МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КОЛЕДЖ КРЕМЕНЧУЦЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до самостійної роботи

з дисципліни «Операційні системи»

галузь знань 12 – Інформаційні технології

спеціальності 123 – Комп'ютерна інженерія

спеціалізації Обслуговування комп'ютерних систем і мереж

Кременчук

2017 рік

РОЗРОБЛЕНО ТА ВНЕСЕНО: цикловою комісією Комп'ютерної техніки коледжу Кременчуцького національного університету імені Михайла Остроградського

Затверджено на засіданні циклової комісії комп'ютерної техніки

Протокол від «_31_» __серпня__ 2017 року №_1_

Голова циклової комісії комп'ютерної техніки

_____ (_Почтовюк С.І._)
(підпис) (прізвище та ініціали)

РОЗРОБНИКИ ПРОГРАМИ:

викладач вищої категорії Луценко Юрій Тарасович

викладач вищої категорії Шинкаренко Людмила Миколаївна

Обговорено та рекомендовано до видання методичною радою коледжу Кременчуцького національного університету імені Михайла Остроградського

Протокол від «_31_» __серпня__ 2017 року №_1_

Голова _____ (_Левченко Р.В._)
(підпис) (прізвище та ініціали)

Тема 1. Вступ. Мета та задачі предмету. Основні концепції операційних систем.

Мета: провести інструктаж з техніки безпеки; ознайомити студентів з поняттям операційної системи та її призначенням; розглянути історію розвитку операційних систем; розглянути класифікацію та функції операційних систем.

План

1. Поняття операційної системи та її призначення.
2. Історія розвитку операційних систем.
3. Класифікація операційних систем.
4. Основні функції операційної системи.

Питання до самоконтролю

1. Які основні функції операційної системи? Чи немає між ними протиріч?
2. Наведіть кілька прикладів просторового і часового розподілу ресурсів комп'ютера. Від чого залежить вибір того чи іншого методу розподілу?
3. У чому полягає основна відмінність багатозадачних пакетних систем від систем з розподілом часу? Як можна в рамках однієї системи об'єднати можливості обох зазначених систем?
4. Чому більшість вбудованих систем розроблено як системи реального часу? Наведіть приклади вбудованих систем, для яких підтримка режиму реального часу не є обов'язковою.
5. Що спільного й у чому відмінності між мережною і розподіленою операційними системами? Яка з них складніша в реалізації і чому?

1. Поняття операційної системи, її призначення та функції.

Причиною появи операційних систем була необхідність створення зручних у використанні комп'ютерних систем. Комп'ютерні системи від самого початку розроблялися для розв'язання практичних задач користувачів. Оскільки робити це за допомогою лише апаратного забезпечення виявилося складно,

були створені прикладні програми. Для таких програм знадобилися загальні операції керування апаратним забезпеченням, розподілу апаратних ресурсів тощо. Ці операції згрупували в рамках окремого рівня програмного забезпечення, який і стали називати *операційною системою*.

Далі можливості операційних систем вийшли далеко за межі базового набору операцій, необхідних прикладним програмам, але проміжне становище таких систем між прикладними програмами й апаратним забезпеченням залишилося незмінним.

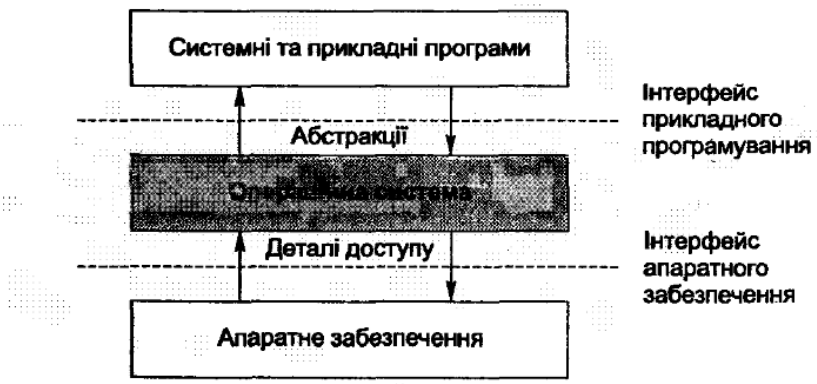
Операційна система (ОС) – це програмне забезпечення, що реалізує зв'язок між прикладними програмами й апаратними засобами комп'ютера.

Операційні системи забезпечують, по-перше, зручність використання комп'ютерної системи, по-друге, ефективність і надійність її роботи.

Перша функція властива ОС як розширеній машині, друга – ОС як розподілювача апаратних ресурсів.

За допомогою операційної системи у прикладного програміста (а через його програми і в користувача) має створюватися враження, що він працює з розширеною машиною.

Апаратне забезпечення комп'ютера недостатньо пристосоване до безпосереднього використання у програмах. Наприклад, якщо розглянути роботу із пристроями введення-виведення на рівні команд відповідних контролерів, то можна побачити, що набір таких команд обмежений, а для багатьох пристроїв – примітивний. Операційна система приховує такий інтерфейс апаратного забезпечення, замість нього програмістові пропонують інтерфейс прикладного програмування, що використовує поняття вищого рівня (їх називають абстракціями).



Взаємодія ОС із апаратним забезпеченням і застосуваннями

Наприклад, при роботі з диском типовою абстракцією є файл. Працювати з файлами простіше, ніж безпосередньо з контролером диска (не потрібно враховувати переміщення головок дисковода, запускати й зупиняти мотор тощо), внаслідок цього програміст може зосередитися на суті свого прикладного завдання. За взаємодію з контролером диска відповідає операційна система.

Виділення абстракцій дає змогу досягти того, що код ОС і прикладних програм не потребуватиме зміни при переході на нове апаратне забезпечення. Наприклад, якщо встановити на комп'ютері дисковий пристрій нового типу (за умови, що він підтримується ОС), всі його особливості будуть враховані на рівні ОС, а прикладні програми продовжуватимуть використовувати файли, як і раніше.

Така характеристика системи називається *апаратною незалежністю*. Можна сказати, що ОС надають апаратно-незалежне середовище для виконання прикладних програм.

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

Розрізняють два основні види розподілу ресурсів. У разі просторового розподілу ресурс доступний декільком споживачам одночасно, при цьому кожен із них може користуватися частиною ресурсу (так розподіляється пам'ять). У разі часового розподілу система ставить споживачів у чергу і згідно

з нею надає їм змогу користуватися всім ресурсом обмежений час (так розподіляється процесор в однопроцесорних системах).

При розподілі ресурсів ОС розв'язує можливі конфлікти, запобігає несанкціонованому доступу програм до тих ресурсів, на які вони не мають прав, забезпечує ефективну роботу комп'ютерної системи.

2. Історія розвитку операційних систем.

Перші операційні системи з'явилися в 50-ті роки і були системами *пакетної обробки*. Такі системи забезпечували послідовне виконання програм у пакетному режимі (без можливості взаємодії з користувачем). У певний момент часу в пам'яті могла перебувати тільки одна програма (системи були однозадачними), усі програми виконувалися на процесорі від початку до кінця. За такої ситуації ОС розглядали просто як набір стандартних служб, необхідних прикладним програмам і користувачам.

Наступним етапом стала *підтримка багатозадачності*. У багатозадачних системах у пам'ять комп'ютера стали завантажувати кілька програм, які виконувалися на процесорі навперемінно. При цьому розвивалися два напрями: багатозадачна пакетна обробка і розподіл часу. У багатозадачній пакетній обробці завантажені програми, як і раніше, виконувалися в пакетному режимі. У режимі розподілу часу із системою могли працювати одночасно кілька користувачів, кожному з яких надавався діалоговий термінал (пристрій, що складається із клавіатури і дисплея).

Підтримка багатозадачності потребувала реалізації в ОС засобів координації задач. Можна виділити три складові частини такої координації.

1. Захист критичних даних задачі від випадкового або навмисного доступу інших задач.
2. Забезпечення обміну даними між задачами.
3. Надання задачам справедливої частки ресурсів (пам'яті, процесора, дискового простору тощо).

Ще одним етапом стала поява ОС *персональних комп'ютерів*. Спочатку ці системи, як і ОС першого етапу, були однозадачними й надавали базовий набір стандартних служб (на цьому етапі важливим було впровадження графічного

інтерфейсу користувача). Подальший розвиток апаратного забезпечення дав змогу використати в таких системах засоби, розроблені для більших систем, насамперед багатозадачність і, як наслідок, координацію задач.

Є правило розвитку ОС для конкретної апаратної платформи: для більшості нових апаратних платформ ОС спочатку створюють як базовий набір стандартних служб, координацію задач реалізують у ній пізніше.

Багато сучасних ОС спочатку розроблялися для персональних комп'ютерів або були перенесені на них з інших апаратних платформ.

3. Класифікація сучасних операційних систем.

Розглянемо класифікацію сучасних операційних систем залежно від області їхнього застосування.

Насамперед відзначимо ОС великих ЕОМ (мейнфреймів). Основною характеристикою апаратного забезпечення, для якого їх розробляють, є продуктивність введення-виведення: великі ЕОМ оснащують значною кількістю периферійних пристроїв (дисків, терміналів, принтерів тощо). Такі комп'ютерні системи використовують для надійної обробки значних обсягів даних, при цьому ОС має ефективно підтримувати цю обробку (в пакетному режимі або в режимі розподілу 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 відіграє мережна підтримка. Є спеціалізовані серверні ОС, з яких виключені елементи, не пов'язані з виконанням їхніх основних функцій (наприклад, підтримка застосувань користувача). Нині 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 фірми Microsoft), інші є спрощеними версіями універсальних ОС. Особлива

увага в персональних ОС приділяється підтримці графічного інтерфейсу користувача і мультимедіа-технологій.

Виділяють також ОС реального часу. У такій системі кожна операція має бути гарантовано виконана в заданому часовому діапазоні. ОС реального часу можуть керувати польотом космічного корабля, технологічним процесом або демонстрацією відеороликів. Існують спеціалізовані ОС реального часу, такі як QNX і V×Works.

Ще однією категорією є вбудовані ОС. До них належать керуючі програми для різноманітних мікропроцесорних систем, які використовують у військовій техніці, системах побутової електроніки, смарт-картах та інших пристроях. До таких систем ставлять особливі вимоги: розміщення в малому обсязі пам'яті, підтримка спеціалізованих засобів введення-виведення, можливість прошивання в постійному запам'ятовувальному пристрої. Часто вбудовані ОС розробляються під конкретний пристрій; до універсальних систем належать Embedded Linux і Windows CE.

4. Функціональні компоненти операційних систем

Операційну систему можна розглядати як сукупність функціональних компонентів, кожен з яких відповідає за реалізацію певної функції системи. Спосіб побудови системи зі складових частин та їхній взаємозв'язок визначає *архітектура операційної системи*.

Однією з найважливіших функцій ОС є виконання прикладних програм. Код і дані прикладних програм зберігаються в комп'ютерній системі на диску в спеціальних виконуваних файлах. Після того як користувач або ОС вирішать запустити на виконання такий файл, у системі буде створено базову одиницю обчислювальної роботи, що називається *процесом* (process).

Процес – це програма під час її виконання.

Операційна система розподіляє ресурси між процесами. До таких ресурсів належать процесорний час, пам'ять, пристрої введення-виведення, дисковий простір у вигляді файлів. При розподілі пам'яті з кожним процесом пов'язується його адресний простір – набір адрес пам'яті, до яких йому дозволено доступ. В адресному просторі зберігаються код і дані процесу. При

розподілі дискового простору для кожного процесу формується список відкритих файлів, аналогічним чином розподіляють пристрої введення-виведення.

Процеси забезпечують захист ресурсів, якими вони володіють. Наприклад, до адресного простору процесу неможливо безпосередньо звернутися з інших процесів (він є захищеним), а при роботі з файлами може бути задано режим, що забороняє доступ до файла всім процесам, крім поточного.

Розподіл процесорного часу між процесами необхідний через те, що процесор виконує інструкції одну за одною (тобто в конкретний момент часу на ньому може фізично виконуватися тільки один процес), а для користувача процеси мають виглядати як послідовності інструкцій, виконувані паралельно. Щоб домогтися такого ефекту, ОС надає процесор кожному процесу на деякий короткий час, після чого перемикає процесор на інший процес; при цьому виконання процесів відновлюється з того місця, де їх було перервано. У багатопроцесорній системі процеси можуть виконуватися паралельно на різних процесорах.

Сучасні ОС крім багатозадачності можуть підтримувати багатопотоковість (multithreading), яка передбачає в рамках процесу наявність кількох послідовностей інструкцій (*потоків*, threads), які для користувача виконуються паралельно, подібно до самих процесів в ОС. На відміну від процесів потоки не забезпечують захисту ресурсів (наприклад, вони спільно використовують адресний простір свого процесу).

Під час виконання програмного коду процесор бере інструкції й дані з оперативної (основної) пам'яті комп'ютера. При цьому така пам'ять відображається у вигляді масиву байтів, кожен з яких має адресу.

Основна пам'ять є одним із видів ресурсів, розподілюваних між процесами. ОС відповідає за виділення пам'яті під захищений адресний простір процесу і за вивільнення пам'яті після того, як виконання процесу буде завершено. Обсяг пам'яті, доступний процесу, може змінюватися в ході виконання, у цьому разі говорять про *динамічний розподіл пам'яті*.

ОС повинна забезпечувати можливість виконання програм, які окремо або в сукупності перевищують за обсягом доступну основну пам'ять. Для цього в ній має бути реалізована технологія віртуальної пам'яті. Така технологія дає можливість розміщувати в основній пам'яті тільки ті інструкції й дані процесу, які потрібні в поточний момент часу, при цьому вміст іншої частини адресного простору зберігається на диску.

Операційна система відповідає за керування пристроями введення-виведення, підключеними до комп'ютера. Підтримка таких пристроїв в ОС звичайно здійснюється на двох рівнях. До першого, нижчого, рівня належать драйвери пристроїв — програмні модулі, які керують пристроями конкретного типу з урахуванням усіх їхніх особливостей. До другого рівня належить універсальний інтерфейс введення-виведення, зручний для використання у прикладних програмах.

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

Сучасні ОС надають великий вибір готових драйверів для конкретних периферійних пристроїв. Що більше пристроїв підтримує ОС, то більше в неї шансів на практичне використання.

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 якого здійснюється за іменем. Термін «файлова система» може вживатися для двох понять: принципу організації даних у вигляді файлів і конкретного набору даних (зазвичай відповідної частини диска), організованих відповідно до такого принципу. У рамках ОС може бути реалізована одночасна підтримка декількох файлових систем.

Файлові системи розглядають на логічному і фізичному рівнях. Логічний рівень визначає зовнішнє подання системи як сукупності файлів (які звичайно перебувають у каталогах), а також виконання операцій над файлами і

каталогами (створення, вилучення тощо). Фізичний рівень визначає принципи розміщення структур даних файлової системи на диску або іншому пристрої.

Мережні системи

Сучасні операційні системи пристосовані до роботи в мережі, їх називають *мережними операційними системами*. Засоби мережної підтримки дають ОС можливість:

- ✓ надавати локальні ресурси (дисковий простір, принтери тощо) у загальне 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, відповідальних за передачу даних між комп'ютерами відповідно до правил, 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. Є також розподілені ОС, які дають змогу об'єднати ресурси декількох комп'ютерів у розподілену систему. Вона виглядає для користувача як один комп'ютер з декількома процесорами, що працюють паралельно. Розподілені та багатопроцесорні системи є двома основними категоріями ОС, які використовують декілька процесорів.

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

Для захисту від несанкціонованого доступу ОС має забезпечувати наявність засобів аутентифікації користувачів (такі засоби дають змогу з'ясувати, чи є користувач тим, за кого себе видає; зазвичай для цього використовують систему паролів) та їхньої авторизації (дозволяють перевірити права користувача, що пройшов аутентифікацію, на виконання певної операції).

Розрізняють два типи засобів взаємодії користувача з ОС: командний інтерпретатор (shell) і графічний інтерфейс користувача (GUI). Командний

інтерпретатор дає змогу користувачам взаємодіяти з ОС, використовуючи спеціальну командну мову (інтерактивно або через запуск на виконання командних файлів). Команди такої мови змушують ОС виконувати певні дії (наприклад, запускати програми, працювати із файлами).

Графічний інтерфейс користувача надає йому можливість взаємодіяти з ОС, відкриваючи вікна і виконуючи команди за допомогою меню або кнопок. Підходи до реалізації графічного інтерфейсу доволі різноманітні: наприклад, у Windows-системах засоби його підтримки вбудовані в систему, а в UNIX вони є зовнішніми для системи і спираються на стандартні засоби керування введенням-виведенням.

Тема: «Архітектура операційних систем».

Мета: ознайомитися з поняттями архітектури операційних систем, ядра системи та системного програмного забезпечення; підходами до реалізації архітектури операційних систем; взаємодії операційної системи та апаратного забезпечення; взаємодії операційної системи та прикладних програм; архітектурою UNIX, Linux та Windows.

План

1. Означення архітектури операційних систем
2. Ядро системи та системне програмне забезпечення
3. Підходи до реалізації архітектури операційних систем
4. Взаємодія операційної системи та апаратного забезпечення
5. Взаємодія операційної системи та прикладних програм
6. Архітектура Windows XP
7. Архітектура UNIX і Linux

Питання для самоконтролю

1. Перелічіть причини, за якими ядро ОС має виконуватися в привілейованому режимі процесора.

2. Чи може процесор переходити у привілейований режим під час виконання програми користувача? Чи може така програма виконуватися виключно в привілейованому режимі?

3. У чому полягає головний недолік традиційної багаторівневої архітектури ОС? Чи має такий недолік архітектура з виділенням рівнів у монолітному ядрі?

4. Чому перехід до використання мікроядрової архітектури може спричинити зниження продуктивності ОС?

5. Автор Linux Лінус Торвальдс стверджує, що мобільність Linux має поширюватися на системи з «прийнятною» (reasonable) архітектурою. Які апаратні засоби повинна підтримувати така архітектура?

6. Наведіть переваги і недоліки реалізації взаємодії прикладної програми з операційною системою в Linux і Windows XP.

7. Чи не суперечить використання модулів ядра принципам монолітної архітектури Linux? Поясніть свою відповідь.

8. Перелічіть переваги і недоліки архітектури ОС, відповідно до якої віконна і графічна підсистеми в Windows XP виконуються в режимі ядра.

9. Чому деякі діагностичні утиліти Windows XP складаються з прикладної програми і драйвера пристрою?

Операційну систему можна розглядати як сукупність компонентів, кожен з яких відповідає за певні функції. Набір таких компонентів і порядок їхньої взаємодії один з одним та із зовнішнім середовищем визначається *архітектурою операційної системи*.

Базові поняття архітектури операційних систем

В ОС насамперед необхідно виділити набір фундаментальних можливостей, які надають її компоненти; ці базові можливості становлять *механізм* (mechanism).

З іншого боку, необхідно приймати рішення щодо використання зазначених можливостей; такі рішення визначають *політику* (policy). Отже, механізм показує, що реалізовано компонентом, а політика – як це можна використати.

Коли за реалізацію механізму і політики відповідають різні компоненти (механізм відокремлений від політики), спрощується розробка системи і підвищується її гнучкість. Компонентам, що реалізують механізм, не повинна бути доступна інформація про причини та цілі його застосування; усе, що потрібно від них, – це виконувати призначену їм роботу. Для таких компонентів використовують термін «*вільні від політики*» (policy-free). Компоненти, відповідальні за політику, мають оперувати вільними від неї компонентами як будівельними блоками, для них недоступна інформація про деталі реалізації механізму.

Прикладом відокремлення механізму від політики є керування введенням-виведенням. Базові механізми доступу до периферійних пристроїв реалізують драйвери. Політику використання цих механізмів задає програмне забезпечення, що здійснює введення-виведення.

Ядро системи. Привілейований режим і режим користувача. Базові компоненти ОС, які відповідають за найважливіші її функції, зазвичай перебувають у пам’яті постійно і виконуються у привілейованому режимі, називають *ядром операційної системи* (operating system kernel).

Існуючі на сьогодні підходи до проектування архітектури ОС по-різному визначають функціональність ядра. До найважливіших функцій ОС, виконання яких звичайно покладають на ядро, належать обробка переривань, керування пам’яттю, керування введенням-виведенням.

Основною характерною ознакою ядра є те, що воно виконується у привілейованому режимі. Для забезпечення ефективного керування ресурсами комп’ютера ОС повинна мати певні привілеї щодо прикладних програм. Треба, щоб прикладні програми не втручалися в роботу ОС, і водночас ОС повинна

мати можливість втрутитися в роботу будь-якої програми, наприклад для перемикання процесора або розв’язання конфлікту в боротьбі за ресурси. Для реалізації таких привілеїв потрібна апаратна підтримка: процесор має підтримувати принаймні два режими роботи – *привілейований* (захищений режим, режим ядра, kernel mode) і *режим користувача* (user mode). У режимі користувача недопустимі команди, які є критичними для роботи системи (перемикання задач, звертання до пам'яті за заданими межами, доступ до пристроїв введення-виведення тощо).

Після завантаження ядро перемикає процесор у привілейований режим і отримує цілковитий контроль над комп’ютером. Кожне застосування запускається і виконується в режимі користувача, де воно не має доступу до ресурсів ядра й інших програм. Коли потрібно виконати дію, реалізовану в ядрі, застосування робить *системний виклик* (system call). Ядро перехоплює його, перемикає процесор у привілейований режим, виконує дію, перемикає процесор назад у режим користувача і повертає результат застосування.

Системний виклик виконується повільніше за виклик функції, реалізованої в режимі користувача, через те що процесор двічі перемикається між режимами.

Для підвищення продуктивності в деяких ОС частина функціональності реалізована в режимі користувача, тому для доступу до неї системні виклики використовувати не потрібно.

Системне програмне забезпечення

Окрім ядра, важливими складниками роботи ОС є також застосування режиму користувача, які виконують системні функції. До такого системного програмного забезпечення належать:

- ~ *системні програми* (утиліти), наприклад: командний інтерпретатор, програми резервного копіювання та відновлення даних, засоби діагностики й адміністрування;
- ~ *системні бібліотеки*, у яких реалізовані функції, що використовуються у застосуваннях користувача.

