконуваний модуль (у машинні коди). Адресний простір виконуваного модуля зазвичай починається з нуля, оскільки транслятор не може знати, починаючи з якої адреси, програма буде реально завантажена в ОП.

Фізичну адресу представляє адреса (номер) байта ОП, з якого починається розміщена в пам'яті змінна або команда після завантаження програми для виконання.

Віртуальним адресним простором називається сукупність віртуальних адрес процесу.

Максимально можливий віртуальний адресний простір у всіх процесів однаково, і визначається розрядністю адреси. При 32-х розрядною ОС воно складає 4 Гбайт, з межами від 00000000_{16} до $FFFFFFFF_{16}$. Не дивлячись на те, що віртуальний адресний простір різних процесів починається з однієї адреси, віртуальні простори єрізними, і при одночасному знаходженні в ОП відображаються на різні ділянки фізичної пам'яті (рис.10.2).

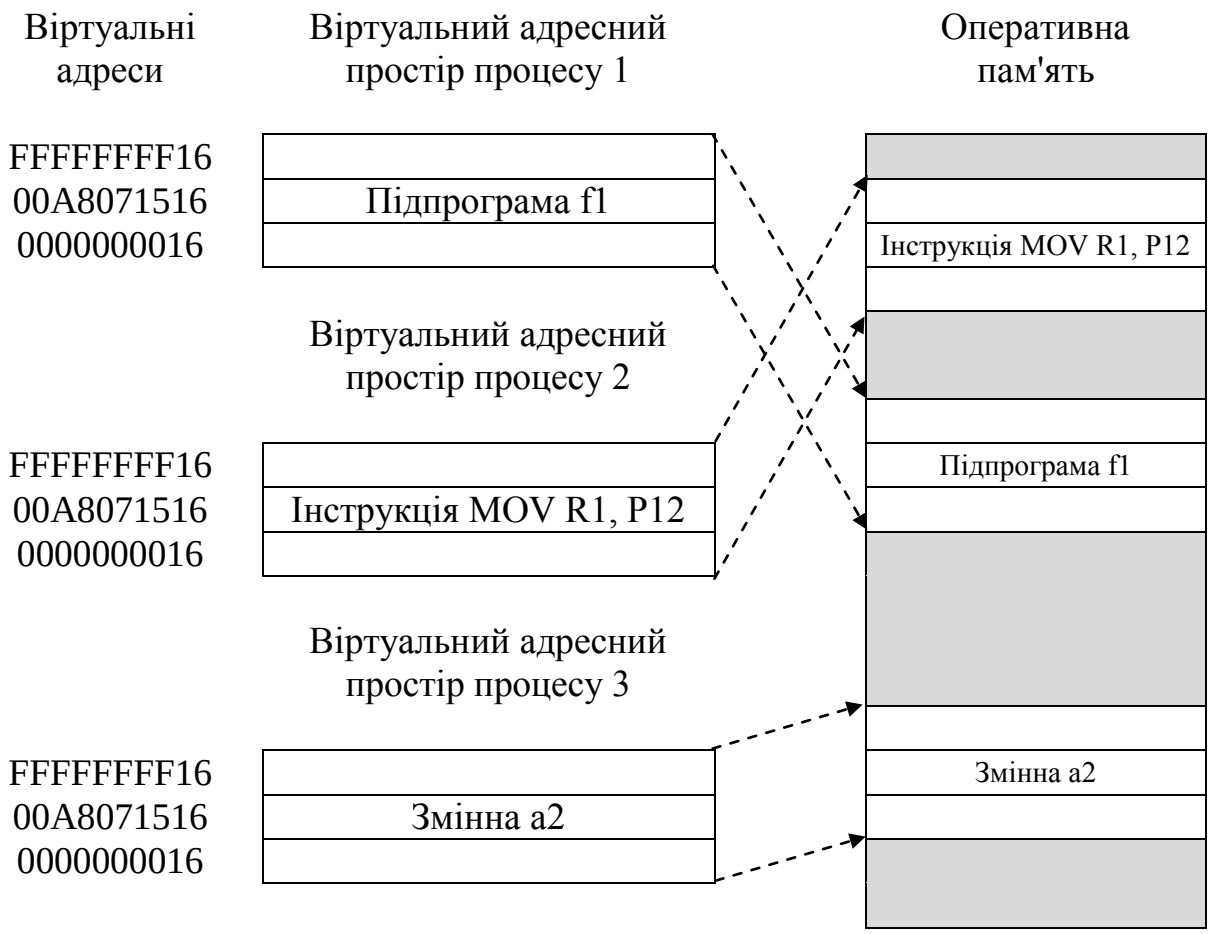


Рис. 10.2 – Віртуальні адресні простори декількох програм

Призначеним віртуальним адресним простором є набір адрес, реально необхідних процесу. Воно може бути збільшене під час виконання процесу (по його запиту для динамічних змінних), але строго контролюється ОС на можливість попадання в область фізичної пам'яті, займаної іншим процесом.

Віртуальний адресний простір в одних ОС представлений лінійною послідовністю адрес (як у фізичній пам'яті). Така структура, коли адреса задається одним числом, називається плоскою. У інших ОС, віртуальний адресний простір ділиться на частини, звані зазвичай сегментами. При цьому, віртуальна адреса зображається парою чисел: (n, m) n – номером сегменту і m – зсувом усередині сегменту. Можливо і більша кількість чисел, що визначають віртуальну адресу (рис.10.3)

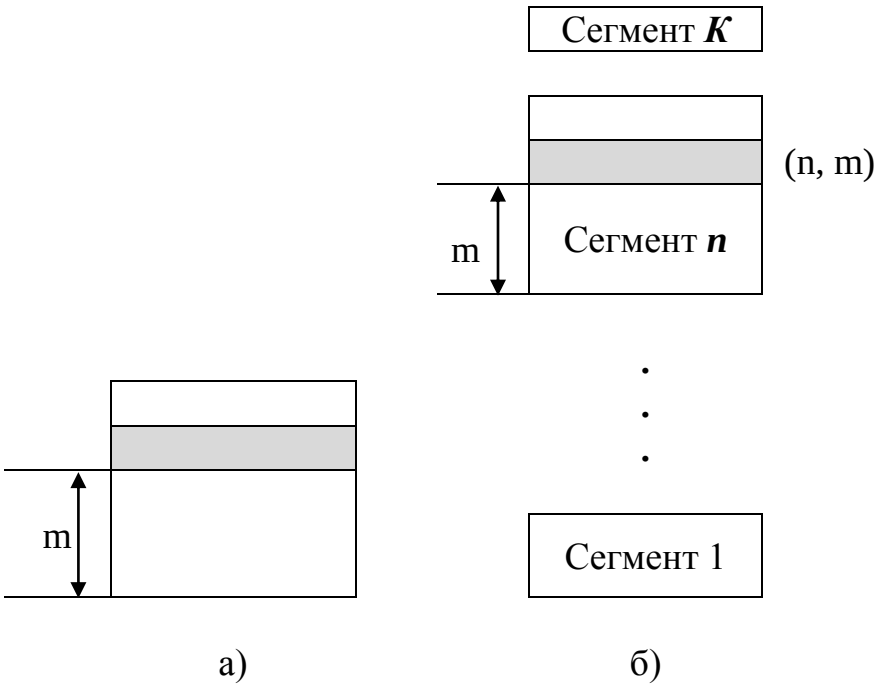


Рис. 10.3 – Типи віртуальних адресних просторів:
плоский (а), сегментований (б)

Якщо переведення символьних імен у віртуальні адреси виконує транслятор, то задача відображення віртуальної адреси на фізичну адресу є функцією ОС. При цьому є два різних підхода до цього процесу.

