МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА ЕЛЕКТРОННОЇ ТЕХНІКИ

КОНСПЕКТ ЛЕКЦІЙ

З ДИСЦИПЛІНИ **«ПРОЕКТУВАННЯ СИСТЕМ РЕАЛЬНОГО ЧАСУ »**

ДЛЯ СТУДЕНТІВ СПЕЦІАЛЬНОСТІ 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ
ДЕННОЇ ТА ЗАОЧНОЇ ФОРМИ НАВЧАННЯ

Луцьк
2023

УДК
К

Конспект лекцій з дисципліни «Проектування систем реального часу» для студентів спеціальності 123 Комп’ютерна інженерія усіх форм навчання [Електронний ресурс] / уклад. : С.О. Ковальов, В.В. Шамаєв. – Луцьк : ДонНТУ, 2023. – 85 с.

Конспект лекцій містить основні теоретичні положення, терміни і визначення з дисципліни «Проектування СРЧ» студентами бакалавріату ДонНТУ, що навчаються за спеціальністю 123 Комп’ютерна інженерія усіх форм навчання. Наведено основи теорії, етапи проектування, методи і алгоритми вирішення практичних задач в системах реального часу.

Конспект лекцій з дисципліни «Проектування систем реального часу» розроблено за участю О.Г. Шевченко. Автори щиро вдячні їй за співпрацю та надані матеріали.

Призначено для бакалаврів спеціальності 123 Комп’ютерна інженерія

Укладачі: С.О. Ковальов, доц., к.т.н., доц. каф. ЕТ.
В.В. Шамаєв, доц., к.т.н., доц. каф. ЕТ.

Рецензент: Н.О. Маслова, доц., к.т.н., доц. каф. ПМІ.

Відповідальний за випуск: О.В. Вовна, проф., д.т.н., зав. каф. ЕТ.

Затверджено навчально-методичним відділом ДонНТУ
Протокол № 4 від 31.01.2023 р.

Розглянуто на засіданні кафедри електронної техніки
Протокол № 7 від 19.01.2023 р.

Зміст:

1. Поняття систем реального часу і ОСРЧ

1.1 Визначення і особливості ОС реального часу

Операційні системи реального часу (**ОСРЧ**) або *Real-time operating system* (**RTOS**) – це тип спеціалізованої ОС, основним призначенням якої є надання необхідного та достатнього набору функцій для роботи систем реального часу та на конкретному апаратно-програмному комплексі. Під реальним часом в ОС розуміють її здатність забезпечити необхідний рівень сервісу в певний проміжок часу. Основне завдання таких систем – це своєчасність (англ. – timeliness) виконання обробки даних. Основною вимогою до ОСРЧ є забезпечення передбачуваності або детермінованості поведінки системи в найгірших зовнішніх умовах, що різко і помітно відрізняється від вимог до продуктивності та швидкодії універсальних ОС. ОСРЧ також повинна мати та забезпечувати передбачувану поведінку за всіма сценаріями системного завантаження (виконання потоків з одночасними перериваннями).

Стандарт POSIX 1003.1 дає таке визначення реального часу: «Реальний час в операційних системах - це здатність операційної системи забезпечити необхідний рівень сервісу в певний проміжок часу». Система повинна встигнути відреагувати на подію, подію на об'єкті, протягом часу, критичного для цієї події. Величина критичного часу для кожної події визначається об'єктом і самою подією, і, природно, може бути різною, але час реакції системи має бути передбачене (обчислено) при створенні системи. Відсутність реакції в передбачений час вважається помилкою для систем реального часу.

Як правило, вживання ОСРЧ пов'язане з апаратурою, об'єктом і подіями, що відбуваються на нім. В порівнянні з ОС спільного призначення це обумовлює корінні відзнаки в структурі системи, у функціях ядра і в побудові системи введення-виводу. І при цьому призначений для користувача інтерфейс ОСРЧ може бути схожий на інтерфейс ОС спільного призначення.

ОС спільного призначення, особливо розраховані на багато користувачів, орієнтовані на оптимальний розподіл ресурсів комп'ютера між користувачами і завданнями (системи розділення часу), В операційних системах реального часу подібне завдання відходить на другий план - все відступає перед головним завданням - встигнути зреагувати на події, що відбуваються на об'єкті. Інша відзнака - вживання операційної системи реального часу завжди

пов'язане з апаратурою, з об'єктом, з подіями, що відбуваються на об'єкті. Система реального часу, як апаратно-програмний комплекс, включає датчики, реєструючи події на об'єкті, модулі введення-виводу, що перетворюють показники датчиків в цифровий вигляд, придатний для обробки цих свідчень на комп'ютері, і, нарешті, комп'ютер з програмою, що реагує на події, що відбуваються на об'єкті.

Одна з корінних зовнішніх відзнак систем реального часу від систем спільного призначення - чітке розмежування систем розробки і систем виконання . Система виконання операційних системах реального часу - набір інструментів (ядро, драйвери, виконувані модулі), що забезпечують функціонування додатка реального часу. Більшість сучасних ведучих операційних систем реального часу підтримують цілий спектр апаратної архітектури, на якій працюють системи виконання (Intel, Motorola, RISC,MIPS, POWERPC, та інші). Це пояснюється тим, що набір апаратних засобів - частина комплексу реального часу і апаратура має бути також адекватна вирішуваному завданню, що тому ведуть операційні системи реального часу перекривають цілий ряд найбільш популярної архітектури, щоб задовольнити самим різним вимогам по частині апаратури. Система виконання операційних системах реального часу і комп'ютер, на якому вона виконується називають "цільовою" (target) системою. Система розробки - набір засобів, який забезпечує створення і відладку додатка реального часу. Самі системи розробки працюють, як правило, в популярних і поширених ОС, таких, як UNIX і Windows. Функціонально засоби розробки операційних систем реального часу відрізняються від звичних систем розробки, таких, наприклад, як Developers Studio, TaskBuilder, оскільки часто вони містять засоби видаленої відладки, засоби профілізації, засоби емуляції цільового процесора, спеціальні засоби відладки взаємодіючих завдань, а інколи і засоби моделювання

Розрізняють два типи ОСРЧ: жорсткого або м'якого реального часу. Першого типа не допускають жодних (майже жодних) затримок реакції системи ні за яких умов, оскільки результати можуть виявитися даремними в разі запізнення або може статися катастрофа в разі затримки реакції (системи управління навігацією, або управління технологічними процесами в промисловості). Системи другого типа характеризуються тим, що затримка реакції не критична, хоча і може привести до збільшення вартості результатів, зниженню продуктивності або якості системи в цілому. Наприклад - робота ме-

режі. Якщо система не встигла обробити черговий прийнятий пакет, це приведе до тайм-ауту на передавальній стороні і повторній посилці. Дані при цьому не втрачаються, але продуктивність мережі знижується. Основна відзнака між системами жорсткого і м'якого реального часу можна виразити так: система жорсткого реального часу Ніколи не запізниться з реакцією на подію, система м'якого реального часу - Не повинна спізнюватися з реакцією на подію.

Більшість ОСРЧ є одночасно вбудовуваними. Поняття вбудовувана операційна система має на увазі можливість її використання у вбудовуваній комп'ютерній системі (Embedded System). Під вбудовуваною комп'ютерною системою зазвичай мається на увазі спеціалізована система, повністю вбудована в деякий пристрій і вирішальна цілком конкретний круг завдань. Типовими сферами вживання вбудовуваних систем є медичне устаткування, побутова електроніка, бортові системи управління. Найважливішою вимогою до вбудовуваної системи в цілому є компактність, а до операційних систем для них - можливість роботи при відносно невеликих ресурсах

Перші комерційні ОСРЧ набули широкого поширення одночасно з появою персональних комп'ютерів на початку 80-х років XX ст. В даний час із понад сотні ОСРЧ, розроблених в різний час, широкого поширення набули лише близько двадцяти. Для порівняння властивостей ОСРЧ різних виробників можна виділити наступні категорії:

- властивості ядра (архітектура, підтримка безлічі процесів і процесорів, стійкість до відмов);
- диспетчеризація (алгоритми планерування, перемикання контексту завдань, динамічна зміна пріоритетів);
- модель процес/нитка/задача (число рівнів пріоритетів, властивості завдання, максимальне число завдань);
- управління пам'яттю (мінімальний і максимальний об'єм оперативною для завдання, підтримка захисту пам'яті);
- управління перериваннями і виключеннями;
- інтерфейс прикладного програмування;
- процес розробки (підтримувані компілятори і мови програмування);
- комерційна інформація (вартість, вартість технічної підтримки).

Сучасні багатоядерні процесори вимагають від виробників ОСРЧ підтримки різної архітектури багатопроцесорної обробки, що у свою чергу спричиняє за собою повсюдне використання спеціалізованих бібліотек і систем паралельного програмування. Залежно від вживаної методології, широкого поширення набули три напрями у взаємодії паралельних завдань. Перше будується на основі концепції обміну повідомленнями, друге - на використанні пам'яті, що розділяється, третє - на основі стандарту POSIX – об’єднує обидва перших.

Найбільш відомим представником першої групи є специфікація MPI (Message Passing Interface) для мов програмування С і Фортран. Перший варіант специфікації MPI був розроблений ще в 1994 р. Специфікація MPI включає близько 200 функцій і реалізована для безлічі компіляторів і операційних систем. Однією з найбільш поширених реалізацій MPI є бібліотека MPICH. Крім того, на ринку представлено декілька комерційних реалізацій MPI, наприклад, MPI/Pro виробництва компанії Verari Systems Software для різних операційних систем, включаючи Windows, Linux, Mac OS X, LYNXOS. Представником другої групи систем паралельного програмування є специфікація OPENMP (Open specifications for Multi-Processing), перша версія якої з'явилася в 1997 р. і призначалася для мови програмування Фортран. У витоків ОРЕПМР стояли компанії IBM, Intel, Sun і Hewlett-Packard. У 1998 р. з'явилися варіанти OPENMP для мов С/С++. Третім стандартом,який застосовується для реалізації паралельних обчислень, є POSIX. В рамках POSIX можливо реалізувати паралельні обчислення як на основі обміну повідомленнями аналогічно MPI, так і на основі пам'яті, що розділяється, як в ОРЕПМР. Природно, в POSIX можлива і будь-яка комбінація цих методів.

Вимоги, що пред'являються до розробки ОСРЧ:

- вимога «живучості» . ОСРЧ повинні забезпечувати високу міру "живучості" системи так, щоб при відмові якої-небудь частини ПЗ інша частина ПЗ продовжувала нормально функціонувати. ОСРЧ повинна гарантувати відсутність спільної відмови системи;
- вимога за якістю ПЗ. ОСРЧ повинні задовольняти жорстким вимогам за якістю ПЗ, що має на увазі відповідність різним галузевим, національним і міжнародним стандартам. ПЗ повинна мати доведена якість і у багатьох випадках повинно бути сертифіковане уповноваженими організаціями;

- вимога по надійності: вірогідність збоїв у ПЗ має бути дуже маленька;
- вимоги по безпеці і секретності даних: у системі мають бути передбачені засоби захисту найбільш важливої інформації.

По критерію зумовленості тимчасових характеристик ОСРЧ можна розділити на системні, спеціалізовані і універсальні.

Спеціалізована ОСРЧ – це система, де конкретні тимчасові вимоги апріорі визначені. Така система на етапі проектування враховує задані тимчасові обмеження. Універсальна ОСРЧ повинна уміти виконувати будь-які (заздалегідь невизначені) завдання реального часу. Розробка таких систем є найскладнішим завданням, хоча зазвичай вимоги, що пред'являються до таких систем м'якше, ніж вимоги до спеціалізованих систем

1.2 Огляд ОСРЧ

Перші комерційні ОСРЧ набули широкого поширення одночасно з появою персональних комп'ютерів на початку 80-х років XX століття.

Вони в основному використовуються для вирішення завдань автоматизації. Це компактні системи, засновані на мікроядерній архітектурі, такі як:

- VxWorks (рік випуску – 1981, остання версія 6.9 – лютий 2011 р.) - що розробляється компанією Wind River Systems (США), орієнтована на використання у вбудовуваних комп'ютерах, що працюють в системах жорсткого реального часу. Використовується:
  - Phoenix Mars Lander — апарат НАСА для вивчення Марса;
  - зонди Spirit, Opportunity і Curiosity, а також апарат Mars Reconnaissance Orbiter використовують VxWorks на платформі POWER. Система використовується і в інших космічних місіях, наприклад Deep Impact;
  - використовують в новітніх авіалайнерах Boeing 787 і Boeing 747-8;
  - комунікаційне устаткування багатьох компаній (наприклад, Nortel, 3COM, Alcatel і ін.);
  - деякі PostScript-принтери;
  - медичне устаткування компанії Siemens AG (зокрема, магнітно-резонансні томографи);
  - системи зберігання ETERNUS DX компанії Fujitsu;
  - останні системи інтерфейсів BMW iDrive;

    - система управління робототехнічними комплексами компанії KUKA;
    - система управління роботами компанії ABB.
- OS-9 (рік випуску – 1979, остання версія 5.0 – 2010 р.) – сімейство багатозадачних, розрахованих на багато користувачів ОСРЧ, розроблених Microware Systems Corporation в 1980х. Є UNIX-подобными. Спочатку працювали на процесорах Motorola 6809. Існують версії для Motorola 68k, POWERPC, x86 і ін. Використовується для інтерактивних і вбудовуваних систем. В наші дні OS-9 належить компанії RadiSys Corporation, розташованій в штаті Орегон (США).
- QNX (рік випуску – 1982), розробник QNX Software Systems, операційна система реального часу, призначена переважно для вбудовуваних систем. Вважається однією з кращих реалізацій концепції мікроядерних операційних систем.

  Приклади вживання QNX за кордоном:
    - Найбільш яскравим прикладом вживання QNX є робота з кредитними картками VISA у всіх регіональних офісах Північної Америки;
    - Управління дорожнім рухом. У канадському місті Оттава-Карлтон (англ. Regional Municipality of Ottawa-Carleton) на базі QNX розроблена система управління рухом міського муніципального транспорту;
    - Управління ядерним реактором. Одним з відділень канадської компанії Atomic Energy of Canada Ltd. розроблена система управління ядерним реактором, яка називається «Розподілена система управління з відкритою архітектурою» (Open Architecture Distributed Control System);
    - Окрім вживання QNX в області управління, вона також успішно використовується і для наукових досліджень: моделювання процесів, відстежування ходу експериментів;
    - Cisco Systems використовує оптимізовану версію мікроядра QNX Neutrino в програмному забезпеченні IOS XR. Програмний пакет IOS XR призначений для управління комутаторами Cisco CRS-1, забезпечує безперервний режим роботи і підтримує розвинені

функції управління терабітними комутаторами з розподіленою архітектурою;

- рішення на базі QNX ліцензовані для використання на більш, ніж 10 мільйонах одиниць техніки від практично всіх провідних виробників автомобілів, включаючи BMW, Chrysler, Daimler, Fiat, Ford, General Motors, Honda, Hyundai, Mazda, Mitsubishi, Nissan, Saab, SsangYong, Toyota і Volkswagen. Зокрема, такі автомобілі випускаються під марками Acura, Alfa Romeo, Audi, Buick, Cadillac, Chevrolet, Dodge, Honda, Hummer, Infiniti, Jeep, Lancia, Mini, Mercedes, Opel, Pontiac, Saturn і іншими.

Приклади вживання QNX в країнах тих, що колись були у складі СНД:

- система «Сиріус-qnx», призначена для оперативного диспетчерського контролю і управління технологічним процесом перекачування нафти. Спільна протяжність системи нафтопроводів в одноніткового виконання складає 3696 км.;
- обчислювальні комплекси для вирішення завдань протиаварійної автоматики і протиаварійного управління різних рівнів в енергетичних системах виробництва ЗАТ «Інститут автоматизації енергетичних систем» (м. Новосибірськ);
- система управління машинами термічного різання металу (МТР), виробництва ВАТ «Парасолька», м. Одеса;
- пульт космонавтів «НЕПТУН-МЕ» транспортного корабля «СОЮЗ-ТМА», НІЇАО, р. Жуковський;
- багатоканальні, що інформаційно-управляють комплекси вогневих стендових випробувань вузлів ракетних двигунів, виконані на QNX 6 і програмному продукті Octavo;
- система автосупроводу електропоїздів «Рух» метрополітену;
- система управління режимами і контролю роботи головних електроприводів стану «Слябінг 1150» на ЗАО «Запоріжсталь» під управлінням QNX4.25, виконана НВО «Донікс» м. Донецьк (Україна).

- LYNXOS (рік випуску – 1988), розроблена для вбудовуваних систем. LYNXOS використовується переважно в авіації, системах управління промисловими процесами і в області телекомунікацій.

- Крім того, кожен з провідних фірм-виробників, що випускають промислові комп'ютери, обов'язково має сьогодні версію своєї операційної системи для роботи в реальному масштабі часу. Для компанії Hewlett-Packard (HP) - це HP RT, для компанії SGI - це ОС REACT, а для систем фірми Motorola - це ціле сімейство різних ОС РВ. Серед них можна назвати LYNXOS компанії Lynx Real-Time Systems Inc. або багатозадачну систему OS-9 фірми Microware Systems Corporation.

Загалом, на сьогоднішній день існує більше 100 комерційних ОСРЧ. Є безліч безкоштовних (або умовно безкоштовних) ОСРЧ і систем, що мають статус дослідницьких або університетських проектів. Ось неповний перелік популярних ОСРЧ: QNX Neutrino, RTOS, RTEMS, ChorusOS, RTX для Windows NT, INtime, TinyOS, OSEK / VDX, Contiki, pSOS, INTEGRITY, LynxOS, Microware OS-9, GRACE-OS, C EXECUTIVE, CMX-RTX, CMX-TINY +, Inferno.

1.3 Основні характеристики ОСРЧ та ОС загального призначення.

Основні порівняльні характеристики ОСРЧ та ОС загального призначення наведено в табл. 1

Таблиця 1.

Порівняльні характеристики ОСРЧ та ОС загального призначення [10]

Завдання ОС	ОС спеціального призначення	ОС загального призначення
Основне завдання	Моментальна реакція на обробку запитів та завдань	Розподілення ресурсів між завданнями та користувачами
Мета (орієнтація)	Обробка зовнішніх запитів	Обробка згідно з діями користувачів
Позиціювання	Створення реального апаратно-програмного середовища реального часу	Створення апаратно-програмного середовища для конкретного користувача
Призначення	Обробка інформації в режимі реального часу	Обробка інформації в заданому режимі на конкретному обладнанні

У своєму розвитку ОСРЧ розроблялися та будувалися на основі монолітної, рівневої (шарової) та архітектури «клієнт-сервер». Монолітна архітектура ОСРЧ визначається як набір модулів, взаємодіючих між собою всередині ядра системи, які надають прикладному ПЗ вхідні інтерфейси для звернень до апаратури (CPU, RAM). Основний недолік цього принципу побудови ОС полягає в досить заниженій передбачуваності її поведінки, викликаної складною взаємодією модулів між собою. Структурну схему монолітної архітектури ОСРЧ наведена на рис. 1.1.