Реалізація архітектури операційних систем.

✓ Монолітні системи. ОС, у яких усі 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 у привілейованому режимі, і помилка в кожному з компонентів є критичною).

Монолітність ядра не означає, що всі його компоненти мають постійно перебувати у пам’яті. Сучасні ОС дають можливість динамічно розміщувати в адресному просторі ядра фрагменти коду (модулі ядра). Реалізація модулів ядра дає можливість також досягти його розширюваності (для додання нової функціональності досить розробити і завантажити у пам'ять відповідний модуль).

✓ Багаторівневі системи. Компоненти багаторівневих ОС утворюють ієрархію рівнів (шарів, layers), кожен з яких спирається на функції попереднього рівня. Найнижчий рівень безпосередньо взаємодіє з апаратним забезпеченням, на найвищому рівні реалізуються системні виклики.

У традиційних багаторівневих ОС передача керування з верхнього рівня на нижній реалізується як системний виклик. Верхній рівень повинен мати права на виконання цього виклику, перевірка цих прав виконується за підтримки апаратного забезпечення. Практичне застосування цього підходу сьогодні обмежене через низьку продуктивність.

Рівні можуть виділятися й у монолітному ядрі; у такому разі вони підтримуються програмно і спричиняють спрощення реалізації системи. У монолітному ядрі визначають рівні, перелічені нижче.

~ Засоби абстрагування від устаткування, які взаємодіють із апаратним забезпеченням безпосередньо, звільняючи від реалізації такої взаємодії інші компоненти системи.

~ Базові засоби ядра, які відповідають за найфундаментальніші, найпростіші дії ядра, такі як запис блоку даних на диск. За допомогою цих засобів виконуються вказівки верхніх рівнів, пов'язані з керуванням ресурсами.

~ Засоби керування ресурсами (або менеджери ресурсів), що реалізують основні функції ОС (керування процесами, пам'яттю, введенням-виведенням тощо).

На цьому рівні приймаються найважливіші рішення з керування ресурсами, які виконуються з використанням базових засобів ядра.

~ Інтерфейс системних викликів, який служить для реалізації зв'язку із системним і прикладним програмним забезпеченням.

Розмежування базових засобів ядра і менеджерів ресурсів відповідає відокремленню механізмів від політики в архітектурі системи. Базові засоби ядра визначають механізми функціонування системи, менеджери ресурсів реалізують політику.

✓ Системи з мікроядром. Один із напрямів розвитку сучасних ОС полягає в тому, що у привілейованому режимі реалізована невелика частка

функцій ядра, які є *мікроядром* (microkernel). Інші функції ОС виконуються процесами режиму користувача (серверними процесами, серверами). Сервери можуть відповідати за підтримку файлової системи, за роботу із процесами, пам'яттю тощо.

Мікроядро здійснює зв'язок між компонентами системи і виконує базовий розподіл ресурсів. Щоб виконати системний виклик, процес (клієнтський процес, клієнт) звертається до мікроядра. Мікроядро посилає серверу запит, сервер виконує роботу і пересилає відповідь назад, а мікроядро переправляє його клієнтові. Клієнтами можуть бути не лише процеси користувача, а й інші модулі ОС.

Рис. 2.1. Виконання системного виклику в архітектурі з мікроядром

Перевагами мікроядрового підходу є:

~ невеликі розміри мікроядра, що спрощує його розробку й налагодження;

~ висока надійність системи, внаслідок того що сервери працюють у режимі користувача й у них немає прямого доступу до апаратного забезпечення;

~ більша гнучкість і розширюваність системи (непотрібні компоненти не займають місця в пам'яті, розширення функціональності системи зводиться до додавання в неї нового сервера);

~ можливість адаптації до умов мережі (спосіб обміну даними між клієнтом і сервером не залежить від того, зв'язані вони мережею чи перебувають на одному комп'ютері).

Головним недоліком мікроядрового підходу є зниження продуктивності. Замість двох перемикань режиму процесора у разі системного виклику

відбувається чотири (два – під час обміну між клієнтом і мікроядром, два – між сервером та мікроядром).

Зазначений недолік є, швидше, теоретичним, на практиці продуктивність і надійність мікроядра залежать насамперед від якості його реалізації. Так, в ОС QNX мікроядро займає кілька кілобайтів пам'яті й забезпечує мінімальний набір функцій, при цьому система за продуктивністю відповідає ОС реального часу.

✓ Концепція віртуальних машин. У системах віртуальних машин програмним шляхом створюють копії апаратного забезпечення (відбувається його емуляція). Ці копії (віртуальні машини) працюють паралельно, на кожній із них функціонує програмне забезпечення, з яким взаємодіють прикладні програми і користувачі.

Уперше концепція віртуальних машин була реалізована в 70-ті роки в операційній системі VM фірми IBM. Розглянемо архітектуру цієї ОС.

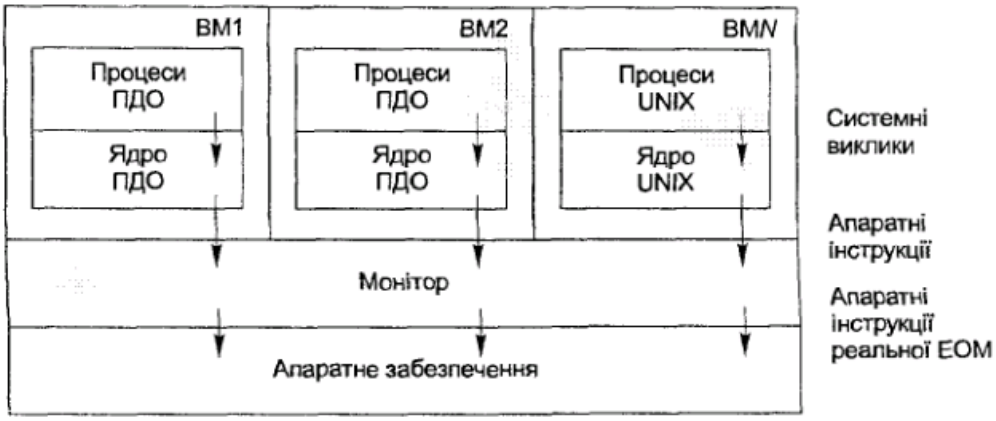


Рис. 2.2. Архітектура VM/370 [44]

Ядро системи, яке називалося монітором віртуальних машин (VM Monitor, МВМ), виконувалося на фізичній машині, безпосередньо взаємодіючи з її апаратним забезпеченням. Монітор реалізовував набір віртуальних машин (ВМ).

Кожна ВМ була точною копією апаратного забезпечення, на ній могла бути запущена будь-яка ОС, розроблена для цієї архітектури. Найчастіше на ВМ встановлювали спеціальну однокористувацьку ОС CMS (підсистема діалогової обробки, ПДО). На різних ВМ могли одночасно функціонувати різні ОС.

Коли програма, написана для ПДО, виконувала системний виклик, його перехоплювала копія ПДО, запущена на відповідній віртуальній машині. Потім ПДО виконувала відповідні апаратні інструкції, наприклад інструкції введення-виведення для читання диска. Ці інструкції перехоплював МВМ і перетворював їх на апаратні інструкції фізичної машини.

Віртуальні машини спільно використовували ресурси реального комп'ютера; наприклад, дисковий простір розподілявся між ними у вигляді віртуальних дисків, названих мінідисками. ОС, запущена у ВМ, використовувала мінідиски так само, як фізичні диски.

Сьогодні концепція віртуальних машин застосовується і в прикладному програмному забезпеченні; опис відповідних рішень (програмних емуляторів апаратного забезпечення, технології керованого коду) можна знайти на сайті супроводу.

Особливості архітектури: Windows. Основні компоненти Windows зображені на рис. 2.4.

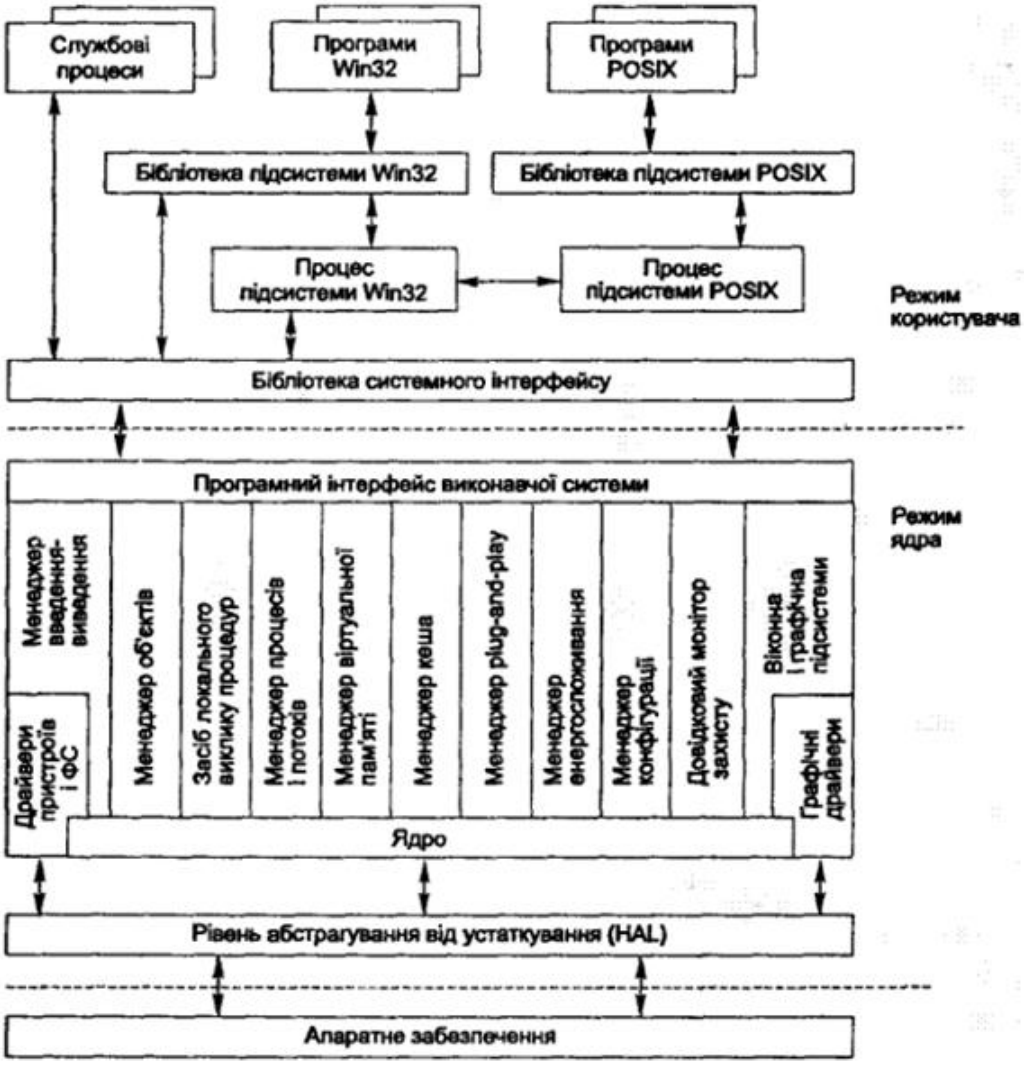


Рис. 2.4. Базові компоненти Windows

Деякі компоненти Windows XP виконуються у привілейованому режимі, інші компоненти – у режимі користувача.

Компоненти режиму ядра. Ядро ОС містить усі компоненти привілейованого режиму, однак у Windows поняття ядра закріплене тільки за одним із цих компонентів.

У Windows реалізовано рівень абстрагування від устаткування (у цій системі його називають *HAL*, (hardware abstraction layer). Для різних апаратних

конфігурацій фірма Microsoft або сторонні розробники можуть постачати різні реалізації HAL.

Хоча код HAL є дуже ефективним, його використання може знижувати продуктивність застосувань мультимедіа. У такому разі використовують спеціальний пакет DirectX, який дає змогу прикладним програмам звертатися безпосередньо до апаратного забезпечення, обминаючи HAL та інші рівні системи.

Ядро. Ядро Windows відповідає за базові операції системи. До його основних функцій належать:

~ перемикання контексту, збереження і відновлення стану потоків;

~ планування виконання потоків;

~ реалізація засобів підтримки апаратного забезпечення, складніших за засоби HAL (наприклад, передача керування оброблювачам переривань).

Ядро Windows відповідає базовим службам ОС і надає набір механізмів для реалізації політики керування ресурсами.

Основним завданням ядра є якомога ефективніше 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 не можна віднести до якогось певного класу ОС. Наприклад, хоча за функціональністю ядро системи відповідає поняттю мікроядра, для самої ОС не характерна класична мікроядрова архітектура, оскільки у привілейованому режимі виконуються й інші її компоненти.

Виконавча система. Виконавча система (ВС) Windows – це набір компонентів, відповідальних за найважливіші служби ОС (керування пам'яттю, процесами і потоками, введенням-виведенням тощо).

Компонентами ВС є передусім базові засоби підтримки. Ці засоби використовують у всій системі.

~ Менеджер об'єктів – відповідає за розподіл ресурсів у системі, підтримуючи їхнє універсальне подання через об'єкти.

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

Інші компоненти ВС реалізують найважливіші служби Windows.

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

~ Менеджер віртуальної пам’яті – реалізує керування пам’яттю в системі, насамперед підтримку віртуальної пам’яті.

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

~ Менеджер кеша – керує кешуванням для системи введення-виведення. Часто використовувані 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 із системним реєстром (registry) – ієрархічно організованим сховищем інформації про налаштування системи і прикладних програм.

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

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

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

~ Менеджер вікон – реалізує високорівневі функції. Він керує віконним виведенням, обробляє введення з клавіатури або миші й 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 тощо).

~ Драйвери графічних пристроїв (відеокарт, принтерів тощо) – відповідають за взаємодію з контролерами цих пристроїв.

Під час створення вікон або елементів керування запит надходить до менеджера вікон, який для виконання базових графічних операцій звертається до GDI.

Потім запит передається драйверу пристрою, а потім – апаратному забезпеченню через HAL.

Компоненти режиму користувача не мають прямого доступу до апаратного забезпечення, їхній код виконується в ізольованому адресному

просторі. Більша частина коду режиму користувача перебуває в динамічних бібліотеках, які у 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 (ntdll.dll). Ця бібліотека надає набір функцій-перехідників, кожній з яких відповідає функція режиму ядра (системний виклик). Застосування зазвичай не викликають такі функції безпосередньо, за це відповідають підсистеми середовища.

Підсистеми середовища надають застосуванням користувача доступ до служб операційної системи, реалізуючи відповідний API.

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

~ процес підсистеми Win32 (csrss.exe), що відповідає, зокрема, за реалізацію текстового (консольного) введення-виведення, створення і знищення процесів та потоків;

~ бібліотеки підсистеми Win32, які надають прикладним програмам функції Win32 API. Найчастіше використовують бібліотеки gdi32.dll (низько рівневі графічні функції, незалежні від пристрою), user32.dll (функції інтерфейсу користувача) і kernel32.dll (функції, реалізовані у ВС і ядрі).

Після того як застосування звернеться до функції Win32 API, спочатку буде викликана відповідна функція з бібліотеки підсистеми Win32.

Підсистема POSIX працює в режимі користувача й реалізує набір функцій, визначених стандартом POSIX 1003.1. Оскільки застосування, або прикладні програми (applications), написані для однієї підсистеми, не можуть використати функції ІНШИХ, у POSIX-програмах не можна користуватися засобами Win32 API (зокрема, графічними та мережними функціями), що знижує важливість цієї підсистеми. Підсистема POSIX не є обов'язковим компонентом Windows XP.

Наперед визначені системні процеси. Ряд важливих процесів користувача система запускає автоматично до закінчення завантаження. Розглянемо деякі з них.

~ Менеджер сесій (Session Manager, smss.exe) створюється в системі першим. Він запускає інші важливі процеси (процес підсистеми Win32, процес реєстрації в системі тощо), а також відповідає за їхнє 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 у підсистему безпеки і в разі успішної його верифікації запускає засоби створення сесії користувача.

~ Менеджер керування службами (Service Control Manager, 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 друкування, постачають разом із системою. Крім того, є багато служб сторонніх розробників; так зазвичай реалізовують серверні застосування (сервери баз даних, веб-сервери тощо).

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.

Об'єктна архітектура Windows. Керування ресурсами у Windows XP реалізується із застосуванням концепції об'єктів. Об'єкти надають універсальний інтерфейс для доступу до системних ресурсів, для яких передбачено спільне використання, зокрема таких, як процеси, потоки, файли і розподілювана пам'ять. Концепція об'єктів забезпечує важливі переваги.

~ Імена об'єктів організовані в єдиний простір імен, де їх легко знаходити.

~ Доступ до всіх об'єктів здійснюється однаково. Після створення нового об'єкта або після отримання доступу до наявного менеджер об'єктів повертає у застосування дескриптор об'єкта (object handle).

~ Забезпечено захист ресурсів. Кожну спробу доступу до об'єкта розглядає підсистема захисту – без неї доступ до об'єкта, а отже і до ресурсу, отримати неможливо.

Менеджер об'єктів відповідає за створення, підтримку та ліквідацію об'єктів, задає єдині правила для їхнього іменування, збереження й забезпечення захисту.

Підсистеми середовища звертаються до менеджера об'єктів безпосередньо або через інші сервіси ВС.

Об'єкти реалізовано як структури даних в адресному просторі ядра. При перезавантаженні системи вміст усіх об'єктів губиться.

Структура заголовка об'єкта. Об'єкти складаються з двох частин: заголовка і тіла об'єкта. У заголовку міститься інформація, загальна для всіх об'єктів, у тілі – специфічна для об’єктів конкретного типу.

До атрибутів заголовка об’єкта належать:

~ ім’я об’єкта і його місце у просторі імен;

~ дескриптор захисту (визначає права, необхідні для використання об’єкта);

~ витрата квоти (ціна відкриття дескриптора об’єкта, дає змогу регулювати кількість об’єктів, які дозволено створювати);

~ список процесів, що дістали доступ до дескрипторів об’єкта.

Менеджер об’єктів здійснює керування об’єктами на підставі інформації з їхніх заголовків.

Об’єкти типу. Формат і вміст тіла об'єкта визначається його типом. Новий тип об'єктів може бути визначений будь-яким компонентом ВС. Існує визначений набір типів об'єктів, які створюються під час завантаження системи (такі об'єкти, наприклад, відповідають процесам, відкритим файлам, пристроям введення-виведення).

Частина характеристик об'єктів є загальними для всіх об'єктів цього типу.

Для зберігання відомостей про такі характеристики використовують спеціальні об'єкти типу (type objects). У такому об'єкті, зокрема, зберігають:

~ ім'я типу об'єкта («процес», «потік», «відкритий файл» тощо);

~ режими доступу (залежать від типу об'єкта: наприклад, для файла такими режимами можуть бути «читання» і «запис»).

Об'єкти типу недоступні в режимі користувача.

Методи об'єктів. Коли компонент ВС створює новий тип об'єкта, він може зареєструвати у диспетчері об'єктів один або кілька методів. Після цього диспетчер об'єктів викликає ці методи на певних етапах життєвого циклу об'єкта. Приклад деяких з методів об’єктів: open – викликається при відкритті дескриптора об’єкта; close – викликається при закритті дескриптора об’єкта; delete – викликається перед вилученням об’єкта з пам’яті.

Покажчики на код реалізації методів також зберігаються в об’єктах типу.

Простір імен об’єктів. Усі імена об'єктів у ВС розташовані в глобальному просторі імен, тому будь-який процес може відкрити дескриптор об'єкта, вказавши його ім'я. Простір імен об'єктів має ієрархічну структуру, подібно до файлової системи. Аналогом каталогу файлової системи в такому просторі імен є каталог об'єктів. Він містить імена об'єктів (зокрема й інших каталогів). Наприклад, деякі наперед визначені імена каталогів: Device – імена пристроїв введення-виведення; Driver – завантажені драйвери пристроїв; ObjectTypes – об'єкти типів.

Простір імен об'єктів, як і окремі об'єкти, не зберігається після перезавантаження системи.

Особливості архітектури: UNIX і Linux

UNIX є прикладом досить простої архітектури ОС. Базова структура класичного ядра UNIX зображена на схемі.

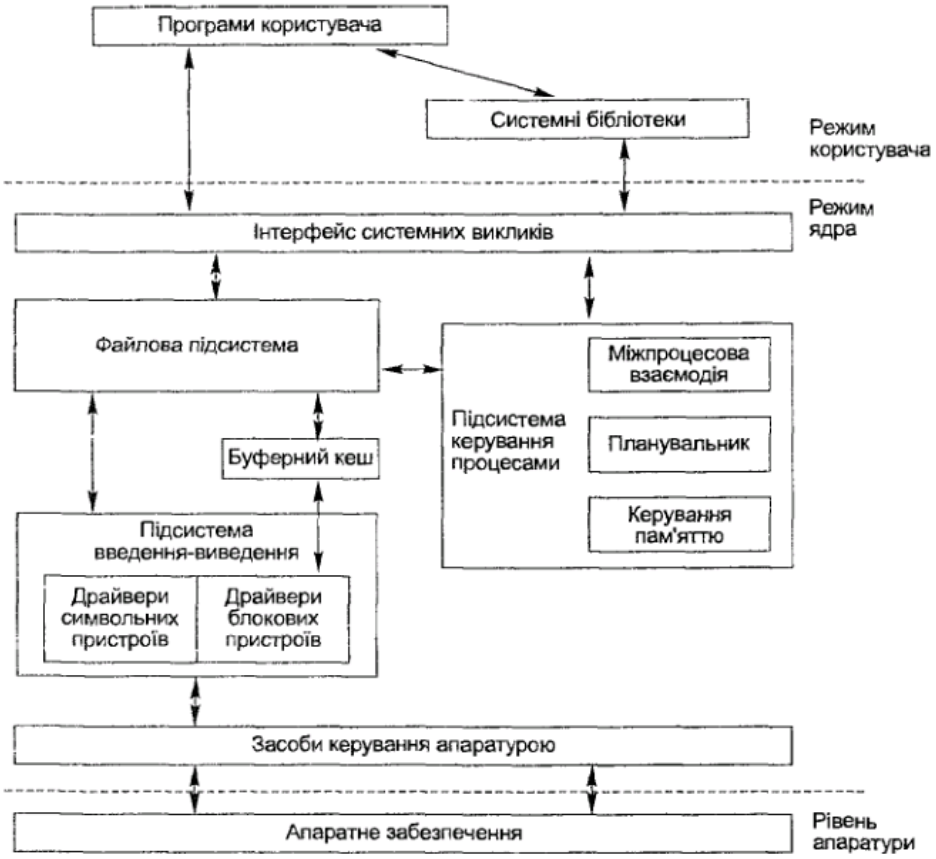


Рис. 2.3. Архітектура UNIX

Система складається із трьох основних компонентів: підсистеми керування процесами, файлової підсистеми та підсистеми введення-виведення.