Перший спосіб полягає в заміні віртуальних адрес на фізичних один раз, у момент завантаження. Це робить програма – завантажувач, на підставі інформації про вільну і зайняту пам'ять у момент завантаження.

Другий спосіб полягає в тому, що програма, хоча і завантажується в пам'ять, починаючи з вільної адреси, усередині команд всі адреси залишаються віртуальними. У момент виконання такої команди, віртуальна адреса в ній замінюється на фізичну (наприклад, додаванням початкового зсуву).

Другий спосіб є більш універсальним, дозволяючи коду процесу в різний час перебувати в різних областях фізичної пам'яті (наприклад, після «збирання сміття» в ОП або після відновлення процесу в ОП після вивантаження на диск). Перший спосіб, зате володіє великою швидкодією,

оскільки перерахунок адрес проводиться один раз при завантаженні. У всіх випадках, механізм перетворення віртуальних адрес у фізичні адреси повинен бути прозорий для програмістів.

Під час виконання, процес застосування часто звертається до функцій ядра ОС, тому, для полегшення переходу від призначеного для користувача потоку команд до модулів ядра, більшість ОС об'єднують віртуальний адресний простір застосування і віртуальний адресний простір ядра ОС в загальний віртуальний адресний простір. Наприклад, в Windows NT і OS/2 віртуальні адреси до 2-х Гбайт віддаються ОС, а від 2-х до 4-х Гбайт – застосуванню. Оскільки при зміні активного застосування ОС повинна залишатися, замінюються тільки сегменти другої половини адресного простору.

У більшості універсальних ОС механізм сторінкової пам'яті застосовується до всіх сегментів призначеної для користувача частини віртуального простору. Системна частина ОС ділиться на області, що витісняються і не витісняються. У частині, що не витісняється, розміщуються модулі ОС, що вимагають швидкого виконання або постійної присутності в ОП, наприклад, диспетчер потоків або код, керівник заміною сторінок пам'яті. В цьому випадку фізична адреса або рівний віртуальному, або легко виходить з віртуального. Решта модулів ОС також піддається сторінковому витісненню і підміні, як і сегменти призначених для користувача процесів.

Лекція 14

10.3 Розподіл пам'яті

В процесі розвитку ОС, вони по-різному вирішували питання розподілу пам'яті. Принциповими моментами були: використання зовнішньої пам'яті для тимчасового витіснення процесів або ні, фіксоване положення програм в пам'яті або переміщуване і так далі Загальна класифікація механізмів управління розподілом пам'яті представлена на рис. 10.4.

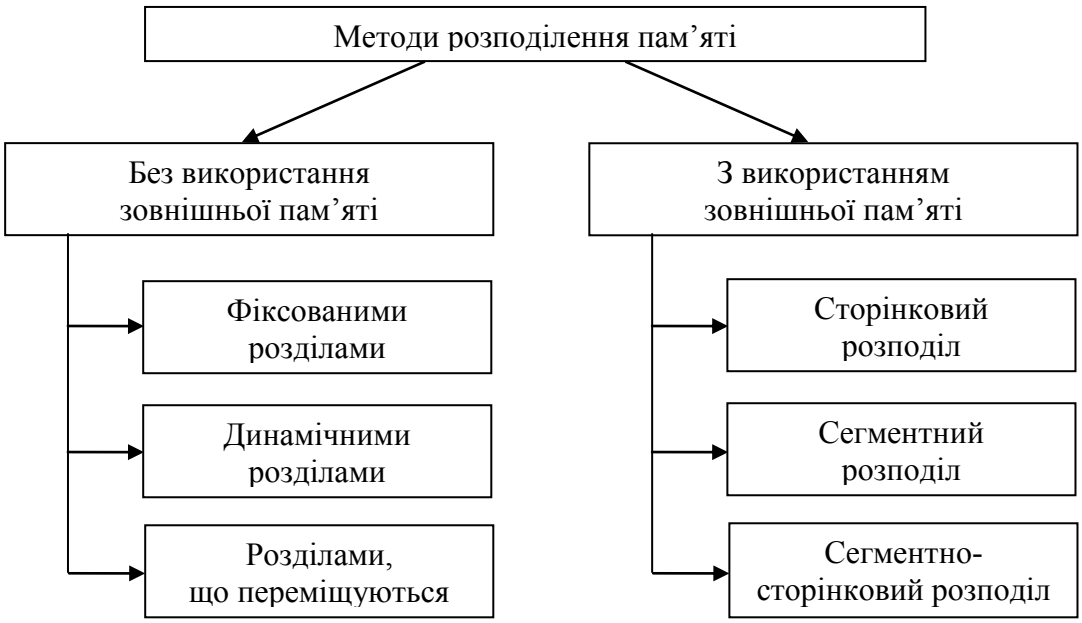


Рис. 10.4 – Класифікація методів розподілу пам’яті

Простий спосіб полягає в розбитті всій ОП на декілька фіксованих розділів. Це виконується у момент завантаження ОС. Кожне чергове завдання поміщається або в загальну чергу, або в чергу замовленого розділу. Якщо в системі підтримується декілька черг, то завдання поміщається в ту чергу, яка відповідає замовленому ресурсу по необхідній пам'яті.

Як тільки звільняється який-небудь розділ, в нього завантажується чергове (або найпріоритетніше) завдання з черги розділу, або відповідне по необхідних ресурсах завдання із загальної черги (рис.10.5).