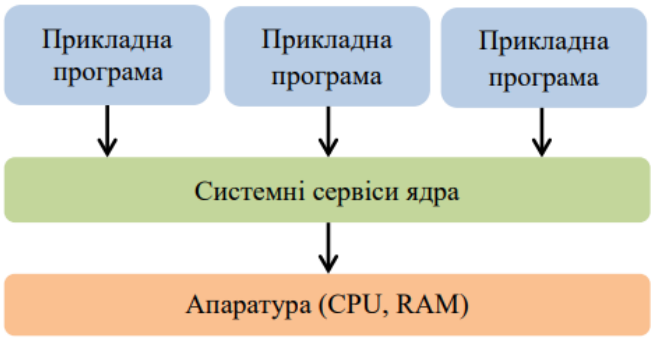

Рис. 1.1 Структурна схема монолітної архітектури ОСРЧ

В рівневій (шарової) архітектури прикладне ПЗ має можливість отримати доступ до апаратури (CPU, RAM) не тільки через ядро системи та її сервіси, а й безпосередньо. У порівнянні з монолітною подібна архітектура забезпечує значно більший ступінь передбачуваності реакцій системи, а також дозволяє здійснювати швидкий доступ прикладних програм до апаратури (CPU, RAM). Головним недоліком таких систем є відсутність багатозадачності. Структурна схема шарової архітектури ОСРЧ наведена на рис. 1.2

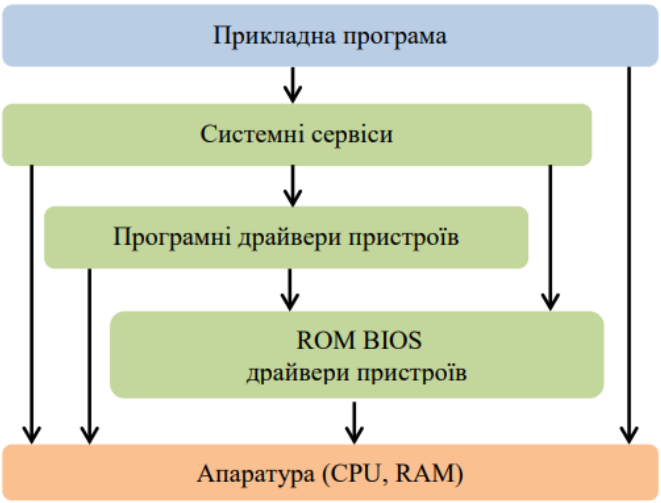

Рис. 2. Структурна схема рівневої (шарової) архітектури ОСРЧ

Згідно з підходами, передбаченими архітектурою «клієнт-сервер», основний її принцип полягає у винесенні сервісів ОС у вигляді серверів на рівень користувача та виконанні мікроядром функцій диспетчера повідомлень між клієнтськими програмами користувача і серверами – системними сервісами. Перевагами саме такої архітектури є: підвищена надійність (кожен сервіс є, по суті, самостійним процесом і його простіше налагодити та відстежити помилки), покращена масштабованість (непотрібні сервіси можуть бути виключені із системи без завдання суттєвої шкоди її працездатності), підвищена відмовостійкість тощо. Структурну схему ОСРЧ архітектури «клієнт-сервер» показано на рис. 1.3.

Рис. 1.3. Структурна схема ОСРЧ архітектура «клієнт-сервер»

1.4 Системи керування у реальному часі (RCS).

Система керування в реальному часі (RCS) є еталонною моделлю архітектури, відповідною для багатьох програмно-інтенсивних проблемних областей управління в реальному часі.

RCS - це архітектура еталонної моделі, яка визначає типи функцій, необхідних в інтелектуальній системі керування в реальному часі, і те, як ці функції пов'язані один з одним.

RCS - це не системний проект, і це не специфікація того, як реалізувати конкретні системи. RCS наказує ієрархічну модель управління, засновану на наборі добре обґрунтованих інженерних принципів для організації системної комплексності. Всі вузли керування на всіх рівнях використовують загальну модель вузлів.[1]

Крім того, RCS надає комплексну методологію проектування, інтеграції та тестування систем управління. Архітектори ітеративно розбивають системні завдання та інформацію на більш тонкі, кінцеві підмножини, які є керованими й ефективними. RCS фокусується на інтелектуальному управлінні, яке адаптується до невизначених і неструктурованих робочих середовищ. Основні проблеми - це сприйняття, знання, витрати, навчання, планування і виконання.[1]

Архітектура еталонної моделі - це канонічна форма, а не специфікація проектування системи. Архітектура еталонної моделі RCS поєднує в собі планування і керування рухом у реальному часі з високорівневим плануванням завдань, розв'язання задач, моделюванням світу, рекурсивною оцінкою стану, тактильною й візуальною обробкою зображень і акустичним сигнатурним аналізом. Фактично, еволюція концепції RCS була обумовлена прагненням включити кращі властивості та можливості більшості, якщо не всіх, інтелектуальних систем управління, відомих в даний час в літературі, від subsumption до SOAR, від blackboards до об'єктно-орієнтованого програмування.[2].

RCS (real-time control system) – це інтелектуальна агентна архітектура, призначена для забезпечення будь-якого рівня інтелектуальної поведінки, включаючи людський рівень продуктивності. RCS був натхненний теоретичною моделлю мозочка, частини мозку, відповідальної за дрібну моторну координацію і контроль свідомих рухів. Спочатку він був розроблений для сенсорно-інтерактивного цілеспрямованого управління лабораторними маніпуляторами. За три десятиліття він перетворився в архітектуру управління у реальному часі для інтелектуальних верстатів, систем автоматизації виробництва та інтелектуальних автономних транспортних засобів.[3]

RCS застосовується до багатьох проблемних областей, включаючи приклади виробництва та приклади систем транспортних засобів. Системи, засновані на архітектурі RCS, були розроблені й впроваджені в різній мірі для широкого спектра застосувань, які включають завантаження і вивантаження деталей і інструментів на верстатах, управління робочими станціями обробки, виконання роботизованого зняття задирок та фаски, а також управління телероботами космічних станцій, декількома автономними підводними апаратами, безпілотними наземними транспортними засобами, системами ав-

томатизації видобутку вугілля, системами обробки пошти й системами автоматизації експлуатації підводних човнів.[2].

1.4.1 Парадигми управління RCS.

RCS розвивався через безліч версій протягом ряду років, оскільки розуміння складності й складності інтелектуальної поведінки збільшилася. Перша реалізація була розроблена для сенсорно-інтерактивної робототехніки Барбарою в середині 1970-х років.[4]

В **RCS-1** акцент робився на об'єднанні команд з сенсорним зворотним зв'язком, щоб обчислити правильну реакцію на кожну комбінацію цілей і станів. Додаток мав керувати роботизованою рукою зі структурованою системою світлового зору в завданнях візуального переслідування.

На розвиток RCS-1 великий вплив зробили біологічні моделі, такі як мозочковая модель арифметичного комп'ютера (CMAC),[6] мозочок.[2] На кожному рівні команда введення ефективно вибирає поведінку, яка управляється зворотним зв'язком в стилі стимул-реакція. CMAC, таким чином, став еталонним зразком будівельного блоку RCS-1, як показано на рисунку 1.4.

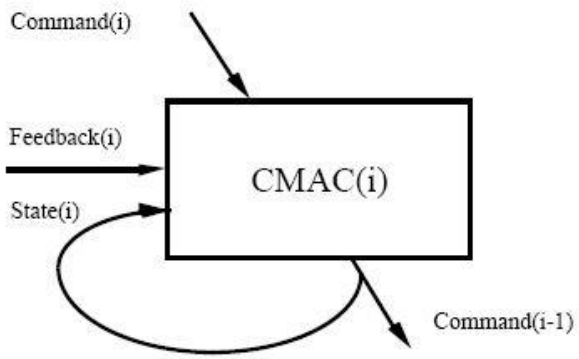

Рисунок 1.4 Парадигма управління RCS-1.

Ієрархія цих будівельних блоків використовувалася для реалізації ієрархії поведінки, що спостерігається Тінбергеном[7] та іншими. RCS-1 багато в чому схожий з архітектурою підзарядки Брукса,[8] за винятком того, що RCS вибирає поведінку до факту через цілі, виражені в командах, а не після факту через субсумпсії.[2]

Наступне покоління, **RCS-2**, було розроблено Барбарою, Фіцджеральдом, Кентом і іншими для управління виробництвом в автоматизованому виробничому дослідному центрі NIST (AMRF) на початку 1980-х рр.[9-11] основний будівельний блок RCS-2 показаний на малюнку 1.5.

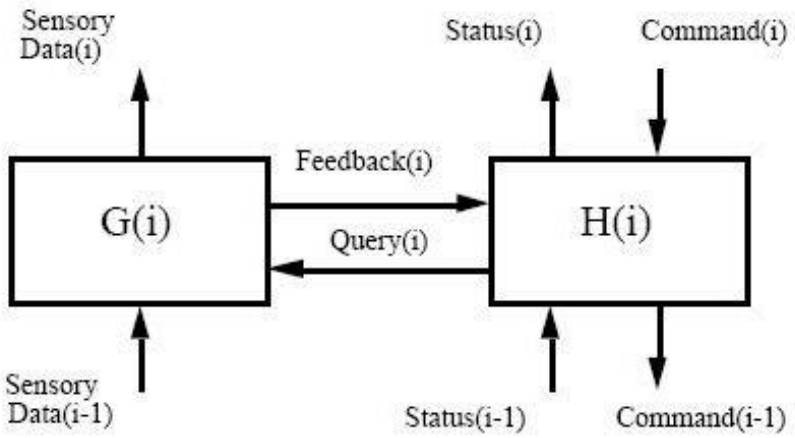

Рисунок 1.5 Парадигма управління RCS-2.

Функція H залишалася виконавцем таблиці станів кінцевого автомата. Новою особливістю CS2 стало включення функції G, що складається з ряду алгоритмів обробки сенсорної, включаючи алгоритми структурованого аналізу світла і великих двійкових об'єктів. RCS-2 був використаний для визначення восьмирівневої ієрархії, що складається з сервоприводу, перетворення координат, E-Move, Завдання, Робочої станції, Стільникової та Об'єктної рівнів управління. Тільки перші шість рівнів були фактично побудовані. Дві з робочих станцій AMRF повністю реалізували п'ять рівнів RCS-2. Система управління армійським польовим вантажно-розвантажувальним роботом (ФМ)[12] також була реалізована в РКС-2, як і проект армійського напівавтономного наземного транспортного засобу (TMAP).[2]

RCS-3 був розроблений для проекту декількох автономних підводних транспортних засобів (MAUV) NBS / DARPA[13] і був адаптований для стандартної моделі NASA / NBS-архітектури системи управління телероботам (NASREM), розробленої для космічної станції Flight Telerobotic Servicer [14]. Основний будівельний блок RCS-3 показаний на малюнку 1.6.

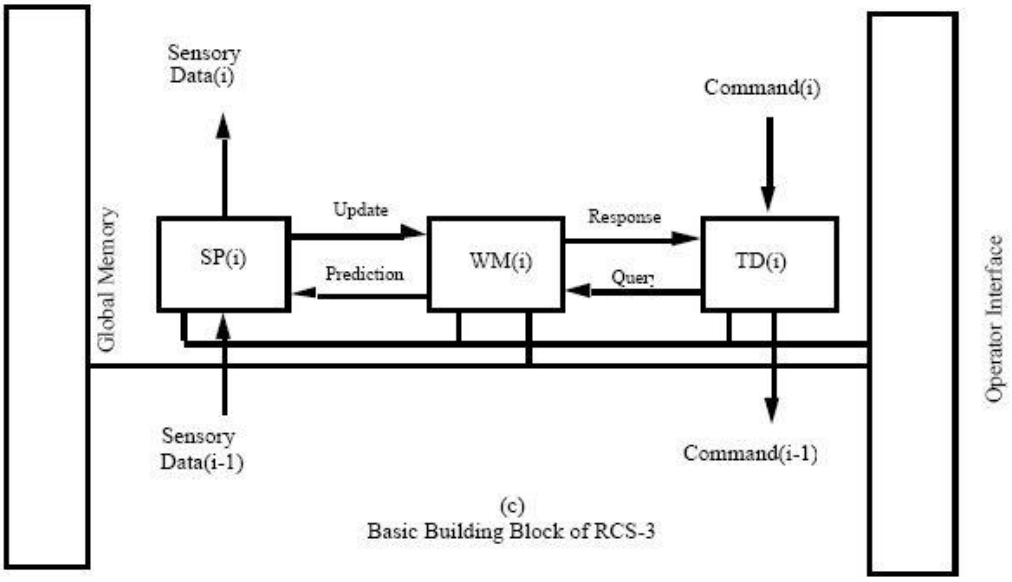

Рисунок 1.6 Парадигма управління RCS-3.

Основними новими функціями, представленими в RCS-3, є модель світу і інтерфейс оператора. Включення моделі світу забезпечує основу для планування завдань і сенсорної обробки на основі моделей. Це призвело до уточнення модулів декомпозиції задач (TD) таким чином, що кожному з них було призначено завдання, а планувальником і виконавцю для кожної з підсистем було призначене завдання. Це приблизно відповідає трирівневій ієрархії управління Сарідіса.[15][2]

RCS-4 розробляється з 1990-х років підрозділом NIST Robot Systems. Основний будівельний блок показаний на малюнку 1.7. Головною новою особливістю RCS-4 є явне представлення системи оцінювальних суджень (VJ). Модулі VJ забезпечують системі управління RCS-4 тип функцій, що надаються біологічному мозку лімбічною системою. Модулі VJ містять процеси, які обчислюють витрати, вигоди й ризики планованих дій, а також визначають цінність об’єктів, матеріалів, території, ситуацій, подій і результатів. Змінні стану значення визначають, які цілі є важливими і які об'єкти або регіони повинні бути охоплені, атаковані, захищені, підтримані або яким-небудь іншим чином залучені. Ціннісні судження, або оцінювальні функції, є невід'ємною частиною будь-якої форми планування або навчання. Застосування оцінювальних суджень до інтелектуальних систем управління розглядав Джордж П'ю.[16] Структура і функція модулів VJ розроблені більш повно, ніж в Albus (1991).[2][17]

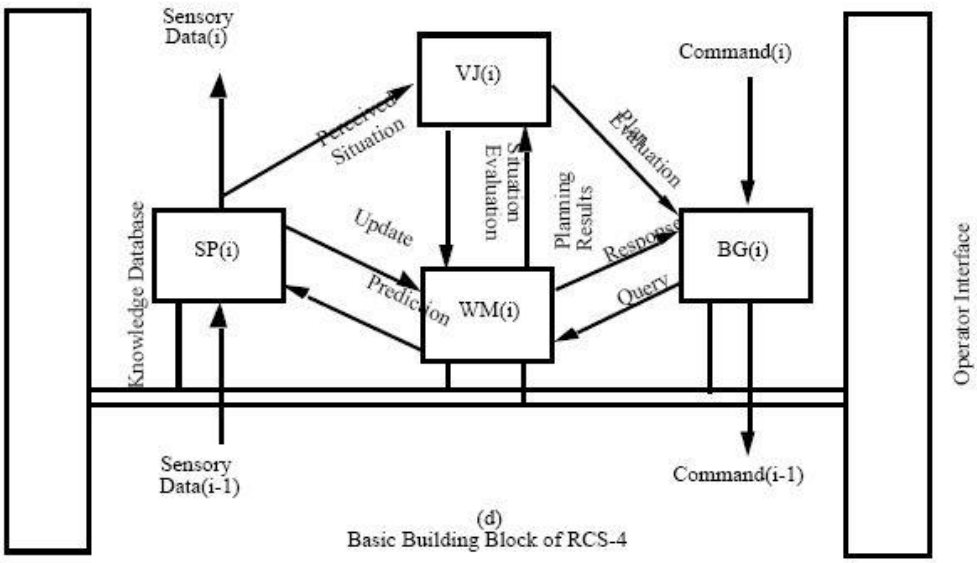

Рисунок 1.7 Парадигма управління RCS-4.

RCS-4 також використовує термін генерація поведінки (BG) замість терміну декомпозиція завдання 5 (TD), як у RCS-3. Мета цієї зміни - підкреслити ступінь автономності прийняття рішень. RCS-4 призначений для вирішення високо автономних завдань в неструктурованих середовищах, де зв'язок з високою пропускною здатністю неможливий, тобто таких, як безпілотні літальні апарати, які працюють на полі бою, глибоко під водою або на далеких планетах. Ці програми вимагають автономних оцінювальних суджень і складного сприйняття можливостей в реальному часі. RCS-3 буде використовуватись для менш складних додатків, таких як виробництво, будівництво або телероботи для ближнього космосу або дрібних підводних операцій, де навколишнє середовище більш структурована, а пропускна спроможність зв'язку з людським інтерфейсом менш обмежена. У цих додатках ціннісні судження часто неявно представляються в процесах планування завдань або у вхідних даних людини-оператора.[2]

1.4.2 Методика RCS проектування системи управління рухом.

На малюнку1.8 наведено приклад методики RCS по проектуванню системи управління автономним рухом по дорозі в умовах повсякденного руху, яка складається з шести етапів.[18]

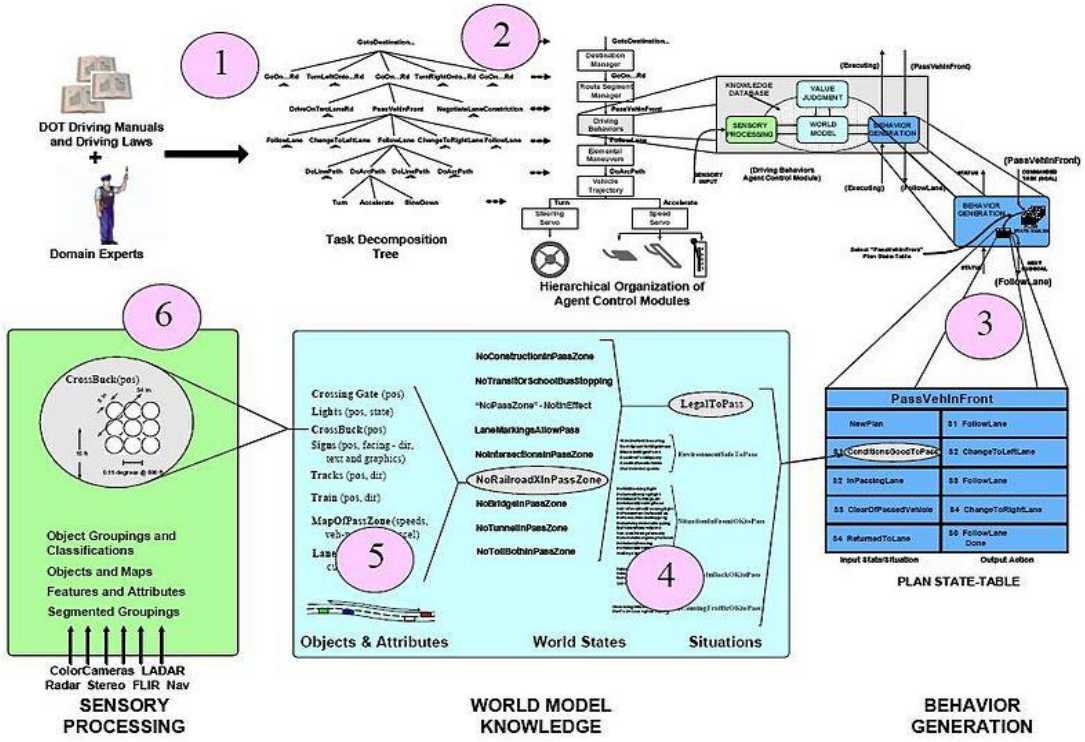

Рисунок 1.8. Шість кроків методології RCS для здобуття та представлення знань

- Крок 1 складається з інтенсивного аналізу знань предметної області з навчальних посібників та експертів з предметної області. Сценарії розробляються і аналізуються для кожного завдання і підзадачі. Результатом цього кроку є структурування процедурних знань в дерево декомпозиції задач з простими завданнями в кожному ешелоні. У кожному ешелоні визначено словник команд (дієслова дій з цільовими станами, параметрами і обмеженнями), що викликає поведінку завдання в кожному ешелоні.[18]
- Крок 2 визначає ієрархічну структуру організаційних підрозділів, які виконуватимуть команди, визначені на кроці 1. Для кожного підрозділу визначено його обов'язки та відповідальність у відповідь на кожну команду. Це аналогічно створенню структури розбивки робіт для проекту розвитку або визначенню організаційної структури для комерційної або військової операції.
- Крок 3 визначає обробку, яка запускається в кожному блоці при отриманні вхідної команди. Для кожної вхідної команди визначається граф станів (або автомат зі змінним або розширеним кінцевим станом), який забезпечує план (або процедуру складання плану) для виконання постав-

леного завдання. Вхідна команда вибирає (або змушує генеруватися) відповідну таблицю станів, виконання якої генерує серію вихідних команд для одиниць в наступному нижньому ешелоні. Бібліотека таблиць станів містить набір залежних від станів процедурних правил, які визначають умови розгалуження задачі й задають параметри переходу стану і команди виведення. Результатом кроку 3 є те, що кожна організаційна одиниця має для кожної вхідної команди таблицю станів впорядкованих виробничих правил, кожна з яких підходить для виконання розширеним кінцевим автоматом (FSA). Послідовність вихідних підкоманд, необхідних для виконання вхідної команди, генерується ситуаціями (тобто умовами розгалуження), які змушують FSA переходити від однієї вихідної підкоманди до наступної.[18]

- На кроці 4 кожна з ситуацій, визначених на Кроці 3, аналізується, щоб виявити їх залежності від станів світу і завдання. Цей крок визначає детальні відносини між сутностями, подіями й станами світу, які призводять до виникнення конкретної ситуації.[18]
- На кроці 5 ми ідентифікуємо і називаємо всі об'єкти і сутності разом з їх конкретними ознаками та атрибутами, які мають відношення до виявлення вищевказаних станів і ситуацій у світі.
- На етапі 6 ми використовуємо контекст конкретних дій задачі для встановлення відстаней і, отже, дозволів, при яких відповідні об'єкти й сутності повинні бути виміряні і розпізнані компонентом сенсорної обробки. Це встановлює набір вимог і / або специфікацій для сенсорної системи для підтримки кожної дії підзадачі.