Підсистема керування процесами контролює створення та вилучення процесів, розподілення системних ресурсів між ними, міжпроцесову взаємодію, керування пам’яттю.

Файлова підсистема забезпечує єдиний інтерфейс доступу до даних, розташованих на дискових накопичувачах, і до периферійних пристроїв. Такий інтерфейс є однією з найважливіших особливостей UNIX. Одні й ті самі системні виклики використовують як для обміну даними із диском, так і для виведення на термінал або принтер (програма працює із принтером так само, як із 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, які значною мірою визначають привілеї користувача в системі.

Підсистема введення-виведення виконує запити файлової підсистеми, взаємодіючи з драйверами пристроїв. В UNIX розрізняють два типи пристроїв: символьні (наприклад, принтер) і блокові (наприклад, жорсткий диск). Основна відмінність між ними полягає в тому, що блоковий пристрій допускає прямий

доступ. Для підвищення продуктивності роботи із блоковими пристроями використовують буферний кеш – ділянку пам'яті, у якій зберігаються дані, зчитані з диска останніми. Під час наступних звертань до цих даних вони можуть бути отримані з кеша.

Сучасні UNIX-системи дещо відрізняються за своєю архітектурою.

~ У них виділено окремий менеджер пам'яті, відповідальний за підтримку віртуальної пам'яті.
~ Стандартом для реалізації інтерфейсу файлової системи є віртуальна файлова система, що абстрагує цей інтерфейс і дає змогу організувати підтримку різних типів файлових систем.
~ У цих системах підтримується багатопроцесорна обробка, а також багатопотоковість.

В ОС Linux можна виділити три основні частини:

~ *ядро*, яке реалізує основні функції ОС (керування процесами, пам'яттю, введенням-виведенням тощо);
~ *системні бібліотеки*, що визначають стандартний набір функцій для використання у застосуваннях (виконання таких функцій не потребує переходу в привілейований режим);
~ *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 перебувають в одному адресному просторі. У ядрі можна виділити кілька 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 та пристроями введення-виведення.
~ *Драйвери пристроїв* – забезпечують безпосередню 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 міжпроцесової взаємодії* – пропонує механізми, які дають змогу різним процесам у системі обмінюватися даними між собою.

Деякі із цих підсистем є логічними компонентами системи, вони завантажуються у пам'ять разом із ядром і залишаються там постійно. Компоненти інших підсистем (наприклад, драйвери пристроїв) вигідно реалізовувати так, щоб їхній код міг завантажуватися у пам'ять на вимогу. Для розв'язання цього завдання Linux підтримує концепцію модулів ядра.

Модулі ядра. Ядро Linux дає можливість на вимогу завантажувати у пам'ять і вивантажувати з неї окремі секції коду. Такі секції називають модулями ядра (kernel modules) і виконують у привілейованому режимі.

Модулі ядра надають низку переваг.

~ Код модулів може завантажуватися в пам'ять у процесі роботи системи, що спрощує налагодження компонентів ядра, насамперед драйверів.

~ З'являється можливість змінювати набір компонентів ядра під час виконання: ті з них, які в цей момент не використовуються, можна не завантажувати у пам'ять.

~ Модулі є винятком із правила, за яким код, що розширює функції ядра, відповідно до ліцензії Linux має бути відкритим. Це дає змогу виробникам апаратного забезпечення розробляти драйвери під Linux, навіть якщо не заплановано надавати доступ до їхнього вихідного коду.

Підтримка модулів у Linux складається із трьох компонентів.

~ Засоби керування модулями дають можливість завантажувати модулі у пам'ять і здійснювати обмін даними між модулями та іншою частиною ядра.

~ Засоби реєстрації драйверів дозволяють модулям повідомляти іншу частину ядра про те, що новий драйвер став доступним.

~ Засоби розв'язання конфліктів дають змогу драйверам пристроїв резервувати апаратні ресурси і захищати їх від випадкового використання іншими драйверами.

Один модуль може зареєструвати кілька драйверів, якщо це потрібно (наприклад, для двох різних механізмів доступу до пристрою).

Модулі можуть бути завантажені заздалегідь – під час старту системи (завантажувальні модулі) або у процесі виконання програми, яка викликає їхні функції.

Після завантаження код модуля перебуває в тому ж самому адресному просторі, що й інший код ядра. Помилка в модулі є критичною для системи.

Особливості системних бібліотек. Системні бібліотеки Linux є динамічними бібліотеками, котрі завантажуються у пам'ять тільки тоді, коли у них виникає потреба. Вони виконують ряд функцій:

~ реалізацію пакувальників системних викликів;

~ розширення функціональності системних викликів (до таких бібліотек належить бібліотека введення-виведення мови С, яка реалізує на основі системних викликів такі функції, як printf());

~ реалізацію службових функцій режиму користувача (сортування, функції обробки рядків тощо).

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

Тема 3. Базова система введення-виведення.

Мета: ознайомити студентів з складовими частинами BIOS; розглянути архітектуру та настройки BIOS.

План

1. Складові частини BIOS.
2. Архітектура BIOS.
3. Настройка BIOS.

Питання для самоконтролю

1. Складові частини BIOS

BIOS – Базова система введення-виведення називається так тому, що включає обширний набір програм введення-виведення, завдяки яким операційна система і прикладні програми можуть взаємодіяти з різними пристроями як самого комп'ютера, так і підключеними до нього. Взагалі кажучи, система BIOS займає особливе місце. З одного боку, її можна розглядати як складову частину апаратних засобів, з другого боку, вона є як би одним з програмних модулів операційної системи. Сам термін BIOS, мабуть, запозичений з операційної системи CP/M, в якій модуль з подібною назвою був реалізований програмно і виконував приблизно подібні дії.

Більшість сучасних відеоадаптерів, а також контроллери накопичувачів мають власну систему BIOS, яка звичайно доповнює системну. В багатьох випадках програми, що входять у конкретну BIOS, замінюють відповідні програмні модулі основної BIOS. Виклик програм BIOS, як правило, здійснюється через програмні чи апаратні переривання.

Система BIOS крім програм взаємодії з апаратними засобами на фізичному рівні містить програму тестування при включенні живлення комп'ютера POST (Power-On-Self-Test, самотестування при включенні живлення комп'ютера). Тестуються основні компоненти, такі як процесор, пам'ять, допоміжні мікросхеми, приводи дисків, клавіатуру і відеопідсистему.

Система BIOS реалізована у вигляді однієї мікросхеми, встановленої на материнській платі комп'ютера.

Оскільки вміст BIOS фірми IBM був захищений авторським правом, тобто його не можна піддавати копіюванню, то більшість інших виробників комп'ютерів була змушена використовувати мікросхеми BIOS незалежних фірм, системи BIOS яких, зрозуміло, були практично повністю сумісні з оригіналом. Найвідоміші з цих фірм три: American Megatrends Inc. (AMI), Award Software і Phoenix Technologies. Конкретні версії BIOS нерозривно пов'язані з набором мікросхем (chipset), що використовується на системній платі. До речі, компанія Phoenix Technologies вважається піонером у виробництві ліцензійно-чистих BIOS. Саме в них вперше були реалізовані такі функції, як задавання типу жорсткого диска, підтримка приводу флоппі-дисків місткістю 1,44 Мбайта і т.д. Більше того, вважається, що процедура POST цих BIOS має наймогутнішу діагностику. Справедливості ради треба відзначити, що BIOS компанії AMI найбільш поширені. За деякими даними, AMI займає близько 60% цього сегменту ринку. Крім того, з програми Setup AMI BIOS можна викликати декілька утиліт для тестування основних компонентів системи і роботи з накопичувачами. Проте при їх використанні особливу увагу слід звернути на тип інтерфейсу, який використовує привід накопичувача.

Система BIOS у комп'ютерах нерозривно пов'язана з SMOS RAM. Під цим розуміється "незмінна" пам'ять, в якій зберігається інформація про поточні показники годинника, значення часу для будильника, конфігурації комп'ютера: кількості пам'яті, типах накопичувачів і т.д. Саме цієї інформації потребують програмні модулі системи BIOS. Своєю назвою SMOS RAM зобов'язана тому, що ця пам'ять виконана на основі КМОП-струкгур (CMOS-Complementary Metal Oxide Semiconductor), які, як відомо, відрізняються малим енергоспоживанням. CMOS-пам'ять енергонезалежна тільки постільки, оскільки постійно підживляється, наприклад, від акумулятора, розташованого на системній платі, або батареї гальванічних елементів, як правило, змонтованої на корпусі системного блоку. Більшість системних плат допускає живлення CMOS RAM як від вбудованого, так і від зовнішнього джерела.

В разі пошкодження мікросхеми CMOS RAM (або розряді батареї чи акумулятора) програма Setup має нагоду скористатися інформацією за

умовчанням (BIOS Setup Default Values), яка зберігається в таблиці відповідної мікросхеми BIOS. До речі, на деяких материнських платах живлення мікросхеми CMOS RAM може здійснюватися як від внутрішнього, так і від зовнішнього джерела. Вибір визначається установкою відповідної перемички.

Програма Setup підтримує установку декількох режимів енергозбереження, наприклад, Doze (дрімаючий), Standby (очікування, або резервний) і Suspend (припинення роботи). Дані режими перераховані в порядку зростання економії електроенергії. Система може переходити в конкретний режим роботи після закінчення певного часу, вказаного в Setup. Крім того, BIOS звичайно підтримує і специфікацію APM (Advanced Power Management). Як відомо, вперше її запропонували фірми Microsoft і Intel. В їх сумісному документі містилися основні принципи розробки технології управління споживаною портативним комп'ютером потужністю.

Задавання повної конфігурації комп'ютера здійснюється не тільки установками з програми Setup, але й замиканням (або розмиканням) відповідних перемичок на системній платі. Призначення кожної з них вказано у відповідній документації.

2. Архітектура BIOS

BIOS – базова система введення-виведення, що зберігається в ПЗП, призначена для ізоляції операційної системи і прикладних програм від специфічних особливостей конкретної апаратури. BIOS містить програмну підтримку стандартних ресурсів PC і забезпечує конфігурацію апаратних засобів, їх діагностику і виклик завантажувача операційної системи. BIOS в значній мірі прив'язана до конкретної реалізації системної платі.

Flash-BIOS зберігається у флеш-пам'яті, що дозволяє обновляти версію BIOS перезаписом її з дискети.

Функції BIOS розділяються на наступні групи:

- Ініціалізація і початкове тестування апаратних засобів – POST (Power On Self Test).
- Настройка і конфігурація апаратних засобів і системних ресурсів – BIOS Setup.

- Завантаження операційної системи з дискових носіїв – Bootstrap Loader.
- Обслуговування апаратних переривань від системних пристроїв (таймера, клавіатури, дисків) – BIOS Hardware Interrupts.
- Відробляння базових функцій програмних звернень (сервісів) до системних пристроїв – BIOS Services.

Всі ці функції виконує системний модуль System BIOS, що зберігається в мікросхемі ПЗП або флеш-пам'яті на системній платі. Більшість сервісних функцій виконується в 16-бітному режимі, хоча деякі нові функції можуть мати і альтернативні виклики для 32-бітного виконання.

Розширення BIOS. Можливості бездискової машини обмежуються тільки функціями BIOS, але їх склад може бути розширений, для чого є спеціальні системні засоби.

Плати адаптерів, встановлених в слоти системної шини, можуть мати мікросхеми ПЗП своєї програмної підтримки – *Additional BIOS* (додаткові модулі BIOS). Їх використовують деякі контроллери жорстких дисків, мережні адаптери з віддаленим завантаженням та інші периферійні пристрої. Для цих модулів в просторі пам'яті зарезервована область C8000h-F4000h. POST сканує цю область з кроком 2 Кбайта у пошуках додаткових модулів BIOS на завершальному етапі виконання (після завантаження векторів переривань покажчиками на власні обробники). Додатковий модуль BIOS відеоадаптера (EGA, VGA, SVGA...) має фіксовану адресу C0000 й ініціалізується раніше (на кроці ініціалізації відеоадаптера).

Додатковий модуль BIOS повинен мати заголовок, вирівняний по межі 2-кілобайтної сторінки пам'яті, тобто біт адреси, що має нульове значення, A[10:0]. Формат заголовка наступний:

- Байт-0=55h, байт-1=AAh – ознака початку модуля.
- Байт-2 – довжина в блоках по 512 байт.
- Байт-3 – точка входу процедури ініціалізації, що закінчується дальнім поверненням (Ret Far).

Коректним вважається модуль, що починається з ознаки 55AAh і нульової суми (по модулю 256) всіх байт в оголошеній області (реальна довжина модуля може перевищувати оголошену, але байт контрольної суми, природно, повинен входити в оголошену область). У разі виявлення коректного модуля POST дальнім викликом (Call Far) викликає процедуру ініціалізації модуля, що починається з 3-ої адреси заголовка модуля. Відповідальність за її коректність повністю лягає на розробника. Процедура може перевизначати вектори переривань, обслуговуваних BIOS. Перевизначивши на себе Bootstrap (INT 19h), можна отримати управління при завантаженні, що й використовується для віддаленого завантаження комп'ютерів через локальну мережу (Remote Boot Reset). Якщо стандартне продовження процедури завантаження не потрібне, а додатковий модуль є, наприклад, управляючою програмою для якого-небудь устаткування, замість процедури ініціалізації в ПЗП може знаходитися і основна програма, що не повертає управління системної послідовності POST.

Вектори переривань BIOS. При ініціалізації таблиці переривань BIOS відповідає за коректне заповнення частини векторів, що мають відношення до апаратних засобів комп'ютера і сервісів BIOS. На частину з них можуть бути просто встановлені заглушки – вектор посилається на код обробника, що містить єдину інструкцію повернення з переривання – IRET.

BIOS ініціалізує вектори *внутрішніх переривань* процесора (виключень), які можуть виникнути в реальному режимі роботи (про виключення захищеного режиму в основному піклується відповідна операційна система). До внутрішніх переривань відносяться наступні (INT 70h – INT 77h – тільки для AT): INT 00h – ділення на 0; INT 01h – покроковий режим; INT 03h – точка зупинки; INT 04h – переповнювання; INT 06h – недопустима команда 286+; INT 07h – виклик відсутнього NPU.

Апаратні переривання включають наступні (INT 70h-77h – тільки для AT): INT 02h – немасковане переривання; INT 08h – таймер 8253/8254; INT 09h – клавіатура; INT 0Ah-0Dh – IRQ2-IRQ5; INT 0Eh – IRQ6 – контроллер гнучких дисків; INT OFh – IRQ7; INT 70h – CMOS-таймер; INT 71h – IRQ9 (перенаправлено на INT 0Ah); INT 72h-74h – IRQ10-IRQ12; INT 75Ь – IRQ13 –

виключення співпроцесора; INT 76h – IRQ14 – контроллер жорстких дисків; INT 77h – IRQ15.

Вектори переривань, що забезпечують *виклики функцій BIOS* (сервісів), включають наступні: INT 05h – друк екрану; INT 10h – відео сервіс; INT 11h – список устаткування; INT 12h – розмір безперервної пам'яті; INT 13h – дискове введення-виведення; INT 14h – обслуговування COM-портів; INT 15h – AT-функції (системний сервіс); INT 16h – клавіатурне введення-виведення; INT 17h – обслуговування LPT-портів; INT 18h – ROM-Basic; INT 19h – початкове завантаження (Bootstrap); INT 1Ah – системний час і 16-бітні виклики сервісів PCI; INT 1Bh – обробник ctrl+break; INT 1Ch – призначена для користувача процедура, що викликається обробником INT 08h (User Timer Interrupt); INT 33h – підтримка миші; INT 67h – EMS-функції.

Декілька векторів використовуються як покажчики на системні таблиці. До *векторів-покажчиків* відносяться наступні: INT 1Dh – відео параметри; INT 1Eh – параметри дискет; INT 1Fh – знакогенератор CGA; INT 41h – параметри HDD 0; INT 46h – параметри HDD 1; INT 43h – знакогенератор EGA; INT 4Ah – будильник користувача.

Як видно з наведених списків, більшість векторів BIOS накладається на область векторів 00-1Fh, зарезервовану фірмою Intel під внутрішні переривання і виключення процесорів. Хоча за часів 8086 з них використовувалася зовсім мала кількість, зарезервованою була оголошена вся вказана область. Проте творці IBM PC "влізли" в цю область, що ускладнило життя системних програмістів, які працюють з більш щедрими на виключення сучасними процесорами.

Області даних BIOS. Окрім векторів переривань, BIOS в оперативній пам'яті має свою область даних *BIOS DATA AREA*, що починається з адреси 400h (відразу за таблицею переривань). Ця адреса в сегментній моделі адресації реального режиму може бути представлена як 0000:0400h або 0040:0000h, що вказує на одну і ту ж фізичну адресу. Призначення комірок даної області розкриває табл. 2.1. BIOS може також використовувати й розширену область даних *EBDA* (Extended BIOS Data Area), яка звичайно розташовується під

верхньою межею (640 Кбайт) стандартної пам'яті. На її положення вказує слово за адресою 40:0Eh, а перший байт цієї області вказує її розмір в одиницях кілобайтів. Ця область використовується для різних семафорів і покажчиків, її розмір звичайно не перевищує 1 Кбайта.

Таблиця 2.1

Призначення комірок BIOS Data Area

Адреса	*Розмір, байт*	*Призначення*
040:000	4×2	Базові адреси портів COM1-COM4
040:008	3×2	Базові адреси портів LPT1-LPT3
040:00E	2	Базова адреса порту LPT4 або адреса сегменту EBDA
040:010	2	Встановлене устаткування
040:013	2	Розмір стандартної пам'яті
040:015	2	Робочі комірки для тестів
040:017	39	Область прапорів і буфер клавіатури
040:03E	1	Біти [0:3] – дисководи, вимагаючі рекалібрування (біт 0=A:, біт 1=B: і т. д.) Біти [4:5] – вибраний дисковод
040:03F	1	Включений мотор дисководів (біт 0-a:, біт 1-в: і т. д.)
040:040	1	Час до відключення мотора (INT 08h вимикає мотор по обнулінню)
040:041	1	Код помилки дискет
040:042	7	Інформація про стан FDC
040:049	1	Поточний активний відеорежим
040:04A	2	Ширина екрану (число колонок символів)
040:04C	2	Розмір зони відеопам'яті, що використовується (в байтах)
040:04E	2	Зміщення активної відеосторінки від відеосегменту
040:050	16	Позиція курсора (8 пар байт; в молодшому байті – колонка, в старшому – ряд)
040:060	2	Розмір курсора (в молодшому байті – останній рядок, в старшому – перша)
040:062	1	Номер активної відеосторінки
040:063	2	Адреса порту відеоконтроллера 6845
040:065	1	Поточне значення 6845 CRTMODE (порт 3×8h)
040:066	1	Поточне значення 6845 CRTPALETTE (порт 3×9h)
040:067	5	Область даних POST
040:068	4	Лічильник переривань від таймера (рахує інтервали 55 мілісекунд)
040:070	1	Переповнювання таймера (перехід через 24 години)
040:071	1	Прапор ctrl+break (біт 7-1 по натисненню)
040:072	2	1234h означає перезавантаження по ctrl+alt+del. Використовується POST
040:074	4	Управління жорстким диском
(0474)	1	Статус останньої операції з жорстким диском

Адреса	*Розмір, байт*	*Призначення*
(0475)	1	Число жорстких дисків
(0477)	1	Порт HDC (XT)
040:078	4×1	Тайм-аут LPT-портів (478h – LPTl, 479h – LPT2...)
040:07C	4×1	Тайм-аут Сом-портів (47Ch – COMl, 47Dh – COM2...)
040:080	2	Зміщення початку клавіатурного буфера (звичайно 01Eh)
040:082	2	Зміщення кінця клавіатурного буфера +1 (звичайно 03Eh)
040:084	1	EGA: максимальне число рядів символів – 1
040:085	2	EGA: число рядків в символі в поточному режимі
040:087	2	EGA: змішана інформація
040:08B	1	AT, PS/2: параметри дисковода (швидкості передачі даних і переміщення головок)
040:08C	1	AT, PS/2: стан HDC
040:08D	1	AT, PS/2: помилки HDD
040:08E	1	AT, PS/2: управління перериваннями від HDD
040:090	1	AT, PS/2: стан носія приводу 0
040:091	1	AT, PS/2: стан носія приводу 1
040:092	1	AT, PS/2: прапор початку операції приводу 0
040:093	1	AT, PS/2: прапор початку операції приводу 1
040:094	1	AT, PS/2: поточний номер циліндра для приводу 0
040:095	1	AT, PS/2: поточний номер циліндра для приводу 1
040:096	1	AT: прапор клавіатури, біт 4-1 (10h) при 101-клавішній клавіатурі
040:097	1	AT: прапори індикаторів клавіатури, біти 0-2 – scrollock, numlock, capslock
040:098	4	AT: покажчик на 8-бітний User Wait Flag (INT 15h Fn 86h)
040:09C	4	AT: мікросекунд до User Wait
040:0A0	1	AT: прапор активності User Wait 1 = зайнятий, 80h = пройшов, 0 = підтверджений
040:0A1	7	AT: резерв для мережних адаптерів
040:0A8	4	EGA: адреса таблиці покажчиків – SAVEPTR
040:0F0	16	Область взаємодії програм
040:100	1	Стан функції друку екрану 00h = Ok; 01h = друк; 0FFh = помилка при друці
040:104	1	"Фантомний" гнучкий диск: 01h = дискета в приводі A: використовується під ім'ям B:
040:110	17	Область інтеPnPетатора Basic
040:130	3	Використовуються командою MODE

3. Настройка BIOS.