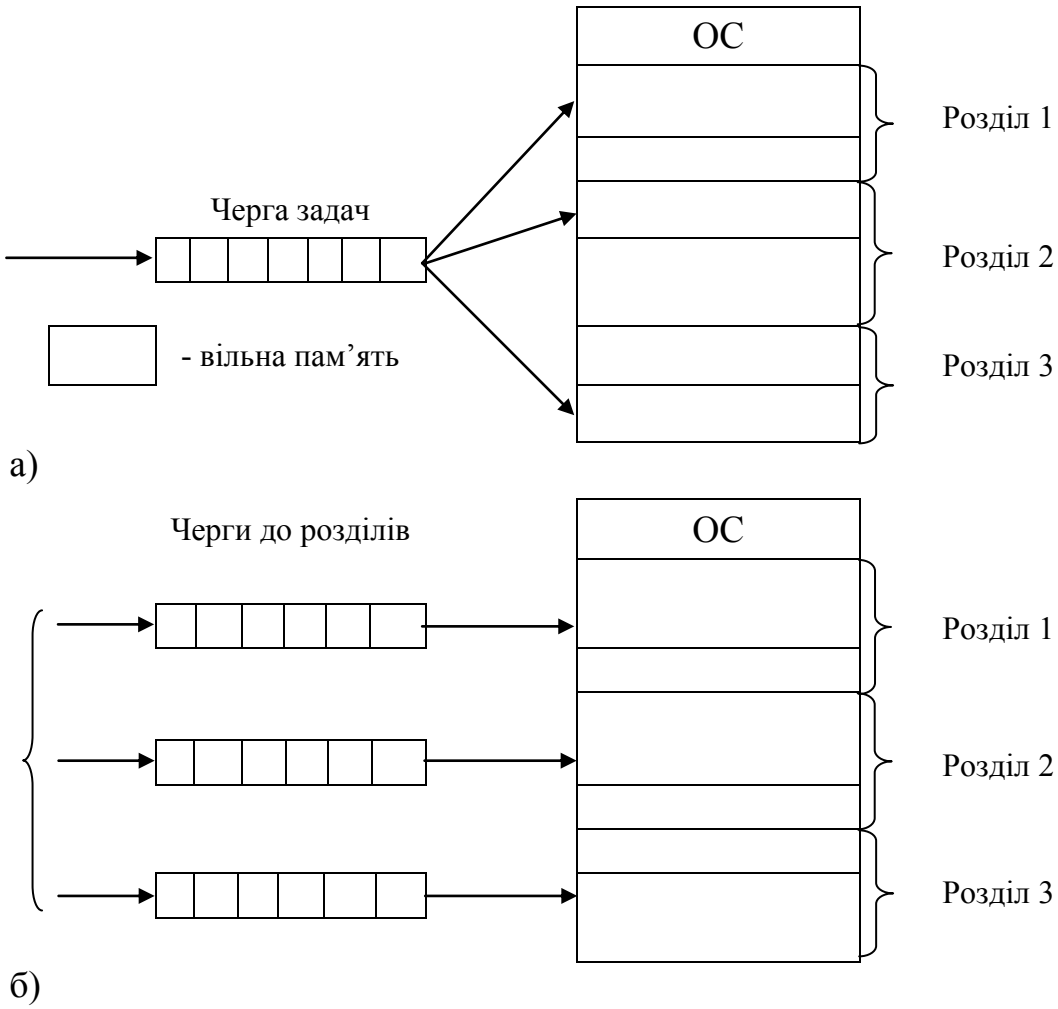


Рис. 10.5 – Розподіл пам’яті фіксованими розділами:
із загальною чергою (а), з окремими чергами (б)

Очевидно, що, хоча це найбільш простій спосіб управління, він абсолютно не гнучкий, і не дозволяє завантажити в пам'ять застосування, що вимагає більше пам'яті, чим найбільший розділ. Крім того, він не дозволяє завантажити багато дрібних застосувань одночасно.

Такий механізм використовувався в старих ОС з фіксованим числом розділів (MFT).

Другий спосіб пропонує виділяти розділи у міру надходження замовлень із заданими вимогами по пам'яті. Зазвичай фіксувалася тільки максимальна кількість застосувань, що одночасно перебувають в пам'яті. Типова кількість – 16 програм. Черговий процес може завантажуватися в пам'ять, тільки коли в ній опиниться безперервна ділянка завдовжки більше

потрібного завданням. Вибір чергового завдання виконувався з єдиної черги по одному з принципів:

- Перший придатний;
- З найвищим пріоритетом;
- Найбільший, такий, що поміщається в наявну ділянку;
- Найменший, такий, що поміщається в наявну ділянку.

Недоліком є значна фрагментизация пам'яті після завершення декількох процесів.

Фрагментізация – це наявність великого числа несуміжних ділянок вільної пам'яті малого розміру.

Такий механізм носить назва MVT (виділення пам'яті динамічними розділами для змінного числа завдань). Він широко використовувався в ОС 70-х років, наприклад в OS/360.

Одним з методів боротьби з фрагментизацією пам'яті стало використання процедури стиснення (дефрагментизації). Вона полягала в перенесенні сегментів процесів так, щоб вільні ділянки пам'яті збиралися на початку або наприкінці фізичної пам'яті машини. Стиснення виконується або при завершенні процесу, або коли не вистачає пам'яті для завантаження чергового процесу.

Такий метод можливий, тільки якщо настроювання віртуальних адрес на фізичну пам'ять проводиться у момент виконання команди, а не при завантаженні переміщаючим завантажувачем.

Процес дефрагментизації займає досить багато часу, тому, незважаючи на ефективне використання пам'яті, часто від нього відмовляються.

Використання зовнішньої пам'яті для тимчасового вивантаження програмного коду або даних невживаного в даний момент процесу дозволяє ефективніше використовувати ОП. При цьому можуть застосовуватися три методи управління пам'яттю.

Сторінковий розподіл.

Сторінкова віртуальна пам'ять організовує переміщення даних між ОП і зовнішньою пам'яттю сторінками – частинами віртуального адресного

простору фіксованого і порівняно невеликого розміру. Віртуальний адресний простір ділиться на віртуальні сторінки (virtual pages), розмірами, що дорівнюють ступені 2 (1024, 4096 і так далі). Наприклад, Pentium дозволяє використовувати сторінка розміром 4Кбайт і 4Мбайт. Для кожного процесу створюється таблиця сторінок, кожній сторінці відповідає структура, що містить номер фізичної сторінки, ознаку модифікації, число звернень до сторінки. При необхідності вибору сторінки, яку потрібно вивантажити на диск, зазвичай використовують ознаку модифікації – якщо не мінялася – не потрібно записувати, а якщо мінялася – вибирають ту, до якої менше зверталися.

Іноді сторінкова організація використовується і без вивантаження на зовнішню пам'ять. Наприклад, в ОС Nowell NetWare 3.x і 4.x вона використовується для дефрагментизации ОП.

Сегментний розподіл.

Сегментна віртуальна пам'ять передбачає переміщення даних сегментами – частинами віртуального адресного простору довільного розміру, отриманими з урахуваннямсмислового значення даних.

При сегментному розподілі пам'яті кожному сегменту можна визначити свої властивості: доступ, приналежність процесу і так далі Віртуальна адреса визначається номером сегменту і зсувом усередині сегменту. Фізична адреса визначається складанням базової (початкового) адреси сегменту і зсуву в сегменті. Сегменти мають різні розміри, базові адреси не є ступенями 2, і обчислення фізичної адреси виконується повільніше, ніж при сторінковій організації.

Сегментно-сторінковий розподіл.

Сегментно-сторінкова віртуальна пам'ять використовує дворівневе ділення: віртуальний адресний простір ділиться на сегменти, а потім сегменти діляться на сторінки. Одиниця переміщення – сторінка.

При цьому методі розподілу, досягаються достоїнства обох методів. Віртуальний адресний простір процесу розділений на віртуальні сегменти,

кожному сегменту визначені права доступу. Всі сегменти утворюють один віртуальний простір. Віртуальна адреса виходить складанням початкової віртуальної адреси сегменту із зсувом, а фізичний – механізмом переходу від віртуальних сторінок до фізичних. При створенні процесу, в пам'ять завантажується тільки частина сторінок, останні завантажуються в міру необхідності.

Лекція 15

10.4 Віртуальна пам'ять

Підміна оперативної пам'яті дисковою називається віртуалізацією.

Віртуальним називається ресурс, який споживачеві представляється таким, що володіє властивостями, якими він насправді не володіє. Використання віртуальної пам'яті дозволяє підвищити рівень мультипрограмування. При використанні інтерактивних процесів, потрібне велике число процесів, щоб забезпечити ефективне використання процесора. При виконанні рахункових задач, таких процесів потрібно не більш 3-4-х.

Віртуалізація ОП виконується програмними засобами ОС і схемами процесора, і включає вирішення наступних задач:

- Розміщення даних в пристроях різного типу, що запам'ятовують, – ОП і на диску;
- Вибір образів процесів або їх частин для переміщення з ОП на диск і назад;
- Переміщення, в міру необхідності, даних між диском і ОП;
- Перетворення віртуальних адрес у фізичних.

Віртуалізація може бути організована на основі двох підходів:

Свопінг – образи процесів вивантажуються на диск і повертаються в ОП цілком.

Віртуальна пам'ять – між ОП і диском переміщаються частини образів процесів (сегменти, сторінки).

Не слід плутати віртуалізацію з *оверлеями* – частинами програми, що завантажуються і вивантажуваними по командах самого процесу для економії займаної оперативної пам'яті.

Свопінг – окремий випадок віртуальної пам'яті, він організований простіше, але володіє двома недоліками: зайвими переміщеннями ОП-ДІСК, і неможливістю виконати програму, розмір якої перевищує розмір ОП.