2. Системи управління технологічними процесами

У загальному випадку під СРЧ розуміється така система, в якій правильність її функціонування залежить не лише від логічної коректності обчислень, але і від часу, за який ці обчислення проводяться.

Тобто для подій, що відбуваються в такій системі, то, коли ці події відбуваються, так само поважно, як логічна коректність самих подій.

Поняття «Реальний час» відноситься до системи в цілому або режиму роботи, в якому обчислення проводяться протягом часу, визначуваного зов-

нішнім процесом, з метою управління або моніторингу зовнішнього процесу за результатами цих обчислень.

Звідси слідує головна властивість систем реального часу: передбачуваність або детермінованість. Лише завдяки цій властивості розробник може гарантувати функціональність і коректність спроектованої системи. При цьому власне швидкість реакції системи важлива лише відносно швидкості протікання зовнішніх процесів, за якими СРЧ повинна стежити, або якими повинна управляти.

З приведених визначень виходить, що СРЧ покликані вирішувати завдання, в яких важливі не лише правильність рішення, але і терміни, в які ці рішення приймаються. У зарубіжній літературі термін, в межах якого має бути прийняте рішення називається «deadline», що може бути переведене як критичний термін обслуговування.

СРЧ в більшості випадків вирішують комбінацію завдань жорсткого і м'якого реального часу, а також завдань, що не мають критичного терміну обслуговування. Інколи завдання може переходити із статусу м'якого реального часу при пропуску деякого терміну обслуговування в статус завдання жорсткого реального часу з призначенням критичного терміну обслуговування.

Система реального часу, як апаратно-програмний комплекс, включає датчики, реєструючи події на об'єкті, модулі введення-виводу, що перетворюють показники датчиків в цифровий вигляд, придатний для обробки цих свідчень на комп'ютері, і, нарешті, комп'ютер з програмою, що реагує на події, що відбуваються на об'єкті.

Загалом у складі СРЧ виділяють три компоненти: прикладне програмне забезпечення, операційна система реального часу (ОСРЧ) і апаратне забезпечення. При розробці СРЧ необхідний ретельний аналіз відповідності характеристик цих трьох компонент вимогам зовнішнього об'єкту, для управління або моніторингу, яким ця СРЧ призначена. Проведення такого аналізу вимагає, щоб тимчасові характеристики всіх компонент системи були добре прогнозованими.

2.1 Класифікація систем управління

Тип системи залежить від того, як приймається рішення по управлінню. Якщо в процесі ухвалення рішення бере участь людина, то така система на-

зивається автоматизованою (АСУ); якщо людина виключена з процесу ухвалення рішення, то це – автоматична система управління (САУ). По фізичній природі об'єкту управління розрізняють:

- системи управління технологічними процесами, де об'єктом управління є фізичний об'єкт;
- системи управління економіко-організаційного типа, в яких об'єктом управління є господарська діяльність.

Останнім часом створюються інтегровані системи управління, в яких присутні об'єкти обох типів.

Іншою ознакою, по якій класифікуються системи, є час ухвалення рішення. У системах управління він залежить від динаміки об'єкту, яка може бути різною: від мілісекунд до годин і днів. Системи управління технологічними процесами завжди функціонують в реальному масштабі часу. Час в таких системах є одним з чинників, що визначають ефективність системи.

Залежно від функцій, що виконуються системою управління, розрізняють види автоматизації: управління, регулювання, контроль, захист і блокування.

Управління є ухваленням рішень і видачею на об'єкт управління сукупності дій, що виконуються на підставі інформації про хід технологічного процесу, з метою підтримки необхідного технологічного режиму або поліпшення його параметрів. Зазвичай передбачається, що управління має на увазі пошук і реалізацію оптимального (по певному критерію) рішення.

Регулювання є ухваленням рішень і видачею на об'єкт управління сукупності дій, що виконуються на підставі інформації про хід технологічного процесу, з метою підтримки параметрів виробничих процесів постійними або такими, що змінюються по заданому закону.

Контроль – це спостереження за параметрами об'єкту управління з метою визначення його стану. Параметри, що підлягають контролю, залежно від їх фізичної природи різні (температура, тиск, витрата палива, число зворотів, сила струму
і тому подібне). Контроль може бути місцевим і дистанційним. Місцевий контроль дає можливість спостерігати за поляганням параметра безпосередньо в контрольованій крапці. При дистанційному контролі за достатком параметрів можна стежити на відстані від контрольованої крапки.

Захист – це комплекс засобів і заходів щодо оберігання агрегатів і установок при порушеннях технологічних режимів.

Блокування це комплекс засобів і заходів, що оберігають ділянку установки або агрегат від неправильних операцій унаслідок неуважності оператора або помилкових команд. Розрізняють дві групи блокування: що заборонено-вирішує і аварійну. Що заборонено-вирішують блокуванням запобігають неправильні включення і відключення механізмів, а також порушення встановленою технологічними вимогами черговості пуску і зупинки різних механізмів. Аварійні блокування призначені для автоматичного послідовного відключення(включення) механізмів відповідно до режиму роботи агрегату, що піддався аварійному відключенню. Блокувальні пристрої застосовують також у випадках, коли потрібно виключити одночасну подачу команд протилежного знаку на один і той же регулюючий орган з різних місць або від автоматичного і дистанційного управління, коли обидва види управління діють паралельно.

Основними складовими автоматизованих систем реального часу є: комплекс технічних засобів, програмне і інформаційне забезпечення.

Для ухвалення рішення по управлінню потрібно знати про стан об'єкту управління і зміни стану зовнішньої середи. Ця інформація вводиться в систему за допомогою вимірювальних засобів, які є джерелами інформації для засобів переробки інформації. Ці засоби реалізують алгоритм ухвалення рішення і формують дії, що управляють, які передаються на виконавчі засоби, безпосередньо пов'язані із змінними параметрами об'єкту управління.

Для того, щоб людина брала участь в процесі ухвалення рішення, йому також потрібно дати можливість стежити за достатком об'єкту і зовнішньої середи. Для цього він забезпечується пристроями відображення інформації, які призначені для перетворення інформації в зручну для сприйняття форму. Людина може втручатися в управління, впливаючи на виконавчі засоби і на засоби переробки

інформації за допомогою засобів, що управляють. Оскільки частини системи можуть знаходитися на значних відстанях один від одного, частина зв'язків є засобами передачі даних спеціальні технічні пристрої, призначені для цих цілей.

Як засоби переробки інформації зазвичай застосовується вузол на основі обчислювальної машини. Тип і структура цього вузла може бути різним

– від простого контролера до кластера, проте у будь-якому випадку його вживання пов'язане з наступними особливостями:

- він повинен стежити за безліччю паралельно протікаючих процесів;
- він повинен обробляти запити, що поступають в довільні моменти часу;
- допустимий час ухвалення рішення звичайний суміжно з часом реалізації алгоритму вироблення такого рішення;
- він повинен задовольняти підвищеним вимогам по надійності і достовірності інформації;
- склад завдань, які вирішує цей вузол, заздалегідь відомий і програмне забезпечення для їх вирішення відлагоджене.

Ці особливості враховуються при побудові програмного забезпечення. Воно включає операційну систему, програми вирішення функціональних завдань, програми контролю і забезпечення стійкості обчислювального процесу. Головне призначення операційної системи – забезпечення паралелізму і обробки заявок, що поступають у випадкові моменти часу. Програми вирішення функціональних завдань виконують основне цільове призначення системи – управління об'єктом. Третя група програм контролює роботу системи.

Інформаційне забезпечення містить дані, необхідні для управління системою.

3. Кількісні критерії оцінки параметрів СРЧ.

Кількісні параметри, що визначають властивості СРЧ, базуються на значеннях інтервалів часу, закладених в основу проектування апаратного і системного ПО системи. Сукупність кількісних параметрів визначають можливості застосовності системи для вирішення завдань реального часу з певними динамічними параметрами. Таким чином, для обґрунтування необхідного складу основних кількісних критеріїв оцінки параметрів СРЧ необхідно розглянути вимоги до параметрів часу з боку вирішуваного єдиного завдання і особливості функціональних залежностей змін стану безлічі зовнішніх компонент (об'єктів управління) від параметра часу.

Головною вимогою з боку вирішуваного завдання до параметрів обчислювальної середи є забезпечення необхідного рівня достовірності отримуваного рішення єдиної задачі в сукупності із зовнішніми компонентами і користувачем. Основним джерелом виникнення погрішностей в середі обчислювача РВ є результати відображення і перетворення безперервних зовнішніх функцій і параметра реального часу в безліч поточних дискретних значень. За кількісний параметр оцінки погрішності відображення станів будь-якої із зовнішньої компоненти, може бути прийнята мінімально допустима величина затримки при виконанні розрахунків в середовищі обчислювача. Ця величина вважається допустимою для систем з конкретно заданими динамічними характеристиками процесів зовнішніх компонент. Таким чином, кількісна оцінка достовірності (погрішності) здобуття єдиного рішення є залежною від сукупності параметрів змін процесів в об'єктах управління у функції часу.

Значення параметрів об'єктів управління засобами обчислювального вузла визначаються на дискретних точках часу, що відстають на певний інтервал – інтервал дискретизації даного параметра. Величина інтервалу дискретизації параметрів визначає величину погрішності його відображення в середовіещ обчислювача. Оскільки дискретизація всіх параметрів зовнішніх компонент здійснюється засобами обчислювача у функції деяких інтервалів часу, то достовірність і точність рішення задачі управління залежать від сукупності дискретизації всіх параметрів об'єктів управління.

Основою відображення (перетворення) параметра реального часу (t) в параметр часу обчислювача і всієї системи (t’), є дискретизація безперервної змінної часу з формуванням кінцевої безлічі дискретних значень. Значення інтервалів часу між сусідніми точками безлічі t’ можуть бути однаковими або визначатися по фактичній тривалості обчислювальних процесів в кожній з точок безлічі. Відповідно до цього виділяють асинхронну і синхронну дискретизацію змінної часу. Кожен із способів дискретизації володіє своїми достоїнствами і недоліками.

При синхронній дискретизації спрощуються всі обчислювальні операції над параметром часу системи, оскільки може бути використана внутрішня шкала часу, одиницею рахунку якого є тривалість інтервалу дискретизації – крок дискретизації.

Спрощуються алгоритми системного ПЗ СРЧ. Внутрішня шкала часу характеризує поточні значення часу у вигляді цілих чисел, кожне з яких є

номером інтервалу від деякого моменту, прийнятого в системі за початок відліку.

Довжина розрядної сітки для представлення значень змінною часу визначається максимальною тривалістю інтервалу рішення єдиної задачі РЧ відносно обраного початку відліку часу (0 ÷ tріш_мах) в числі кроків дискретизації і таким чином може мати мінімально можливе значення.

Недоліком синхронного способу дискретизації змінної часу є обмеження на мінімальне значення погрішності фіксації моментів настання зміни полягань в зовнішніх компонентах. Ця величина визначається тривалістю вибраного кроку дискретизації.

При асинхронній дискретизації значення моментів часу можуть визначатися точніше. Єдиним обмеженням, що визначає погрішність відліку часу, є характеристики фізичного пристрою відліку часу - таймера у складі мікропроцесору обчислювача СРЧ.

Недоліком асинхронного способу дискретизації змінної часу є збільшення довжини розрядної сітки для представлення змінних часу, значення якої визначається сукупністю параметрів лічильника таймера і максимальною тривалістю інтервалу рішення (tріш_мах) єдиної задачі РЧ. Ускладнюються всі обчислювальні операції над параметрами часу в системі, оскільки необхідно додатково враховувати циклічний характер відліку часу у внутрішніх лічильниках таймера, а отже збільшуються витрати системного часу.

В більшості випадків для рахунку і обліку параметра часу в СРЧ використовується синхронна дискретизація. У ряді випадків для підвищення точності реєстрації деяких подій, для них значення моментів часу доповнюється поточними значеннями параметрів таймера на момент настання даної події. Такий підхід дозволяє використовувати переваги синхронної дискретизації із збереженням для ряду параметрів можливості фіксації моментів часу з максимальною точністю.

Незалежно від способу обліку параметрів часу, в більшості СРЧ, що розробляються, облік і контроль параметрів системного часу здійснюється у відносних одиницях, відносно умовно прийнятого початку відліку. Для розрахунків абсолютних показників часу для значень моментів часу або тривалості інтервалів часу, необхідно зберігати в параметрах системи значення абсолютного часу точки умовного початку відліку внутрішнього відносного си-

стемного часу і абсолютне значення тривалості інтервалу дискретизації параметра реального часу. Таким чином забезпечується однозначна конвертація параметрів внутрішнього системного значення часу в еквівалентні значення реального часу в прийнятих абсолютних одиницях його рахунку.

В процесі проектування програмного забезпечення СРЧ спочатку необхідно розрахувати і обґрунтувати величини інтервалів дискретизації функцій станів об'єктів управління і на підставі цього визначити величину основного критерію – величину інтервалу дискретизації змінної часу (Δtсист).

Другою обов'язковою вимогою до апаратно-програмних засобів проектованої СРЧ є забезпечення здобуття достовірних результатів на всьому інтервалі рішення єдиної задачі. Тому другим кількісним критерієм оцінки параметрів СРЧ є величина максимально інтервалу часу, на якому може продовжуватися здобуття достовірних результатів рішення. Відповідно до даного критерію і обраного способу дискретизації параметра часу, визначається розрядна сітка для представлення всіх параметрів часу.

Розглянемо методику набуття оптимального значення даного параметра для проектованої СРЧ.

У більшості сфер застосування СРЧ є основою для побудови автоматизованих систем контролю і управління процесами реального часу. При цьому в більшості випадків здобуття рішення повинне здійснюватися на інтервалах існування або актуальності процесу, що автоматизується. Значення таких інтервалів залежно від особливостей конкретного вживання складають величини від декількох діб до декількох років.

Вибір величини розрядної сітки і способу представлення параметрів часу додатково визначає необхідна розмірність масивів і структур для зберігання оперативних даних алгоритмічної обробки і способів доступу до них, що забезпечує мінімальний час вибірки – записи. Відповідно до цього найбільш оптимальним є пошук мінімально можливої розрядності представлення значень часу. При цьому для кожної структурної одиниці вибирається своя мінімальна розрядна сітка для змінних часу.

Основою структуризації інтервалу рішення є представлення процесу вирішення у вигляді безлічі подій, що характеризують зв'язні технологічні процеси, які виділяються в окремі технологічні завдання – технологічні цикли. Кожен технологічний цикл характеризується завершенням деякого техно-

логічного процесу. Для здобуття достовірного рішення єдиної задачі на інтервалі технологічного циклу всі події в межах даного циклу мають бути помітні за часом. Розрізнювальність подій є обов'язковою вимогою до здобуття рішення. Розрізнювальність подій за часом забезпечується величиною розрядності достатньої для відображення максимальної тривалості технологічного циклу. Розрізнювальність подій що належать різним технологічним циклам не потрібний, оскільки єдине завдання вирішується в межах кожного з циклів. Таким чином, для забезпечення процесу рішення єдиної задачі вистачає для змінних часу резервувати розрядну сітку достатню для представлення тривалості інтервалу часу, який характеризує технологічний цикл максимальної величини.

Організація обчислювальних процесів з розрахунком і використанням значень тривалості інтервалів між окремими подіями технологічного циклу повинна враховувати можливі варіанти взаємного розташування інтервалів технологічних циклів на осі часу рішення єдиної задачі. Інтервали можуть слідувати один за одним із зазорами, тривалість яких може бути значною або «перекриватися». В разі пересічень інтервалів повинна дотримуватися умова унікальності подій на інтервалі пересічення. Ця умова означає, що всі події, що виникають на інтервалі перекриття, відповідно до свого вигляду можуть бути однозначно віднесені до циклу, що завершується, або до нового циклу. Ця вимога повинна забезпечуватися відповідними обмеженнями на параметри технологічних процесів в зовнішніх компонентах.

Наявність пересічень інтервалів технологічних циклів в обчислювальній середі найпростіше враховується з використанням формату відліку внутрішнього системного часу у вигляді циклічного лічильника без примусового обнулення. Дана особливість накладає відповідні вимоги на алгоритми розрахунку тривалості інтервалів між окремими подіями кожного з циклів.

Величина інтервалу технологічного циклу є другою за значимістю кількісною характеристикою кожної СРЧ. Неправильний вибір даного інтервалу може привести до помилкового рішення, оскільки буде виконана спроба в середі якого-небудь обробника реалізувати розрахунок однієї з функцій на основі неіснуючого в системі інтервалу між подіями, належним різним циклам.

3.1 Методика розрахунку головного кількісного критерію СРЧ

Розрахунок інтервалу дискретизації змінної часу нерозривно пов'язаний з особливостями дискретизації параметрів зовнішніх компонент при їх віддзеркаленні в середовище обчислювача СРЧ.

Величина інтервалу дискретизації для цифрового обчислювача може розглядатися як період організації обміну інформацією з деякою зовнішньою компонентою (tобм). Величина інтервалу обміну визначає величину погрішності контролю значення функції стану відповідної зовнішньої компоненти. З врахуванням конкретного математичного опису рішення єдиної задачі в цифровому середовищі СРЧ, аналітично або апріорі може бути встановлено функціональний зв'язок значень безлічі погрішностей представлення змінних зовнішніх компонент з погрішністю отримуваного при цьому рішення. Співвідношення зв'язності параметричних погрішностей з погрішністю спільного рішення шляхом рішення зворотної задачі приводить до знаходження допустимих значень погрішностей по окремих параметрах при відомому значенні допустимого значення погрішності рішення спільної задачі. Вихідною інформацією для розрахунку необхідного значення величин параметрів періодів обміну, є задане значення допустимої погрішності дискретизації по цих параметрах. Для кожного з параметрів на основі значення допустимої погрішності дискредитації (δfi) і заданій залежності зміни даного параметра в часі ($fi(t)$), необхідно визначити максимально допустиму тривалість інтервалу обміну – інтервалу дискретизації (tобмі), при якій зміна параметра Δfi на інтервалі tобмі не перевищить величини δfi. Залежно від динамічних властивостей функцій зміни значень кожного з параметрів (Δfi) при однаковій величині інтервалу дискретизації (Δt) має різну величину. Величина зміни значення функції на інтервалі визначається швидкістю зміни значень в області даної крапки. Значення tобмі для кожної функції визначається в області крапки з максимальним значенням швидкості зміни параметра в часі, таким чином, що б зміна функції в цій області не перевищувала величину допустимої погрішності дискретизації даної змінної. Пошук такої крапки необхідно здійснювати індивідуально для кожної функції на всьому інтервалі участі її в здобутті рішення єдиної задачі. Критерієм знаходження задовольняючого значення tобм є співвідношення отримуваних в цих точках змін функцій з початково заданими значеннями (δfi).

Таким чином визначається безліч інтервалів часу, на яких системні засоби СРЧ повинні забезпечувати управління здобуттям рішення поставленої задачі. Для коректного управління процесами дискредитації по всій безлічі змінних, необхідно, щоб відлік часу в системі, а відповідно і інтервал дискретизації змінної часу (Δtсист), здійснювався із значенням хоч би на порядок менше наймінімальнішого значення з безлічі знайдених $tобм_i$.

$$Δtсист ≤ tобм\_min / 10. \quad (3.1)$$

Значення кроку дискретизації (Δtсист) є основним кількісним параметром, що характеризує можливості кожної конкретною СРЧ достовірно здійснювати рішення єдиної задачі (з погрішністю не гірше (δF) в комплексі із зовнішніми компонентами, динамічні параметри функцій яких забезпечуються даним значенням Δtсист.

Значення параметра Δtсист є мінімально помітним інтервалом часу в системі і абсолютним значенням тривалості одиниці внутрішнього відліку часу. Δtсист є мінімальним інтервалом розрізнюваності подій в часі.

Методика розрахунку величини Δtсист залежить від способу завдання функцій, що описують зміни параметрів зовнішньою компоненти в часі (fi(t)). У свою чергу, спосіб завдання цій функцій залежить від рівня формалізації даних процесів і наявності їх математичних моделей.