Базова система введення-виведення *BIOS* є ключовим елементом системної плати, без якого всі її чудові компоненти є лише набором дорогих "залізяк". BIOS, користуючись засобами, що надаються чіпсетом, управляє всіма компонентами і ресурсами системної плати. З цього виходить, що

використовувана версія BIOS дуже сильно прив'язана до чіпсета, і, крім того, вона повинна знати особливості вмкористовуваних компонентів (процесор, пам'ять, інтегровані контроллери). Код BIOS зберігається в мікросхемі енергозалежної постійної *(BIOS)* або флеш-пам'яті (Flash BIOS). З погляду регулярної роботи, тип носія BIOS принципового значення не має. З погляду модифікованості флеш-пам'ять має явну перевагу – можливість модернізації прямо в комп'ютері, іноді, правда, що обертається недоліком. Визначити, який носій BIOS використовується на даній системній платі можна, знявши наклейку з мікросхеми (на ній звичайно надруковані вихідні дані BIOS) і прочитавши позначення: 28Fxxx – флеш-пам'ять 12 В; 29Cxxx – флеш-пам'ять 5 В; 29LVxxx – флеш-пам'ять 3 В (рідкісний варіант); 28Cxxx – EEPROM, близька по властивостях до флеш-пам'яті; 27Cxxx – EPROM, записувана на програматорі і стирана ультрафіолетом (якщо є скляне віконце); PH29EE010 – ROM фірми SST, перезаписується аналогічно флеш-пам'яті; 29EE011 – флеш-пам'ять 5В фірми Winbond; 29C010 – флеш-пам'ять 5 В фірми Atmel.

Причин узятися за модернізацію BIOS може бути декілька, наприклад:

- Некоректна робота в деяких режимах (наприклад, мимовільний перехід в енергозберігаючий режим, що виражається в зупинках вінчестера, гасінні екрану або раптовому різкому зниженні продуктивності неначебто нормально функціонуючого комп'ютера). По мірі виявлення помилок виробник випускає нові версії BIOS (можливо, і з новими помилками).
- Неузгодженість драйверів BIOS з вимогами нових версій ОС.
- Отримання нових функціональних можливостей, підвищення продуктивності.
- Бажання мати найсвіжішу версію (для любителів експериментувати на собі).
- Стерти конфігураційну інформацію в NVRAM (включаючи і ESCD), якщо для цієї мети немає перемикача або опції в BIOS Setup. Утиліта перепрограмування флеш-пам'яті виконує цю дію автоматично або пропонує його виконати зі свого меню.

Нову версію BIOS краще всього отримувати від виробника системної плати, велика колекція версій і утиліт доступна в мережі Інтернет за адресою http://www.sysdoc.pair.com. Фірми-розробники BIOS (наприклад, AMI, Award) нові версії BIOS для кінцевих користувачів не поставляють. Свої нові продукти з інструментальними засобами вони поставляють розробнику системної плати, який проводить остаточну "підгонку" BIOS під конкретну модель плати, особливості якої він знає краще за всіх. В першому наближенні BIOS різної системної плати з однаковими або близькими чіпсетами можуть виявитися (або показатися) сумісними – принаймні, при включенні виводиться заставка, проходить POST і навіть завантаження. Проте при більш ретельному тестуванні може виявитися, наприклад, що неможливо звернутися до дисків (гнучких або жорстких), не працюють порти, доступна не вся пам'ять і т.п. Добре, якщо при цьому вдасться завантажити утиліту перепрограмування BIOS, щоб повернутися до старої (*заздалегідь збереженої*!) *версії*.

Утиліти перезапису флеш-пам'яті прив'язані до підтримуваних типів мікросхем енергозалежної пам'яті, системної плати (чіпсетів) і виробників (іноді й версій) BIOS. Звичайно не вдається штатним способом (у комп'ютері) переписати BIOS із зміною виробника (Award, AMI, Phoenix). Як варіант можлива заміна (хоча б тимчасова) мікросхеми BIOS на зняту з аналогічної системної плати, але якщо мікросхема припаяна, а не встановлена в ліжечко, процедура заміни сильно ускладнюється. Сміливо займатися перепрограмуванням BIOS можна, тільки коли ви маєте доступ до програматора і мікросхема BIOS встановлена в ліжечку.

Якщо нова версія BIOS не дозволяє завантажити комп'ютер, ряд системних плат дозволяє включити режим відновлення (*Boot Block Recovery*). Для цього на платі повинен бути спеціальний перемикач або джампер. В режимі відновлення працює тільки дисковод, в який необхідно встановити спеціальну дискету з файлом-образом BIOS. При цьому "повідомлення" користувачу можуть зводитися до підморгування індикатором дисковода і гудкам динаміка. Мова цих повідомлень повинна наводитися в описі системної плати. Іноді режим відновлення включається автоматично (якщо Boot Block

отримує управління на початку POST завжди, він може оцінити коректність вмісту основного блоку ПЗП і при необхідності включити режим відновлення).

Якщо ж після невдалого перепрограмування режим відновлення не рятує (або відсутній), а доступного програматора немає, тобто хоча й ризикований, але можливий варіант "гарячої заміни" BIOS. Для цього з аналогічної працездатної системної плати витягують мікросхему BIOS, встановлюють її замість зіпсованої, включають і завантажують комп'ютер як для режиму перезапису BIOS. При цьому в Setup повинне бути дозволено вживання тіньовій пам'яті для області системної BIOS. Далі, не вимикаючи живлення (небезпечно, але в безвихідному становищі можна ризикнути) замінюють мікросхему на невірно записану і виконують процедуру перезапису. Комп'ютер продовжує працювати, оскільки код BIOS виконується з тіньової області ОЗП. Файл-образ для перезапису може бути отриманий як копія "рятівної" мікросхеми, зроблена тією ж програмуючою утилітою.

Кажучи про недоліки флеш-BIOS, мається на увазі небезпеку втрати працездатності системної плати не тільки через необачні дії користувача, що модернізує BIOS, але й нове "поле діяльності" для вірусів. Стерти BIOS, знаючи роботу чіпсета і конкретної мікросхеми пам'яті, можна навіть наладчиком DEBUG. Парольний (програмний) захист перезапису може бути зламаний, а надійний апаратний захист (необхідністю подачі високої напруги для стирання і програмування, а також сигнал захисту запису) є далеко не в усіх мікросхем енергозалежної пам'яті і системної платі.

Тест початкового включення POST. Після включення живлення, апаратного скидання від кнопки RESET або натисненні комбінації клавіш Ctrl+Alt+Del процесор переходить до виконання коду початкового самотестування *POST* (Power-On Self Test), що зберігається в мікросхемі BIOS. POST виконує тестування процесора, пам'яті і системних засобів введення-виведення, а також конфігурація всіх програмно-керованих апаратних засобів системної плати. Частина конфігурації виконується однозначно, частина управляється джамперами системної плати, але ряд параметрів дозволяє або навіть вимагає конфігурації за бажанням користувача. Для цих цілей служить

утиліта *Setup,* вбудована в код BIOS. Після тестування і конфігурації (що включає настройку пристроїв PnP), POST ініціалізує завантаження операційної системи.

При проходженні кожної секції POST записує її код (номер) в *діагностичний регістр.* Цей регістр фізично розташовується на спеціальній діагностичній платі, встановлюваній в слот системної шини. Плата містить 8-бітний регістр з світловою (двійковою або шістнадцятковою) індикацією стану бітів. В просторі введення-виведення регістр займає одну адресу, залежну від архітектури PC (точніше, версії BIOS): ISA, EISA – 80h, ISA-Compaq – 84h, ISA-PS/2 – 90h, MCA-PS/2 – 680h, деякі моделі EISA – 300h (часто пишуть те ж і в 80h). По індикаторах плати можна визначити, на якій секції зупинився POST, і визначити причину несправності. Проте для використовування такої діагностики необхідна, по-перше, сама плата-індикатор, і по-друге, "словник" несправностей – таблиця, специфічна для версії BIOS і системної плати.

Під час виконання POST може видавати діагностичні повідомлення у вигляді послідовності коротких і довгих звукових сигналів, а після успішної ініціалізації графічного адаптера короткі текстові повідомлення виводяться на екран монітора.

Звична послідовність кроків POST: тестування регістрів процесора; перевірка контрольної суми BIOS; перевірка і ініціалізація таймера 8253/8254, портів 8255; після цього кроку доступна звукова діагностика (табл. 3.1); перевірка і ініціалізація контроллерів DMA 8237; перевірка регенерації пам'яті; тестування 64 Кбайт нижньої пам'яті; завантаження векторів переривання і стека в нижню область пам'яті; ініціалізація відеоконтроллера – на екрані з'являється заставка Video BIOS, звичайно з вказівкою моделі відеокарти і об'ємом встановленої відеопам'яті.

Таблиця 3.1

Звукова діагностика POST

*Сигнал**	*Помилка*	*Можливі дії*
1д2к	Не знайдений графічний адаптер	Встановити (переставити) адаптер
1д3к	Не підключений монітор (для системної платі з вбудованим	Підключити монітор, перевірити включення термінаторів на

*Сигнал**	*Помилка*	*Можливі дії*
	графічним адаптером)	моніторі
1дХк	Помилка графічного адаптера (Х залежить від версії Video BIOS)	Встановити (переставити) адаптер
1к	Помилка регенерації DRAM – встановлено некоректне значення періоду регенерації або несправний контроллер регенерації	Спробувати встановити настройки Setup за умовчанням, замінити DRAM. Якщо не допомагає – несправність у системній платі
2к	Помилка паритету DRAM (відсутній в платі, що не підтримує контроль паритету)	Замінити (переставити) пам'ять
3к	Помилка перших 64 Кбайт DRAM	Замінити (переставити) пам'ять
4к	Помилка системного таймера	Ремонт системної платі
5к	Помилка процесора	Замінити процесор
6к	Помилка управління GateA20 (контроллер 8042)	Переустановити або замінити IC контроллера клавіатури
7к	Помилка захищеного режиму	Ремонт системної платі
8к	Помилка відеопам'яті	Замінити відеопам'ять (графічний адаптер)
9к	Помилка контрольної суми BIOS	Замінити (перезаписати) BIOS
10к	Помилка CMOS (звернення до комірки 0Fh)	Ремонт системної плати
11к	Помилка кеш-пам'яті	Замінити кеш-пам'ять, перевірити її швидкодію і настройки Setup при відключеному кеші

* 1д 2к – один довгий сигнал, за яким слідують два коротких.

Після успіху цього кроку зображення на екрані змінюється заставкою системної BIOS з лічильником об'єму тестованої динамічної пам'яті. Тепер діагностичні повідомлення виводяться на екран (табл. 3.2). POST продовжує роботу, виконуючи наступні кроки: тестування повного об'єму ОЗУ; тестування клавіатури; тестування CMOS-пам'яті і годинника; ініціалізація COM і LPT портів; ініціалізація і тест контроллера НГМД; ініціалізація і тест контроллера НЖМД; сканування області додаткового BIOS; виклик Bootstrap (INT 19h) – завантаження операційної системи, при неможливості – спроба запуску ROM Basic (Int 18h), при невдачі – зупинка процесора з повідомленням "System Halted" (система зупинена).

Таблиця 3.2

Діагностичні повідомлення POST

Повідомлення	*Причина і можливі дії*
PRESS A KEY TO REBOOT	Пропозиція перезавантаження після натиснення будь-якої клавіші супроводжує повідомлення про помилку, знайдену POST
SYSTEM HALTED, (CTRL+ALT+DEL) TO REBOOT	Зупинка комп'ютера після виявлення серйозної помилки. Можливе тільки перезавантаження по Crtl+Alt+Del, апаратному скиданню або повторному включенню живлення
CMOS Battery State Low CMOS BATTERY HAS FAILED	Впала напруга живлення CMOS. Перевірити напругу на батареї при вимкненому живленні комп'ютера (повинне вище 3 В), перевірити установку джампера 2-3 на роз'ємі зовнішньої батареї. Замінити батарею
CMOS Checksum Failure CMOS CHECKSUM ERROR	Помилка контрольної суми CMOS. Може бути викликана проблемами з живленням CMOS, вживанням непридатною завантажуваною утилітою SETUP, дією вірусу. Виконати "штатний" SETUP
CMOS System Optons Not Set	Не встановлені опції Setup. Виконати SETUP
CMOS Time and Date Not Set	Не встановлений годинник і календар. Виконати SETUP і задати час і дату
Display Switch Not ProperDISPLAY SWITCH IS SET INCORRECTLY	Перевірити положення перемикача типу графічного адаптера (Color/Mono), що є на більшості старих системних плат
DISPLAY TYPE HAS CHANGED SINCE LAST BOOT	З моменту попереднього завантаження змінився тип графічного адаптера (монітора). Виконати SETUP і змінити (підтвердити новий) тип адаптера
Keyboard is locked ... Unlock it	Клавіатура заблокована ключем. Повернути ключ (якщо не допомагає, перевірити правильність приєднання ключа до роз'єму системної плати).
Keyboard Error K/B Interface Error KEYBOARD ERROR OR NO KEYBOARD PRESENT	Помилка клавіатури. Перевірити підключення роз'єму, перемикача XT/AT на клавіатурі, замінити клавіатуру. Цю перевірку можна подавити установкою "Keyboard Not Installed" в SETUP (опція є не у всіх версіях, той же ефект дає установка "HALT ON ALL, BUT KEYBOARD" в опції "Halt on Error")
DISK BOOT FAILURE, INSERT SYSTEM DISK AND PRESS ENTER	Немає доступного завантажувального пристрою (гнучкий, жорсткий диск, CD-ROM, мережний адаптер з мікросхемою BOOT ROM) з дійсним завантажувальним записом. Встановити завантажувальну дискету в A:, перевірити контроллер, конфігурацію і підключення диска C:
Invalid Boot Diskette Diskette Boot Failure	Неможливо завантажити ОС з дискети (немає завантажувального сектора). Замінити дискету

Повідомлення	*Причина і можливі дії*
No ROM Basic	Немає пристрою, з якого можна завантажити ОС (гнучкий, жорсткий диск, CD-ROM, мережний адаптер з мікросхемою BOOT ROM), а інтерпретатор Basic в ROM відсутній (був у перших моделях PC). Підключити і конфігурувати завантажувальний пристрій
DISKETTE DRIVES OR TYPES MISMATCH ERROR – RUN SETUP	Тип дисковода (A: або B:) не співпадає із записом в CMOS. Виконати Setup і задати правильні типи дисководів
FDD Controller Failure FLOPPY DISK CNTRLR ERROR OR NO CNTRLR PRESENT	Помилка контроллера накопичувачів на гнучких дисках (дисководів, кабелів). Перевірку можна відключити, встановивши в Standard Setup для дисків A: і B: значення Not Installed (None). Якщо контроллера немає, повинно бути встановлено значення Not Installed (None)
HDD Controller FailureERROR INITIALIZING HARD DRIVE CONTROLLER	Помилка контроллера накопичувачів на жорстких дисках (дисководів, кабелів). Перевірку можна відключити, встановивши в Standard Setup для всіх жорстких дисків (двох або чотирьох) значення Not Installed.
3: (D:) Drive Error 3: (D:) Drive Failure ERROR ENCOUNTERED INITIALIZING HARD DRIVE	Неможливе звернення до диска C: (D:). Невірно встановлені параметри в Setup, джампери на накопичувачах, інтерфейсні кабелі, невідформатований диск або несправний дисковод
CMOS Memory Size MismatchMEMORY SIZE HAS CHANGED SINCE LAST BOOT	Розбіжність розміру пам'яті, визначеної POST, із значенням, записаним в CMOS. Звичайно відбувається при додаванні або видаленні додаткових модулів пам'яті, але може вказувати і на несправність пам'яті. Ввійти в SETUP (пункт STANDARD SETUP) і вийти зі збереженням результатів в CMOS. Для EISA може бути потрібно виконання ECU
On Board Parity Error Board Parity Error Parity Error Memory Parity Error at XXXXRAM PARITY ERROR – CHECKING FOR SEGMENT	Помилка паритету пам'яті, встановленої на системній платі (On Board), платі розширення (Board) або без вказівки місцезнаходження. Збійна адреса XXXX може бути визначена не завжди. Повідомлення може бути викликано і вірусом
PRESS Fl TO DISABLE NMI, F2 TO REBOOT	Пропозиція продовжити роботу із забороненим контролем паритету (заборонене NMI), натискуючи F1, або перезавантажити комп'ютер, натискуючи F2. Може з'являтися при виявленні помилки паритету пам'яті
Memory Address Error at XXXX Memory Verify	Помилка пам'яті за адресою XXXX. Локалізувати і замінити модуль (мікросхему) пам'яті

Повідомлення	*Причина і можливі дії*
Error at XXXX	
Address Line Short	Замикання адресних ліній мікросхем або модулів пам'яті. Переставити (замінити) мікросхеми або модулі DRAM
Cache Memory Bad, do Not Enable Cache!	Помилка кеш-пам'яті. Усунути помилку (замінити або переставити мікросхеми) або заборонити зовнішній (External або L2) кеш в Setup
I/O Card Parity Error at XXXX	Помилка, знайдена на платі розширення (сигнал подається по лінії IOCHK)
DMA Bus Time-out	Пристрій в режимі DMA затримує цикл шини більше, ніж на 7,8 мкс. Причина – несправність плати розширення або системної плати
EISA Configuration is Not Complete	Не повністю задана конфігураційна інформація EISA-система може бути завантажена в режимі ISA для виконання конфігурації утилітою ECU (EISA Configuration Utility)
Invalid EISA Configuration	Конфігураційна інформація EISA недійсна. Система може бути завантажена в режимі ISA для виконання конфігурації утилітою ECU
EISA CMOS Checksum Failure EISA Configuration Checksum Error	Помилка контрольної суми визначеної CMOS-пам'яті конфігурації пристроїв EISA, можливо, через батарею. Система може бути завантажена в режимі ISA для виконання конфігурації утилітою ECU
EISA CMOS Inoperational	Помилка доступу (читання-запис) до додаткової CMOS-пам'яті конфігурації пристроїв EISA, можливо через батарею
Expansion Board not ready at Slot X	Плата розширення в слоті X (EISA) не готова. Перевірити плату і конфігурацію
ID information mismatch for Slot X Wrong Board in Slot X	Ідентифікатор встановленої плати розширення EISA не співпадає із записом в CMOS для цього слота
Slot X Should Be Empty But EISA Board FoundSlot X Not Empty	Слот X шини EISA повинен бути порожнім, але знайдена плата. Виконати конфігурацію утилітою ECU
Slot X Should Have EISA Board But Not Found	Для слота X шини EISA призначена плата, але вона не знайдена. Виконати конфігурацію утилітою ECU
Invalid Configuration Information for Slot X	Некоректна інформація конфігурації для платі розширення EISA в слоті X. Виконати конфігурацію утилітою ECU
BUS Timeout NMI at Slot X	Помилка тайм-ауту звертання по системній шині для плати в слоті X
Fail-Safe Timer NMI	Відбулося переривання від таймера, контролюючого граничний час розтяжки шинного циклу
INTR #1 Error	Помилка контроллера переривань #1 (відповідає за лінії IRQ 0-7)

Повідомлення	*Причина і можливі дії*
INTR #2 Error	Помилка контроллера переривань #2 (відповідає за лінії IRQ 8-15)
8042 Gate A20 Error!	Несправність роботи вентиля лінії A20 (Gate A20) в мікросхемі контроллера клавіатури 8042. Можна обійти, встановивши в Setup опцію Gate A20 Control в значення Fast (управління від чіпсета)
DMA #1 En-or, DMA Error	Помилка контроллера DMA (може бути викликана платами розширення)

При завантаженні системи в разі готовності дисковода A: в пам'ять завантажується перший сектор диска і йому передається управління; при неготовності A: завантажується Master Boot диска C: і йому передається управління. Master Boot завантажує Boot Sector активного розділу в пам'ять і передає управління на його початкову адресу.

Спроба завантаження з дискети може блокуватися або виконуватися тільки після невдалої спроби завантаження з жорсткого диска при відповідному завданні параметра "Boot Sequence" в Setup.

Послідовність завантаження може змінюватися додатковим BIOS мережного адаптера у разі віддаленого завантаження по мережі.

У процесі роботи POST використовуються комірки CMOS 0Fh (Shutdown Flag) – ідентифікатори стану перед початком тесту і BIOS DATA AREA [0:0472] – тип рестарту (1234h=CTRL+ALT+DEL – "теплий" старт, 4321h – скидання зі збереженням пам'яті). Це дозволяє розрізняти причини рестарту (перезавантаження, вихід із захищеного режиму 286 і т. д.) для обходу деяких секцій POST.

В AT результати проходження тестів заносяться в CMOS OEh – Post Diagnostic Status Byte.

Конфігурація комп'ютера – BIOS Setup. Всі сучасні комп'ютери мають утиліту Setup, вбудовану в BIOS. Утиліта *BIOS Setup* має інтерфейс у вигляді меню, іноді навіть віконний з підтримкою миші. Віконний інтерфейс в даному випадку дратує, оскільки замість швидкого входу в текстове меню комп'ютер довго шукає підключену мишу, після чого виводить вікна в режимі графіки низького дозволу (дань сумісності). При цьому ніяких принципово нових

можливостей (в порівнянні з текстовим режимом і управлінням від клавіатури) не з'являється.