10.5 Кеш

Вся пам'ять ЕОМ побудована за ієрархічним принципом. Вона складається з декількох пристроїв, що відрізняються об'ємом, часом доступу, вартістю. Ієрархія пам'яті відображена на рис. 10.6.

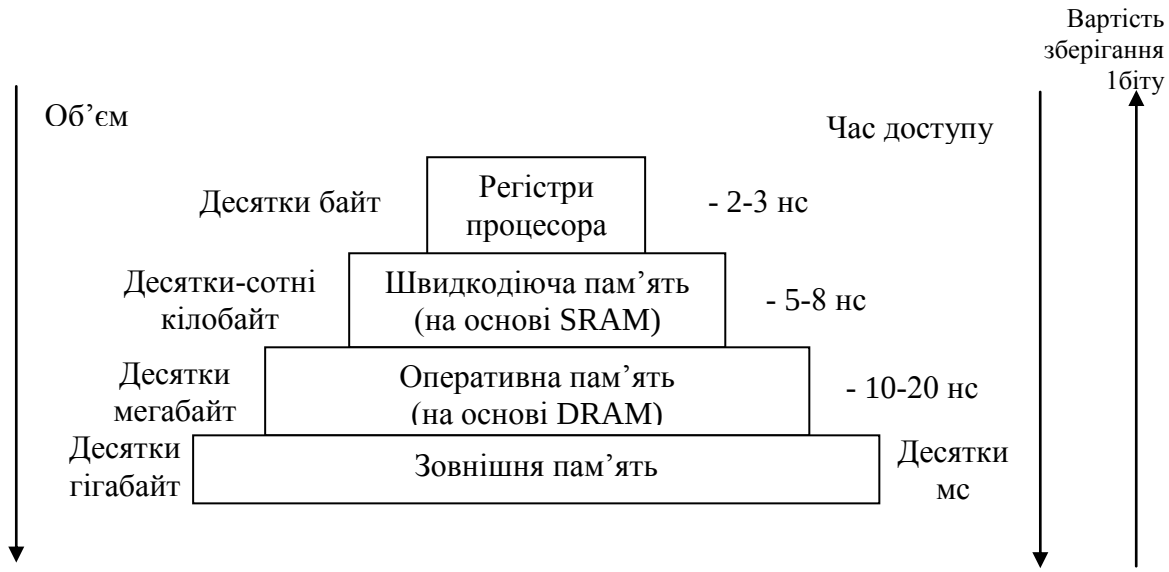


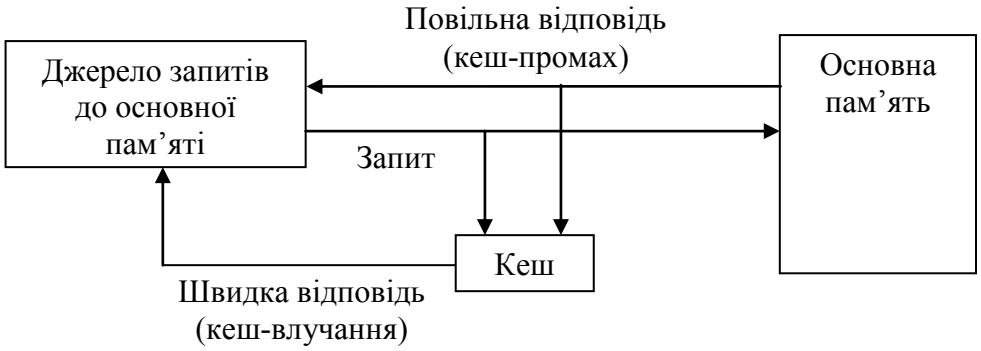
Рис.10.6 – Ієрархія запам’ятовуючих пристроїв

Із зменшенням часу доступу зростає вартість зберігання біта інформації. Щоб частково зменшити час доступу до всіх пристроїв, починаючи з другого зверху шару, використовується механізм кешування.

Кеш-пам'ять, або просто *кеш* – це спосіб сумісного функціонування двох типів пристроїв (що відрізняються часом доступу і вартістю зберігання інформації), що запам'ятовують, який за рахунок динамічного копіювання часто використовуваної інформації з «повільного» ЗУ в «швидке» ЗУ дозволяє, з одного боку, зменшити середній час доступу, з іншої – заощадити на дорожчій швидкій пам'яті.

Кеш повинен бути прозорий для програм, все виконується автоматично, засобами апаратури і ОС.

Кешем також називають швидке ЗУ, у відмінності від повільного. Для диска роль кеша грає ОП, для ОП кешем виступає швидка пам'ять (рис. 10.7).



Структура кеш-пам'яті

Адреса даних в основної пам'яті	Данні	Управляюча інформація

Рис. 10.7 – Схема функціонування кеш-пам'яті

Кеш-пам'ять є таблицею записів, кожна з яких може містити зведення про елемент даних з 3-х частин:

1. Значення адреси елементу даних, яке він має в ОП.
2. Само значення елементу даних.
3. Додаткову інформацію: ознака модифікації і ознака дійсності (зайнятості рядка).

Якщо при зверненні до кеша (за адресою із запиту) дані знайшлися – це кеш-попадання, інакше – кеш-промах. У сучасних реалізаціях імовірність кеш-попадання > 90%. Це пояснюється *часовою локальністю* і *просторовою локальністю* даних програм.

Часова локальність – якщо відбулося звернення за деякою адресою, то наступне звернення за цією адресою з великою імовірністю відбудеться найближчим часом.

Просторова локальність – якщо відбулося звернення за деякою адресою, то з високим ступенем імовірності найближчим часом відбудеться звернення до сусідніх адрес.

При читанні даних, якщо вони не в кеші, паралельно з вибором з ОП проводиться запис в кеш. При записі даних, якщо вони поза кешем, запис йде в ОП, якщо в кеші – запис або в кеш і в ОП (*наскрізний запис*) або тільки в кеш, але із зміною ознаки модифікації (*зворотний запис*). У останньому випадку, якщо необхідно звільнити модифікований запис кеша для іншого запису, запис, що видаляється, заноситься в оперативний запис.

Відзначимо, що кеш може бути дворівневий, при цьому кеш 1-го рівня має менший об'єм, але є більш швидким, розміщується всередині процесора, а кеш 2-го рівня грає таку ж роль для кеша 1-го рівня, як ОП для кеша 2-го рівня.

Лекція 16

Тема 11 Файлова система сучасної ОС

Введення і виведення – допоміжні функції обчислювальної системи. (Головна функція – перетворення інформації). Проте, якщо головну функцію реалізують, в основному, застосування, то введення і виведення «лягає на плечі» операційної системи, оскільки це обмін інформацією між ОП і зовнішніми пристроями ЕОМ. Проблему введення-виведення зазвичай розглядають такою, що складається з 2-частей: роботи власне підсистеми введення-виведення і способів організації інформації на зовнішніх пристроях (ЗП) – файлової системи.

11.1 Введення-виведення

Основними компонентами підсистеми введення-виведення є драйвери, керівники зовнішніми пристроями. З деякою умовністю, до неї можна віднести і диспетчер переривань, оскільки джерелом переривань, в основному, є зовнішні пристрої.

Задачами ОС по управлінню файлами і пристроями, є:

- Організація паралельної роботи пристроїв введення-виведення і процесора;
- Узгодження швидкостей обміну і кешування даних при обміні;
- Розподіл пристроїв і даних між процесами;
- Забезпечення зручного логічного інтерфейсу між ЗП і рештою частини системи (ядром обчислювальної системи);
- Підтримка розширюваної бібліотеки драйверів;
- Підтримка декількох файлових систем;
- Підтримка синхронних і асинхронних операцій введення-виведення.