Функції можуть задаватися аналітично або представлені у вигляді осцилограм або таблиць, отриманих емпірично.

Вихідними даними для виконання розрахунків є :

- опис функцій зміни параметрів в часі для всіх зовнішніх компонент – {fi(t)};
- безліч допустимих значень погрішностей дискретизації функцій – {δfi};
- тривалість інтервалів участі всіх функцій (fi) в здобутті рішення єдиної задачі – tріш.

У загальному випадку методика включає наступні етапи:

1) На першому етапі розрахунків для кожної із зовнішніх функцій визначається точка або інтервал з максимальною швидкістю зміни змінної. Якщо функція задана аналітично по першій похідній визначаються точки екст-

ремуму. Екстремум - найбільше абсолютне значення і визначить точку з максимальною швидкістю зміни змінної в часі (f'max). Якщо функція задана таблично, пошук інтервалу з максимальною швидкістю зміни змінних реалізується чисельним методом з апроксимацією точкового завдання функції (Vmax). Для збільшення точності розрахунків у кожному конкретному випадку можуть використовуватися лінійні або нелінійні способи апроксимації. При дослідженні функції можуть бути варіанти, коли екстремум першої похідної або розрахункові значення функції будуть нескінченно великими (значення прагнуть в нескінченність). В цьому випадку розрахунки обмежуються досягненням значення функції fmax, яке визначається з умови діапазонів зміни параметрів фізичних процесів в конкретних пристроях. Інтервал і точка з максимальною швидкістю знаходитимуться на одному з кордонів інтервалу дослідження функції (Vmax). В результаті виконання першого етапу отримуємо множину максимальних значень швидкостей зміни функцій (f'max або Vmax) і значень моментів часу, в яких вони отримані.

2) На другому етапі в області знайдених точок визначаємо множину значень $tобм_i$ дискретизації або обробки значень кожній з функцій.

- для аналітичного завдання функції:

$$tобм_i = \delta fi / f'max;$$

- для табличного завдання функції:

$$tобм_i = \delta fi / Vmax.$$

В результаті 2-го етапу визначається безліч $\{ tобм_i \}$.

3) На третьому етапі визначаємо мінімальну величину інтервалу обміну (tобм_ min) з безлічі допустимих значень інтервалів обміну всіх функцій $\{ tобм_i \}$.

4) На останньому етапі визначаємо максимально допустиме значення інтервалу дискретизації змінної часу (Δtсист).

$$\Delta tсист = tобм\_min / 10.$$

4. Типи подій в СРЧ

4.1 Періодичні і асинхронні події

Основні обчислювальні процеси в СРЧ реалізуються як функції параметра часу, а також сукупності параметрів, що характеризують стан зовніш-

ніх компонент і інтерфейсу користувача. Особливістю організації таких процесів є необхідність ініціювати виконання розрахункових залежностей лише в моменти вступу нових значень параметрів цих компонент або команд користувача.

Інформація, що поступає, може носити кількісний характер або сигналізувати про якісні зміни стану компонентів. Кількісна інформація поступає в обчислювальне середовище відповідно до параметрів дискретизації безперервних функцій, тобто в моменти, визначені періодами обміну. Обчислювальні процеси обробки таких змін повинні запускатися по подіях завершення інтервалів часу дискретизації функцій від зовнішніх компонент. Обробка якісних змін достатків зовнішніх компонент завжди ініціюється асинхронно в моменти виникнення подій, що характеризують такі зміни.

Таким чином, основою організації розрахункових процесів в середовищі обчислювача СРЧ є організація обробки подій. Способи управління обробкою подій визначаються вимогами достовірного рішення єдиної задачі і залежать від видів оброблюваних подій і сукупності обмежень на параметри часу в процесі виконання обробки.

Всі процеси в зовнішніх компонентах характеризуються сукупністю змін параметрів і станів. Зміна параметрів відбувається безперервно в часі, і характеризує кількісні зміни в стані об'єкту. Для обліку кількісних безперервних змін параметрів в програмному середовищі обчислювача СРЧ системне ПЗ РЧ реалізує дискретизацію змінних станів, що поступають, і формує періодичні події, що вимагають проблемної обробки. Періодичні події забезпечують обробку змін параметрів проблемними завданнями.

Моменти настання періодичних подій завжди синхронні з періодом дискретизації змінної часу -Δtсист.

В процесі рішення єдиної задачі в окремих компонентах або у всьому об'єкті в цілому можуть виникати якісні зміни станів. Це стрибкоподібні зміни, які виникають асинхронно в часі.

При асинхронному надходженню сигналу про якісну зміну стану системне ПЗ (супервізор РЧ) отримує управління шляхом фізичного переривання всіх обчислювальних процесів і за результатами обробки може передати управління відповідному прикладному обробникові. Це сигнали проблемних

переривань. Сигнали проблемних переривань поступають в систему за допомогою введення ініціативних сигналів.

При асинхронній передачі управління прикладному обробникові момент часу надходження події може фіксуватися з дискретністю прийнятою у всій системі і для всіх типів обробників (Δtсист), або з дискретністю мінімального рахунку одиниць лічильником таймера (Ттм), відповідного частоті на вході таймера (fтм), Ттм = 1 / fтм. У останньому випадку збільшується точність визначення моменту настання події.

В більшості випадків використовують основний відлік часу. І лише для особливо актуальних подій додатково зчитують поточне значення лічильника таймера. Поточне значення лічильника таймера характеризує момент настання переривання на інтервалі основної одиниці рахунку.

Різні види оброблюваних подій РЧ розрізняються по події, що ініціює, і за способом реакції на неї при передачі управління прикладному обробникові. Так періодичні події ініціюються по завершенню призначеного інтервалу часу за допомогою сигналу таймера. Прикладом періодичних подій є реалізація функції дискретизації безперервних значень параметрів зовнішніх компонент. По завершенню інтервалу часу можуть призначатися і асинхронні події. Прикладом таких подій є обробка тайм-аутів, тобто функцій контролю тривалості деяких процесів.

Події про якісні зміни ініціюються сигналами проблемних переривань. Під час надходження таких подій супервізор визначає і передає управління відповідному завданню проблемної обробки. Таким чином, кожна подія однозначно відповідає певному сигналу, що ініціює, і завданню проблемної обробки.

4.2 Актуальність подій

Відповідно до типа подій і їх функціональної особливості, окремі події можуть володіти властивістю актуальності не на всьому інтервалі рішення єдиної задачі, а лише на окремих її частинах. Відповідно до розбиття всього інтервалу вирішення на сукупність інтервалів технологічних циклів, актуальність всіх подій повинна визначатися в межах окремих технологічних циклів.

Деяка подія перестає бути актуальним, якщо сукупність поточних станів зовнішніх компонент або інтерфейсу користувача приводять до режи-

му рішення єдиної задачі, при якому в алгоритмічній обробці не беруть участь модулі обробки даної події.

Весь інтервал рішення єдиної задачі і відповідно інтервали технологічних циклів для кожної події (кожного завдання його обробки) розбивається на ряд інтервалів активності. Кожен з обробників повинен мати хоча б один інтервал активності на всьому інтервалі рішення єдиної задачі.

Кожен інтервал активності характеризується рядом параметрів. Основними параметрами кожного з інтервалів активності, є значення його меж:

- Тпоч – момент часу початку інтервалу активності;
- Ткін – момент завершення інтервалу активності.

Чисельні значення меж можуть визначатися на початку технологічного циклу, або задаватися в процесі рішення задачі, але значення, що при цьому задається, повинне визначати момент часу правіше поточного значення часу (в майбутньому від поточного значення). Межі інтервалів можуть задаватися одним з 3-х способів:

- у вигляді чисельної константи, значення якої залишається незмінним на всьому інтервалі технологічного циклу або всьому інтервалі рішення;
- у вигляді чисельної змінної, початкове значення якої не визначене і числове значення задається лише в процесі обробки інших подій;
- у вигляді чисельної змінної, початкове значення якої не визначене і числове значення задається лише по зовнішній події з інтерфейсу користувача.

При завданні будь-якому з кордонів інтервалу активності по події, значення кордонів набуває рівним поточному значенню часу настання даної події.

Кожен інтервал активності є одним з параметрів події, для якої він визначений і відповідно цій події, параметром призначеного обробника (завдання обробки). Всі події поза інтервалом активності не приводять до виклику відповідного обробника. Поза інтервалами активності відповідні завдання обробки не активні, а на кордонах інтервалу супервізор змінює їх стани з врахуванням поточного стану обчислювального процесу.

Вся сукупність подій за ініціативою завершення інтервалів часу обробляється призначеними проблемними завданнями – проблемними обробни-

ками. Організації управління завданнями обробки по кожній події здійснюється відповідно до безліччю параметрів, сукупність яких має бути достатньою для виконання супервізором РЧ функцій управління викликом обробників і контролю їх функціонування.

Достатньою сукупністю є множина, що містить:

- параметри, що відображають особливості подій, що ініціюють;
- сукупність вимог і умов можливості виконання обробки події, що настала;
- сукупність вимог до контролю обмежень на інтервали часу знаходження завдання обробки в певних станах;
- параметри оцінки поточного стану завдання обробки.

По особливостях подій, що ініціюють, виділяють періодичну обробку і асинхронну. При періодичній обробці завдання викликається після завершення інтервалу часу – періоду, який ініціюється супервізором періодично. Кожен подальший інтервал відліку часу запускається після завершення попереднього.

Асинхронна обробка запускається після завершення інтервалу часу, відлік якого ініціювався за деякими зовнішніми умовами або подіями. Після завершення обробки асинхронної події повторний виклик обробника ініціюється асинхронно і не залежить від завершення попередньої обробки

На етапі проектування ПЗ СРЧ розробляється функціонально повна множина станів, в яких може перебувати завдання-обробник. Як обов'язкові елементи цієї множини виступають:

- завдання в стані чекання виклику на виконання;
- завдання в стані виконання;
- завдання в стані переривання.

У одному з цих станів завдання може знаходитися лише якщо воно активне, тобто на інтервалі від Тпоч до Ткін. За межами вказаного інтервалу завдання вважається не активним і управління отримати не може.

Після настання моменту Тпоч супервізор переводить завдання - обробник в стан активності. Після цього, якщо дане завдання визначене як періодичне, супервізор викликає (намагається викликати) з вказаним періодом. Для завдань асинхронної обробки вирішується реакція на зовнішні події, що ініціюють. Після настання події по завершенню інтервалу часу або надхо-

дженню сигналу проблемного переривання, супервізор перевіряє можливість виклику завдання обробки на виконання. Основною умовою можливості виклику завдань на виконання є стан всієї системи – вільна від виконання завдань обробки РЧ. Якщо виклик можливий, то завдання переводиться в стан виконання, якщо стан системи не дозволяє передати управління даному завданню обробки, то завдання переводиться в стан чекання виклику. При подальших викликах супервізора по сигналах таймера (з періодом Δtсист), контролюється зміна стану системи і після звільнення системи синхронно з черговим Δtсист, переводить завдання в стан виконання – тимчасова точка фактичного виклику завдання на виконання може бути відмінна від раніше розрахованої планової. При цьому змінюється (призначається) супервізором новий стан для всієї системи – виконується обробка завдання . При знаходженні системи в стані виконання, супервізор контролює тривалість роботи завдань. В процесі виконання завдання процесорний ресурс може на деяких інтервалах надаватися іншим завданням – переривання інтервалу виконання. На цих інтервалах дане завдання переходить в стан «переривання обробки».

В стані роботи завдання супервізор може контролювати два параметри: тривалість роботи (Троб) і тривалість виконання (Твик):

- Троб – тривалість інтервалу часу від перекладу завдання в стан виконання до моменту завершення завданням своєї роботи і звільнення процесорного ресурсу;
- Твик – сукупність тривалості інтервалів, на яких завдання дійсно займало процесорний ресурс, тобто реалізувала свій алгоритм.

Значення Троб і Твик відрізняються на значення тривалості інтервалу часу, в перебігу якого уривалося виконання завдання з наданням процесорного ресурсу приоритетнішим завданням обробки. В тому випадку, якщо переривання роботи завдання відсутні, параметри Троб і Твик рівні.

Вся розглянута сукупність параметрів є необхідною множиною значень, які характеризують індивідуальні особливості кожного із завдань обробки. Відповідно, ці значення необхідні супервізору для організації обчислювальних процесів обробки подій в режимі РЧ кожного з проблемних завдань.

5. Спільні принципи розробки інтерфейсу користувача.

Для більшості СРЧ головний екран додатка повинен в комплексі представляти стан об'єкту управління, а по запитах оператора - необхідну детальну інформацію. Зовнішній вигляд головного вікна додатка може сильно залежати від типа об'єкту управління і апаратної реалізації автоматизованої системи управління. При вживанні монітора з сенсорним екраном класичний стиль додатка не підійде, необхідно проектувати вікна додатка з великими кнопками і можливо без рядка меню.

Якість роботи СРЧ в значній мірі визначається тим, наскільки адекватно сприймає оператора інформацію, що поступає, і наскільки своєчасно він на неї реагує. При достатньому рівні підготовленості персоналу основним чинником, що впливає на роботу оператора, є якість організації його взаємодії з системою, тобто її інтерфейс. В той же час, навіть в разі ухвалення правильного рішення оператор може допустити так звану функціональну помилку (натиснути не ту клавішу, вибрати не ту команду) і так далі. Небезпека функціональних помилок істотно зростає в стресових ситуаціях. Іншими словами, якість роботи СРЧ залежить від форми представлення інформації про поточну ситуацію в системі і від доступних операторові засобів дії на виконавчі компоненти системи.

Таким чином, при розробці призначеного для користувача інтерфейсу СРЧ основна увага має бути приділена наступним питанням:

- детальному проектуванню сценарію діалогу з метою вибору оптимальних маршрутів переміщення оператора по дереву діалогу, а також запобігання ситуаціям, які можуть зажадати перезапуску системи.
- реалізації засобів динамічної зміни структури діалогу залежно від поточної ситуації, що складається в системі.
- ретельному вибору візуальних атрибутів інформації, що відображується, у тому числі вибиранню засобів залучення уваги користувача (оператора).

При цьому повинна забезпечуватися властивість природності інтерфейсу СРЧ. Мається на увазі наступне. У багатьох системах управління технологічними процесами за роки їх існування була сформована оптимальна структура засобів індикації і контролю, а також відповідна неї система умовних

позначень, використовувана на операторських пультах. При створенні робочих місць операторів враховувалися результати вельми глибоких ергономічних дослідженні. Тому при проектуванні інтерфейсу автоматизованих робочих місць (АРМ) на базі ПЕОМ доцільно зберегти основну схему візуалізації процесів, що протікають в даній системі управління. Невипадково практично всі інструментальні засоби, призначені для розробки інтерфейсу систем диспетчерського управління містять графічні бібліотеки, що дозволяють відтворювати на екрані монітора мнемосхеми ідентичні тим, що були на колишніх пультах . Такий підхід дозволяє використовувати при роботі оператора не лише візуальні, але і інші (в першу чергу - звукові) засоби індикації і залучення уваги.

Ефективність роботи користувача визначається не лише функціональними можливостями наявних в його розпорядженні апаратних і програмних засобів, але і доступністю для користувача цих можливостей. У свою чергу, повнота використання потенційних можливостей наявних ресурсів залежить від якості призначеного для користувача інтерфейсу. Створення якісного інтерфейсу вимагає значно більшого, ніж просто дотримання деяких інструкцій.

5.1 Принципи побудови інтерфейсу

Принципи побудови інтерфейсу користувача СРЧ базуються на спільних концепціях створення інтерфейсних застосувань, але з урахуванням специфіки сфери вживання. Серед спільних принципів слід виділити:

- природність інтерфейсу.;
- узгодженість інтерфейсу.;
- дружність інтерфейсу;
- принцип «зворотного зв'язку»;
- простота інтерфейсу;
- видимість інтерфейсу;
- спроможність інтерфейсу.

Природний інтерфейс — такий, який не вимушує користувача істотно змінювати звичні для нього способи рішення задачі. Це, зокрема, означає, що повідомлення і результати, що видаються додатком, не повинні вимагати додаткових пояснень, що інформація відображується на екрані у вигляді, при-

датному для безпосереднього використання. Не слід заставляти користувача додатково обробляти цю інформацію, наприклад, уточнювати по довідниках значення кодів, проводити які-небудь перетворення, перерахунки і тому подібне. Формат для виведення дати, часу і інших подібних стандартизованих даних має бути загальноприйнятим, а не індивідуальним для даної системи. Загальноприйнята система поєднання великих і малих букв в тексті покращує його сприйняття. Доцільно також зберегти систему позначень і термінологію, використовувані в даній наочній області. Використання знайомих користувачеві понять і образів забезпечує інтуїтивно зрозумілий інтерфейс при виконанні його завдань.

Узгодженість інтерфейсу дозволяє користувачам переносити наявні знання на нові завдання, освоювати нові аспекти швидше, і завдяки цьому фокусувати увагу на вирішуваному завданні, а не витрачати час на з'ясування відмінностей у використанні тих або інших елементів управління, команд і так далі Забезпечуючи спадкоємність отриманих раніше знань і навиків, узгодженість робить інтерфейс впізнаним і передбаченим. Узгодженість важлива для всіх аспектів інтерфейсу, включаючи імена команд, візуальне представлення інформації і поведінку інтерактивних елементів. Для реалізації властивості узгодженості в створюваному програмному забезпеченні, необхідно враховувати його різні аспекти.

Дружність інтерфейсу. Користувачі зазвичай вивчають особливості роботи з новим програмним продуктом методом проб і помилок. Ефективний інтерфейс повинен брати до уваги такий підхід. На кожному етапі роботи він повинен вирішувати лише відповідний набір дій і запобігати користувачам про ті ситуації, де вони можуть пошкодити системі або даним; ще краще, якщо у користувача існує можливість відмінити або виправити виконані дії. Навіть за наявності добре спроектованого інтерфейсу користувачі можуть робити ті або інші помилки. Ці помилки можуть бути як «фізичного» типа (випадковий вибір неправильної команди або даних) так і «логічного» (ухвалення неправильного рішення на вибір команди або даних). Ефективний інтерфейс повинен дозволяти запобігати ситуаціям, які, ймовірно закінчаться помилками

Принцип «зворотного зв'язку». Необхідно забезпечувати зворотний зв'язок для дій користувача. Кожна дія користувача повинна отримувати ві-

зуальне, а інколи і звукове підтвердження того, що програмне забезпечення сприйняло введену команду; при цьому вигляд реакції, по можливості, повинен враховувати природу виконаної дії. Зворотний зв'язок ефективний в тому випадку, якщо він реалізується своєчасно, тобто як можна ближче до точки останньої взаємодії користувача з системою.

Простота інтерфейсу. Інтерфейс має бути простим. Один з можливих шляхів підтримки простоти — відображення на екрані інформації, мінімально необхідної для виконання користувачем чергового кроку завдання. Багатослівні командні імена або повідомлення, непродумані або надлишкові фрази ускладнюють користувачеві отримання істотної інформації. Інший шлях до створення простого, але ефективного інтерфейсу — розміщення і представлення елементів на екрані з урахуванням їх смислового значення і логічного взаємозв'язку. Це дозволяє використовувати в процесі роботи асоціативне мислення користувача.

Видимість інтерфейсу. Інтерфейс повинен своєчасно інформувати користувача про: поточний стан системи і зміну стану в результаті дій користувача; способи управління і дії на систему. В разі відсутності інформації про стан системи зростає кількість помилок, що допускаються при роботі. Інформування користувача про стан системи включає у тому числі і надання різного роду зворотному зв'язку: підсвічування об'єкту, індикацію про те, що дія користувача сприйнята і знаходиться в обробці і тому подібне У разі, коли відсутня видима інформація про способи дії на систему, людині доводиться виконувати зайву роботу, пов'язану із здобуттям цієї інформації, що навряд чи сприяє концентрації на власних завданнях. Елемент управління можна вважати видимим, якщо він безпосередньо доступний для сприйняття або був сприйнятий настільки недавно, що ще не встиг вийти з короткочасної пам'яті. При цьому елемент має бути не просто видимим в принципі, але видний там, де його чекає побачити користувач. Можна передбачити, що саме унаслідок видимості графічні інтерфейси стали набагато популярнішими ніж консольні. Хоча останні можуть бути цілком зручні для роботи, але «поріг входження» для них набагато вищий.