Для *входу в Setup* під час виконання POST з'являється пропозиція натискувати клавішу Del. Іноді для цього використовується комбінація Ctrl+Alt+Esc, Esc, Ctrl+Esc, бувають і екзотичні варіанти (натискувати клавішу F12 в ті секунди, коли в правому верхньому кутку екрану видний прямокутник). Деякі версії BIOS дозволяють увійти в Setup по комбінації Ctrl+Alt+Esc у будь-який момент роботи комп'ютера. Пропозиція (і спосіб – натиснення F1 чи F2) входу в Setup з'являється, якщо POST знайде помилку устаткування, яка може бути усунена за допомогою Setup. Утримання клавіші Ins під час POST у ряді версій BIOS дозволяють встановити настройки за умовчанням, відміняючи всі "прискорювачі". Це допомагає відновити працездатність після надмірно агресивних спроб "розігнати" комп'ютер.

Меню утиліти Setup, способи переміщення по пунктах і вибору параметрів залежать від схильностей виробника і версії BIOS, але вони зрозумілі з короткого пояснення на екрані. Натиснення F1 або Alt+H викликає коротку контекстну довідку, звичайно пов'язану з навігацією. Смислових пояснень значення параметрів вона не дає. Склад керованих параметрів, детальна і гнучкість управління варіюється від гранично докладних, в яких може заплутатися і досвідчений користувач, до гранично коротких. Що краще – справа смаку. Нижче наведемо пояснення поширених установок. В конкретній версії вони відрекомендовані, звичайно ж, лише вибірково. Деякі установки можуть називатися і не зовсім так, як вказано в таблицях, але бути співзвучними (в англійському варіанті). За період розвитку PC деякі терміни набули нового значення – якщо раніше під типом мікросхем пам'яті (DRAM Type) мали на увазі об'єм мікросхем (64K, 25K, 1M), то тепер це FPM, EDO, BEDO і SDRAM. У зв'язку з цим можливе двояке тлумачення деяких параметрів, але не можна обійняти неосяжне і перерахувати всі існуючі на цей час параметри настройки. Можливі пункти головного меню Setup наведені в табл. 3.3.

Опція "*Auto Configuration with BIOS Defaults*" дозволяє встановити набір параметрів, що забезпечує нормальну роботу системної плати. При цьому не зачіпається дата, час, параметри гнучких і жорстких дисків. Це є початковою точкою для оптимізації установок, на якій можна і зупинитися.

Таблиця 3.3

Пункти головного меню Setup

Пункт меню	*Призначення*
Standard CMOS Setup	Установка стандартних параметрів CMOS
Advanced CMOS Setup	Установка розширеного набору параметрів
Chipset Setup	Управління особливостями чіпсета (дозвіл прогресивних властивостей і настройки параметрів)
Power Management	Управління режимом енергозбереження
PCI/PnP Setup Plug And Play	Конфігурація розподілу ресурсів
Peripherial Setup	Конфігурація периферії системної плати
Change Password	Зміна (установка) пароля
Optimal	Установка оптимальних параметрів
Auto Configuration with BIOS Defaults	Установка "нормальних" параметрів
Auto Configuration with Power-on Defaults Fail-Safe Утримання Del при включенні	Установка "консервативних" параметрів
Write to CMOS and Exit Save and Exit	Вихід зі збереженням нових установок
Do Not Write to CMOS and Exit Exit without Saving	Вихід без збереження (залишаються колишні значення)

Опція "*Auto Configuration with Power-on Defaults*" – встановлює найконсервативніші значення параметрів: кешування заборонено на обох рівнях, тимчасові діаграми самі розтягнуті і т.п. Якщо системна плата не працює і з такими установками, необхідно перевірити її апаратну конфігурацію – установку джамперів, знімних елементів (процесор, пам'ять, кеш і т. п.). Якщо в Setup не ввійти, ті ж значення параметрів можна набути, утримуючи клавішу Del (іноді Ins) під час включення комп'ютера, або для цих цілей є спеціальний

перемикач (джампер) на системній платі. Спосіб порятунку залежить від версії BIOS і моделі системної плати.

Вибрані установки зберігаються при виході з Setup (за бажанням користувача) і починають діяти з моменту початку наступного виконання POST. Таким чином, якщо немає упевненості в правильності вибраних установок, можна вийти з Setup без збереження нових значень.

Вибрані значення установок рекомендується зберегти на папері. На жаль, функція друку екрану по клавіші PrintScreen з Setup працює не завжди (до ініціалізації під час завантаження LPT-порт може утримувати низький рівень на виході 1МГГ#, що не дозволяє принтеру друкувати). Але витрати часу навіть на рукописне збереження параметрів допоможуть згодом заощадити час, сили і нерви в критичній ситуації, оскільки деякі установки можуть приводити до повної втрати працездатності комп'ютера (на щастя, тимчасової – до виправлення значень).

Невдалі параметри конфігурації (або забутий пароль) при неможливості входу в Setup можна скинути відключенням живлення CMOS (замиканням контактів 3, 4 роз'єму акумулятора). В деяких системах пароль скидається тільки спеціальною перемичкою. Проте в CMOS зберігаються зовсім не всі параметри – частина їх міститься в NVRAM, яку відключенням батареї змінити (очистити) неможливо за визначенням (це дійсно енергозалежна пам'ять). Добре, якщо на системній платі є спеціальний перемикач для обнулення NVRAM (або, хоча б, ESCD). Такий перемикач спрацьовує, якщо в положенні "очищення" на системну плату буде подано живлення (разом з сигналом апаратного скидання). Після обнулення плату включають з нормальним положенням перемикача. Зрідка зустрічається опція Setup, призначена для обнулення NVRAM. Якщо явних способів очищення немає, то при необхідності залишається скористатися утилітою перепрограмувало флеш-BIOS (NVRAM звичайно є областю мікросхеми-носія флеш-BIOS). Проте для цього необхідно мати файл-образ BIOS і утиліту програмування. Записом некоректної інформації в NVRAM іноді грішить Windows 95 при установці. Це може

привести до втрати працездатності окремих вузлів і навіть плати в цілому, не усуненої ніякими настройками Setup.

Пункт *Standard CMOS Setup,* що практично завжди є в головному меню утиліти (в старих машинах він просто єдиний), відноситься до настройки параметрів, що зберігаються в елементах стандартної пам'яті CMOS. Це одна з небагатьох областей Setup, в якій все досить просто і зрозуміло (табл. 3.4).

У цьому ж розділі виводиться і об'єм встановленої оперативної пам'яті. В старих комп'ютерах можна (і потрібно) було розподіляти наявну пам'ять, що перевищує стандартні 640 або 512 Кбайт, між Extended і Expanded. Нові версії BIOS тільки показують об'єм, знайдений тестом, і його розподіл між стандартною пам'яттю (Conventional Memory або Base Memory) і розширеної (Extended Memory). Проте при зміні розміру фізично встановленої пам'яті деякі версії BIOS фіксують цю зміну під час POST і вимагають входу в Standard Setup тільки для того, щоб побачити, згодитися і зберегти зміни в CMOS. Інші версії виконують цю модифікацію вмісту CMOS автоматично, не привертаючи уваги користувача.

Іноді в цьому ж меню знаходиться режим реакції тесту POST на помилки конфігурації (чи зупинятися з повідомленням або по можливості ігнорувати). Тут же може бути і дозвіл контролю паритету оперативної пам'яті.

Вживання зовнішніх утиліт (SETUP.COM та ін.), якими заповнювали комірки CMOS в перших моделях AT, в сучасних комп'ютерах може привести до втрати інформації в CMOS через розбіжність правил підрахунку контрольної суми.

Набір опцій *розширеного Setup* залежить від версії BIOS. Ці опції включають:

- Управління параметрами клавіатури, послідовністю завантаження (C:, A:, CD-ROM); "обмін" гнучких дисків (Swap Floppy); дозвіл тіньової пам'яті, контролю паритету; конфігурація кеш-пам'яті, вбудованої периферії і ін.
- Управління тимчасовими параметрами (частоти синхронізації і кількість тактів очікування) циклів шин, оперативної і кеш-пам'яті.

• Вбудовані утиліти автоматичного визначення типів IDE-дисків, тестування і форматування дисків.

• Група опцій безпеки (Security) – установка пароля на вхід в систему і Setup, обмеження доступу до гнучких дисків.

• Засоби антивірусного захисту – попередження про спробу запису в Boot-сектор і перевірка при завантаженні на збіг його контрольної суми з еталонним значенням, що зберігається в CMOS.

Група *Power Management* або *Green Options* управляє параметрами системи енергозбереження (час і рівні "засипання" і події, що викликають "пробудження" системи).

Таблиця 3.4

Параметри Standard CMOS Setup

Опція	*Можливі значення*	*Призначення*
Date (month/date/year) and Time	Дата (місяць, день і рік) і час	Встановлює значення годинника-календаря в CMOS, з яких ОС бере поточні значення при завантаженні, після чого ОС звичайно відлічує час самостійно
Daylight Saving	Enable/ Disable	Дозвіл перемикання на літній/зимовий час в останню неділю жовтня і квітня
Hard disk C		
Type	1-47, Not Installed, Auto	Тип: 0 або Not Installed застосовується за відсутності жорсткого диска (для його відключення), а також для дисків SCSI.
		Типи 1-46 задають фіксовані параметри, 47 – параметри, визначувані користувачем або утилітою IDE Autodetection. Іноді під параметри користувача виділяють також і тип 46. Значення Auto дозволяє автоматично визначати тип підключеного диска у момент завантаження. Кожному типу відповідають свої значення параметрів Cyln, Head, WPcom, LZone, Sect, Size і режиму трансляції секторів
- Cylinder (Cyl)	1-65535	Кількість циліндрів диска. Для дисків IDE задається логічне значення числа циліндрів
- Heads (Hd)	1-16	Кількість головок диска. Для дисків IDE задається логічне значення числа головок
- Write Precomp (Wpcomp, WP)	-1-65535	Номер циліндра, починаючи з якого включається передкомпенсація запису

Опція	*Можливі значення*	*Призначення*
		(для старих дисків MFM і RLL). Значення -1 або 65535 відміняє передкомпенсацію. Диски IDE цей параметр ігнорують
- Landing Zone (Lzone, LZ)		Номер циліндра для паркування головок (для старих дисків MFM і RLL, що не виконували паркування автоматично). Звичайно використовували значення 0 або співпадаюче з числом циліндрів. Сучасні диски цей параметр ігнорують
- Sectors (Sec, SPT)	1-63	Число секторів на треку. Для дисків MFM типово 17, для RLL – 26. Для дисків IDE задається логічне значення числа секторів
- Size (Capacity)	Мбайт	Автоматично підраховується по формулі Size – (Hds × Cyl × Sect × 512) / 1024
- Normal/LBA/ Large		Режим трансляції адреси сектора
- Block Mode, 32Bit Mode, PIO Mode.	Enable/ Disable	Режими контроллера
Hard disk D type	1-47, Not Installed, Auto	Тип другого жорсткого диска (аналогічно першому)
Primary Master, Primary Slave, Secondary Master і Secondary Slave	1-47, Not Installed, Auto	Тип жорстких дисків, підключених до первинного і вторинного контроллерів IDE, розташованих на системній платі PCI. Аналогічно попередньому
Floppy drive A	360 (5"), 720, 1,44, 2,88, None (Not Installed)	Тип НГМД з фізичною адресою A: (підключеного до роз'єму з переверненою частиною шлейфу)
Floppy drive B	360 (5"), 720, 1,44, 2,88, None (Not Installed)	Те ж для накопичувача B: (підключеного до роз'єму з неперевернутою частиною шлейфу)
Primary display	MDA (Mono), CGA40, CGA80, VGA/PGA/EGA або Absent	Тип графічного адаптера і його режим при завантаженні. В системі з двома адаптерами дозволяє вибрати тип первинного (використовуваного через BIOS) графічного адаптера. Іноді дозволяє завантажувати комп'ютер і без графічного адаптера
Keyboard System Keyboard	Installed/Not installed Present /Absent	Дозвіл виконання в POST-тесту клавіатури. Вибір Not Installed дозволяє завантажувати комп'ютер (наприклад, сервер) з від'єднаною клавіатурою
Hit Del Message Display	Enable/Disable	Виведення запрошення до входу в Setup "Hit Del if you want to run Setup"
Halt on Error	Список	Дозвіл зупинки POST по помилках (чекаючи натиснення Fl).

Опція	*Можливі значення*	*Призначення*
		Виключення зупинки помилково клавіатури також дозволяє завантажувати комп'ютер з від'єднаною клавіатурою

Setup систем з шинами MCA, EISA, PCI має спеціальні опції конфігурації системних ресурсів, що надаються периферійним контроллерам.

З утиліт обслуговування жорстких дисків – *Hard Disk Utility* – до дисків IDE можна сміливо застосовувати лише автоматичне розпізнавання типу диска *Auto Detect Hard Disk (IDE Autodetection).* Автоматично певні логічні параметри можуть бути занесені у відповідні поля опису диска типу 47, але можливе і їх редагування. Для дисків розміром більше 528 Мбайт пропонується вибір режиму LBA або Large Disk. Для спрощення переносимості жорстких дисків рекомендується погоджуватися з параметрами, пропонованими за умовчанням. Проте для дисків, що раніше використалися з типом 1-46, параметри, визначувані автоматично, можуть не співпасти з параметрами, з якими він форматувався і використовувався. В результаті з нього може відмовитися завантажуватися ОС (легка і очевидна нестиковка), але розбіжність може виявитися і пізніше втратою даних.

Форматування жорстких дисків утилітами BIOS Setup застосовне лише для дисків з контроллерами ST506 (MFM, RLL). Форматування IDE-дисків може привести до непередбачуваних результатів (в кращому разі "розумний" диск проігнорує цю операцію, в разі середньої важкості загубляться оптимальні параметри і диск працюватиме повільно, в гіршому разі диск перестане працювати взагалі).

Для низькорівневого форматування (LOW LEVEL FORMAT) є опції Auto Interleave (автоматичного визначення чинника чергування) і Media Analyse (аналізу поверхні). При форматуванні вся інформація на диску втрачається.

Параметри розширеної конфігурації (Advanced Setup) кочують з однієї групи в іншу навіть у різних версіях BIOS одного виробника, тому в нижченаведених таблицях вони групуються можливо і не так, як в якому-небудь конкретному варіанті.

Відносно нескладні настройки, що не вимагають знання тимчасових параметрів компонентів, наведені в табл. 3.5.

Таблиця 3.5

Загальні опції Advanced Setup

Опція	*Можливі значення*	*Призначення*
Above 1MB Memory Test	Enable/Disable	Дозвіл тестування пам'яті вище 1 Мбайт. Заборона прискорює проходження POST – при цьому пам'ять вище 1 Мбайт не тестується, а тільки ініціалізувався
Quick Boot Quick Power On Self Test	Enable/Disable	Дозвіл швидкого завантаження (пам'ять понад 1 Мбайт не тестується, готовність HDD перевіряється без очікування)
Memory Test Tick Sound	Enable/Disable	Дозвіл звукового супроводу тесту пам'яті.
Wait for Fl If Any Error	Yes/No	Очікування натиснення F1 помилки, знайденої POST. Може бути заданий список (або, навпаки, виключення) помилок, що вимагають натиснення клавіші. Заборона зупинки за помилкою клавіатури використовується для серверів, що працюють з від'єднаною клавіатурою
System Boot Num Lock	On/Off	Стан індикатора NumLock і використання цифрового поля клавіатури після завантаження
Floppy Drive Swap	Enable/ Disable	Дозвіл взаємної заміни імен дисків A: і B:. При дозволі фізичний диск B: (підключений до "прямого" роз'єму) отримує логічне ім'я A: і з нього стає можливим завантаження, фізичний диск A: стає логічним B:. Параметри дисководів, задані в Standard Setup, відносяться до фізичних імен
Floppy Drive Seek at Boot	Enable/ Disable	Дозвіл перевірки позиціювання голові до НГМД перед завантаженням і визначення типу накопичувача (40 або 80 доріжок). При завантаженні не з гнучкого диска економить час (пошук нульової доріжки виконується при першому зверненні ОС до дисковода)
SystemBoot Sequence	Список з пристроїв A:, C:, D:, E:, F:, CDROM, LS120	Послідовність опитування дисків при завантаженні. Традиційно BIOS при готовності дисковода A: починає завантаження з дискети, а при

Опція	*Можливі значення*	*Призначення*
		неготовності гнучкого диска починає завантаження з жорсткого диска (послідовність A:, C:). Зміна послідовності є одним із засобів захисту від несанкціонованого доступу до комп'ютера і оберігає від випадкового завантаження з дискети, залишеної в приводі. Крім того, завантаження із зміненою послідовністю проходить швидше, особливо якщо заборонити "Floppy Drive Seek at Boot"
Password Checking Option Security Option	None/ Setup/ Always	Перевірка пароля при вході. Захищає від несанкціонованого доступу або зміни настройок комп'ютера. Якщо пароль не заданий користувачем явно, то для AMI BIOS пароль за умовчанням "AMI", для AWARD BIOS – "BIOSTAR" або "AWARDSW"
Typematic Rate Programming	Enable / Disable	Дозвіл програмування параметрів автоповтору клавіатури
Typematic Rate Delay (msec)	250-1000 mc	Затримка до початку автоповторів при утриманні клавіші. Вибирається по темпераменту користувача
Typematic Rate (Chars/Sec)	6-30 символов/с Fast/Slow	Частота автоповтору символу натискуючої клавіші
System Boot CPU speed	HIGH/LOW	Початкова швидкість процесора, встановлювана при початку завантаження. В будь-який момент часу швидкістю можна управляти з клавіатури: Ctrl+Alt "+" включає високу швидкість, Ctrl+Alt "–" включає низьку (працює не зі всіма версіями BIOS)
Mouse Support	Enable/Disable	Дозвіл роботи порту PS/2 Mouse
IRQ12/Mouse Function	Enable/Disable	Дозвіл використання IRQ12 для PS/2 Mouse
Boot Sector Virus Protection Virus Warning	Enable/ Disable	Дозвіл антивірусного захисту. Рекомендується заборонити і користуватися резидентними антивірусними засобами відповідно до ОС, що використовується
External CacheMemory	Enable/ Disable	Дозвіл роботи зовнішнього (L2) кеша. Комп'ютер із забороненим кешуванням працює суттєво повільніше. Дозвіл роботи кеша при його фізичній відсутності (або несправності) звичайно приводить до "зависань". "Розумна" версія BIOS не

Опція	*Можливі значення*	*Призначення*
		підключає відсутній або несправний кеш, навіть якщо він дозволений користувачем
Internal Cache Memory CPU Internal Cache Internal Cache	Enable/ Disable	Дозвіл роботи внутрішнього (L1) кеша (для 486+ і деяких 386 процесорів). Аналогічно попередній опції
External Cache Write Policy External Cache	WriteBack (WB)/ WriteThrough(WT)/ Disabled	Політика запису зовнішнього кеша: WT – крізна, WB – зворотна. Тут же може бути і заборона зовнішнього кеша. В більшості випадків ефективність кеша WB помітно вище, але його реалізація складніше і кількість можливих джерел неприємностей більше. Вибирається відповідно до можливостей системної плати (чіпсета)
Internal Cache Write Policy Internal Cache	WriteBack (WB)/ WriteThrough(WT)/ Disabled	Політика запису внутрішнього кеша: WT – крізна, WB – зворотна. Аналогічно попередній опції, але вибирається відповідно до можливостей не тільки чіпсета, але і процесора
Memory Parity Error Check	Enable/Disable	Дозвіл контролю паритету пам'яті. Якщо системна плата підтримує контроль паритету, а всі встановлені модулі мають контрольні біти (і відповідну конфігурацію), варто дозволити (продуктивність не страждає, а надійність виграє). Повідомлення типа PARITY ERROR AT Оабо 0123 SYSTEM HALTED, які при цьому можуть з'являтися, повинні радувати конкретністю вказівки на несправність пам'яті (замість загадкових зависань)
ECC Memory	Режим	Встановлюється залежно від вживаних модулів пам'яті
Memory Remapping Memory Relocation	Enable/Disable	Дозвіл переміщення 384 Кбайт ОЗУ з області A0000-FFFFF за межу 1 Мбайт. Дозволяє використовувати цю пам'ять як розширену, але суперечить використанню тіньової пам'яті (Shadow ROM, RAM) навіть в малих об'ємах (область звичайно переміщається тільки цілком). Нові системи звичайно переміщення і не пропонують (дуже дрібний шматок, щоб з ним возитися), на старих системах переміщення іноді працює лише при 1 Мбайт встановленої пам'яті

Опція	*Можливі значення*	*Призначення*
Memory Hole [at ...]	Disabled, 512-640K або 15-16MB	Неадресована область пам'яті. Вибір 512-640K зменшує об'єм основної (conventional) пам'яті, вибір 15-16MB не дозволяє використовувати пам'ять більше 15 Мбайт. Звичайно заборонено (Disabled)
Shadow Memory Cacheable	Enable/Disable	Дозвіл кешування тіньової пам'яті (другий виток прискорення доступу до вмісту ROM). Позитивний ефект дає не завжди
Video ROM Shadow COOO,32K	Enable/Disable	Дозвіл вживання тіньової пам'яті на область BIOS графічного адаптера. Рекомендується дозволити, оскільки суттєво підвищує продуктивність роботи з екраном через BIOS. При цьому звичайно всі старші 384 Кбайт пам'яті з першого мегабайта перестануть бути доступними як ОЗП. Іноді затінювання Video ROM приводить до "зависань". Якщо вживання тіньової пам'яті не прискорює роботу графічних функцій, слід перевірити збіг вказаної області з адресами BIOS встановленої графічної карти. BIOS графічного адаптера, інтегрованого в системну плату, звичайно знаходиться в області System BIOS
Adaptor ROM Shadow C800, 16K	Enable/Disable	Дозвіл вживання тіньової пам'яті на область BIOS додаткового адаптера. Рекомендується дозволити, якщо функції цього розширення BIOS інтенсивно використовуються (недоцільне затінювання BOOT ROM адаптерів локальних мереж, код яких використовується лише одноразово при завантаженні). При цьому звичайно всі старші 384 Кбайт пам'яті з першого мегабайта перестануть бути доступними як ОЗП
Adaptor ROM Shadow CC00.16K	Enable/Disable	Те ж (часто використовується для адаптерів жорстких дисків, SCSI)
Adaptor ROM Shadow D000,16K	Enable/Disable	Те ж (часто використовується для адаптерів ЛВС)
xxOO, 16K	Shadow/ Cache/Disable	Дозвіл тіньової пам'яті або кешування вказаної області адрес (області, починаючи з C000, C400...DC00).