Розглянемо ці задачі.

Організація паралельної роботи пристроїв введення-виведення і процесора.

Кожний ЗП має спеціальний електронний блок управління – контроллер. Драйвер (системний програмний модуль) передає контроллеру інформацію, що управляє, визначає, що робити з даними. При виведенні, драйвер також передає ці дані, при введенні – приймає їх від контроллера. Під управлінням контроллера, ЗП може якийсь час функціонувати самостійно, без участі драйвера, таким чином, процесор може бути використаний для виконання інших процесів.

Від підсистеми введення-виведення потрібно так спланувати роботу задіяних ВУ (точніше за їх контроллери), щоб з одного боку, забезпечить максимальну пропускну спроможність системи в цілому, з іншого – щоб забезпечити прийнятний час реакції кожного драйвера на незалежні події контроллера. Крім того, необхідно мінімізувати завантаження процесора задачами введення-виведення, звільнивши його для призначених для користувача процесів.

Відзначимо, що підсистема введення-виведення працює в системі реального часу. Для забезпечення прийнятного часу реакції, всі драйвери (або їх частини) розподіляються по декількох пріоритетах. Перемикання виконує загальний диспетчер переривань ОС.

Узгодження швидкостей обміну і кешування даних при обміні. Забезпечується буферизацією даних і синхронізацією доступу до буферів. У системах, де потрібна висока швидкість передачі даних, велика частина ОП віддається не під застосування, а під буфери даних. Проте, частіше роль таких буферів виконує не ОП а диск, і файли буфера носять назву «спул-файлов». Типовий процес спулінга – виведення даних на принтер, коли на диску утворюється черга принт-файлов.

Часто для підвищення швидкості роботи ЗП і розвантаження ОП, буфер організовується на самому контроллері. Приклад – відеокарта з відеопам'яттю, часто сумірною за об'ємом з ОП.

Паралельно з питаннями узгодження швидкостей, буферизація вирішує і задачу прискорення роботи по введенню-виведенню за рахунок кешування

даних. Наприклад, у всіх операціях введення-виведення створюється кеш на диску.

Розподіл пристроїв і даних між процесами

Пристрої введення-виведення можуть надаватися як в монопольне, так і в сумісне використання. При цьому ОС повинна контролювати доступ процесам, як і до інших ресурсів, шляхом перевірки прав користувача або групи користувачів, від імені яких діє процес.

Зазвичай послідовний або паралельний порти надаються в монопольне використання, а диск – в те, що розділяється.

Забезпечення зручного логічного інтерфейсу між ВУ і рештою частини системи

Різноманітність пристроїв введення-виведення робить особливо актуальною функцію ОС по створенню єдиного інтерфейсу між застосуваннями і ЗП. Звичайно це потримало файлової моделі інтерпретації ЗП. Файлова модель, однак, є дуже бідною для багатьох пристроїв, тому вона є базовою, над якою підсистема введення-виведення будує змістовнішу модель пристроїв конкретного вигляду.

Підтримка розширюваної бібліотеки драйверів

Достоїнством будь-якої ОС є наявність в ній широкого набору драйверів. Приклад: OS/2 потерпіла поразку Windows 3.x через відсутність драйверів. Щоб не було нестачі в драйверах, повинен бути відкритим інтерфейс між драйверами і частинами ОС, з якими вони взаємодіють. Тоді їх можуть розробляти виготівники апаратури.

Драйвери взаємодіють, з одного боку, з модулями ядра, з іншої – з контроллерами ВУ. Тому існує 2 типи інтерфейсів: «драйвер-ядро (DKI) » і «драйвер-пристрій» (DDI - Driver Device Interface). Перший повинен бути стандартизован, другий, – бажано відокремлений від апаратури примітивними функціями ОС.

Драйвери бажано уміти завантажувати і вивантажувати з ОП у міру потреби.

Підтримка декількох файлових систем

Дані на дисках організовуються у файлові системи (ФС), властивості ФС багато в чому визначають властивості самої ОС. Популярність ФС приводить до її міграції з однієї ОС в іншу. Тому, підтримка декілька ФС також важлива, як і підтримка різних пристроїв.

Підтримка синхронних і асинхронних операцій введення-виведення

Підсистема введення-виведення повинна надавати своїм клієнтам – застосуванням і модулю ядра можливість виконувати як синхронні, так і асинхронні операції введення-виведення. Зазвичай системні виклики введення-виведення виконуються в синхронному режимі, а внутрішні виклики ядра – в асинхронному режимі.

11.2 Логічна організація файлової системи

Одним з основних завдань ОС є надання користувачам зручності при роботі з наборами даних, що зберігаються на дисках. Для цього ОС підміняє фізичну структуру даних на диску логічною моделлю. Остання пропонує розглядати набори даних у вигляді дерева каталогів, в якому самі набори даних є кінцевими елементами – «листям». При цьому набори даних отримують назву «файлу». Як і файлова система в цілому, кожен файл має фізичну структуру і логічну організацію.

Файл – це іменована область зовнішньої пам'яті, в яку можна записати і з якої можна прочитати дані.

Призначенням файлів є:

- Довготривале і надійне зберігання інформації;
- Сумісне використання інформації декількома застосуваннями (користувачами).

Файлова система – це частина ОС, що включає сукупність всіх файлів на диску, набори службових структур даних (каталоги, дескриптори файлів, списки вільних кластерів диска) і комплекс програмних засобів для операцій з файлами (створення, читання, видалення і так далі).

Файлова система грає роль проміжного шару, що екранує всі складнощі фізичної організації файлів на диску і дозволяє застосуванням працювати з більш простій логічною моделлю даних.

Завдання, які повинна вирішувати ФС, залежать від типу ОС. У найбільш простій (однозадачній) ОС, наприклад, MS DOS, ці завдання включають:

- Іменування файлів;
- Програмний інтерфейс для застосувань
- Відображення логічної моделі ФС на фізичну організацію носія;
- Стійкість ФС до збоїв живлення, помилок апаратних і програмних засобів обчислювальної системи.

У багатозадачних однокористувачевих ОС (наприклад, OS/2, Win9x) з'являється ще задача сумісного використання файлу декількома процесами, тобто використання файлу як ресурсу, що розділяється.

У багатокористувачевих ОС з'являється задача захисту файлів одного користувача від іншого, тобто розмежування доступу до файлів.

Нарешті, ФС у складі мережної ОС додатково з'являються файлові мережні служби, і в задачі ФС входять:

- Реалізація сервера мережної файлової системи;
- Реалізація клієнта мережної файлової системи;
- Забезпечення інтерфейсу мережної файлової системи;
- Підтримка протоколу клієнт-серверної файлової системи.

11.3 Логічна побудова файлів

Всі файли діляться на :

1. Звичайні файли – що містять довільну інформацію. ОС не контролює їх зміст. Єдині файли, які ОС повинна розпізнавати і контролювати, – це файли з виконуваними модулями даної ОС.

2. Каталоги – це особливі файли фіксованої структури, що створюються самою ОС, і що містять інформацію про інші файли

(звичайних або каталогах), зареєстровані в даному каталозі. Каталоги встановлюють відповідність між іменами файлів і їх характеристиками.

3. Спеціальні файли – це фіктивні файли, що асоціюються з ПВВ і використовувані для уніфікації механізмів доступу до файлів і ЗП. Спеціальні файли дозволяють користувачам виконувати команди введення-виведення на ПВВ у формі читання або запису даних у файл.

Сучасні ФС підтримують і інші типи файлів, які тут не розглядаються.