Спроможність інтерфейсу. Елемент управління є не лише видимим, але і спроможним, якщо поодинці його вигляду користувач може визначити, як саме з ним взаємодіяти. Таким чином, принцип спроможності доповнює

принцип видимості. Взаємодія з подібними по вигляду елементами і об'єктами повинна відбуватися завжди одноманітно, інакше виникатимуть помилки неоднозначності. Прикладом може бути інтерфейс, в якому в одній ситуації елемент, що виглядає як кнопка, необхідно натискувати, а в іншій ситуації елемент, що виглядає так само, необхідно перемістити.

Для якісного сприйняття інформації, велике значення має раціональне розміщення даних на екрані. Необхідна щільність розташування даних — поняття суб'єктивне. Вона залежить від конкретного користувача і вирішуваного завдання. Проте існують деякі правила, регулюючі щільність розташування даних на екрані (або в межах вікна):

- фрагменти тексту повинні розташовуватися на екрані так, щоб погляд користувача переміщався в потрібному напрямі;
- вміст полів повинен не «притискатися» до краю екрану, а розташовуватися біля його горизонтальних або вертикальних меж;
- меню, що містить відносно невеликий об'єм інформації, повинне зміщуватися в ліву верхню частину екрану;
- щоб підкреслити симетрію, вміст і найменування полів, що відносяться до однієї групи, повинні вирівнюватися по вертикалі. По можливості необхідно вирівнювати всі логічно зв'язані групи даних.

Інший набір рекомендацій визначається чинниками, пов'язаними з право / лівою асиметрією головного мозку людини. Відомо, що ліва і права півкулі по-різному беруть участь в сприйнятті і переробці інформації. Зокрема, при запам'ятовуванні слів провідну роль грає ліва півкуля, а при запам'ятовуванні образів активніше праве. Інформація з правої частини екрану поступає безпосередньо в ліву півкулю, а з лівої частини — в праву (природно, при бінокулярному зору оператора). У зв'язку з цим можна рекомендувати текстові повідомлення групувати справа, а зображення — зліва. У деяких людей це розподіл функцій півкуль протилежний, у жінок асиметрія виражена слабкіше, ніж у чоловіків. Цей факт ще раз підтверджує необхідність індивідуалізації характеру відображення інформації. Облік право / лівої асиметрії пам'яті має істотне значення, якщо інтервали дотримання повідомлень не перевищують 10с. Тому приведені рекомендації обов’язково слід враховувати в інтерфейсах програм, що працюють в режимі реального часу.

Раціональне розміщення даних на екрані є найбільш важливим, але не єдиним методом забезпечення зручності і природності призначеного для користувача інтерфейсу. Сучасні монітори надають в розпорядження розробника різні методи виділення інформації, що виводиться, на екрані.

Виділення інформації — це використання таких атрибутів, які дозволяють привернути увагу користувача до деякої області екрану. У якості таких атрибутів можуть виступати: колір символів, колір фону, рівень яскравості, мерехтіння і вживання різних шрифтів для символів, що виводяться. Часто для виділення інформації використовують підкреслення, вивід в інверсному вигляді, різні рамки і «тіні». Ефект вживання цих атрибутів різний, а їх поєднання — часто непередбачуване і залежить від індивідуальних особливостей користувачів.

Для концентрації уваги пропонується відокремлювати вміст від форми. Для цього застосовуються два методи — прямокутників і виділених точок.

При використанні методу прямокутників після розбиття екрану на поля кожне з них заповнюється інформацією і відділяється від інших по всьому периметру, принаймні, одним пропуском. Через центр екрану (умовно) проводяться осі, що дозволяють оцінити збалансованість розміщення полів.

Метод виділених точок дозволяє визначити число і розміщення областей екрану, до яких буде притягнена увага користувача (із-за підвищеної яскравості, кольору або мерехтіння символів). Для цього кожна область, що вимагає підвищеної уваги, моделюється групою символів, відмінних від пропуску.

Розглянуті методи дозволяють усунути грубі помилки у форматуванні екрану, проте кращий спосіб оцінити його якість — дати можливість потенційному користувачеві попрацювати з системою.

6. Етапи проектування призначеного для користувача інтерфейсу

При проектуванні призначеного для користувача інтерфейсу необхідно визначити:

- структуру діалогу;
- можливий сценарій розвитку діалогу;

- вміст повідомлень і даних, якими можуть обмінюватися оператор з СРЧ (семантику повідомлень);
- візуальні атрибути інформації, що відображується (синтаксис повідомлень).

6.1 Вибір структури діалогу

Вибір структури діалогу — це перший з етапів, який має бути виконаний при розробці інтерфейсу. Розглянемо деякі види діалогу.

Діалог типа «питання - відповідь». Структура діалогу із звичайним інтерв'ю. Система бере на себе роль інтерв'юера і отримує інформацію від користувача у вигляді відповідей на питання. Це найбільш відома структура діалогу; всі діалоги, керовані комп'ютером, в тій або іншій мірі складаються з питань, на які користувач відповідає. Проте в структурі «питання - відповідь» цей процес виражений явно. У кожній точці діалогу система виводить як підказку одне питання, на яке користувач дає одну відповідь. Залежно від отриманої відповіді система може вирішити, яке наступне питання ставити. Ця структура надає природний механізм введення як повідомлень (команд), що управляють, так і даних. Жодних обмежень на діапазон або типи вхідних даних, які можуть оброблятися, не накладається. Існують системи, відповіді в яких даються на природній мові, але частіше використовуються пропозиції з одного слова з обмеженою граматикою.

Діалог у вигляді питань і відповідей достатньою мірою забезпечує підтримку користувача, оскільки навіть коротке навідне питання при розумній побудові може бути самопояснюючим. Структура діалогу не гарантує мінімального об'єму введення, що оцінюється по кількості натиснень клавіш, проте при відповідному підборі скорочень можна зменшити будь-яку надмірність. В той же час, ця структура володіє одним істотним недоліком. Навіть якщо введення відбувається досить швидко, для людини, яка вже знає, які питання ставить система і які відповіді потрібно на них давати, відповідати на всю серію питань досить утомливо.

З появою графічного інтерфейсу структура «питання-відповідь» дещо застаріла, проте у неї є певні достоїнства. Ця структура може задовольнити вимоги різних користувачів і типів даних. Зокрема, така структура особливо доречна при реалізації діалогу з безліччю «відгалужень», тобто в тих випадках, коли на кожне питання передбачається велике число відповідей, кожне з

яких впливає на те, яке питання буде поставлено наступним. З цієї причини дана структура часто використовується в експертних системах, але не в системах управління технологічними процесами.

Діалог на основі меню. Меню є, мабуть, найбільш популярним варіантом організації запитів на введення даних під час діалогу, керованого комп'ютером. Існує декілька основних форматів представлення меню на екрані:

- список об'єктів, обираних прямою вказівкою, або вказівкою номера (або мнемонічного коду);
- меню у вигляді блоку даних;
- меню у вигляді рядка даних;
- меню у вигляді піктограм.

Меню у вигляді рядка даних може з'являтися вгорі або внизу екрану і часто залишається в цій позиції впродовж всього діалогу. Таким чином, за допомогою меню зручно відображувати можливі варіанти даних для введення, доступних у будь-який час роботи з системою. Якщо в системі є чимала різноманітність варіантів дій, організовується ієрархічна структура з відповідних меню. Додаткові меню у вигляді блоків даних «спливають» на екрані в позиції, визначуваній поточним положенням покажчика, або «випадають» безпосередньо з рядка меню верхнього рівня. Ці меню зникають після вибору варіанту.

Меню у вигляді піктограм є безліччю об'єктів вибору, розкиданих по всьому екрану. Часто об'єкти вибору містять графічне представлення варіантів роботи.

Користувач діалогового меню може вибрати потрібний пункт, вводячи текстовий рядок, який ідентифікує цей пункт, вказуючи на нього безпосередньо або переглядаючи список і вибираючи з нього. Система може виводити пункти меню послідовно, при цьому користувач вибирає потрібний йому пункт натисненням клавіші. Меню можна з рівним успіхом застосовувати для введення як повідомлень, що управляють, так і даних. Прийнятна структура меню залежить від його розміру і організації, від способу вибору пунктів меню і реальної потреби користувача в підтримці з боку меню.

Структура типа меню є найбільш природним механізмом для роботи з пристроями вказівки і вибору: меню є зображення тих об'єктів, які вибира-

ються користувачем. Якщо діалог складається виключно з меню, можна реалізувати послідовний інтерфейс, при якому користувач застосовує лише пристрої для вказівки; проте така постійність рідко досяжна на практиці. Слід також відмітити, що, хоча робота з цими пристроями не вимагає професійного володіння клавіатурою, для підготовленого користувача це не найшвидший спосіб вибору з меню. Замість вказівки користувач може повідомити про свій вибір введенням відповідного ідентифікатора.

Меню — це найбільш зручна структура діалогу для непідготовлених користувачів; жорстка черговість відкриття і ієрархічна вкладеність меню може викликати роздратування професіонала, уповільнювати його роботу. Традиційна структура меню недостатньо гнучка і не повною мірою узгоджується з методами адаптації діалогу, такими, наприклад, як випереджаюче введення, за допомогою якого можна прискорити темп роботи підготовленого користувача.

Діалог на основі екранних форм. Як структура типа «питання — відповідь», так і структура типа меню передбачають обробку на кожному кроці діалогу єдиної відповіді. Діалог на основі екранних форм допускає обробку на одному кроці діалогу декількох відповідей. На практиці форми використовуються в основному там, де облік якої-небудь діяльності вимагає введення досить стандартного набору даних. Людина, що заповнює форму, може вибирати послідовність відповідей, тимчасово пропускати деяке питання, повертатися назад для корекції попередньої відповіді і навіть «порвати бланк» і почати заповнювати новий. Користувач працює з формою до тих пір, поки не заповнить її повністю і не передасть системі. Програмна система може перевіряти кожну відповідь безпосередньо після введення або почекати і вивести список помилок лише після заповнення форми цілком. У деяких системах інформація, що вводиться користувачем, стає доступною лише після натиснення клавіші «введення» після закінчення заповнення форми. Питання про те, чи треба перевіряти відповідь безпосередньо або відкласти перевірку до закінчення введення всіх відповідей, вирішити непросто: повідомлення про помилки, що виводяться безпосередньо після відповіді, можуть відвернути увагу, але можуть зробити і позитивний вплив. У тих випадках, коли інформація для введення вибирається з деякого цілісного документа, перевірку краще відкласти до кінця заповнення форми, щоб не переривати процес вве-

дення; якщо ж такої цілісності немає, то перевірку слід виконувати відразу після введення відповіді (після заповнення чергового поля).

Якщо зустрілася яка-небудь помилка, додаток не повинен заново виводити порожню форму; виводиться форма з попередніми відповідями і допущеними помилками. Новий «бланк» видається лише в разі відповідного запиту користувача. Таким чином, цю структуру доречно застосовувати там, де джерелом даних служить існуюча вхідна («паперова») форма документа. Не обов'язково, щоб зовнішній вигляд цих форм збігався (це навіть може погіршити сприйняття даних на екрані), але всі елементи даних, що вводяться, повинні розташовуватися в тому ж відносному порядку і мати такий же формат, що і у вихідному документі.

Часто всі необхідні одиниці введення не можна відображувати одночасно в межах одного екрану (або вікна), і їх необхідно розділити на групи, які відображуються на послідовності екранів (вікон). Поважно, щоб це розбиття зберігало логічні зв'язки і не приводило до розділення зв'язаних частин документа.

Структура діалогу на основі екранної форми забезпечує високий рівень підтримки користувача: для кожного питання форми можуть бути передбачені повідомлення про помилки і довідкова інформація. Користувачеві можна також надати допомогу, включивши деякі елементи формату відповіді в питання або в полі відповіді. Ця структура дозволяє підвищити швидкість введення даних в порівнянні із структурою типа «питання — відповідь» і маніпулювати ширшим діапазоном вхідних даних, ніж меню; крім того, з нею можуть працювати користувачі будь-якої кваліфікації. Оскільки ця структура має послідовну, а не деревовидну організацію, вона у меншій мірі личить для роботи в режимі вибору варіантів. Ще однією сферою застосування екранних форм є завдання параметрів запитів в базах даних. Цей механізм інколи називають запитом за зразком .

Одним з типів заповнення форм є також багатоваріантні меню. У таких меню користувачеві надається список варіантів і він не обмежений можливістю єдиного вибору; можна вказати декілька варіантів.

Діалог на основі командної мови. Структура діалогу на основі командної мови настільки ж поширена, що і структура типа меню. Вона дуже часто використовується в операційних системах і розташовується на іншому кінці

спектру структур діалогу по відношенню до структури типа меню. Історично це перша з реалізованих структур діалогу. При такій організації діалогу система не виводить нічого, окрім постійної підказки (запрошення на введення команди), яка означає готовність системи до роботи. Кожну команду вводять з нового рядка і зазвичай закінчують натисненням клавіші «введення». Відповідальність за правильність команд, що задаються, лягає на користувача. Система інформує про неможливість виконання невірної команди. Подібно до меню, діалог на базі команд зручний для введення повідомлень, що управляють, проте він забезпечує ширші можливості вибору в будь-якій точці діалогу і не вимагає ієрархічної організації обслуговуючих його програм.

Система може підтримувати чималу кількість команд, але на практиці слід обмежувати їх число, щоб не перенавантажувати пам'ять користувача. Структура на базі командної мови не відрізняється хорошою підтримкою користувача і придатна в основному для підготовлених фахівців. Перед використанням такої системи необхідно пройти курс вчення і надалі вивчати особливості роботи по документації, а не на практиці. Більш того, оскільки системі невідомо, що має намір робити користувач, важко запропонувати якунебудь реальну допомогу в процесі роботи, окрім видачі довідок спільного характеру.

Оскільки дана структура передбачає великий об'єм матеріалу, що запам'ятовується, імена команд слід вибирати так, щоб вони несли смислове навантаження і легко запам'ятовувалися. Розробник повинен прагнути уникати зайвої функціональності, що походить від бажання створити власну команду для кожної функції, виконуваною системою, тобто не варто створювати безліч всіляких команд з функціями, що часто перекриваються. Такі «благі» наміри нерідко приводять до появи великої кількості ключових слів для позначення команд і синтаксичних правил, багато хто з яких рідко використовується і лише ускладнює роботу.

Діалог повинен управляти даними. У інтерфейсах на основі мов команд це зазвичай досягається за допомогою складених командних рядків, де ключове слово для позначення команди (що робити) передує списку параметрів (вхідним даним). Параметри в списку можна задавати в одній з двох форм — в позиційній або ключовій. У першому випадку призначення параметра визначається по його місцю в командному рядку. В разі ключових параметрів

кожне значення передує певним ідентифікатором, який визначає його призначення.

Позиційні параметри зменшують об'єм інформації, що вводиться, але їх істотним недоліком є те, що значення, що вводяться, повинні вказуватися в строго певному порядку, порушення якого погано діагностується системою і може спричинити серйозні наслідки. Завдання позиційних параметрів ускладнюється, якщо їх список досить великий. Цей недолік прагнуть компенсувати, дозволяючи опускати незмінні параметри, вводячи два роздільники один за одним.

Ключові параметри зменшують навантаження на пам'ять користувача в тому відношенні, що відпадає необхідність в запам'ятовуванні порядку їх дотримання; крім того, можна опускати необов'язкові параметри. З іншого боку, в цьому випадку користувачеві необхідно запам'ятати безліч ключових слів, а розробникові підібрати для них «осмислені» імена. Цей підхід також вимагає більшого часу роботи системи, щоб розпізнати ключові слова, задані в довільному порядку.

Багато командних мов підтримують макроси, які розширюють функціональні можливості діалогу без збільшення кількості команд. Макрос містить декілька окремих командних рядків. При зверненні до нього в процесі діалогу окремі рядки команд макросу виконуються одна за одною, неначебто вони вводилися з клавіатури. Це істотно скорочує діалогові сеанси, послідовності команд, які, часто повторюються і оформляються у вигляді макросів.

Структура на основі мови команд по своїх можливостях найшвидша і гнучкіша зі всіх структур діалогу. Більшість призначених для користувача інтерфейсів на базі «природної» мови реалізуються за допомогою мов команд з дуже великим набором ключових слів. Проте ця структура не забезпечує користувача підтримкою, тому навіть підготовлені користувачі вважають, що дуже складно використовувати всі закладені в ній можливості. Більшість же користувачів добре знайома лише з вельми обмеженим набором засобів, з яким вони працюють регулярно.

6.2 Сценарій розвитку діалогу

Спосіб опису сценарію діалогу залежить від міри його складності. Існуючі методи опису сценаріїв можна розділити на дві великі групи: неформальні і формальні методи. Головне достоїнство формальних методів полягає в тому, що вони дозволяють автоматизувати як проектування діалогу, так і його модифікацію відповідно до характеристик користувача. В даний час найширше використовуються формальні методи опису сценаріїв на основі мереж Петрі і їх розширень, а також на основі систем представлення знань .

Сценарій діалогу дозволяє описати процес взаємодії користувача з додатком на рівні вирішуваної їм прикладного завдання. У зв'язку з цим при розробці сценарію діалогу повинні враховуватися такі психофізіологічні особливості потенційних користувачів, як моторні навики, час реакції, сприйнятливість колірної гамми і так далі, що визначає темп ведення діалогу.

Час відповіді системи визначається як інтервал між подією і реакцією системи на нього. Дана характеристика інтерфейсу визначає затримку в роботі користувача при переході до виконання наступного кроку завдання. Вимоги до часу відповіді залежать від того, що чекає користувач від роботи системи, і від того, як взаємодія з системою впливає на виконання його завдань.

6.3. Візуальні атрибути відображення

До візуальних атрибутів інформації, що відображується, відносяться:

- взаємне розташування і розмір об'єктів, що відображуються;
- колірна палітра;
- засоби залучення уваги користувача.

Проектування розміщення даних на екрані передбачає виконання наступних дій:

- визначення складу інформації, яка повинна з'являтися на екрані;
- вибір формату представлення цієї інформації;
- визначення взаємного розташування даних (або об'єктів) на екрані;
- вибирання засобів залучення уваги користувача;
- розробка макету розміщення даних на екрані;
- оцінка ефективності розміщення інформації.

Загальні принципи розташування інформації на екрані повинні забезпечувати для користувача:

- можливість перегляду екрану в логічній послідовності;
- простоту вибору потрібної інформації;
- можливість ідентифікації зв'язаних груп інформації;
- розрізнюванність виняткових ситуацій (повідомлень про помилки або запобігань);
- можливість визначити, яка дія з боку користувача потрібна для продовження виконання завдання.

Питання про те, яка інформація підлягає відображенню, вирішується залежно від специфіки виконуваного користувачем завдання. Тут істотну роль грає правильне розбиття завдання на операції (етапи), що не вимагають одночасної присутності великого об'єму даних на екрані. Ця умова витікає з такої психофізіологічної особливості людини, як обмеженість його короткочасної пам'яті, здатної зберігати одночасно не більш п’яти - дев'яти об'єктів. Якщо вся інформація вихідного документа не поміщається на одному екрані, деякі елементи даних можуть повторюватися на інших екранах для збереження цілісності і послідовності обробки. Як правило, повторювана інформація не повинна міняти свого розташування на всіх кроках виконання завдання.

7. Особливості графічного інтерфейсу користувача

У шведському стандарті на призначений для користувача інтерфейс систем управління електростанціями є пункт, що передбачає, що вимагає термінової уваги оператора інформація повинна супроводжуватись символом червоного кольору, помітним не менше чим з двох метрів від монітора. Якщо ця вимога не була виконана, то при пропуску оператором аварійної ситуації відповідальність за те, що сталося автоматично покладається на розробника призначеного для користувача інтерфейсу. Щоб не опинитися в положенні такого розробника, при виборі візуальних атрибутів елементів інтерфейсу слід враховувати як вимоги наявних стандартів, так і фізіологічні особливості зору оператора.

7.1. Особливості візуального сприйняття інформації

Прийом візуальної інформації містить ряд елементарних процесів: виявлення, розрізнення, пізнання і декодування. На виконання цих процесів основний вплив роблять наступні характеристики зору оператора: яскравість; просторові; тимчасові; колірного сприйняття. Всі вони в значній мірі залежать від розмірів і властивостей випромінювання об'єктів, що відображуються на екрані.

Характеристики яскравості визначають розмір зони бачення об'єкту, що світиться, а також швидкість і безпомилковість обробки інформації, що світиться. Необхідно також враховувати, що необхідна гострота зору при сприйнятті світлих об'єктів в 3-4 рази нижче чим, для темних; світлі об'єкти на темному фоні виявляються легше, ніж темні на світлому.