Опція	*Можливі значення*	*Призначення*
		Недопустимо затінювання пам'яті, що розділяється (буферів мережних адаптерів і інтелектуальних графічних адаптерів)
System ROM Shadow FOOO, 64K System BIOS Shadow	Enable/Disable	Дозвіл вживання тіньової пам'яті на область системної BIOS. Для DOS і Windows підвищує продуктивність – рекомендується дозволити, при цьому звичайно всі старші 384 Кбайт пам'яті з першого мегабайта перестануть бути доступними як ОЗП. Іноді дозволяється по окремості областями з кроком адрес в 400h – має сенс затінювати BIOS цілком
Numeric Processor Test	Enable/Disable	Дозвіл тестування математичного співпроцесора (і визначення його присутності)
CPU Selection	Auto або тип процесора	Вибір типу встановленого процесора
Weitek Coprocessor	Enable/Disable	Дозвіл роботи співпроцесора Weitek (якщо він установлений на платі)
Fast Gate A20 Option Gate A20 Option	Enable/Disable Normal/Fast	Вибір швидкого способу перемикання вентиля лінії A20, що використовується в реальному режимі процесора. Стандартний спосіб (через контроллер клавіатури 8042) працює повільно, прискорений (Fast) іноді викликає проблеми сумісності ПО і системної плати, оскільки його реалізація специфічна для кожного типу чіпсета
Keyboard Reset Control	Enable/Disable	Дозвіл управління апаратним скиданням процесора через контроллер клавіатури. Служить для перемикання із захищеного режиму в реальний для процесора 286
Fast Decode	Enable/Disable	Дозвіл швидкого (апаратного, виконуваного спеціальною логікою) декодування команди формування сигналу скидання процесора, що виробляється контроллером клавіатури. На AT застосовувалося для прискорення переходу в реальний режим із захищеного. На машинах 386+ ця назва може відноситися і до настройки декодування адреси шини ISA, що дозволяє прискорити обмін
Hard Disk Type 47 RAM Area	0:300	Область розміщення параметрів жорсткого диска типу 47 (User Defined).

Опція	*Можливі значення*	*Призначення*
		Значення за умовчанням не підходить для деяких мережних ОС
OS/2 Compatible Mode	Enable/Disable	Включення режиму сумісності з IBM OS/2

Група параметрів, які задають "тонкі" настройки (режими і тимчасові діаграми), вимагає більш глибокого знання функціонування підсистем комп'ютера. Загальні принципи настройки такі: чим вищі частоти, менші коефіцієнти ділення і кількості тактів очікування (Wait States), тим вища продуктивність підсистеми і комп'ютера, що зачіпає, в цілому, якщо підсистема використовується інтенсивно. Межі прискорення визначаються швидкодією і кількістю вживаних компонентів і можуть бути виявлені емпірично. Проте можливі побічні ефекти, коли "розгін" однієї підсистеми приводить до непрацездатності іншої, на перший погляд і не сильно з нею зв'язаною. Багато груп параметрів мають *загальну опцію автоконфігурації* (Automatic Configuration). Дозвіл автоконфігурації – типової установки таких параметрів, як коефіцієнти ділення частоти, кількості тактів очікування і т. п., – дозволяє встановити якщо і не оптимальну, то в більшості випадків цілком нормально працюючу конфігурацію. Заборона дозволяє встановити ці параметри вручну (даючи користувачу додаткову можливість помилитися).

На сучасній системній платі завжди розташований високоефективний контроллер IDE, параметрами якого управляють з Setup (табл. 3.6). Включення його прогресивних режимів ще не означає підвищення швидкості обміну з дисками – для цього необхідна ще й програмна підтримка коректними драйверами з боку ОС, і підтримка цих режимів власне накопичувачами.

Таблиця 3.6

Параметри настройки контроллера IDE

Опція	*Можливі значення*	*Призначення*
IDE DMA Transfer Mode	Disabled, Type B (для EISA) Standard (для PCI)	Режим DMA при передачі даних по IDE (може не підтримуватися деякими накопичувачами CD-ROM)
IDE Multiple Sector Mode	Enable/Disable або число	Дозвіл (вказівка максимальної кількості секторів) мультисекторного режиму передачі

IDE Block ModeIDE Multi Block Mode	Enable/Disable	Дозвіл блокового (мультисекторного) режиму передачі
IDE PIO:	Auto, 0-4	Режим передачі IDE (PIO Mode). Обмежує максимальний режим, пропонований контроллером (пристрій обмежить його своїми можливостями). Автоматичне узгодження режимів працює не зі всіма пристроями, тому іноді доводиться його явно обмежувати. Може задаватися окремо для кожного каналу або пристрою
IDE 32-bit Transfer	Enable/Disable	Дозвіл 32-бітного звернення до регістра даних IDE (при цьому за одну 32-бітну операцію процесора по шині IDE послідовно передаються два 16-бітні слова). Прискорює обмін з дисками, але може бути джерелом проблем при некоректних драйверах

Конфігурації підлягають і периферійні пристрої, розташовані на системній платі, – контроллери гнучких дисків, портів і т.п. Їх опції зібрані в групу *Peripheral Setup* (табл. 3.7).

Таблиця 3.7

Опції конфігурації вбудованої периферії

Опція	*Можливі значення*	*Призначення*
Onboard FDC	Enable/Disable	Дозвіл роботи контроллера FDD (якщо FDC і HDC IDE розташовані на різній платі, можливий некоректний Відробляння зміни гнучкого диска)
Onboard Serial Porti	3F8h, 3E8h, Disabled	Адреса або заборона роботи першого COM-порту, встановленого на системній платі
Onboard Serial Port2	28h, 2E8h, Disabled.	Те ж для другого порту
Onboard Parallel Port	378h, 278h, Disabled.	Адреса або заборона роботи LPT-порту, встановленого на системній платі
Parallel Port Mode	Standard (Normal, Std, SPP, Compatible) BiDirectional EPP ECP ECP+EPP 1284 Compliance Fast Centronics	Режим роботи LPT-порту

Опція	*Можливі значення*	*Призначення*
Parallel Port DMA	Disabled, DMA Ch (channel) 0, DMA Ch 1, DMA Ch 3	Заборона на використання DMA, або номер каналу DMA, що використовується в режимі ECP

Група опцій *Power Management* (Green Function) управляє системою зниження енергоспоживання. Різні режими зниження активності (і споживання) включаються через заданий інтервал витримки неактивності користувача (клавіатура, миша) або підсистеми (відсутність звернень до жорсткого диска). В нормальний режим комп'ютер переходить по певних заданих подіях. Некоректна настройка і помилки в BIOS можуть приводити до несподіваного різкого зниження продуктивності. Найпростішим виходом з такої ситуації є заборона режимів зниження споживання, проте для комп'ютерів з автономним живленням енергозбереження досить суттєве. Визначені наступні режими роботи комп'ютера:

- Full On Mode: Режим повної потужності.
- Doze Mode: Зниження активності на 80% – помірне пониження частоти процесора.
- Standby Mode: Зниження активності на 92% – пониження частоти процесора до мінімуму.
- Suspend Mode: Зниження активності на 99%. Процесор зупинений і переривання не відпрацьовує. З цього стану комп'ютер виходить досить довго (одиниці секунд).

Поведінка монітора і жорсткого диска в різних режимах може задаватися довільно. Опції управління енергоспоживанням наведені в табл. 3.8.

Таблиця 3.8

Опції управління енергоспоживанням (Power Management)

Опція	*Можливі значення*	*Призначення*
Power Management	Enable/Disable, Max. Saving Mode, Min. Saving Mode, User Defined Mode	Дозвіл управління енергоспоживанням, вибір режимів максимального, мінімального збереження або визначуваного користувачем
PM Control APM	Enable/Disable	Дозвіл управління енергоспоживанням від системи APM
Green PC Monitor	Disabled, Standby,	Режим, в який переводиться монітор при

Опція	*Можливі значення*	*Призначення*
Power State	Suspend	тривалій неактивності користувача, або заборона управління монітором (Disabled)
Power Down Mode Monitor Power State	Disabled, Standby, Suspend	Режим, в який переводиться монітор в стані PoweDown, або заборона управління монітором (Disabled)
InstantON Support	Enable/Disable	Дозвіл швидкого включення на повну потужність
Video Method	Blank Screen V/H Sync+Blank	Метод виключення монітора: тільки гасіння зображення або ще й зупинка синхронізації (по протоколу DPMI)
Doze Mode Control	Частота процесора, режим монітора, диска	Опис режиму Doze Mode
Standby Mode Control	Частота процесора, режим монітора, диска	Опис режиму Standby Mode
Syspend Mode Control	Частота процесора, режим монітора, диска	Опис режиму Syspend Mode
Inactivity Mode Control	Частота процесора, режим монітора, диска	Опис режиму Inactivity Mode
Hard Disk Power Down Mode	Disabled, Standby, Suspend	Режим, в який переводиться HDD по таймеру
Green Timer Main Board	Disabled або час (1-15 хвилин)	Заборона або час витримки таймера вклю-чення режиму зниженого споживання
Doze Timer	Disabled або час (1-15 хвилин)	Заборона або витримка перед зниженням активності на 80%
Standby Timer	Disabled або час (1-15 хвилин)	Заборона або витримка перед зниженням активності на 92%
Suspend Timer	Disabled або час (1-15 хвилин)	Заборона або витримка перед зниженням активності на 99%. З цього стану комп'ютер виходить досить довго
Standby to Suspend Timeout	Disabled або час (1-15 мін)	Заборона або витримка неактивності в стані Standby, після якої система переводиться в режим Suspend, або заборону цього переходу
Full-On to Standby Timeout (Min)	Disabled або час (1-15 мін)	Заборона або витримка неактивності, після якого проводиться перехід з режиму повної потужності в Standby
HDD Standby TimerHard Disk Timeout (Min)	Disabled або час (1-15 хвилин)	Заборона або витримка перед зупинкою жорсткого диска. Оптимальне значення підбирається з урахуванням об'єму встановленого ОЗП, операційної системи і властивостей додатків, що

Опція	*Можливі значення*	*Призначення*
		використовуються. Іноді викликає неприємності при зверненні до веденого (Slave) диска при "заснулому" ведучому "Master"
Slow Clock Ratio	1:1, 1:2, 1:4, 1:8, 1:16, 1:32, 1:64 або 1:128	Коефіцієнт зниження ефективної тактової частоти процесора в стані Power Down (насправді частота процесора не міняється, але застосовується переривчаста синхронізація)
Display Activity	Monitor або Ignore	Дозвіл переведення в режим Standby по неактивності монітора (відсутності змін зображення)
Power Down & Resume Events: IRQ3...IRQ 15	Monitor / Ignore або On/Off	Відстеження переривань як подій, що означають активність для режимів управління споживанням
Monitor Event In Full On Mode: LPT port Activity COM port Activity ISA Master Activity VESA Slave Acivity IDE Activity Floppy Activity VGA Activity Keyboard Activity Mouse Activity	Enable/Disable	Відстеження подій, що означають активність для режимів управління споживанням. Будь-яка з відстежуваних подій скидає таймер перемикання режиму (перемикання відбувається, якщо відповідний таймер встигає долічити до заданого значення)
Wake Events	Список переривань і (або) функціональних вузлів	Список подій, що "будять"

Тема № 4. Файлові системи операційних систем.

Мета: ознайомити студентів з поняттям та завданнями файлової системи; розглянути основні види файлових систем.

План

1. Файлова система. Завдання файлової системи.
2. Файлові системи Fat.
3. Файлова система NTFS.

Питання для самоконтролю

1. Що таке файлова система?
2. Які завдання виконує файлова система?
3. Які файлові системи підтримує Windows XP? Перечисліть їх, вкажіть переваги та недоліки кожної. Порівняйте.
4. Які обмеження мають файлові системи?
5. Яку структуру має файлова система NTFS?
6. Які можливості файлової системи NTFS?

Файлова система

Файлова система (англ., file system) – регламент, що визначає спосіб організації, зберігання і іменування даних на носіях інформації. Вона визначає формат фізичного зберігання інформації, яку прийнято групувати у вигляді файлів.

Конкретна файлова система визначає розмір імені файлу (теки), максимальний можливий розмір файлу і розділу, набір атрибутів файлу. Деякі файлові системи надають сервісні можливості, наприклад, розмежування доступу або шифрування файлів.

Файлова система зв'язує носій інформації. з одного боку і API (інтерфейс програмування додатків) для доступу до файлів – з іншою. Коли прикладна програма звертається до файлу, вона не має жодного уявлення про те, яким чином розташована інформація в конкретному файлі, так само. як і на якому фізичному типові носія (CD, жорсткому диску, магнітній стрічці, блоці флеш-пам'яті або другому носії) він записаний. Все, що знає програма – це ім'я файлу, його розмір і атрибути. Ці дані вона отримує від драйвера файлової системи.

Саме файлова система встановлює, де і як буде записаний файл на фізичному носієві (наприклад, жорсткому диску).

З точки зору операційної системи (ОС), весь диск є набір кластерів розміром від 512 байт і вище. Драйвери файлової системи організовують кластери у файли і каталоги (що реально є файлами, які містять список файлів в цьому каталозі). Ці ж драйвери відстежують, які з кластерів в даний час використовуються, які вільні, які помічені як несправні.

Проте файлова система не обов'язково безпосередньо пов'язана з фізичним носієм інформації. Існують віртуальні файлові системи, а також мережеві файлові системи, які є лише способом доступу до файлів, що знаходяться на видаленому комп'ютері.

Завдання файлової системи

Основні функції будь-якої файлової системи націлені на вирішення наступних завдань:

- ✓ іменування файлів;
- ✓ програмний інтерфейс роботи з файлами для додатків;
- ✓ відображення логічної моделі файлової системи на фізичну організацію зберігання даних;
- ✓ організація стійкості файлової системи до збоїв живлення, помилок апаратних і програмних засобів;
- ✓ вміст параметрів файлу, необхідних для правильної його взаємодії з іншими об'єктами системи (ядро, додатки та ін.).

У розрахованих на багато користувачів системах з'являється ще одне завдання: захист файлів одного користувача від несанкціонованого доступу іншого користувача, а також забезпечення спільної роботи з файлами, наприклад, при відкритті файлу одним з користувачів, для інших цей же файл тимчасово буде доступний в режимі «лише читання».

Windows XP підтримує наступні файлові системи: Fat12, Fat16, Fat32 і NTFS. Крім того, дві файлові системи підтримуються на пристроях CD-ROM і DVD: Compact Disc File System (**CDFS**) і Universal Disk Format (**UDF).**

Windows XP підтримує розподілену файлову систему (Distributed File System, **DFS**) і шифруючу файлову систему (Encrypting File System, **EFS**). Хоча **DFS** і **EFS** і названі "файловими системами", вони не є такими в строгому розумінні цього терміну. Так, **DFS** є розширенням мережевого сервісу, що дозволяє об'єднати в єдиний логічний том мережеві ресурси, розташовані на різних комп'ютерах в розділах, які можуть мати різні файлові системи. (**DFS** відноситься до серверних технологій) Що стосується **EFS** – це надбудова над **NTFS**, яка доповнює **NTFS** можливостями шифрування даних.

Файлові системи FAT

FAT (найчастіше в главі мається на увазі Fat16) є простою файловою системою, розробленою для невеликих дисків і простих структур каталогів. Її назва походить від назви методу, вживаного для організації файлів – таблиця розміщення файлів (File Allocation Table, FAT). Ця таблиця розміщується на початку тому. В цілях захисту тому на нім зберігаються дві копії FAT. В разі пошкодження першої копії FAT дискові утиліти (наприклад, Scandisk) можуть скористатися другою копією для відновлення тому. Таблиця розміщення файлів і кореневий каталог повинні розташовуватися по строго фіксованих адресах, щоб файли, необхідні для запуску системи, були розміщені коректно.

За принципом побудови FAT схожа на зміст книги, оскільки операційна система використовує її для пошуку файлу і визначення кластерів, які цей файл займає на жорсткому диску. Спочатку компанія Microsoft розробила FAT для управління файлами на дискетах, і тільки тоді прийняла її як стандарт для управління дисками в MS-DOS. Спочатку для дискет і невеликих жорстких дисків (менше 16 Мбайт) використовувалася 12-розрядна версія FAT (так звана Fat12). У MS-DOS v. 3.0 була введена 16-розрядна версія FAT для більших розміром дисків. У Windows XP система Fat12 застосовується лише на гнучких дисках і томах розміром менше 16 Мбайт. Наприклад, всі 3.5-дюймові дискети ємкістю 1,44 Мбайт форматуються для Fat16.

У Windows Nt/2000 Fat16 працює точно так, як і в MS-DOS, Windows 3.1 і Windows 95/98. Підтримка цієї файлової системи включена в Windows XP, оскільки вона сумісна з більшістю операційних систем інших фірм-

постачальників програмного забезпечення. Окрім цього, вживання Fat16 забезпечує можливість оновлення попередніх версій операційних систем сімейства Windows.

Не можна використовувати Windows Nt/2000/xp спільно з програмними засобами, що здійснюють розбиття диска на томи і стискування дисків за допомогою драйверів пристроїв, які завантажуються з MS-DOS. Наприклад, якщо потрібно мати доступ до розділу або логічного диска FAT, працюючи під управлінням Windows Nt/2000/xp, не слід застосовувати для них такі засоби стискування, як Doublespace (MS-DOS 6.0) або Drivespace (MS-DOS 6.22). Для сканування і відновлення томів FAT, використовуваних Windows Nt/2000/xp, рекомендується ввести в командному рядку команду chkdsk. Ця програма об'єднує функціональні можливості, властиві програмам MS-DOS Chkdsk і Scandisk, включаючи сканування поверхні жорсткого диска. Якщо потрібно виконати сканування поверхні диска, дайте з командного рядка команду Chkdskr.

Файлова система Fat32

32-розря дна файлова система Fat32 з'явилася в Windows 95 Osr2 і підтримується в Windows 98/ME і Windows 2000/XP. Вона забезпечує оптимальний доступ до жорстких дисків, підвищуючи швидкість і продуктивність всіх операцій введення/виводу. Fat32 є вдосконаленою версією FAT, призначеною для використання на томах, об'єм яких перевищує 2 Гбайт.

Для забезпечення максимальної сумісності з існуючими прикладними програмами, мережами і драйверами пристроїв, Fat32 була реалізована з мінімумом можливих змін в архітектурі і внутрішніх структурах даних. Всі утиліти Microsoft, призначені для роботи з дисками (Format, Fdisk, Defrag і Scandisk), були перероблені для забезпечення підтримки Fat32.

Таблиця 1. Порівняння характеристик Fat16 і Fat32 в операційних системах.

FAT16	FAT32

FAT16	FAT32
Підтримується більшістю операційних систем, в числі яких MS-DOS, Windows 9x/ME, Windows NT, Os/2 і UNIX	На даний момент підтримується лише операційними системами Windows 98 (і Windows 98 Second Edition), Windows 95 Osr2, Windows ME і Windows 2000/XP
Ефективна лише на логічних дисках, розмір яких не перевищує 256 Мбайт	Не підтримуються диски, розмір яких менше 32 Мбайт
Підтримує стискування диска за допомогою таких утиліт, як Drvspace	Не підтримує стискування диска
Обмежена за розміром до 65524 кластерів. Кожен кластер має фіксований розмір залежно від розміру логічного диска. Обмеження по кількості кластерів, і їх розміру (32 Кбайт) приводять до загального обмеження за розміром диска (не більше 2 Гбайт). Окрім цього, Fat12/16 зазвичай має обмеження по кількості файлів і тек, які можуть міститися в кореневому каталозі (залежно від диска максимальне значення вагається від 200 до 400)	Максимальний розмір кластера—16 Кбайт, максимальний розмір диска — 32 Тбайт
Оскільки із збільшенням розміру диска розмір кластера Fat16 збільшується, зберігання файлів на таких дисках стає неефективним. Наприклад, якщо файл розміром 10 Кбайт зберігається в кластері розміром 32 Кбайт, то 22 Кбайт дискового простору не використовуються	Для дисків розміром менше 8 Гбайт розмір кластера — 4 Кбайт

Fat32 забезпечує наступні переваги в порівнянні з колишніми реалізаціями FAT:

~ **Підтримка дисків розміром до 2 Тбайт.** Слідує, відзначити, що команда format, включена в Windows XP, не дозволяє форматувати для використання Fat32 томи, розмір яких перевищує 32 Гбайт. Тому при форматуванні томів об'ємом більше 32 Гбайт слід використовувати файлову систему NTFS. Проте драйвер Fastaft, що є у складі Windows XP, дозволяє

вмонтовувати і підтримувати будь-які томи Fat32, у тому числі і такі, об'єм яких перевищує 32 Гбайт. За винятком згаданого обмеження Fat32 в Windows XP працює точно так, як і в Windows 95 Osr2 і Windows 98.

~ **Ефективніше витрачання дискового простору**. Fat32 використовує дрібніші кластери, що дозволяє підвищити ефективність використання дискового простору на 10-15% в порівнянні з FAT.

~ **Підвищена надійність і швидше завантаження програм**. На відміну від Fat12 і Fat16, Fat32 володіє можливістю переміщати кореневий каталог і використовувати резервну копію FAT, якщо перша копія отримала пошкодження. Крім того, завантажувальний сектор Fat32 був розширений в порівнянні з Fat16 і містить резервні копії життєво важливих структур даних. Підвищена стійкість Fat32 обумовлена саме цими чинниками.