У більшості ОС файлові системи будуються за ієрархічним принципом: кожен файл або каталог (окрім кореневого каталога) зареєстрований тільки в одному каталозі. Прикладами таких ОС є MS DOS, OS/2,Win 9x, Win NT . Але можуть існувати і мережні файлові структури, коли один і той же файл зареєстрований в двох і більш каталогах. Прикладом є ОС UNIX.

У ФС використовуються наступні поняття імен файлів:

Просте (коротке) ім'я ідентифікує файл в межах свого каталогу. У різних ОС різні обмеження на довжину і допустимий алфавіт для імен. У DOS – це не більше 8 символів на ім'я і 3-х - на розширення, в Windows (32-х розр.) – до 255 символів на ім'я. У ієрархічних ФС різним файлам можна мати однакові імена, тільки якщо вони знаходяться в різних каталогах. Для однозначної ідентифікації використовуються повні імена.

Повне ім'я – це ланцюжок простих символьних імен всіх каталогів, через які проходить шлях від кореня до даного файлу. Повне ім'я – складене, імена розділяються межу собою прямим або зворотним слешем. У ієрархічній системі між файлом і повним ім'ям є пряма однозначна відповідність «один файл – одне повне ім'я». У ФС з мережною структурою, одному файлу може відповідати декілька повних імен. У них є відповідність «один файл – декілька повних імен», в будь-якому разі, повне ім'я однозначно визначає файл.

Відносне ім'я – це складене ім'я, що показує шлях від поточного каталогу до файлу. Поняття поточного каталогу визначається до ОС як повного імені каталогу, яке приписується за умовчанням до короткого імені для отримання повного імені файлу. Поточний каталог задає (вибирає) користувач за допомогою команди ОС.

У деяких ОС є можливість привласнити одному файлу декілька простих імен (псевдонімів). В цьому випадку, для однозначного визначення файлу, ОС повинна привласнювати файлам також *унікальні імена.* Наприклад, в UNIX для цього використовується номер індексного дескриптора файлу.

Оскільки у ВС може бути декілька дисків, однозначність повних імен задається або додаванням імені диска, або монтуванням файлової структури одного диска як каталог на інший диск, званий в цьому випадку системним. Так, наприклад, часто робиться в ОС UNIX.

У ФС кожному файлу приписується деякий набір атрибутів. Як атрибути можуть виступати:

- Тип файлу (звичайний, каталог, спеціальний і так далі);
- Власник файлу;
- Творець файлу;
- Пароль доступу до файлу;
- Інформація про дозволені операції з файлом;
- Часи створення останньої модифікації, останнього звернення;
- Поточний розмір;
- Максимальний розмір;
- Ознаки:
    1. «тільки для читання»;
    2. «прихований файл»;
    3. «системний файл»;
    4. «архівний файл»;
    5. «двійковий/символьний»;

6. «тимчасовий файл»;
7. блокування.

- Довжина запису;
- Покажчик на ключове поле в записі;
- Довжина ключа.

У різних ОС набори атрибутів відрізняються один від одного. Крім того, іноді атрибути зберігаються з ім'ям файлу в каталозі (MS DOS, Windows), іноді в каталозі окрім імені зберігається тільки посилання на рядок таблиці з характеристиками файлу. Наприклад, у файловій системі UFS операційної системи UNIX в каталозі зберігається тільки ім'я файлу і номер індексного дескриптора цього файлу, а всі атрибути занесені в таблицю дескрипторів. Підхід, коли ім'я файлу відокремлене від його атрибутів, робить систему гнучкішої.

З погляду логічної організації побудови окремого файлу, вони діляться на файли прямого доступу (з однаковою довжиною і типом запису), файли послідовного доступу (з довільною довжиною і ознаками кінців записів) і індексно-послідовного доступу, що складаються з 2-х частин, – індексною (прямого доступу), у формітаблиці індекс-адреса і послідовною, такою, що містить записи довільної довжини.

Лекція 17

11.4 Фізична організація файлової системи

Фізична організація зберігання файлів на диску зовсім не відповідає логічній організації ФС. Принципи розміщення файлів, каталогів і службової інформації на диску описуються фізичною організацією ФС. Оскільки основним пристроєм зберігання інформації є жорсткий незнімний диск (вінчестер), далі фізична організація розглядається стосовно нього.

Диск зазвичай складається з пакету пластин на загальній осі, круглі пластини покриті феромагнітним шаром, над ними розташовані магнітні зчитуючі і записуючі голівки.

Низькорівневе форматування створює на диску доріжки і сектори. Останні – зазвичай по 512 байт. Це найменша одиниця обміну, що адресується, даними на диску. Число секторів на кожній доріжці однаково. Циліндр – сукупність доріжок на одному радіусі. Для адресації даних необхідно задати номер циліндра, номер голівки (поверхні) і номер сектора. Низькорівневе форматування не залежить від ОС. Нумерація доріжок (циліндрів) починається з 0 від зовнішнього радіусу.

ОС зазвичай користується власними одиницями дискового простору – кластерами. Кожному файлу виділяється один або декілька кластерів на диску. Розмітка диска під конкретний тип ФС виконується високорівневим форматуванням. При цьому на диск обов'язково записується завантажувач – програма читання інформації з цього диска.

Перед високорівневим форматуванням, диск може бути розбитий на частини – розділи (логічні пристрої, томи). На фізичному пристрої може бути декілька ФС, але в кожному розділі – тільки одна ФС. Найчастіше, один розділ – один логічний пристрій, але може бути декілька розділів (не обов'язково на одному фізичному диску), об'єднаних в один логічний пристрій. Прикладом є RAID-массив.

Розділи мають блоки однакового розміру, але кластери можуть відрізнятися. Розмір кластера – від 512 байт до 64 Кбайт.

11.5 Фізична організація і адресація файлу

Критеріями фізичної організації побудови файлу є:

- Швидкість доступу до даним;
- Об'єм адресної (службовою) інформації;
- Ступінь фрагментованості диску;
- Максимально можливий розмір файлу.

Основними можливими варіантами розміщення є:

- Безперервне розміщення в послідовних кластерах диска;
- Розміщення у вигляді списку зв'язаних кластерів;
- Розміщення з використанням зв'язаного списку індексів (FAT);
- Розміщення з адресацією файлу у вигляді списку кластерів файлу (UFS – UNIX, NTFS – Win NT/200). У останньому випадку, замість переліку кластерів використовується перелік ділянок, кожен з яких задається парою чисел – номером 1-го кластера і числом підряд кластерів, що йдуть.

Недоліки є у кожного методу (1- фрагментизація; 2- складності доступу до внутрішніх кластерів і не кратність довжини даних ступеня 2; 3 – швидкість читання; 4 – змінна довжина адресації).

Найбільш поширеними є ФС типу FAT, UFS і NTFS.

11.6 Фізична організація FAT

Логічний розділ складається з:

- Завантажувального сектора з програмою початкового завантаження (512 байт);
- Основній копії FAT (розмір залежить від об'єму диска);
- Копії FAT;

- Кореневого каталогу фіксованого розміру (32 сектори – 16 Кбайт), куди можна занести 512 записів про набір даних. Кожен запис – 32 байти;
- Область даних – вся решта область розділу.

FAT підтримує 2 типи набору даних – файл і каталог. Пам'ять розподіляється тільки з області даних, порціями в кластер.

Таблиця FAT – масив індексних покажчиків, кожному покажчику відповідає один кластер. Покажчик може мати значення:

- Кластер вільний;
- Кластер належить файлу і не останній;
- Кластер – останній у файлі;
- Кластер збійний;
- Кластер резервний.