Просторові характеристики впливають на виявлення, розрізнення і пізнання об'єктів. При вирішенні практичних завдань необхідно враховувати наступні положення:

- основну інформацію про об'єкт несе його контур; час розрізнення і пізнання контури об'єкту збільшується із збільшенням його складності.
- при розрізненні складних контурів безпомилковість вища, ніж при розрізненні простих.
- вирішальне значення в сприйнятті форми об'єктів має співвідношення «фігура/фон».
- мінімальний розмір об'єкту повинен обиратися для заданих рівнів контрасту і яскравості; зменшення значень цих параметрів вимагає збільшення кутових розмірів об'єкту.
- для підвищення вірогідності розрізнення потрібне збільшення кутових розмірів для простих фігур на 20...25%, а для знаків типа букв і цифр — в два рази.
- для розрізнення положення фігури відносно вертикальної або горизонтальної осі порогова величина виявлення має бути збільшена в 3 рази (поріг виявлення темного об'єкту на світлому фоні складає 1 кутову секунду).

За наявності на екрані рухомих об'єктів слід враховувати ряд додаткових чинників. Наприклад, при переміщенні точкового об'єкту із швидкістю

0,25 градус/с його безперервний рух сприймається як дискретний, при швидкості 0,25...4 градус/с — як безперервний, а при швидкості більше 4 градус/с зображення зливається в суцільну смугу.

Існує три види руху, що здається:

- сприйняття переміщення сигналу з одного положення в інше при послідовному пред'явленні двох ідентичних сигналів від різних об'єктів;
- зміна розмірів об'єкту, що здається, при послідовній появі двох об'єктів, що мають ідентичні контури;
- зміна розмірів об'єкту, що здається, при зміні яскравості самого об'єкту або фону.

Тимчасові характеристики. Зорове сприйняття об'єкту, що світиться, формується в людини-оператора з деякою затримкою по відношенню до початку дії зорового подразника і його припинення, що обумовлює ряд особливостей функціонування зорового аналізатора. Ці особливості виявляються як при сприйнятті одиночних світлових сигналів, так і їх послідовності. Знання тимчасових характеристик зору дозволяє обґрунтовано вибирати час експозиції сигналів для забезпечення їх мінімальної розрізнюваності і тимчасових інтервалів пред'явлення сигналів в послідовності.

Характеристики колірного сприйняття. Кольори розрізняються тоном, світлиною і насиченістю. Число помітних відтінків кольору по всьому спектру при яскравості не менше 10 кд/м2 і максимальній насиченості дорівнює приблизно 150. Розрізнення мір насиченості вагається від 4 (для жовтого) до 25 (для червоного). При ізольованому пред'явленні чоловік точно ідентифікує не більше 10-12 колірних тонів, а в комбінації з іншими кольорами - не більше восьми.

Зміна яскравості об'єкту впливає на сприйняття його кольору. Із зменшенням яскравості від 180 до 0,5 кд/м^2 відбувається зменшення світлини і поступове знебварвлення жовтого і синього кольорів, а спектр стає трибарвним: червоно-зелено-фіолетовим. Сприйняття кольору залежить також від кутових розмірів об'єкту: із зменшенням розміру змінюється видима яскравість і спотворюється колірність. До найбільшої зміни схильні жовтий і синій кольори.

Для систем реального часу основним критерієм вибору кольорів символів, що відображуються на екрані, і повідомлень є гострота розрізнення. Вона максимальна для символів білого кольору і мінімальна для символів, що мають крайні кольори спектру. Хоча білий колір найбільш простий у вживанні і його часто використовують, якнайкращим в цьому відношенні є жовто-зелений колір, який по насиченості мало відрізняється від білого, але має максимальну видимість; червоний, фіолетовий і синій кольори не рекомендується використовувати для відображення символів або об'єктів складної конфігурації.

При узгодженні кольорів символів і фону слід враховувати, що сприйняття символів максимальне для контрастних кольорів (тобто що відноситься до протилежних меж спектру). При контрастності менше 60% читаність символів різко погіршується. Встановлені наступні допустимі комбінації кольору символу з кольором фону (в порядку убування чіткості сприйняття):

- синій на білому;
- чорний на жовтому;
- зелений на білому;
- чорний на білому;
- білий на синьому;
- зелений на червоному;
- червоний на жовтому ;
- червоний на білому ;
- оранжевий на чорному;
- чорний на пурпурному;
- оранжевий на білому;
- червоний на зеленому.

При одночасному вступі двох або декількох сигналів (повідомлень) на їх сприйняття оператором впливають наступні чинники: вибірковість уваги, абсолютна і відносна інтенсивність сигналів, взаємне розташування на екрані, міра синхронності сигналів, об'єм інформації, що поступає, і швидкість її вступу.

7.2 Способи передачі смислового вмісту

Поряд з розглянутими вище характеристиками важливе значення для ефективної роботи оператора має спосіб передачі смислового вмісту інформації, що відображується на екрані. Цей спосіб може базуватися на використанні одного з чотирьох типів знакових систем (або їх комбінації) - буквеної, піктографічної, цифрової, геометричної.

При виборі знакової системи слід враховувати:

- легкість пізнання і декодування знаків;
- необхідну тривалість безпомилкової роботи оператора, у тому числі в умовах стресу;
- рівень перешкодостійкості системи;
- швидкість запам'ятовування і тривалість збереження алфавіту знакової системи в оперативній і довготривалій пам'яті оператора.

Як інтегральна характеристика знакової системи може використовуватися коефіцієнт оперативності коду, що є відношенням часу пізнання символу (знаку) до часу його декодування. У стресових ситуаціях числа до 3-розрядних включно доцільно представляти на екрані в текстовій формі (тобто словами). В той же час, основні властивості об'єкту або опис необхідних дій ефективніше відображувати у вигляді піктограми. Найбільш значимі характеристики об'єкту повинні кодуватися (відображуватися) його контуром, а внутрішніми деталями - другорядні. При цьому система пізнавальних ознак форми знаку, що обрана для певних характеристик об'єкту, повинна застосовуватися для всього алфавіту знакової системи.

При розробці знакової системи слід враховувати, що симетричні символи легше засвоюються людиною-оператором і міцніше зберігаються в короткочасній і довгостроковій пам'яті. Як розрізняльні ознаки для знаків в межах одного алфавіту не рекомендується використовувати:

- число елементів в знаку;
- геометричні розміри знаку (принаймні, більше двох варіантів);
- відзнака за принципом «позитив-негатив» і «пряме-дзеркальне відображення».

7.3 Недовантаження і перевантаження

Важливим чинником, що впливає на ефективність роботи оператора, є підтримка всіх його «підсистем» на необхідному рівні готовності протягом достатньо довгого часу. У зв'язку з цим додатково до розглянутих мають бути вирішені дві взаємозв'язані проблеми: запобігання як «сенсорного голоду», так і надмірного сенсорного перевантаження оператора.

Для боротьби з перевантаженням досить враховувати загальні рекомендації по розміщенню інформації на екрані, а також характеристики зору оператора.

«Сенсорний голод» може бути обумовлений надмірним штучним зниженням динамічності відображення поточної ситуації на екрані, а також світло - і звукоізоляцією робочого місця. У зв'язку з цим для запобігання «сенсорного голоду» має бути введена певна надмірність представленої на екрані інформації в порівнянні з мінімально необхідною. Наприклад, на екрані можуть з'являтися повідомлення, що вимагають тієї або іншої реакції оператора, але управління, що реально не впливають на процес, Інший спосіб боротьби з «сенсорним голодом» заснований на використанні декількох форм представлення інформації (тобто на вживанні мультимедійних технологій). Йдеться в першу чергу про доповнення візуальної інформації звуковим супроводом. Така організація інформації, що надається операторові, дозволяє вирішити відразу декілька проблем. По-перше, вгамувати його «сенсорний голод». По-друге, при дублюванні інформації по декількох сенсорних каналах скорочується час реакції оператора. По-третє, такий підхід дозволяє збільшити об'єм інформації, що одночасно приймається. І, нарешті, залучення додаткових сенсорних каналів дозволяє розвантажити той, по якому інформація поступає найінтенсивніше.

В той же час, вживання мультимедійних технологій значно ускладнює інтерфейс, що вимагає при його проектуванні вирішення додаткових завдань. Основна з них - узгодження інформації, що поступає по різних каналах, за часом і за змістом. При недотриманні цієї умови ефект може виявитися прямо протилежним очікуваному. При одночасному вступі декількох сигналів, що вимагають виконання різних дій, час реакції оператора збільшується; практично неминуче також інформаційне перевантаження оператора і, як наслідок, — його підвищена стомлюваність.

У системах реального часу участь оператора в процесі управління виявляється головним чином при виникненні нештатних, аварійних або критичних ситуацій. В той же час, саме такі ситуації викликають у людини дискомфортний або навіть стресовий стан. У зв'язку з цим особливе значення для СРЧ має проблема реалізації засобів підтримки користувача. Очевидно, що провідну роль тут повинні грати контекстно-залежна допомога і допомога, визначувана завданням. При цьому доцільно передбачити два способи надання допомоги: по запиту оператора і автоматично, наприклад, після закінчення деякого допустимого часу чекання реакції оператора на виниклу ситуацію. З тих же позицій слід розглядати і можливість документування дій оператора. Для повноцінного аналізу дій оператора потрібно не лише реєструвати перелік команд, що виконувалися, і значення регульованих параметрів, але і формувати знімки екрану у відповідні моменти часу. При реалізації такої можливості необхідно враховувати технічні характеристики засобів виводу () . Якщо для експрес-аналізу використовується чорно-білий друк, то це накладає додаткові обмеження па вибір колірної палітри екрану і інформації, що відображується на нім.

У зв'язку з підвищеними вимогами, що пред'являються до надійності і швидкодії систем реального часу, при їх створенні значно зростає роль етапу макетування призначеного для користувача інтерфейсу і його узгодження з потенційними користувачами.

8. Спільні положення теорії планування завдань

Планування виконання завдань — одна з ключових концепцій в багатозадачності і багатопроцесорності, як в операційних системах спільного призначення, так і в операційних системах реального часу. Планування полягає в пошуку додатків, готових до виконання, на основі властивостей цих додатків.

У СРЧ планувальник завдань повинен забезпечити відробіток процесів протягом заданих тимчасових проміжків. Це критично для підтримки коректної роботи системи реального часу. Процес є більш великомасштабним представленням завдання, оскільки позначає незалежний модуль програми або весь виконуваний файл цілком з його адресним простором, станом регістрів процесора, лічильником команд, кодом процедур і функцій та додатковими системними даними. Кожен процес містить як мінімум один потік, при

цьому максимальна кількість потоків в межах одного процесу в більшості ОС обмежено лише об'ємом оперативної пам'яті обчислювального комплексу. Потоки, що належать одному процесу, розділяють його адресний простір, тому вони можуть легко обмінюватися даними, також час перемикання між такими потоками (тобто час, за який процесор переходить від виконання команд одного потоку до виконання команд іншого) виявляється значно меншим, ніж час перемикання між процесами. У зв'язку з цим в завданнях реального часу паралельно виконувані завдання прагнуть максимально компонувати у вигляді потоків, що виконуються в межах одного процесу.

8.1. Диспетчеризація потоків

Кожен потік має важливу властивість, на підставі якої ОС приймає рішення про те, коли надати йому час процесора. Ця властивість називається пріоритетом потоку і виражається цілочисельним значенням. Кількість пріоритетів (або рівнів пріоритетів) визначається функціональними можливостями ОС. Потік може знаходитися в одному з наступних станів:

- активний потік – це той потік, який в даний момент виконується системою;
- потік в стані готовності – потік, який може виконуватися і чекає своєї черги;
- блокований потік – потік, який не може виконуватися по деяких з причин (наприклад, чекання події або звільнення потрібного ресурсу).

Методи диспетчеризації, тобто надання різним потокам доступу до процесора, в загальному випадку можуть бути розділені на дві групи.

До першої відносяться випадки, коли всі потоки, які розділяють процесор, мають однаковий пріоритет, тобто їх важливість з точки зору системи однакова:

- FIFO (First In First Out) – перший увійшов перший вийшов;
- карусельна багатозадачність (round robin)

FIFO - першою виконується завдання, яке перше увійшло до черги, при цьому воно виконується до тих пір, поки не закінчить свою роботу або не бу-

де заблокована в очікуванні звільнення деякого ресурсу або події. Після цього управління передається наступному в черзі завданню.

Карусельна багатозадачність. При цьому методі диспетчеризації в системі визначається спеціалізована константа, що визначає тривалість безперервного виконання потоку, т.з. квант часу виконання. Таким чином, виконання потоку може бути перерване або закінченням його роботи, або блокуванням в очікуванні ресурсу або події, або завершенням кванта часу. Після цього управління передається наступному в черговості потоку.

Після закінчення часу останнього потоку управління передається першому потоку, що знаходиться в стані готовності. Таким чином, виконання кожного потоку розбите на послідовність тимчасових циклів.

Поява другої групи методів диспетчеризації пов'язана з необхідністю розподілу часу процесора між потоками, що мають різну важливість, тобто різний пріоритет. У таких випадках для потоків з рівним пріоритетом використовується один з вказаних вище методів диспетчеризації, а передача управління між різно пріоритетними потоками здійснюється одним з наступних методів:

- якщо в стан готовності переходять два потоки з різними пріоритетами, то процесорний час передається тому, в якого вищий пріоритет;
- адаптивна багатозадачність;
- витісняюча пріоритетна багатозадачність.

Перший метод називається пріоритетною багатозадачністю, але його використання у такому вигляді пов'язане з рядом складнощів. За наявності в системі однієї групи потоків з одним пріоритетом і іншої групи з іншим, нижчим пріоритетом, при карусельній диспетчеризації кожної групи в системі з пріоритетною багатозадачністю потоки низькопріоритетної групи можуть взагалі не дістати доступу до процесора.

Адаптивна багатозадачність широко застосовується в інтерфейсних системах. Суть цього методу полягає в тому, що пріоритет потоку, що не виконується деякий період часу підвищується на одиницю. Відновлення вихідного пріоритету відбувається після виконання потоку в перебігу одного кванта часу або при блокуванні потоку. Таким чином, при карусельній багатозадач-

ності, черга (або «карусель») пріоритетніших потоків не може повністю заблокувати виконання черги менш пріоритетних потоків.

У завданнях реального часу пред'являються специфічні вимоги до методів диспетчеризації, оскільки передача управління потоку повинна визначатися критичним терміном його обслуговування (т.з. deadline-driven scheduling). Найбільшою мірою цій вимозі відповідає витісняюча пріоритетна багатозадачність. Суть цього методу полягає в тому, що як тільки потік з вищим, ніж в активного потоку, пріоритетом переходить в стан готовності, активний потік витісняється (тобто з активного стану примусово переходить в стан готовності) і управління передається пріоритетнішому потоку.

На практиці широко застосовуються як комбінації описаних методів, так і різні їх модифікації. У СРЧ, в контексті завдання диспетчеризації декілька різнопріоритетних потоків, дуже важливою є проблема розподілу пріоритетів так, щоб кожен потік уклався в свій термін критичного обслуговування. Якщо всі потоки системи укладаються в свої терміни критичного обслуговування, то говорять, що система диспетчеризуєма .

8.2 Алгоритми планування

У зв'язку з вище згаданими проблемами планування в ОСРЧ використовуються два підходи до планування – статичні алгоритми планування (RMS – Rate Monotonic Scheduling) і динамічні алгоритми планування (EDF – Earliest Deadline First) [ ].

RMS використовується для формального доказу умов передбачуваності системи. Для реалізації цієї теорії необхідне планерування на основі пріоритетів, що переривають виконання поточного завдання (preemptive priority scheduling). У теорії RMS пріоритет заздалегідь призначається кожному процесу (потоку). Процеси (потоки) повинні задовольняти наступним умовам:

- процес має бути завершений за час його періоду;
- процеси не залежать один від одного;
- кожному процесу потрібний однаковий процесорний час на кожному інтервалі;
- в неперіодичних процесів немає жорстких термінів;
- переривання процесу відбувається за обмежений час.

Процеси виконуються відповідно до пріоритетів. При плануванні RMS перевага віддається завданням з найкоротшими періодами виконання.

У EDF пріоритет привласнюється динамічно, і найбільший пріоритет виставляється процесу, в якого залишився найменший час виконання. При великих завантаженнях системи в EDF є переваги перед RMS.

Зазвичай в ОСРЧ використовується планування з пріоритетами, що переривають виконання поточного завдання, яке засноване на RMS. Пріоритетне переривання виконання є невід'ємній складовій ОСРЧ, оскільки в системі реального часу повинні існувати гарантії того, що подія з високим пріоритетом буде оброблена перед подією нижчого пріоритету. Все це веде до того, що ОСРЧ потребує не лише механізму планування на основі пріоритетів, що переривають обслуговування, але також і у необхідній кількості рівнів пріоритетів, оскільки для організації паралельного виконання декількох потоків частенько необхідне розділення цих потоків по мірі важливості (або критичному терміну обслуговування). Серед сукупності завдань, що паралельно виконуються, виділяються потоки жорсткого реального часу, потоки м'якого реального часу і потоки, не критичні до часу обслуговування. Кожна з вказаних груп повинна мати свій рівень пріоритетів, до того ж потоки жорсткого реального часу, у ряді випадків, повинні мати індивідуальні значення пріоритетів. Практичний досвід розробки систем реального часу говорить, що збільшення кількості різнопріоритетних потоків приводить до непрозорості і непередбачуваності системи [ ].

ОСРЧ повинна володіти розвиненою системою пріоритетів. По-перше, це потрібно тому, що система сама може розглядатися як набір серверних застосувань, що підрозділяються на потоки, і декілька високих рівнів пріоритетів має бути виділене системним процесам і потокам. По-друге, в складних застосуваннях необхідно всі потоки реального часу поміщати на різні пріоритетні рівні, а потоки не реального часу поміщати на один рівень (нижче, ніж будь-які потоки реального часу). При цьому потоки не реального часу можна обробляти в режимі циклічного планерування (RRS – round-robin scheduling), при якому кожному процесу надається квант часу процесора, а коли квант закінчується, контекст процесу зберігається, і він ставиться в кінець черги. У багатьох ОСРЧ для планування завдань на одному рівні використовується RRS. Пріоритетний рівень 0 зазвичай використовується для холостого режиму.

При плануванні на основі пріоритетів необхідно вирішити дві обов'язкові проблеми:

- забезпечити виконання процесу з найвищим пріоритетом;
- не допустити інверсії пріоритетів, коли завдання з високими пріоритетами чекають ресурси, захоплені завданнями з нижчими пріоритетами.

Для боротьби з інверсією пріоритетів в ОСРЧ часто використовується механізм спадкоємства пріоритетів, проте при цьому доводиться відмовлятися від планування на основі RMS, оскільки пріоритети стають динамічними.

На сьогоднішній день існує ряд інструментів математичного аналізу, що дозволяють розподілити пріоритети між декількома потоками так, що б вони гарантовано виконувалися в свої критичні терміни обслуговування. Якщо ж для даного набору потоків це неможливо, то результати математичного аналізу покажуть, які саме потоки мають критичне відношення терміну обслуговування до часу виконання. Математичний апарат дозволяє провести таке дослідження для випадку періодичних критичних часів обслуговування. Проте для його вживання аналізовані потоки повинні мати унікальні значення пріоритетів, визначувані періодом кожного потоку. У зв'язку з цією вимогою розробники ОСРЧ закладають в своїх системах чималу кількість пріоритетів. Для прикладу в QNX 6.x їх 64, а в VxWorks – 256 [ ].

Таким чином, ОСРЧ повинна мати чималу (визначається масштабом завдання) кількість пріоритетів (рекомендується до 128 значень). Природно, що в прикладних завданнях необхідно украй обережно використовувати потоки з різними пріоритетами і, по можливості прагнути до мінімізації їх кількості. Велика кількість різнопріоритетних потоків може привести не лише до втрати передбачуваності системи, але і до проблем синхронізації при доступі до ресурсів, що розділяються.

8.3. Моделі захисту пам'яті

Перемикання між потоками є часозатратною операцією і залежить від конфігурації пам'яті, тобто від моделі захисту пам'яті. Розглянемо чотири найбільш поширених в ОСРЧ моделі захисту пам'яті.

- модель без захисту
- модель захисту «система/користувач»;

- модель захисту «користувач/користувач»;
- модель захисту віртуальної пам’яті.