Файлова система NTFS

Файлова система Windows NT (NTFS) забезпечує таке поєднання продуктивності, надійності і ефективності, якого неможливо добитися за допомогою будь-якої з реалізації FAT (як Fat16, так і Fat32). Основними цілями розробки NTFS були забезпечення та швидкісне виконання стандартних операцій над файлами (включаючи читання, запис, пошук) і надання додаткових можливостей, включаючи стискування і відновлення пошкодженої файлової системи на великих дисках.

NTFS володіє характеристиками захищеності, підтримуючи контроль доступу до даних і привілеї власника, що грають виключно важливу роль в забезпеченні цілісності життєво важливих конфіденційних даних. Теки і файли NTFS можуть мати призначені їм права доступу незалежно від того, є вони загальними чи ні. NTFS – єдина файлова система в Windows Nt/2000/xp, яка дозволяє призначати права доступу до окремих файлів. Проте, якщо файл буде скопійований з розділу або том NTFS в розділ або на том FAT, всі права доступу і інші унікальні атрибути, властиві NTFS, будуть втрачені.

У Windows 2000 була введена нова версія NTFS – NTFS 5.0. Нові структури даних, що з'явилися у складі цієї реалізації, дозволяють використовувати оригінальні можливості Windows 2000, наприклад, квоти на

використання диска для кожного користувача, шифрування файлів, відстеження посилань, точки переходу (junction points), вбудовані набори властивостей (native properly sets). Крім того, додавати додатковий дисковий простір до томів NTFS 5.0 можна без перезавантаження. Windows XP не пропонує принципових новин для цієї файлової системи, проте деякі поліпшення все ж з'явилися.

NTFS – найкращий вибір для роботи з томами великого об'єму. При цьому слід врахувати, що якщо до системи пред'являються підвищені вимоги (до яких належать забезпечення безпеки і вживання ефективного алгоритму стискування), то частина з них можна реалізувати лише за допомогою NTFS. Тому у ряді випадків потрібно використовувати NTFS навіть на невеликих томах.

Структура тома NTFS

Файлова система NTFS, як і FAT, використовує кластери як фундаментальну одиницю дискового простору. У NTFS розмір кластера за умовчанням залежить від розміру тому. Програма Disk Administrator дозволяє встановлювати розмір кластера до 4 Кб. Якщо для форматування тома NTFS використовується програма Format, що запускається з командного рядка, то потрібний розмір кластера можна вказати як параметр цієї команди. Допустимі розміри кластерів приведені в наступній таблиці:

Форматування тома для використання файлової системи NTFS приводить до створення декількох системних файлів і головної таблиці файлів (Master File Table, MFT). MFT містить інформацію про всі файли і теки, що є на томі NTFS.

Основну інформацію про том NTFS містить завантажувальний сектор розділу (Partition Boot Sector), який починається з сектора 0 і може мати довжину до 16 секторів. Він складається з двох структур:

~ Блоку параметрів BIOS. Ця структура містить інформацію про будову тому і структури файлової системи.

~ Коди, яка описує, як знайти і завантажити файли для будь-якої із завантажуваних операційних систем.

Сумісність операційної системи з файловою.

Операційна система	**Файлова система**		
	NTFS	**FAT**	**FAT32**
MS DOS, Windows 3.x і Windows95 (версії до Osr2)	Ні	Так	Ні
Windows 95 Osr2, Windows 98 і Windows ME	Ні	Так	Так
Windows NT 3.51	Так (окрім NTFS 5.0)	Так	Ні
Windows NT 4.0	Так (з Service Pack 4)	Так	Ні
Windows 2000/XP	Так	Так	Так

Обмеження файлових систем.

Обмеження	**NTFS**	**FAT і Fat32**
Розміри тому	Мінімальний розмір тому складає приблизно 8 Мбайт. На практиці рекомендується створювати томи, розмір яких не перевищує 2 Тбайт. За допомогою NTFS не можна форматувати дискети	FAT підтримує різні розміри томів – від об'єму дискет до 4 Гбайт. Fat32 підтримує томи об'ємом від 2 Гбайт до 2 Тбайт. Працюючи під управлінням Windows XP для Fat32 можна відформатувати томи, об'єм яких не перевищує 32 Гбайт.

Розміри файлів	Теоретично розмір файлу може складати 16 екзабайт (263-1)	FAT підтримує файли розміром не більше 2 Гбайт. Fat32 підтримує файли розміром не більше 4 Гбайт.

Розміри кластерів

При форматуванні дискові томи розмічаються на кластери – це мінімальний простір, що виділяється на диску для файлів. Для будь-якої файлової системи розмір кластера за замовчуванням визначається розміром тому.

Розміри кластерів за замовчуванням для Fat16, Fat32 і NTFS

Розмір диска	Розмір кластерів **FAT16**	Розмір кластерів **FAT32**	Розмір кластерів **NTFS**
513-1024 Мбайт	**16 Кбайт**	**4 Кбайт**	**1 Кбайт**
1025-2048 Мбайт **(2 Гбайт)**	**32 Кбайт**	**4 Кбайт**	**2 Кбайт**
2049-4096 Мбайт **(4 Гбайт)**	**64 Кбайт**	**4 Кбайт**	**4 Кбайт**
4097-8192 Мбайт **(8 Гбайт)**	Не підтримується	**4 Кбайт**	**4 Кбайт**
8193-16384 Мбайт **(16 Гбайт)**	Не підтримується	8 Кбайт	**4 Кбайт**
16385-32768 Мбайт **(32 Гбайт)**	Не підтримується	**16 Кбайт**	**4 Кбайт**
От 32 Гбайт	Не підтримується	Не підтримується	**4 Кбайт**

Можливості NTFS.

Деякі з можливостей, що забезпечуються лише файловою системою NTFS, перераховані нижче:

~ Можливість індивідуальної установки прав доступу до конкретних файлів і каталогів. Це дозволяє встановити, які користувачі і групи мають доступ до файлу або теки, з вказівкою типа доступу. NTFS забезпечує ширший діапазон типів прав доступу, чим FAT, і права доступу до файлів і каталогів можна встановлювати на індивідуальній основі.

~ Файлова система NTFS володіє вбудованими засобами забезпечення відновлюваності, і тому ситуації, коли користувач повинен запускати на томі NTFS програму відновлення диска, досить рідкі. Навіть в разі краху системи NTFS має можливість автоматично відновити несуперечність файлової системи, використовуючи журнал транзакцій і інформацію контрольних крапок.

~ Структура тек файлової системи NTFS дозволяє істотно прискорити доступ до файлів в теках великого об'єму в порівнянні з системою FAT.

~ Поліпшення процедури стискування. Використання можливостей стискування NTFS дозволяє здійснювати стискування окремих тек і файлів. За допомогою цих засобів можна читати стислі файли і писати в них без необхідності виклику програми, що виконує декомпресію.

Файлові системи операційних систем класу Unix

Структура файлової системи.Інформація на дисках розміщується блоками, по 512 байт в кожному блоці. Диск розбивається на наступні області:

Невикористовуваний блок
Суперблок
i – вузол 1
i – вузол 2
i – вузол 3
: :
i – вузол n
Блок з даними файла
Блок з даними файла
Блок з даними файла
Вільний блок
Файл
Вільний блок
: :

на диску невикористовуваний блок; керуючий блок або суперблок, у якому вміщується розмір диска і границі інших областей; і-список, який складається з описів файлів, що називаються і-вузлами; область для збереження вмісту файлів.

Кожний і-вузол вміщує: ідентифікацію власника; ідентифікацію групи власника; біти захисту; фізичні адреси на диску, де знаходиться вміст файлу; розмір файлу; час створення файлу; час останнього використання файлу (modification time); час останньої зміни атрибутів (change time); число зв'язків-посилань, що вказують на файл; індикацію, чи є файл директорією, звичайним файлом чи спеціальним файлом.

Слідом за і-списком йдуть блоки, призначені для збереження вмісту файлів. Простір на диску, що залишився вільним від файлів, утворює зв'язаний список вільних блоків.

Таким чином, файлова система UNIX являє собою структуру даних, яка розміщується на диску і утримує керуючий суперблок, в якому визначена файлова система в цілому; масив і-вузлів, де визначені всі файли у файловій системі; самі файли; і, нарешті, сукупність вільних блоків. Виділення простору під дані здійснюється блоками фіксованого розміру.

Кожний файл однозначно ідентифікується *старшим номером пристрою, молодшим номером пристрою та і-номером* (індексом і -вузла даного файлу в масиві і-вузлів). Коли викликається драйвер пристрою, за старшим номером індексується масив вхідних точок в драйвери. За молодшим номером драйвер вибирає один пристрій із групи ідентичних фізичних пристроїв.

Файл-директорія, у якому перераховані імена файлів, дозволяє установити відповідність між іменами і самими файлами. Директорії утворюють деревоподібну структуру. На кожний звичайний файл або файл пристрою можуть бути посилання в різних вузлах цієї структури. У непривілейованих програмах запис у директорії не дозволена, але при наявності відповідних дозволів вони можуть читати їх. Додаткових зв'язків між директоріями немає.

Велике число системних директорій система UNIX використовує для своїх власних потреб. Одна з них, коренева директорія, є базою для всієї структури директорій, і, "відштовхуючися" від неї, можна знайти розміщення усіх файлів. В інших системних директоріях вміщуються програми і команди, що надаються користувачам, і файли пристроїв.

Імена файлів задаються послідовністю імен директорій, що розділяються похилою рискою ("/") і приводять до кінцевого вузла (листа) деякого дерева. Якщо ім'я файлу починається з косої риски, то пошук по дереву починається в кореневій директорії. Якщо ж ім'я файлу не має на початку косої риски, то пошук починається з поточної директорії. Імена файлів, що починаються з "../", мають на увазі початок пошуку в директорії, розташованій на один рівень вище поточної. Ім'я файлу personal вказує на елемент personal у поточній директорії. Ім'я файлу /work/office/personal приводить до пошуку директорії work у кореневій директорії. Потім до пошуку директорії office у директорії work і, нарешті, до пошуку елемента ppersonal у директорії office. Коса риска / позначає кореневу директорію. У приведеному прикладі знайшла відображення типова ієрархічна структура файлової системи.

Файл, що не є директорією, може зустрічатися в різних директоріях, можливо, під різними іменами. Це називається зв'язуванням. Елемент у директорії, що відноситься до одного файлу, називається зв'язком. У системі UNIX усі такі зв'язки мають рівний статус. Файли не належать директоріям. Скоріше, файли існують незалежно від елементів директорій, а зв'язки в директоріях вказують дійсно на фізичні файли. Файл "зникає", коли видаляється останній зв'язок, що вказує на нього. Біти захисту, задані в зв'язках, можуть відрізнятися від бітів у вихідному файлі. Таким чином, вирішується проблема вибіркового обмеження доступу до файлів. З кожним підтримуваним системою пристроєм асоціюється один або більша кількість спеціальних файлів. Операції вводу-виводу для спеціальних файлів здійснюються так же, як і для звичайних дискових файлів, з тією лише різницею, що ці операції активізують відповідні пристрої. Спеціальні файли звичайно знаходяться в

довіднику /dev. На спеціальні файли можуть указувати зв'язки так же, як на звичайні файли.

Від файлової системи не потрібно, щоб вона вся повністю розміщувалася на тому пристрої, де знаходиться корінь. Запит від системи mount (на установку носіїв і т.п.) дозволяє вбудовувати в ієрархію файлів файли на змінних томах. Команда mount має кілька опцій, але обов'язкових аргументів у стандартного варіанта її використання два: імена файлу блокового пристрою та ім'я каталогу. У результаті виконання цієї команди файлова система, розташована на вказаному пристрої, підключається до системи таким чином, що її вміст замінює собою вміст заданого в команді каталогу. Тому для *монтування* відповідного тому звичайно використовують порожній каталог. Команда unmount виконує зворотну операцію – "від'єднує" (розмонтує) файлову систему, після чого диск із даними можна фізично добути із системи.

Захист файлів

Захист файлів здійснюється за допомогою номера, що ідентифікує користувача, і установки десяти бітів захисту - атрибутів доступу. Права доступу поділяються на три типи: читання (read), запис (write) і виконання (execute). Ці типи прав доступу можуть бути надані трьом класам користувачів: власнику файлу, групі, до якої входить власник, і всім іншим користувачам. Дев'ять з цих бітів керують захистом по читанню, запису і виконанню для власника файлу, інших членів групи, до якої входить власник, і всіх інших користувачів. Файл завжди зв'язаний з визначеним користувачем – своїм власником – і з визначеною групою, тобто в нього є UID (user ID, ідентифікатор користувача) і GID (group ID, ідентифікатор групи). Змінювати права доступу до файлу дозволено тільки його власнику. Змінити власника файлу може суперкористувач, групу – суперкористувач або власник файлу.

Програма, що виконується в системі, завжди запускається від імені деякого користувача і деякої групи (зазвичай – основної групи цього користувача), але зв'язок процесів з користувачами і групами організований складніше: тут розрізняються ідентифікатор для доступу до файлової системи (FSUID – file system access user ID, FSGID – file system access group ID) і

ефективний ідентифікатор (EUID – effective user ID, EGID – effective group ID), а при доступі до файлів враховуються ще і повноваження (capabilities), присвоєні самому процесу.

При створенні файл одержує UID, що співпадає з FSUID процесу, який його створює, і, як правило, GID, що співпадає з FSGID цього процесу. Атрибути доступу визначають, що дозволено робити з даним файлом даній категорії користувачів. Є всього три операції: читання, запис і виконання. При створенні файлу (чи ще одного імені для вже існуючого файлу) модифікується не сам файл, а каталог, у якому з'являються нові посилання на вузли. Видалення файлу полягає у видаленні посилання. Таким чином, право на створення або видалення файлу – це право на запис у каталог. Право на виконання каталогу інтерпретується як право на пошук у ньому (проходження через нього). Воно дозволяє звернутися до файлу за шляхом, що вміщує даний каталог, навіть тоді, коли каталог не дозволено читати і тому список всіх його файлів недоступний.

Крім трьох названих основних атрибутів доступу існують додаткові, що використовуються в наступних випадках. Атрибути SUID і SGID мають значення при запуску програми на виконання: вони вимагають, щоб програма виконувалася не від імені користувача (групи), що її запустив, а від імені власника (групи) того файлу, у якому вона знаходиться. Формально кажучи, якщо файл програми має атрибут SUID (SGID), то FSUID і EUID (FSGID і EGID) відповідного процесу не успадковуються від процесу, що запустив його, а співпадають з UID (GID) файлу. Завдяки цьому користувачі отримують можливість запустити системну програму, яка створює свої робочі файли в закритих для них каталогах. Крім того, якщо процес створює файл у каталозі, який має атрибут SGID, то файл одержує GID не за FSGID процесу, а за GID каталогу. Це зручно для колективної роботи: всі файли і підкаталоги в каталозі автоматично виявляються належними до однієї і тієї ж групи, хоча створювати їх можуть різні користувачі. Є ще один атрибут – CVTX, який тепер відноситься до каталогів. Він показує, що з каталогу, який має цей атрибут, посилання на файл може видалити тільки власник файлу.

Існують дві стандартні форми запису прав доступу – символьна і восьмерична. Символьна являє собою ланцюжок з десяти знаків, перший з який не відноситься до прав, а позначає тип файлу. Використовуються наступні позначення: "-" – звичайний файл; "d" – каталог (директорія); "c" – символьний пристрій; "b" – блоковий пристрій; "p" – іменований канал (named pipe); "s" – "гніздо" (socket[1]); "I" – символічне посилання.

Далі слідують три послідовності, кожна з трьох символів, що відповідають правам користувача, групи і всіх інших. Наявність права на читання позначається буквою "r", на запис – "w", на виконання – "x", відсутність будь-якого права – знаком "-" у відповідній позиції. Наявність атрибута SUID (SGID) позначається заголовною буквою "S" у позиції права на виконання для власника (групи), якщо виконання не дозволене, і прописною буквою "s", якщо дозволене. Восьмеричний запис – це шестизначне число, перші два знаки якого позначають тип файлу і досить часто опускаються, третя цифра – атрибути GUID (4), SGID (2) і SVTX (1), а останні три – відповідно права власника, групи і всіх інших. Очевидно, що право на читання можна представити числом "4", право на запис – числом "2", а право на виконання кодується як "1". Наприклад, стандартний набір прав доступу для каталогу /tmp у символьній формі виглядає як drwxrwxrwt, а у восьмеричній – як 041777 (це каталог; читання, запис і пошук дозволені всім; встановлений атрибут SVTX). А набір прав -r-S-xw-, чи в числовому вигляді – 102412, буде означати, що це звичайний файл, власнику дозволяється читати його, але не виконувати і не змінювати; користувачам із групи файлу (за винятком власника) – виконувати (причому під час роботи програма отримає права власника файлу), але не читати і не змінювати; а всім іншим – змінювати, але не читати і не виконувати. Більшість програм створюють файли з дозволом на читання і запис для всіх користувачів, а каталоги – з дозволом на читання, запис і пошук для всіх користувачів. Цей вихідний набір атрибутів логічно додається до "маски користувача" – *user file-creation mask,* скорочено umask, що зазвичай обмежує доступ. Наприклад, наступні значення для umask u=rwx, g=rwx, o=r-x слід розуміти так: у власника і групи залишається повний набір прав, а всім іншим

забороняється запис. У восьмеричному виді воно запишеться як 002 (перша цифра – обмеження для власника, друга – для групи, третя – для інших, заборона читання – 4, запису – 2, виконання – 1). Власник файлу може змінити права доступу до нього командою chmod.

Тема 5. Планування процесів. Дисципліни планування.

Мета: ознайомити студентів з поняттям процесу та потоку, розглянути загальні принципи, види та стратегії планування; ознайомитися з планування в Linux та Windows XP.

План

1. Означення процесу та потоку.
2. Загальні принципи, види та стратегії планування.
3. Алгоритми планування.
4. Планування в Linux.
5. Планування у Windows XP.

Питання для самоконтролю

1. Що таке процес?
2. Що таке потік?
3. Назвіть складові частини процесу.
4. Назвіть основні елементи процесу.
5. Які елементи містить потік?
6. Що таке планування процесів?
7. Назвіть основні види планування.

Процеси і потоки в сучасних ОС. У сучасній операційній системі одночасно виконуються код ядра (що належить до його різних підсистем) і код програм користувача. При цьому відбуваються різні дії: одні програми і підсистеми виконують інструкції процесора, інші зайняті введенням-виведенням, ще деякі очікують на запити від користувача або інших застосувань. Для спрощення керування цими діями в системі доцільно виділити набір елементарних активних елементів і визначити інтерфейс взаємодії ОС із цими елементами. Коли активний елемент системи зв'язати із програмою, що виконується, ми прийдемо до поняття процесу.

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

Програма – це деяка послідовність машинних команд, що зберігається на диску, в разі необхідності завантажується у пам'ять і виконується. Можна сказати, що під час виконання програму представляє процес.

Однозначна відповідність між програмою і процесом встановлюється тільки в конкретний момент часу: один процес у різний час може виконувати код декількох програм, код однієї програми можуть виконувати декілька процесів одночасно.

Для успішного виконання програми потрібні певні ресурси. До них належать:

~ ресурси, необхідні для послідовного виконання програмного коду (передусім процесорний час);

~ ресурси, що дають можливість зберігати інформацію, яка забезпечує виконання програмного коду (регістри процесора, оперативна пам'ять тощо).

Ці групи ресурсів визначають дві складові частини процесу:

~ послідовність виконуваних команд процесора;

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

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

Потоком (потік керування, нитка, thread) називають набір послідовно виконуваних команд процесора, які використовують загальний адресний простір процесу. Оскільки в системі може одночасно бути багато потоків, завданням ОС є організація перемикання процесора між ними і планування їхнього виконання.

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

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

Захищеність адресного простору процесу є його найважливішою характеристикою. Код і дані процесу не можуть бути прямо прочитані або перезаписані іншим процесом; у такий спосіб захищаються від багатьох програмних помилок і спроб несанкціонованого доступу. Природно, що неприпустимим є тільки прямий доступ (наприклад, запис у пам'ять за допомогою простої інструкції перенесення даних); обмін даними між процесами принципово можливий, але для цього мають бути використані спеціальні засоби, які називають засобами міжпроцесової взаємодії. Такі засоби складніші за прямий доступ і працюють повільніше, але при цьому забезпечують захист від випадкових помилок у разі доступу до даних.

На відміну від процесів потоки розпоряджаються загальною пам'яттю. Дані потоку не захищені від доступу до них інших потоків за умови, що всі вони виконуються в адресному просторі одного процесу. Це надає додаткові можливості для розробки застосувань, але ускладнює програмування.

Захищений адресний простір процесу задає абстракцію виконання коду на окремій машині, а потік забезпечує абстракцію послідовного виконання команд на одному виділеному процесорі.

Адресний простір процесу не завжди відповідає адресам оперативної пам'яті.

Наприклад, у нього можуть відображатися файли або регістри контролерів введення-виведення, тому запис за певною адресою в цьому просторі призведе до запису у файл або до виконання операції введення-виведення. Таку технологію називають відображенням у пам'ять (memory mapping).

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

- В однозадачних системах є тільки один адресний простір, у якому в кожен момент часу може виконуватися один потік.
- У деяких вбудованих системах теж є один адресний простір (один процес), але в ньому дозволене виконання багатьох потоків. У цьому разі

можна організовувати паралельні обчислення, але захист даних застосувань не реалізовано.

❖ У системах, подібних до традиційних версій UNIX, допускається наявність багатьох процесів, але в рамках адресного простору процесу виконується тільки один потік. Це традиційна однопотокова модель процесів. Поняття потоку в даній моделі не застосовують, а використовують терміни «перемикання між процесами», «планування виконання процесів», «послідовність команд процесу» тощо (тут під процесом розуміють його єдиний потік).

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

Надалі для позначення послідовності виконуваних команд вживатимемо термін «потік», за винятком ситуацій, коли обговорюватиметься реалізація моделі процесів.

Складові елементи процесів і потоків. До елементів процесу належать:

~ захищений адресний простір;

~ дані, спільні для всього процесу (ці дані можуть спільно використовувати всі його потоки);

~ інформація про використання ресурсів (відкриті файли, мережні з'єднання тощо);

~ інформація про потоки процесу.