Розмір покажчика буває 12, 16 або 32 розряди, що дозволяє посилатися на 4096, 65536 або більше 4 млр. кластерів.

FAT12 – для дисків до 16 Мбайт, FAT16 – для дисків до 512 Мбайт (максимально -4Гбайт), FAT32 – для дисків з практично необмеженим розміром (до 128 Тбайт). Використовуваний метод зберігання адресної інформації не дуже надійний, оскільки збій в списку індексів призводить до втрати всього хвоста файлу.

У FAT16, як і FAT 12, використовуються імена 8+3 символу, але в ОС Win32 можливе застосування довгих імен, з 255 символів UNICODE (по 2 байти на символ). Перші 8+3 символу зберігаються в цьому випадку також в основному записі каталога а подальші символи – по 13 штук (26 байт) + 6 службових байт – окремими порціями за основним записом. Аналогічно влаштована ФС FAT32.

Лекція 18

11.7 Фізична організація NTFS

NTFS з'явилася пізніше FAT і HPFS (для OS/2). Відмінні риси цієї системи:

- Підтримка великих дисків і файлів (об'ємом до 2^{64} байт);
- Відновлюваність після збоїв і відмов програм і апаратури управління дисками;
- Висока швидкість операцій;
- Відносно низький рівень фрагментизації;
- Гнучка структура атрибутів файлів, можливість додавати нові типи атрибутів;
- Підтримка довгих символьних імен;
- Контроль доступу до файлів і каталогів.

Базовою одиницею розподілу дискового простору є відрізок – безперервна послідовність логічних (фізичних) кластерів (LCN). Оскільки файл може складатися з декількох відрізків, для нього визначено поняття віртуального кластера (VCN) – відносного порядкового номера кластера усередині файлу.

На томі (логічному диску) розміщуються тільки файли або частини файлів. Каталоги представляють такі ж файли (у тому числі і кореневий). На початку тому розміщується завантажувальний сектор, потім головна таблиця файлів (Master File Table – MFT). MFT – це файл, що містить по одній або більше записів (розміром, зазвичай, 2^K) для кожного файлу. У нульовому записі – відомості про сам файл MFT. У подальших 15 записах - відомості про стандартні службові файли. Далі йдуть відомості про всі створювані на томі файли.

Як системні файли, окрім самої MFT, виступають:

- Копія перших трьох записів MFT;

- Файл журналу транзакцій, використовуваний для відновлення файлової системи після збою;
- Файл з ім'ям тому, версією NTFS, іншою інформацією про том;
- Файл з таблицею визначення атрибутів. Для кожного атрибуту зберігається ім'я типу, номер, опис;
- Файл кореневого каталогу;
- Файл з бітовою картою логічних кластерів тому (по 1 біту на кластер, для визначення зайнятий він або вільний);
- Файл з адресою завантажувального сектора тому;
- Файл із списком поганих кластерів тому;
- Файл із таблицею квот – кількості дискового простору, що виділяється кожному користувачеві;
- Файл з таблицею перетворення регістра символів для кодування Unicode.

Чотири записи (11-15) в MFT зарезервовано для подальших модифікацій NTFS.

Кожен файл том ідентифікується номером, співпадаючим з порядковим номером запису про нього в MFT. Кожен файл складається з групи атрибутів, причому і ім'я файлу і дані файлу і стандартна інформація про нього розглядаються як атрибути. Таким чином, будь-який файл складається тільки з набору атрибутів.

Кожен атрибут містить заголовок (що складається з типу, довжини і імені) і значення. Деякі атрибути є системними, вони визначені спочатку, користувач може додати свої атрибути. У системний набір атрибутів входять:

- Attribute List – список атрибутів (використовується рідко, якщо список атрибутів не поміщається в перший запис MFT для файлу;
- File Name – ім'я файлу – містить довге ім'я файлу в Unicode і номері запису в таблиці MFT для батьківського каталогу. Якщо файл зафіксований в декількох каталогах, то у нього буде декілька атрибутів типу Name, поодинці для кожного каталогу;

- MS-DOS Name – ім'я файлу у форматі MS-DOS (8+3);
- Version – номер останньої версії файлу;
- Security Description – атрибут зберігає інформацію про захист файлу - список прав доступу ACL і поле аудиту – вказівки, які операції з файлом потрібно реєструвати;
- Volume Version – версія тому (тільки для системних файлів);
- Volume Name – ім'я тому (тільки для системних файлів);
- MFT bitmap – карта використання кластерів тому (тільки для системного файлу);
- Index Root – Корінь двійкового дерева для пошуку файлів в каталозі (тільки для каталогів);
- Index Allocation – розміщення продовжень каталогу, коли він не поміщається в один запис (тобто для великих каталогів);
- Standard Information – стандартна інформація про файл – зокрема, час створення, останній модифікації файлу і так далі.

Якщо дані файлу займають мало місця (в межах 1500 байт) те весь він поміщається в запис MFT разом з рештою атрибутів. Такі файли називаються резидентними, а запис в MFT складається з 4-х атрибутів:

1. SI – стандартній інформації;
2. FN – імені файлу;
3. Data – власне даних файлу;
4. SD – дескриптора безпеки.

Якщо він довший, то запис містить засоби пошуку відрізків, складових файл. Проте атрибути SI, FN і SD містяться завжди. Великий файл замість даних містить списки відрізків, де розташовані дані, дуже великі дані замість атрибуту дані містить номер запису MFT, де містяться списки відрізків файлу, а надвеликі файли замість атрибуту даних містять список записів MFT, в яких зберігаються списки відрізків з даними файлу.

Кожен каталог NTFS складається з резидентної частини, що зберігається в записі MFT, і можливо (для довгих каталогів) додаткових

відрізків зовні MFT. При цьому в резидентній частині залишаються тільки імена файлів, що завершують кожен відрізок каталогу, але доповнюють посиланням на початок такого відрізка. Відзначимо, що атрибути SI, FN і SD також зберігаються в резидентній частині каталогу.

11.8 Файлові операції

Будь-яка файлова система повинна надавати (користувачам) набір операцій з файлами у формі системних викликів. При будь-яких видах робіт з файлами, ОС повинна виконати групу універсальних дій для кожного файлу і якісь унікальні операції. Універсальні дії включають:

1. Пошук характеристик файлу по його символьному імені;
2. Копіювання характеристик файлу в ОП;
3. Перевірку прав користувача на виконання запитаних операцій;
4. Очищення області пам'яті, тимчасово зайнятої прочитаними характеристиками.

До унікальних для кожного файлу операцій відносяться читання, запис, видалення файлу або його частини, читання і зміна атрибутів і так далі.

ОС можуть виконувати послідовності дій над файлами двома способами:

- Для кожної операції виконуються як універсальні, так і унікальні дії. Це схема без запам'ятовування стану файлу. Вона стійкіша до збоїв в роботі і тому іноді використовується в розподілених файлових системах (наприклад, в Network File System компанії Sun).
- Всі універсальні дії для файлу виконуються на початку і наприкінці роботи з ним, а для кожної проміжної операції виконуються тільки унікальні операції. Цей спосіб економічніший і швидший, використовується в більшості сучасних систем.

Приклад роботи за першим способом:

open read1 close open read2 close open read3 close

Приклад роботи за другим способом:

open read1 read2 read3 close.

При другому способі, універсальні дії об'єднуються в системні виклики OPEN (що виконує дії 1-3) і CLOSE, що виконує дію 4.