Перша модель - адресні простори системи та користувача не захищені один від одного, використовується два сегменти пам'яті: для коду і даних; при цьому від системи не вимагається жодного управління пам'яттю, не потрібний MMU (memory management unit – спеціальний апаратний пристрій для підтримки управління віртуальною пам'яттю).

Модель захисту система/користувач – системний адресний простір захищений від адресного простору користувача. Системні і призначені для користувача процеси виконуються загалом в віртуальному адресному просторі, при цьому потрібний MMU. Захист забезпечується сторінковим механізмом захисту. Розрізняються системні і призначені для користувача сторінки. Призначені для користувача застосування ніяк не захищені один від одного. Процесор знаходиться в режимі супервізора, якщо поточний сегмент має рівень 0, 1 або 2. Якщо рівень сегменту – 3, то процесор знаходиться в призначеному для користувача режимі. У цій моделі необхідно чотири сегменти – два сегменти на рівні 0 (для коду і даних) і два сегменти на рівні 3. Механізм сторінкового захисту не додає накладних витрат, оскільки захист перевіряється одночасно з перетворенням адреси, яке виконує MMU; при цьому ОС не потребує управління пам'яттю.

Модель захисту користувач/користувач –к моделі система/користувач додається захист між призначеними для користувача процесами; потрібний MMU. Як і в попередній моделі, використовується механізм сторінкового захисту. Всі сторінки позначаються як привілейовані, за винятком сторінок поточного процесу, які позначаються як призначені для користувача. Таким чином, потік, що виконується, не може звернутися за межі свого адресного простору. ОС відповідає за оновлення прапора привілейованої для конкретної сторінки в таблиці сторінок при перемиканні процесу. Як і в попередній моделі використовуються чотири сегменти.

Модель захисту віртуальної пам'яті – кожен процес виконується в своїй власній віртуальній пам'яті, потрібний MMU. В кожного процесу є свої власні сегменти і, отже, своя таблиця описувачів. ОС несе відповідальність за підтримку таблиць описувачів. Простір, що адресується, може перевищувати

розміри фізичної пам'яті, якщо використовується сторінкова організація пам'яті спільно з підкачуванням. Проте в системах реального часу підкачування зазвичай не застосовується із-за її непередбачуваності. Для вирішення цієї проблеми доступна пам'ять розбивається на фіксоване число логічних адресних просторів рівного розміру. Число процесів, що одночасно виконуються, в системі стає обмеженим.

Фундаментальна вимога до пам'яті в системі реального часу полягає в тому, що час доступу до неї має бути обмеженим (або, іншими словами, передбачуваним). Прямим слідством стає заборона на використання для процесів реального часу техніки виклику сторінок за запитом (підкочування з диска). Тому системи, що забезпечують механізм віртуальної пам'яті, повинні вміти блокувати процес в оперативній пам'яті, не допускаючи підкачування.

Якщо підтримується сторінкова організація пам'яті (paging), відповідне відображення сторінок у фізичні адреси має бути частиною контексту процесу. Інакше знову з'являється непередбачуваність, неприйнятна для ОСРЧ.

Для процесів, що не є процесами жорсткого реального часу, можливе використання механізму динамічного розподілу пам'яті, проте при цьому ОСРЧ повинна підтримувати обробку тайм-ауту на запит пам'яті, тобто обмеження на передбачений час чекання.

У звичайних ОС при використанні механізму сегментації пам'яті для боротьби з фрагментацією застосовується процедура ущільнення після збірки сміття. Проте такий підхід непридатний в середі реального часу, оскільки під час ущільнення переміщувані завдання не можуть виконуватися, що веде до непередбачуваності системи. У системах жорсткого реального часу зазвичай застосовується статичний розподіл пам'яті.

9. Планування періодичних завдань

Для СРЧ, що застосовуються в обробці періодичних подій, в 1970 році Ліу і Лейленд запропонували математичний апарат, що дозволяє визначити, чи є система диспетчеризуємою [ ]. Цей апарат називається «Частотно монотонний аналіз» (ЧМА) (Rate Monotonic Analyzing).

Ефективність цього математичного апарату привела до того, що ЧМА був прийнятий як стандарт такими організаціями як USA Department of Defense, Boeing, General Dynamics, Magnavox, Mitre, NASA, Panamax, і ін. Також серед організацій, ЧМА, що встановили, як стандартний засіб аналізу і розробки систем жорсткого реального часу можна відзначити IBM Federal Sector Division, US Navy і European Space Agency [7].

9.1. Теорія частотно-монотонного аналізу

Теорія частотно-монотонного аналізу (ЧМА) з'явилася в контексті диспетчеризації завдань в системі, де кінцева кількість періодичних завдань спільно використовує єдиний процесор. Теорія ЧМА забезпечує правила, за якими відбувається аналіз, чи дійсно дані завдання можуть бути диспетчеризуєми згідно їх характеристикам.

Планування завдань в системах реального часу істотно відрізняється від традиційних форм планування. Це означає, що терміновіші завдання повинні отримувати пріоритет у виконанні в порівнянні зі всіма останніми завданнями, що працюють в системі не лише в той момент, коли потрібно вибрати чергове завдання із списку завдань, але також і тоді, коли одне із завдань вже виконується, але новий запит поступає від завдання з вищим пріоритетом. У відзнаці від стандартного значення пріоритету (рівня важливості привласненого об'єкту), термін пріоритет використовується тут як рівень терміновості, привласнений завданню.

Ситуація, в якій термінове завдання видаляє завдання, що виконується в даний час, відома як витіснення.

Розуміння витіснення є вельми важливим для розуміння того, як можна скоротити час реакції системи на зовнішні події. Система, в якій потрібне завершення виконання поточного завдання, перш ніж може початися виконання інший, може опинитися значно повільніше за систему, в якій поточне завдання може бути витиснена, визволяючи процесор для терміновішого завдання, яке запитало доступу до формального закінчення роботи поточного завдання. Ця особливість визначає основну властивість системи реального часу – реактивність, що означає, як швидко після настання події система зможе почати обробку цієї події.

В цілому, реактивність системи може бути визначена як властивість, що характеризує час, який протікає від виникнення специфічної події до початку його обробки в найгіршому випадку. Ця властивість визначає, як швидко система реагує на події - тобто, як швидко вона може сприймати їх - і є одним з компонентів повного часу реакції системи.

На додаток до реактивності, важливою властивістю системи є своєчасність - або як швидко (тобто як своєчасно) вже отримані події можуть бути оброблені. Своєчасність - властивість, яка характеризується найгіршим часом обробки вже отриманої події.

Розглянемо ситуацію, коли є три завдання і лише два рівні пріоритету, як показано в Таблиці 9.1.

Таблиця 9.1

Ілюстрація проблеми, пов'язаної з недостатньою кількістю рівнів пріоритету:

Завдання і	Пріоритет	Тс	Ткрит
1	2	2	10
2	1	6	20
3	1	3	6

де: Тс – час виконання завдання; Ткрит – критичний час обслуговування (максимальний час чекання виклику завдання, після якого вона не може бути виконана).

Хай завдання 1 і 2 повинні запускатися раніше, ніж завдання 3. В цьому випадку, згідного того, що в завдання 1 пріоритет вище, управління отримує завдання 1. У момент часу 1 завдання 3 не може отримати управління, оскільки його пріоритет нижчий, ніж в активного процесу .По його закінченню (момент часи 2) запускається завдання 2, а не завдання 3, хоча у них і рівні пріоритети, але час чекання завдання 2 більше, ніж завдання 3. У результаті, завдання 3 не отримує управління взагалі, оскільки перевищений для неї параметр крайнього терміну виконання.

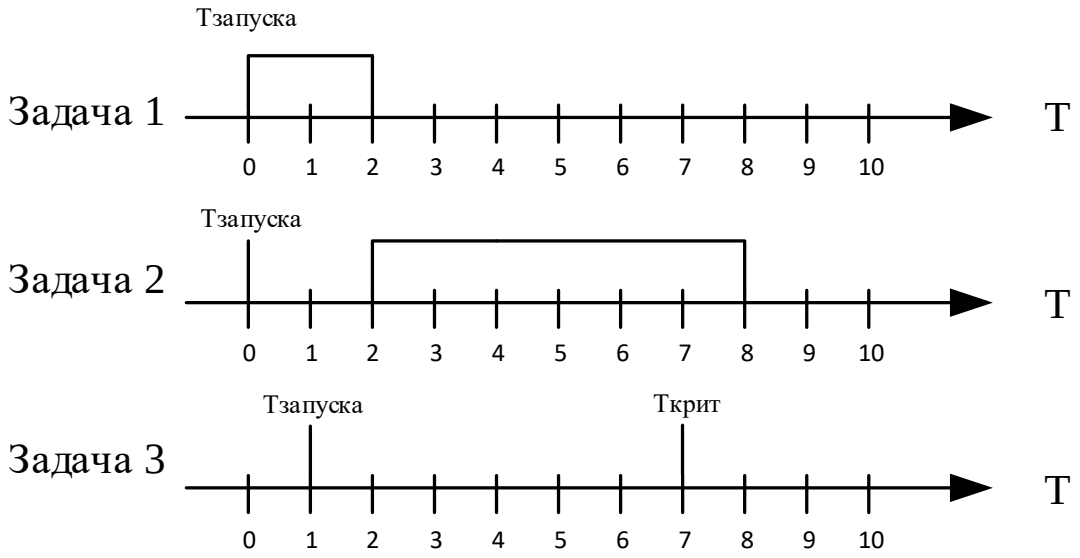

Рисунок 9.1 - Ілюстрація недостатньої кількості рівнів пріоритету

Крім того, лише з двома рівнями пріоритету, може бути важко знайти значення пріоритету, яке вирішує проблему всіх крайніх термінів за гірших умов прибуття. Наприклад, якщо завдання 3 буде має найвищий пріоритет (рівний 2), а завдання 2 і 1 мають низький пріоритет (рівний 1), то запуск завдань 3 і 2 трохи раніше, ніж завдання 1 фактично не дасть можливості їй виконається в її крайній термін (просто, бо час виконання завдання 3 + завдання 2 складає 9 одиниць, і завдання 1 не зможе виконати свої 2 одиниці до крайнього терміну, який дорівнює 10).

Призначення пріоритету рівним 2 лише до завдань 2 і 3 вирішує проблему, навіть якщо всі завдання, що виконуються періодично, з періодами рівними крайнім термінам. Може допомогти надання додаткового рівня пріоритету. Це пояснює, чому призначення пріоритетів настільки важливе в системах реального часу. Якщо кількість рівнів пріоритету достатня, природно буде призначати пріоритет на підставі важливості завдання. Приклад 2 показує, чому пріоритети, засновані на важливості завдань можуть не працювати.

Приклад 2. Всі завдання в Таблиці 9.2 прибувають приблизно в той же самий час і виконуються лише однин раз. Якщо Завдання 1 прибуває першим, воно виконується 6 одиниць, і фактично не дозволяє (із-за свого вищого пріоритету) іншим двом завданням дістати доступ до процесора. Таким чином, з такими величинами пріоритету, інші завдання не можуть виконатися в їх критичні терміни. Проте, якщо привласнити найнижчий пріоритет найважливішому завданню, тоді всі завдання укладаються в їх критичні терміни обслуговування.

Таблиця 9.2

Ілюстрація призначення пріоритету, заснованого на важливості завдання

Завдання і	Пріоритет (важливість)	Тс	Ткрит
1	Високий	6	10
2	Середній	2	5
3	Низький	2	4

Теорія частотно-монотонного аналізу пропонує критерій для призначення пріоритету для взаємно незалежних завдань, що періодично виконуються. Основний принцип частотно-монотонного аналізу може бути виражений таким чином: чим коротше період завдання, тим вище її пріоритет.

Отже, цей алгоритм названий частотно-монотонною диспетчеризацією, бо він призначає пріоритети завданням на основі монотонної функції їх періодів (частот) .

Ключовим поняттям, що лежить в основі теорії ЧМА, є параметр використання процесора, названий коефіцієнтом (чинником) використання. Коефіцієнт використання визначається відношенням між часом використання ресурсу і спільним часом доступності ресурсу. Цей коефіцієнт може бути розрахований індивідуально для кожного об'єкту, що конкурує за ресурс або кумулятивний, як сума для всіх об'єктів, що використовують ресурс. Проте, необхідно відзначити, що поняття коефіцієнта використання не універсальне, і система з низьким коефіцієнтом використання не може гарантувати диспетчеризуємість, тоді як завдання в іншій системі, з коефіцієнтом використання рівним 1 (тобто в системі, в якій використання досягає 100 відсотків) все ще можуть бути такими, що диспетчеризуються.

Найявнішим прикладом погано розробленої в цьому відношенні системи, буде система, в якій коефіцієнт використання процесора більше 1. Така система досягає насиченості, і не кожен модуль може дістати доступ до процесора тоді, коли це необхідно. Коефіцієнт використання має бути менше або рівний 1, щоб гарантувати своєчасний доступ до процесора для кожного модуля.

Вищезазначена вимога про те, що коефіцієнт використання має бути менше або рівним 1, є необхідною вимогою диспетчеризуємості системи. Для призначення пріоритетів на основі періодів завдань нетривіальна достатня умова диспетчеризуємості була введена алгоритмом частотно-монотонної диспетчеризації (ЧМД) [ ].

9.2 Частотно-монотонна диспетчеризація. Теорема 1.

Для безлічі N незалежних періодичних завдань, де Ci і Ti = 1,2.. N, є часом виконання і періодом, відповідно, і виходячи з того, що критичний термін виконання завдання дорівнює його періоду, завдання диспетчеризуємо на підставі частотно-монотонної диспетчеризації, якщо виконується наступна умова [ ]:

$$\sum_{i=1}^{n} C_i / T_i \le N\left(2^{1/N} - 1\right) \tag{9.1}$$

Ця теорема говорить, що, якщо вищезазначена умова виконана, то всі завдання завершаться вчасно (до їх крайнього терміну виконання, який дорівнює їх періоду).

Значення $N(2^{1/N}-1)$ прагне до Ln 2 ≈ 0.693 при N, прагнучому до нескінченності. Так само, якщо сума використань менше ніж деяка верхня межа, то всі конкретні завдання (цілий набір завдань) можуть бути диспетчеризовані на підставі ЧМА із завершенням всіх завдань, що гарантується, до їх критичного терміну виконання (якщо відповідні припущення виконані). Приклади 3 і 4 ілюструють значення Теореми 1.

Приклад 3. У Таблиці 9.3 представлений набір з шести завдань, що характеризуються часом виконання Тс і періодом Тп. Згідно Теоремі 1, сукупний коефіцієнт використання процесора, сума Тс/Тп, має бути менше ніж верхня межа $N(2^{1/N}-1)$ для гарантії диспетчируємості на підставі ЧМА. З таблиці видно, що набір перших чотирьох задач (або менш) – диспетчеризуєм на підставі ЧМА, бо коефіцієнт використання менше верхньої межі. Проте, набір зі всіх шести завдань не може бути диспетчеризуєм на підставі алгоритму ЧМД, бо для двох завдань, що залишилися, коефіцієнт використання більше верхньої межі (0.812> 0.743 і 0.892> 0.735 відповідно).

Таблиця 9.3

Характеристики шести завдань, що наведені в Прикладі 3

Завдання i	Tc	Tп	Tc/Tп	Коефіцієнт використання	Межа
1	3	10	0.300	0.300	1.000
2	4	20	0.200	0.500	0.828
3	5	40	0.125	0.625	0.779
4	6	60	0.100	0.725	0.756
5	7	80	0.087	0.812	0.743
6	8	100	0.080	0.892	0.735

Приклад 4. У Таблиці 9.4, представлений інший набір з семи завдань. Перші шість завдань диспетчеризуємі згідно Теоремі 1; проте, при додаванні завдання 7 виникає проблема, при чому не поважно, що завдання в таблиці не впорядковані згідно величині їх періодів. При найближчому розгляді видно, що найбільший вклад до коефіцієнта сукупного використання центрального процесора (0.250 і 0.200) вносять завдання 6 і 7, відповідно. Проектувальник може перевірити можливість скорочення часу виконання завдання 6 або збільшення періоду завдання 7, щоб змусити їх вписатися в схему і таким чином усунути проблему. Наприклад, збільшення періоду завдання 7 до Tc = 100 вирішило б проблему (бо 0.653 <0.724).

Таблиця 9.4

Характеристики семи завдань, що наведені в Прикладі 4

Завдання i	Tc	Tп	Tc/Tп	Коефіцієнт використання	Межа
1	10	500	0.020	0.020	1.000
2	15	300	0.050	0.070	0.828
3	12	100	0.120	0.190	0.779
4	5	150	0.033	0.223	0.756
5	20	200	0.100	0.323	0.743
6	50	200	0.250	0.573	0.735
7	8	40	0.200	0.773	0.724

У двох наведених вище прикладах Теорема 1 не виконується. Строго кажучи, Теорема 1 не визначає необхідну умову. Якщо нерівність цієї теореми не виконується, то не можна сказати, чи дійсно набір завдань не диспетчеризуєм. Необхідний деякий додатковий критерій, щоб визначити диспетчеризуємість в цьому випадку.

9.3 Теорема про час завершення. Теорема2.

Для набору N незалежних періодичних завдань, де Ci і Ti, i = 1, 2 ., N, є їх часом виконання і періодом відповідно і, приймаючи, що критичний термін виконання завдання дорівнює періоду завдання, завдання диспетчеризуєми на основі ЧМА (тобто вони завжди виконуватимуться в їх критичний термін), тоді і лише тоді, коли виконується наступна умова [ ]:

$$\forall \min \sum_{j=1}^{i} \frac{C_j}{lT_{jk}} \left| lT_k / T_j \right\rangle \le 1 \tag{9.2}$$

де мінімум розрахований як, і

$W_i = \left\{ (k,l), 1 \le k \le i, l = 1, ..., \left\langle T_i / T_k \right| \right\}$

Позначення і означають найбільше ціле число, менше ніж або рівне вираженню і найменше цілому числу, більше або рівне вираженню – тобто функції округлення до цілого вгору і вниз – відповідно.

Приклад 5. З метою ілюстрації практичного застосування Теореми 2, проаналізуємо диспетчеризуємість завдань, представлених в Таблиці 9.5. Ясно, що ці три завдання - не диспетчеризуєми згідно ЧМА Теоремі 1, бо коефіцієнт сукупного використання центрального процесора для цих трьох завдань більш ніж верхня межа (0.800> 0.779).

Таблиця 9.5

Характеристики трьох завдань, що наведені в Прикладі 5

Завдання і	Тс	Тп	Тс/Тп	Коефіцієнт використання	Межа
1	25	50	0.200	0.200	1.000
2	25	100	0.400	0.600	0.828
3	50	150	0.300	0.800	0.779

Щоб перевірити диспетчеризуємість цих трьох завдань з Таблиці 9.5, необхідно використовувати рівняння Теореми 2, лише для i = 3, бо перші два завдання диспетчеризируємі згідно Теоремі 1.

Таким чином, один з чотирьох індексів в цьому рівнянні є фіксований (i = 3). При розрахунку відповідних сум, інший індекс, j, змінюється від 1 до 3. Індекс до повинен змінитися від 1 до i = 3, і для кожного до, індекс, l повинен змінити від 1 до 3.

Таким чином, значення всіх чотирьох індексів задаються таким чином:

i=3

j=1 3

k=1 ... 3

l=1 . . .

Далі:

k=1: = = 2; де, l = 1 ., 2

k=2: = = 2; де, l = 1 ., 2

k=3: = = 1; де, l = 1.

Таблиця 9.6

Нерівності, отримані на основі Теореми 2

K	*l*	*A*
1	1	>1
2	1	>1
3	1	>1
1	2	≤1
2	2	>1

де: *k, l* – значення індексів, а A – значення лівої частини нерівності 9.2.

З вищезгаданого ясно, що, для *k* = 1 і *l* = 2, умова Теореми 2 виконується. Таким чином, набір трьох завдань з Таблиці 9.6 диспетчеризуєм на підставі алгоритму ЧМД.

Суть теореми полягає в тому, що для визначення диспетчеризуємості, досить і необхідно перевірити, чи може кожне завдання завершити своє виконання перед першим критичним терміном. Технічно, щоб здійснити це, необхідно перевірити множину нерівностей для всіх можливих точок диспетчеризації для кожного залученого завдання. Де точки диспетчеризації – це всі моменти часу, в які період будь-якого завдання закінчується. Ця перевірка має бути зроблена для всіх точок диспетчеризація, які відносяться до першого періоду кожного завдання, бо гарантується, що, якщо завдання виконується в її перший критичний термін, вона завжди виконуватиметься у всі інші критичні терміни.