Потік містить такі елементи:

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

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

Види паралелізму. Можна виділити такі основні види паралелізму:

~ паралелізм багатопроцесорних систем;

~ паралелізм операцій введення-виведення;

~ паралелізм взаємодії з користувачем;

~ паралелізм розподілених систем.

Перший з них є справжнім паралелізмом, тому що у багатопроцесорних системах інструкції виконують декілька процесорів одночасно. Особливості використання такого паралелізму будуть розглянуті в розділі 20. Інші види паралелізму можуть виникати і в однопроцесорних системах тоді, коли для продовження виконання програмного коду необхідні певні зовнішні дії.

Паралелізм операцій введення-виведення. Під час виконання операції введення-виведення (наприклад, обміну даними із диском) низькорівневий код доступу до диска і код застосування не можуть виконуватись одночасно. У цьому разі застосуванню треба почекати завершення операції введення-виведення, звільнивши на цей час процесор. Природним вважається зайняти на цей час процесор інструкціями іншої задачі.

Багатопотокове застосування може реалізувати цей вид паралелізму через створення нових потоків, які виконуватимуться, коли поточний потік очікує операції введення-виведення. Так реалізується асинхронне введення-виведення, коли застосування продовжує своє виконання, не чекаючи на завершення операцій введення-виведення.

Паралелізм взаємодії з користувачем. Під час інтерактивного сеансу роботи користувач може виконувати різні дії із застосуванням (і очікувати негайної реакції на них) до завершення обробки попередніх дій. Наприклад, після запуску команди «друкування документа» він може негайно продовжити введення тексту, не чекаючи завершення друкування.

Щоб розв'язати це завдання, можна виділити окремі потоки для безпосередньої взаємодії із користувачем (наприклад, один потік може очікувати введення 3 клавіатури, інший – від миші, додаткові потоки – відображати інтерфейс користувача). Основні задачі застосування (розрахунки, взаємодія з базою даних тощо) у цей час виконуватимуть інші потоки.

Паралелізм розподілених застосувань. Розглянемо серверне застосування, яке очікує запити від клієнтів і виконує дії у відповідь на запит. Якщо клієнтів багато, запити можуть надходити часто, майже водночас. Якщо тривалість

обробки запиту перевищує інтервал між запитами, сервер буде змушений поміщати запити в чергу, внаслідок чого знижується продуктивність. При цьому використання потоків дає можливість організувати паралельне обслуговування запитів, коли основний потік приймає запити, відразу передає їх для виконання іншим потокам і очікує нових.

Загальні принципи, види та стратегії планування

Можливість паралельного виконання потоків залежить від кількості доступних процесорів. Якщо процесор один, паралельне виконання неможливе принципово (у кожен момент часу може виконуватися тільки один потік). Якщо кількість процесорів N>1, паралельне виконання може бути реалізоване тільки для N потоків (по одному потокові на процесор).

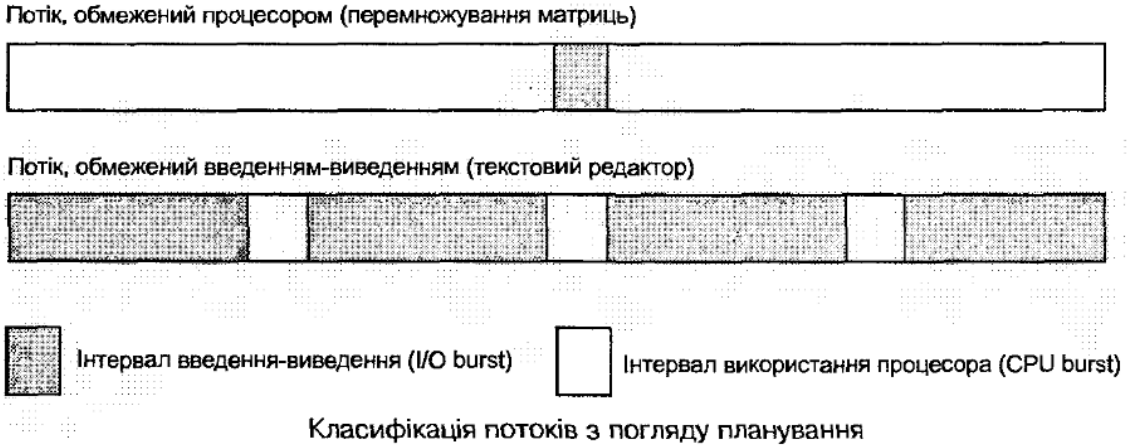
Якщо потоків у системі більше, ніж доступних процесорів, ОС повинна *розв'язувати задачу планування* (scheduling). Головна мета планування для однопроцесорної системи полягає у такій організації виконання кількох потоків на одному процесорі, за якої у користувача системи виникало б враження, що вони виконуються одночасно.

Це означення може бути розширене на багатопроцесорні системи у разі виникнення задачі планування, коли кількість потоків перевищує кількість доступних процесорів.

Особливості виконання потоків. З погляду планування виконання потоку можна зобразити як цикл чергування періодів обчислень (використання процесора) і періодів очікування введення-виведення. Інтервал часу, упродовж якого потік виконує тільки інструкції процесора, називають *інтервалом використання процесора* (CPU burst), інтервал часу, коли потік очікує введення-виведення, – *інтервалом введення-виведення* (I/O burst). Найчастіше ці інтервали мають довжину від 2 до 8 мс.

Потоки, які більше часу витрачають на обчислення і менше – на введення-виведення, називають обмеженими можливостями процесора (CPU bound). Вони активно використовують процесор. Основною їхньою

характеристикою є час, витрачений на обчислення, інтервали використання процесора для них довші. Потоки, які більшу частину часу перебувають в очікуванні введення-виведення, називають обмеженими можливостями введення-виведення (I/O bound). Такі потоки завантажують процесор значно менше, а середня довжина інтервалу використання процесора для них невелика. Що вища тактова частота процесора, то більше потоків можна віднести до другої категорії.



Класифікація потоків з погляду планування

Слід розрізняти механізми і політику планування. До механізмів планування належать засоби перемикання контексту, засоби синхронізації потоків тощо, до політики планування – засоби визначення моменту часу, коли необхідно перемкнути контекст. Ту частину системи, яка відповідає за політику планування, називають планувальником (scheduler), а алгоритм, що використовують при цьому, – алгоритмом планування (scheduling algorithm).

Є різні критерії оцінки політики планування, одні з них застосовні для всіх систем, інші – лише для пакетних систем або лише для інтерактивних.

Сьогодні найчастіше використовують три критерії оцінки досягнення мети.

✓ Мінімальний час відгуку. Це найважливіший критерій для інтерактивних систем. Під часом відгуку розуміють час між запуском потоку (або введенням користувачем інтерактивної команди) і отриманням першої відповіді. Для сучасних систем прийнятним часом відгуку вважають 50-150 мс.

✓ Максимальна пропускна здатність. Це кількість задач, які система може виконувати за одиницю часу (наприклад, за секунду). Такий критерій доцільно застосовувати у пакетних системах; в інтерактивних системах він може бути

використаний для фонових задач. Щоб підвищити пропускну здатність, необхідно:

~ скорочувати час даремного навантаження (наприклад, час, необхідний для перемикання контексту);

~ ефективніше використати ресурси (для того, щоб а ні процесор, а ні пристрої введення-виведення не простоювали).

✓ Третім критерієм є справедливість, яка полягає в тому, що процесорний час потокам виділяють відповідно до їхньої важливості. Справедливість забезпечує такий розподіл процесорного часу, що всі потоки просуваються у своєму виконанні, і жоден не простоює. Відзначимо, що реалізація справедливої політики планування не завжди призводить до зменшення середнього часу відгуку. Іноді для цього потрібно зробити систему менш справедливою.

Види планування. Розрізняють планування довготермінове (long-term scheduling), середньотермінове (medium-term scheduling) і короткотермінове (short-term scheduling).

Засоби *довготермінового планування* визначають, яку з програм треба завантажити у пам'ять для виконання. Таке планування називають також статичним, оскільки воно не залежить від поточного стану системи. Воно відігравало важливу роль у пакетних системах, коли заздалегідь відомо, які процеси повинні бути виконані і можна скласти розклад виконання задач. В інтерактивних системах (наприклад, у системах з розподілом часу) завантаження процесів у пам'ять здійснюють переважно користувачі, і це плануванню не підлягає; тому в них зазвичай використовують спрощену стратегію довготермінового планування. Система дає можливість створювати процеси і потоки до досягнення деякої максимально можливої межі, після чого подальші спроби створити новий процес або потік спричинятимуть помилку. Така стратегія ґрунтується і на психології користувачів, які, почуваючи себе некомфортно в перевантаженій системі, можуть переривати роботу з нею, що призводить до зниження навантаження.

Засоби *середньотермінового планування* керують переходом потоків із призупиненого стану в стан готовності й назад. Відразу ж зазначимо, що керуючі блоки готових до виконання потоків організуються у пам'яті в структуру, яку називають чергою готових потоків (ready queue). Докладніше розглянемо цю чергу під час вивчення короткотермінового планування.

Перехід потоку в призупинений стан можуть викликати такі фактори:

~ очікування операції введення-виведення;

~ очікування закінчення виконання іншого потоку (приєднання);

~ блокування потоку через необхідність його синхронізації з іншими потоками.

Зазвичай для коректної організації такого очікування, крім черги готових потоків, реалізують додатковий набір черг. Кожна така черга пов'язана з ресурсом, який може викликати очікування потоку (наприклад, із пристроєм введення-виведення); ці черги ще називають чергами планування (scheduling queues) або чергами очікування (wait queues). Середньотерміновий планувальник керує всіма цими чергами, переміщаючи потоки між ними та чергою готових потоків.

Короткотермінове планування. Короткотермінове планування, або планування процесора (CPU scheduling), є найважливішим видом планування. Воно дає змогу відповісти на два базових запитання.

~ Коли перервати виконання потоку?

~ Якому потокові з числа готових до виконання потрібно передати процесор у цей момент?

Короткотерміновий планувальник – це підсистема ОС, яка в разі необхідності перериває активний потік і вибирає з черги готових потоків той, що має виконуватися. До його продуктивності ставлять найвищі вимоги, бо він отримує керування дуже часто. Виділяють також диспетчер (dispatcher), який безпосередньо передає керування вибраному потокові (перемикає контекст).

Формат черги готових потоків залежить від реалізації короткотермінового планування. Така черга може бути організована за принципом FIFO, бути чергою із пріоритетами, деревом або невпорядкованим зв'язним списком.

Усі стратегії й алгоритми планування, які ми будемо розглядати далі, належать до короткотермінового планування.

Стратегії планування. Витісняльна і невитісняльна багатозадачність. Перед тим як розглянути основні стратегії планування, перелічимо варіанти передачі керування від одного потоку до іншого:

~ після того, як потік перейшов у стан очікування (наприклад, під час введення-виведення або приєднання);

~ після закінчення виконання потоку;

~ явно (потік сам віддає процесор іншим потокам на час, поки він не зайнятий корисною роботою);

~ за перериванням (наприклад, переривання від таймера дає змогу перервати потік, що виконується довше, ніж йому дозволено).

Останній варіант відрізняється від інших тим, що потік не може контролювати, коли настане час передачі керування, за це відповідає планувальник операційної системи. Залежно від підтримки такого варіанта передачі керування розрізняють дві основні стратегії планування потоків – витісняльну і невитісняльну багатозадачність.

При витісняльній багатозадачності (preemptive multitasking) потоки, що логічно мають виконуватися, можуть бути тимчасово перервані планувальником ОС без їхньої участі для передачі керування іншим потокам. Переривання виконання потоку й передачу керування іншому потокові найчастіше здійснюють в обробнику переривання від системного таймера. Така стратегія реалізована в усіх сучасних ОС, і тому буде розглянута в цьому розділі докладніше.

При невитісняльній багатозадачності (non-preemptive multitasking) потоки можуть виконуватися упродовж необмеженого часу й не можуть бути перервані ОС. Для невитісняльної багатозадачності передача керування за останнім варіантом не реалізована, і потоки самі повинні віддавати керування ОС для передачі іншим потокам або, принаймні, переходити у стан очікування. Якщо якийсь потік забуде або не зможе це зробити, наприклад займе процесор нескінченним циклом, інші потоки не зможуть продовжувати свою роботу.

Таку стратегію було реалізовано в ОС Novell NetWare (наприклад, у версії 3.11, яку широко використовували в 90-х роках XX ст).

Природно, що реалізація невитісняльної багатозадачності в загальному випадку робить систему досить нестабільною (будь-яке некоректно написане застосування користувача може спричинити «зависання» всієї системи). Практика показує, що невитісняльна багатозадачність у системах із застосуваннями користувача не може бути реалізована.

Таку стратегію, проте, можна використати в системах, де всі застосування виконуються в режимі ядра і фактично є системними драйверами. Для розробки таких застосувань необхідна висока кваліфікація програмістів, вимоги до надійності застосувань можна порівняти з вимогами до самої ОС. При цьому простота реалізації та відсутність зовнішніх переривань потоків від планувальника ОС може підвищити продуктивність системи для обмеженого кола задач.

Алгоритм планування дає змогу короткотерміновому планувальникові вибирати з готових до виконання потоків той, котрий потрібно виконувати наступним. Можна сказати, що алгоритми планування реалізують політику планування.

Залежно від стратегії планування, яку реалізують алгоритми, їх поділяють на витісняльні та невитісняльні. Витісняльні алгоритми переривають потоки під час їхнього виконання, невитісняльні – не переривають. Деякі алгоритми відповідають лише одній із цих стратегій, інші можуть мати як витісняльний, так і невитісняльний варіанти реалізації.

Планування за принципом FIFO. Розглянемо найпростіший («наївний») невитісняльний алгоритм, у якому потоки ставлять на виконання в порядку їхньої появи у системі й виконують до переходу в стан очікування, явної передачі керування або завершення. Чергу готових потоків при цьому організовують за принципом FIFO, тому алгоритм називають алгоритмом FIFO.

Як тільки в системі створюється новий потік, його керуючий блок додається у хвіст черги. Коли процесор звільняється, його надають потоку з голови черги.

У такого алгоритму багато недоліків:

~ він за визначенням є невитісняльним;

~ середній час відгуку для нього може бути доволі значним (наприклад, якщо першим надійде потік із довгим інтервалом використання процесора, інші потоки чекатимуть, навіть якщо вони самі використовують тільки короткі інтервали);

~ він підлягає ефекту конвою (convoy effect).

Ефект конвою можна пояснити такою ситуацією. Припустимо, що в системі є один потік (назвемо його T_{cpu}), обмежений можливостями процесора, і багато потоків T_{io}, обмежених можливостями введення-виведення. Рано чи пізно потік T_{cpu} отримає процесор у своє розпорядження і виконуватиме інструкції з довгим інтервалом використання процесора. За цей час інші потоки T_{io} завершать введення-виведення, перемістяться в чергу готових потоків і там чекатимуть, при цьому пристрої введення-виведення простоюватимуть. Коли T_{cpu} нарешті заблокують і відбудеться передача керування, всі потоки T_{io} швидко виконають інструкції своїх інтервалів використання процесора (у них, як ми знаємо, такі інтервали короткі) і знову перейдуть до введення-виведення. Після цього T_{cpu} знову захопить процесор на тривалий час і т. д .

Кругове планування. Найпростішим для розуміння і найсправедливішим витісняльним алгоритмом є алгоритм кругового планування (round-robin scheduling). У середні віки терміном «round robin» називали петиції, де підписи йдуть по колу, щоб не можна було дізнатися, хто підписався першим (ця назва свідчить, що для такого алгоритму всі потоки рівні).

Кожному потокові виділяють інтервал часу, який називають квантом часу (time quantum, time slice) і упродовж якого цьому потокові дозволено виконуватися. Коли потік усе ще виконується після вичерпання кванта часу, його переривають і перемикають процесор на виконання інструкцій іншого потоку. Коли він блокується або закінчує своє виконання до вичерпання кванта часу, процесор теж передають іншому потокові. Довжина кванта часу для всієї системи однакова.

Такий алгоритм реалізувати досить легко. Для цього черга готових потоків має бути циклічним списком. Коли потік вичерпав свій квант часу, його переміщують у кінець списку, туди ж додають і нові потоки. Перевірку вичерпання кванта часу виконують в обробнику переривання від системного таймера.

Єдиною характеристикою, яка впливає на роботу алгоритму, є довжина кванта часу. Тут слід дотримуватися балансу між часом, що витрачається на перемикання контексту, і необхідністю відповідати на багато одночасних інтерактивних запитів.

Є різні способи розв'язання проблеми голодування. Наприклад, планувальник може поступово зменшувати пріоритет потоку, який виконують (такий процес називають старінням), і коли він стане нижче, ніж у наступного за пріоритетом потоку, перемкнути контекст на цей потік. Можна, навпаки, поступово підвищувати пріоритети потоків, які очікують.

Планування на підставі характеристик подальшого виконання. Важливим класом алгоритмів планування з пріоритетами є алгоритми, в яких рішення про вибір потоку для виконання приймають на підставі знання або оцінки характеристик подальшого його виконання.

Насамперед слід відзначити алгоритм «перший – із найкоротшим часом виконання» (Shortest Time to Completion First, STCF), коли з кожним потоком пов'язують тривалість наступного інтервалу використання ним процесора і для виконання щоразу вибирають той потік, у якого цей інтервал найкоротший. У результаті потоки, що захоплюють процесор на коротший час, отримують під час планування перевагу і швидше виходять із системи.

Алгоритм STCF є теоретично оптимальним за критерієм середнього часу відгуку, тобто можна довести, що для вибраної групи потоків середній час відгуку в разі застосування цього алгоритму буде мінімальним порівняно з будь-яким іншим алгоритмом. На жаль, для короткотермінового планування реалізувати його неможливо, тому що ця реалізація потребує передбачення очікуваних характеристик (у розділі 9 буде показано, що це не останній теоретично оптимальний алгоритм із таким недоліком). Для довготермінового

планування його використовують досить часто (у цьому разі, ставлячи задачу на виконання, оператор повинен вказати очікуваний момент її завершення, на який система буде зважати під час вибору). Зазначимо також, що оптимальність такого алгоритму невіддільна від його «несправедливості» до потоків із довшими інтервалами використання процесора.

Для короткотермінового планування може бути реалізоване наближення до цього алгоритму, засноване на оцінці довжини чергового інтервалу використання процесора з урахуванням попередніх інтервалів того самого потоку.

Витісняльним аналогом STCF є алгоритм «перший – із найкоротшим часом виконання, що залишився» (Shortest Remaining Time to Completion First, SRTCF).

Його відмінність від SCTF полягає в тому, що, коли в чергу готових потоків додають новий, у якого наступний інтервал використання процесора коротший, ніж час, що залишився до завершення виконання поточного потоку, поточний потік переривається, і на його місце стає новий потік.

Багаторівневі черги зі зворотним зв'язком. Алгоритм багаторівневих черг зі зворотним зв'язком (multilevel feedback queues) є найуніверсальнішим алгоритмом планування (за допомогою налаштування параметрів його можна звести майже до будь-якого іншого алгоритму), але при цьому одним із найскладніших у реалізації.

З погляду організації структур даних цей алгоритм схожий на звичайний алгоритм багаторівневих черг: є кілька черг готових потоків із різним пріоритетом, при цьому потоки черги із нижчим пріоритетом виконуються, тільки коли всі черги верхнього рівня порожні.

Відмінності між двома алгоритмами полягають у тому, що:

~ потокам дозволено переходити з рівня на рівень (із черги в чергу);

~ потоки в одній черзі об'єднуються не за пріоритетом, а за довжиною інтервалу використання процесора, потоки із коротшим інтервалом перебувають у черзі з більшим пріоритетом.

Усередині всіх черг, крім найнижчої, використовують кругове планування (у найнижчій працює FIFO-алгоритм). Різні черги відповідають різній довжині кванта часу – що вищий пріоритет, то коротший квант (звичайно довжина кванта для сусідніх черг зменшується удвічі). Якщо потік вичерпав свій квант часу, він переміщається у хвіст черги із нижчим пріоритетом (і з довшим квантом).

У результаті потоки з коротшими інтервалами (наприклад, обмежені введенням-виведенням) залишаються з високим пріоритетом, а потоки з довшими інтервалами подовжують свій квант часу. Можна також автоматично переміщати потоки, які давно не отримували керування, із черги нижнього рівня на рівень вище.

Лотерейне планування. Лотерейне планування (lottery scheduling) – це останній алгоритм, який ми розглянемо. Він простий для розуміння і легкий у реалізації, разом з тим має великі можливості.

Ідея лотерейного планування полягає у тому, що:

~ потік отримує деяку кількість лотерейних квитків, кожен з яких дає право користуватися процесором упродовж часу T;

~ планувальник через проміжок часу T вибирає випадково один лотерейний квиток;

~ потік, «що виграв», дістає керування до наступного розіграшу.

Лотерейне планування дає змогу:

~ емулювати кругове планування, видавши кожному потокові однакову кількість квитків;

~ емулювати планування із пріоритетами, розподіляючи квитки відповідно до пріоритетів потоків;

~ емулювати SRTCF – давати коротким потокам більше квитків, ніж довгим (якщо потік отримав хоча б один квиток, він не голодуватиме);

~ забезпечити розподіл процесорного часу між потоками – дати кожному потокові кількість квитків, пропорційну до частки процесорного часу, який потрібно йому виділити (наприклад, якщо є три потоки, і треба, щоб потік A

займав 50 % процесора, а потоки B і C – по 25 %, можна дати потоку A два квитки, а потокам B і C – по одному);

~ динамічно міняти пріоритети, відбираючи і додаючи квитки по ходу.

Хоча більшість цих задач може бути розв'язана лотерейним плануванням лише приблизно, з деякою ймовірністю, на практиці отримують цілком задовільні результати. При цьому що довше працює система, то ближчими будуть результати до теоретичних значень (за законом великих чисел). Насправді лотерейне планування використовує той факт, що вся ідеологія планування значною мірою є евристичною, оскільки не можна точно передбачити характер поведінки потоків у системі.