При відкритті файлу, в ОП формується так званий дескриптор файлу, в якому відображують деякі характеристики поточного стану файлу (чи був він змінений, чи заблокований він якимсь процесом, скільки процесів з ним працюють і так далі) Окрім дескриптора, при кожному новому відкритті файлу, в ОП формується структура **file**, в якій відбиваються покажчики, як на дескриптор файлу, так і на процес, що його відкрив. Там же зберігається тип дозволених операцій (читання, запис.), зсув у файлі (номер запису) і інша інформація. Зазвичай, для структур **file** в ОС відводиться фіксована область ОП, тому число одночасно відкритих файлів обмежене.

У ОС передбачені стандартні файли введення, виведення і повідомлення про помилки. Ці файли зазвичай автоматично відкриваються для кожного застосування, що запускається, і зв'язуються з системними пристроями введення і виведення, якими є клавіатура і дисплей. Проте є можливість перевизначити системні пристрої (файли) введення і виведення, що робиться як в UNIX, OS/2, так і в Windows символами > для виведення і < для введення.

Наприклад, команда

Dir>C:\list_c.txt виведе зміст поточного каталогу не на дисплей, а в текстовий файл list_c.txt у кореневий каталог диску С.

11.9 Контроль доступу до файлів

Файли – найбільш часто використовуваний вид ресурсів, що розділяються. ОС повинна контролювати доступ користувачів до цих ресурсів. Користувачі виступають суб'єктами доступу, ресурси, що розділяються, – об'єктами. Для кожного типу ресурсів існують свої набори можливих операцій. Для файлів це читання, запис, видалення, виконання, зміна атрибутів.

У Windows NT для будь-яких ресурсів, що розділяються, створюється уніфікована структура – об'єкт безпеки і для контролю доступу до таких ресурсів використовується загальний модуль, що входить в ядро, – менеджер безпеки.

Як суб'єкти доступу можуть виступати як окремі користувачі, так і групи користувачів. У кожного об'єкту доступу існує власник (окремий користувач або група). Власник має право виконувати з об'єктом будь-які допустимі операції. Окрім власника, існує особливий користувач, який має все має рацію по відношенню до будь-яких об'єктів системи. Він називається «Адміністратором».

Є два підходи до визначення прав доступу

- Вибірковий доступ, коли сам власник визначає допустимі операції з об'єктами. В цьому випадку немає ієрархії користувачів, а власник або адміністратор можуть надавати будь-якому користувачеві або групі будь-які права доступу до об'єкту.
- Мандатний доступ – коли система визначає права доступу до об'єкту по тому, до якої групи входить користувач. Права всіх членів групи однакові, вони можуть користуватися правами користувачів нижчих в ієрархії груп, але не можуть надавати свої права жодним членам нижчих груп користувачів.

Другий вид – надійніший, але менш гнучкіший, використовується в спеціалізованих системах з підвищеними вимогами до захисту інформації. У універсальних системах використовується перший підхід.

При вході в систему кожного зареєстрованого користувача породжується процес-оболонка, що містить ідентифікатори користувача, як власні, так і груп, в яких він полягає. У свою чергу, для кожного файлу або каталогу створюються *списки управління доступом (Access Control List – ACL).* Формат списку управління доступом представляє набір ідентифікаторів користувачів, з переліком допустимих операцій для кожного ідентифікатора. Будь-який процес, породжений користувачем, отримує «по

спадку» від оболонки список ідентифікаторів користувача. Якщо цей процес спробує виконати відкриття файлу для певних видів операцій, відбувається порівняння списку ідентифікаторів процесу з ACL і якщо для будь-якого ідентифікатора в ACL необхідна операція дозволена, доступ буде отримано.

У Windows NT реалізований гнучкий підхід отримання прав доступу. За умовчанням, адміністратор працює на рівні укрупнених операцій з файлами: *читати*, *писати* і *виконати*, але можливий перехід на рівень елементарних операцій: читати (R), писати (W), виконати (X), видалити (D), змінити список дозволених операцій (P), стати власником файлу (O).

Як вже згадувалося раніше, для будь-яких об'єктів, що розділяються, в Windows NT створюється об'єкт безпеки, контроль доступу до об'єкту виконується централізованим монітором безпеки (Security Reference Monitor). Крім того, в цій ОС є велика кількість зумовлених суб'єктів доступу – стандартних груп користувачів, заздалегідь наділених певними правами. Завданням адміністратора є віднесення знов реєстрованого користувача до тієї або іншої групи (або декільком групам). Звичайно, адміністратор може створити нові групи, або змінити права у тих, що існують.

Windows NT підтримує три класи операцій доступу, суб'єктів, що відрізняються типом, і об'єктів, беруть участь в операції.

- Дозволи (Permissions) – безліч операцій, які можуть бути визначені для суб'єктів всіх типів по відношенню до об'єктів будь-якого типу: файлам, каталогам, принтерам, сторінкам пам'яті і так далі.
- Має (User rights) рацію – визначаються для суб'єктів типу група на виконання деяких системних операцій: установку системного часу, архівацію файлів, виключення комп'ютера і тому подібне Тут виступає особливий об'єкт доступу – вся ОС в цілому. В основному має рацію, а не дозволи відрізняють одну вбудовану групу від іншої. Деякі права вбудованої групи також є вбудованими – їх не може змінити адміністратор.

- Можливості (User abilities) – визначаються для окремих користувачів на виконання дій, пов'язаних з формуванням їх операційного середовища, наприклад зміна головного меню, право використовувати пункт меню Run і так далі За умовчанням, можливості відкриті, але адміністратор може їх заборонити.

При вході користувача в систему, створюється т.з. токен доступу (access token) – структуру, що містить список ідентифікаторів користувача і його груп, список управління доступом ACL за умовчанням (для дозволів, що приписуються створюваним користувачем нових об'єктів), список прав користувача на виконання системних дій. У відмінності від UNIX, в Windows NT існує як список дозволених, так і список заборонених для користувача операцій.

Для файлів передбачено 3 стандартних дозволи для Windows NT або 5 (для Windows XP).

Дозвіл	Елемент. має рацію	Опис функцій (що вирішується)
Читання (Read)	R	Тільки читання (для Windows XP)
Читання і виконання (Read&Execute)	RX	Читання і виконання
Запис (Write)	W	Створення нового файлу, запис в той, що існує. Читання – ні (для Windows XP)
Змінити (Modify)	RWD	Читання запис і видалення файлу
Повний доступ	все	Все, зокрема зміна дозволів і прав володіння.

Окрім стандартних, існують спеціальні дозволи. Їх кількість різна для файлів і каталогів в різних ОС, наприклад для Windows XP їх 13 штук.

Правила для використання дозволів:

- Користувач не може працювати з файлом і каталогом, якщо не має явного дозволу або не входить до групи, що має явний дозвіл.

- Дозволи мають накопичувальний ефект (суммация дозволів по групах) окрім дозволу No Access, який відміняє всі дозволи.

За замовчанням, в Windows Explorer відображаються стандартні права, а перехід до індивідуальних відбувається тільки при виконанні деяких дій.

ЛІТЕРАТУРА

1. Шеховцов В. А. Операційні системи / В. А. Шеховцов – К.: Вид. гр. BHV, 2005. – 576 с.

2. Alapati Sam R. Modern Linux Administration» / Sam R Alapati , O‘Reilly Media , 2016 — 500 p.

3. Barrett D. Linux. Pocket Guide. Essential Commands / D. Barrett 3rd edition, O‘Reilly. 2016, – 272 p.

4. Linux Distributions – Facts and Figures. – Режим доступу: http://distrowatch.com/stats.php?section=popularity.

5. Readers' Choice Awards 2010. – Режим доступу: http://www.linuxjournal.com/content/readers-choice-awards-2010.

6. Five Best Virtual Machine Applications. – Режим доступу: http://lifehacker.com/5715803/best-virtual-machine-application-virtualbox .