10. Планування асинхронних завдань

СРЧ оперує двома типами завдань: періодичними і аперіодичними. Тому для будь-якої теорії диспетчеризації важливо розглядати неперіодичні події і їх події. Проте, складність аналізу викликана тим, що для більшості неперіодичних подій, частота їх появи є необмеженою; тобто вони можуть поступати пакетами. Тому необхідно робити деякі припущення про моменти вступу запитів на виконання завдань, якщо вони є неперіодичними.

Якщо є як періодичні, так і аперіодичні завдання, то алгоритм монотонних частот необхідно узагальнити. Аперіодичне завдання активізується у випадкові моменти часу (наприклад, моменти виникнення подій від зовнішніх компонент СРЧ) і виконується протягом деякого проміжку часу Т.

Виконання завдання повинно бути завершено до наступної події активізації. Тому, аперіодичне завдання може бути розглянута як періодичне з періодом, рівним мінімальному часу між виникненням подій, які активізують виконання аперіодичного завдання.

Дане припущення знайшло своє відображення в алгоритмі відомому під назвою «алгоритм спорадичного сервера» . Згідно цього алгоритму кожному аперіодичному завданню виділяється квота процесорного часу, яка може бути використана у будь-який момент впродовж умовного періоду.

Таким чином, аперіодичним завданням допустимо призначати різні рівні пріоритетів відповідно до еквівалентних умовних періодів і розглядати їх як періодичні.

При проектуванні СРЧ пріоритети завданням обох видів необхідно призначати (по можливості) відповідно до алгоритмів монотонних частот. Простіше за всього його застосовувати до періодичних завдань. Для аперіодичних потрібно оцінити час між подіями у гіршому разі і призначити їм частотно-монотонні пріоритети, але вище, ніж в періодичних. Вищий пріоритет слід призначати завданню, враховуючи критерій його важливості в системі.

Алгоритм починається з використання Теореми 1 і, якщо всі умови виконані (вершина 2), алгоритм завершується видачею відповідного повідомлення (вершина 6).

Інакше визначаються всі точки диспетчеризації, починаючи з часу 0 до кінця першого періоду самих низькочастотних завдань (вершина 3).

Для всіх точок диспетчеризації, знайдених у вершині 3, створюються нерівності, які мають зліва суму всіх можливих часів виконання завдань, які можуть бути активізовані перед цією точкою диспетчеризації, а справа лише значення часу, відповідне цій точці диспетчеризації (вершина 4).

Перевіряється, чи є значення зліва меншими або рівними відповідним ним значенням справа. Якщо хоч би одно з цих нерівностей виконується, то набір завдань диспетчеризується згідно другій теоремі ЧМА. Інакше набір завдань не диспетчеризуєм згідно ЧМА (вершини 5-7).

10.1. Алгоритм перевірки на диспетчеризуємість за допомогою ЧМА

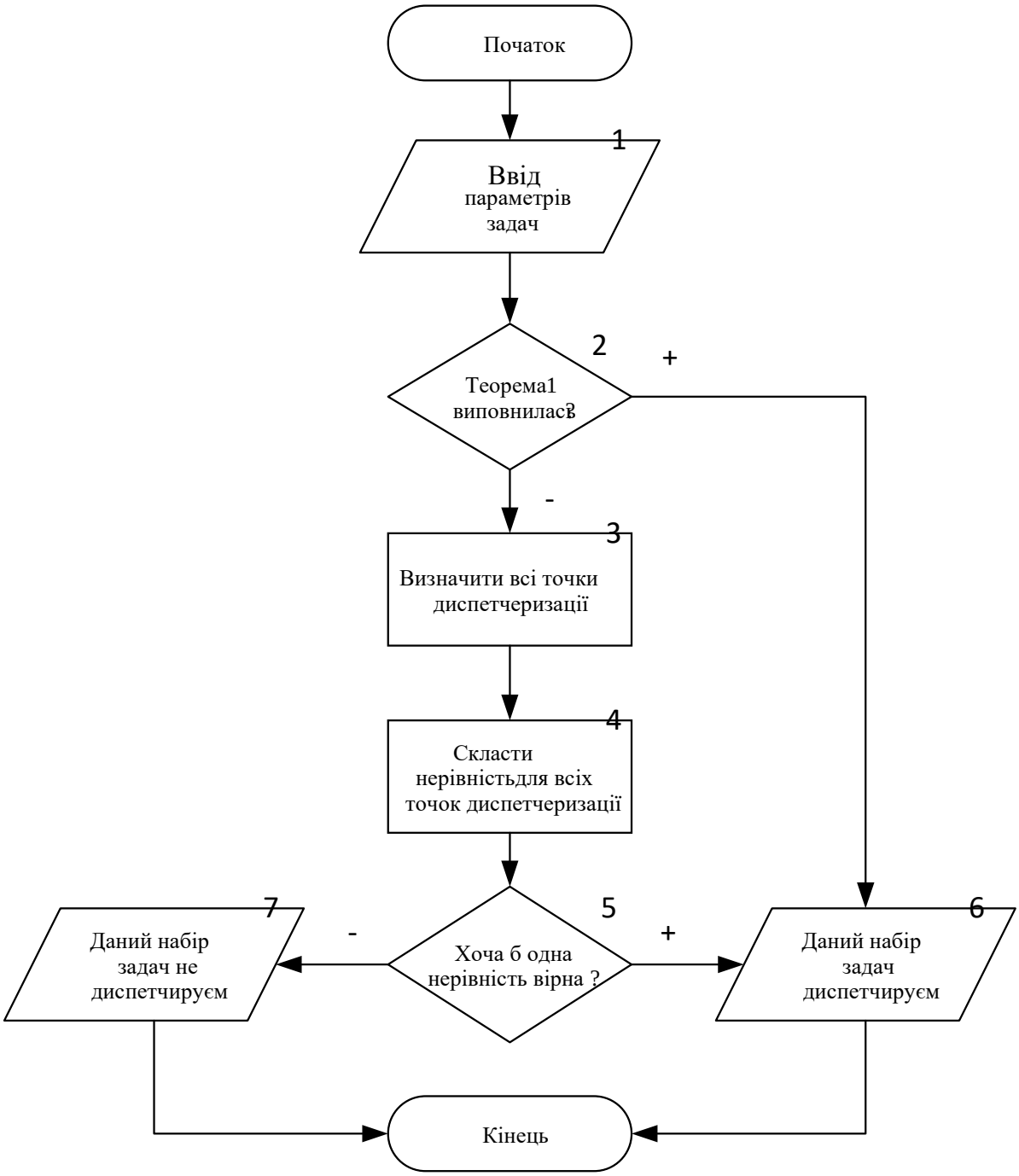

Рисунок 10.1 - Алгоритм перевірки на диспетчеризуємість за допомогою ЧМА

Приклад 1. Проаналізуємо набір завдань з характеристиками, представленими в Таблиці 10.1.

Таблиця 10.1

Набор завдань з коефіцієнтом використання рівним 1

Завдання і	Тс	Тп	Тс/Тп	Коефіцієнт використання	Межа
1	1	4	0.25	0.25	1.000
2	2	5	0.40	0.65	0.828
3	7	20	0.35	1.00	0.779

Перші два завдання - диспетчеризуємо згідно ЧМА Теоремі 1, але третє завдання – ні. Тому, застосуємо Теорему 2.

Визначимо всі точки диспетчеризації з часу 0 до 20 (кінець періоду завдання найнижчої частоти). Точки диспетчеризації :

для Завдання 1: 0, 4, 8, 12, 16, 20
для Завдання 2: 0, 5, 10, 15, 20
для Завдання 3: 0, 20

На осі часу, ці точки диспетчеризації з'являються в порядку, представленому на Рисунку 10.2.

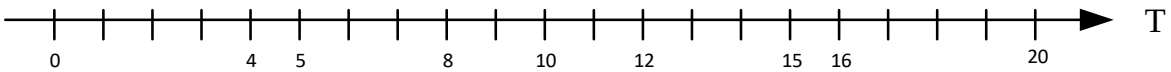

Рисунок 10.2 - Точки диспетчеризації для трьох завдань з Прикладу 1.

Опишемо нерівності для всіх точок диспетчеризації. Нерівності:

C1 + C2 + C3 ≤T1
2 C1 + C2 + C3 ≤T2
2 C1 + 2 C2 + C3≤2*T1
3 C1 + 2 C2 + C3 ≤2 *2
3 C1 + 3 C2 + C3 ≤3*T1
4 C1 + 3 C2 + C3 ≤3*T2
4 C1 + 4 C2 + C3 ≤4* T1
5 C1 + 4 C2 + C3 ≤T3

Замінимо фактичними значеннями всі змінні і перевіримо чи виконується хоч би одна з нерівностей.

1+2+7>4

2*1+2+7>5

2*1 + 2*2+7>2*4

3*1 + 2*2+7>2*5

3*1 + 3*2+7>3*4

4*1 + 3*2+7>3*5

4*1 + 4*2+7>4*4

5*1 + 4*2+7=20

Виходячи з розрахунків набір завдань з Таблиці 10.1 є таким, що диспетчеризується, згідно Теоремі 2, бо одна з нерівностей для третього завдання виконується.

10.1. Типи асинхронних подій і стратегії їх обробки

Основною відзнакою в організації обробки проблемних переривань є відсутність планування моменту виклику завдання на виконання. Момент виклику визначається відповідно до настання деякої події в середовищі зовнішніх компонент. Такі події характеризують виникнення якісних змін станів в сукупності зовнішніх компонент зв'язаного об'єкту.

У складі СРЧ організація обробки проблемних переривань повинна забезпечувати особливі властивості для завдань їх обробки. Основною властивістю є мінімізація затримки часу між моментами настання подій і початком виконання завдань, призначених для обробки даних подій. Моментом планового виклику завдання обробки проблемного переривання вважається момент настання події – надходження сигналу проблемного переривання.

У середовищі обчислювача СРЧ проблемні події представляються у вигляді сигналів переривання. Дані сигнали поступають з боку зовнішніх компонент, які викликають переривання поточних обчислювальних процесів і забезпечують введення сигналів про проблемні події в апаратно-програмне середовище обчислювача СРЧ. У середовищі обчислювача відповідно до таких сигналів повинні викликатися спеціально призначені проблемні завдан-

ня. Інформаційні канали, що забезпечують введення і обробку в обчислювачі сигналів проблемних переривань, називаються ініціативними.

В більшості випадків множина параметрів необхідних для обробки періодичних подій по завершенню інтервалів часу повністю покривають множину параметрів необхідне для характеристики завдань обробки проблемних переривань. По відношенню до завдань даного типа супервізор виконує функції управління викликів на виконання і контролю достатків завдань відповідно до поточної зміни параметра внутрішнього часу системи і значень параметрів, що характеризують особливості кожного з завдань обробки.

Залежно від особливостей, що ініціюють проблемну обробку подій, завданням обробки можуть призначатися різні способи управління їх викликом на виконання. Відповідно до призначених способів управління викликом завдань, супервізором реалізується різні стратегії і алгоритми забезпечення даної функції. Основних стратегій в організації управління викликом завдань проблемної обробки виділяють дві: негайна обробка; приведена обробка.

Негайна обробка забезпечує мінімальні затримки за часом між подією, що ініціює, і викликом завдання його проблемної обробки. Використання стратегії негайної обробки накладає ряд обмежень на реалізацію програмної обробки подій по даній стратегії. Наявність і особливості обмежень обґрунтовуються необхідністю виклику на виконання завдання обробки по сигналу переривання. Таким чином, до таких завдань повинні пред'являтися вимоги, як до завдань обробки переривань. Інтервали виконання саме такий завдань і приводять до наявності інтервалів переривання роботи завдань обробки інтервалів часу, тобто до відмінності значень $t_{роб}$ і $t_{вик}$.

Завдання негайної обробки є в системі найбільш пріоритетними і викликаються з перериванням поточних процесів обчислювача.

Основними вимогам до програмно-алгоритмічній реалізації завдань негайної обробки є:

- мінімальні інтервали часу виконання програми;
- заборона звернень до системних або призначених для користувача функцій, які не володіють властивістю реєнтерабельності.

Перша вимога забезпечує мінімізацію затримок за часом, а тим самим і порушень параметрів РЧ по відношенню до всієї множини інших завдань обробки подій РВ.

Друга вимога захищає програмне середовище від непередбачуваних наслідків з повною втратою достовірності отримуваних результатів рішення єдиної задачі. Відповідно до цієї вимоги істотно обмежуються функціональні можливості виконання алгоритмічної обробки і взаємодії з користувачем і периферійним устаткуванням в середовищі завдання негайної обробки.

У зв'язку з необхідністю виконання вказаних вимог, при програмуванні завдань негайної обробки в більшості випадків обмежуються мінімальними об'ємами обчислень. Як правило, основою таких обчислень є дії із збереження значень моментів настання подій або ряду інших параметрів, які характеризують стан на момент настання події. Всі ці значення передаються для подальшої обробки. При цьому повна обробка подій виконується іншим завданням, яке призначається як завдання за часом і виклик його ініціюється управлінням Тпоч інтервалу активності з обробника негайного переривання.

Додатковими параметрами, необхідними супервізору для управління викликом обробників, відповідних перериванням з негайною обробкою, є значення пріоритетів, як між окремими завданнями негайної обробки, так і по відношенню до інших видів завдань проблемної обробки.

Стратегія приведеної обробки дозволяє реалізувати обробку проблемних переривань, не накладаючи на їх програмування обмежень і вимог, характерних для реалізації негайної обробки. Виклик таких завдань на виконання здійснюється супервізором не безпосередньо по сигналах (подіям) проблемних переривань, а синхронно з виконанням деяких умов, які аналізуються в середовищі супервізора періодично по $\Delta t_{сист}$. Зрештою виклик обробника після настання події «приводиться» до істинності деякої умови – умови приведення. Функціональна залежність для умов приведення може призначатися індивідуально для кожного завдання або формулюватися для груп завдань, що об'єднуються функціонально або за типом подій.

11. Синхронізація виконання завдань

11.1. Інверсія пріоритетів

У Теоремах 1 та 2 ЧМА присутнє припущення, яке є досить непрактичним, а саме- незалежність завдання. Насправді, сама суть існування завдання - взаємодія з іншими завданнями через просту синхронізацію за допомогою деякого механізму (зазвичай, через передачу повідомлень або загальнодоступну пам'ять). Проте, ця взаємодія веде до явища, відомого як інверсія пріоритетів. Інверсія пріоритетів відбувається, коли низькопріоритетне завдання перешкоджає високопріоритетнішому завданню виконатися, бо високопріоритетніше завдання блоковане на ресурсі, використовуваному завданням з низьким пріоритетом.

Одне з вирішень даної проблеми полягає в тому, що б заборонити витіснення завдання, яке знаходиться в критичній секції, проте це прийнятно лише в тому випадку, якщо час перебування в критичній секції невелик, інакше низькопріоритетне завдання заблокує виконання високопріоритетного. Для запобігання затримці високопріоритетного завдання застосовується алгоритм корекції пріоритетів. На час перебування в критичній секції низькопріоритетного завдання його пріоритет піднімається до максимального з пріоритетів завдань, що блокуються. Даний алгоритм був запропонований в 1990 року і відомий під назвою « протокол граничного пріоритету»[].

Друга проблема. Яка може виникнути іменується « тупикова ситуація», коли два завдання хочуть розділити два ресурси.

Розглянемо три завдання - завдання High, Завдання Medium, і Завдання Low - де назви завдань передають їх пріоритети і Задачам High і Low необхідно використовувати спільний ресурс винятковим способом, що не витісняється. Допустимо, наприклад, що Задача Low може працювати першою, а обидва інших завдання знаходяться в даний час в стані чекання (момент часу 0 на Рисунку 11.1).

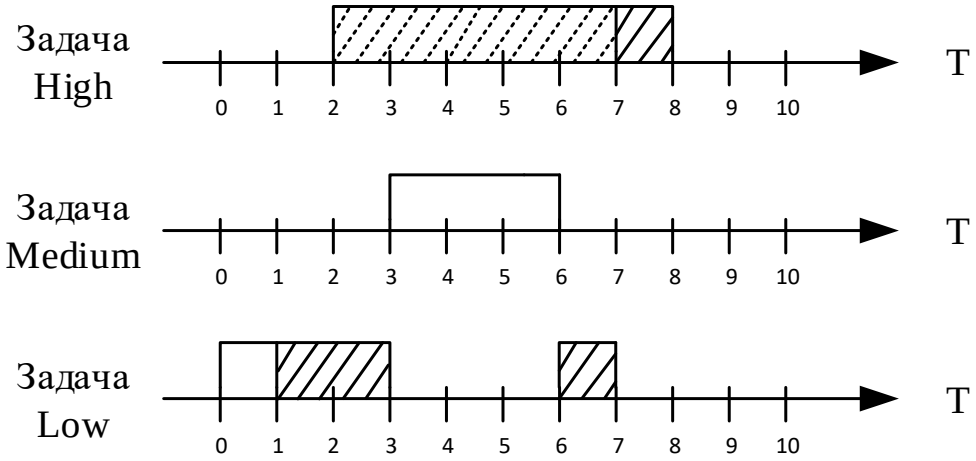

Рисунок 11.1 - Ілюстрація інверсії пріоритетів.

Якщо Задача Low починає використовувати ресурс, починаючи з часу 1 (на Рисунку 11.1 це відмічено штрихуванням з нахилом праворуч) вона входить в критичну секцію і зазвичай використовує деякий механізм синхронізації, щоб перешкоджати іншим завданням виконати той же самий код (критична секція - частина коди, яка вимагає виконання виключно одним завданням до завершення цієї секції). Якщо Задача High хоче використовувати той же самий ресурс, починаючи з часу 2 (відмічено крапками), Завдання Low не може вивантажитися, бо ресурс захищений під час її виконання. Отже, Завдання Low продовжує виконуватися, а Задача High чекає. Це називають прямим блокуванням.

Проте, завдання Medium може не потребувати своєї роботи в захищеному ресурсі (час 3). Оскільки пріоритет Задачи Medium вище чим Завдання Low, вона звичайно витісняє Задачу Low і виконується, таким чином вимушуючи Задачу High чекати продовження інтервалу часу (протягом часу 3-6), навіть при тому, що пріоритет Задачи Medium нижче чим в Завдання High. Цей вигляд блокування називають, непрямим блокуванням. Після того, як Задача Medium завершилася, Завдання Low поновлюється у момент часу 6. Завдання High може виконатися лише після того, як Задача Low закінчила використовувати ресурс (під час 7). Теоретично, особливо якщо є багато завдань з пріоритетом Завдання Medium (між Задачей High's і Завданням Low's), ця ситуація може тривати невизначено довго, таким чином непередбачуване блокуючи Завдання High. Рішення в даному прикладі, полягає в тому, щоб збільшити пріоритет Завдання Low, так, щоб він тимчасово став ви-

соким на час використання ресурсу, до якого Задача High просить доступ. Тоді, жодне інше завдання, окрім високопріоритетніших чим Завдання High, не може блокувати Завдання Low, і вона може безпечно завершити використання ресурсу. Таке збільшення називають спадкоємством пріоритету.

11.2 Частотно-монотонний алгоритм з врахуванням блокувань

Теорема 1 частотно-монотонній диспетчеризація з врахуванням залежності між виконанням завдань буде представлена таким чином:

Для набору N незалежних періодичних завдань, де Ci, Ti, і Bi, i = 1,2.... N, є часом виконання, періодом і часом блокування в самому гіршому випадку, відповідно, а критичний термін виконання дорівнює періоду завдання, завдання диспетчеризуєми згідно ЧМА, якщо наступна умова виконується:

$$\forall_{n,} 1 \le n \le N \sum_{i=1}^{n} C_i / T_i + B_n / T_n \le n(2^{1/n} - 1). \tag{11.1}$$

Теорема 2 частотно-монотонного алгоритму, враховуючи можливі блокування завдань може бути сформульована наступним чином:

Для набору N незалежних періодичних завдань, де Ci, Ti, і Bi, i= 1,2 . . . . N, є часом виконання, періодом, і часом блокування в найгіршому випадку відповідно, а критичний термін виконання дорівнює періоду завдання, завдання диспетчеризуеми згідно ЧМА, якщо наступна умова виконується:

$$V_{i\min}\left[\sum_{j=1}^{i-1}\frac{C_j}{lT_k}\left[lT_k / T_j\right] + \frac{C_i}{lT_k} + \frac{B_i}{lT_k}\right] \le 1, \tag{11.2}$$

де мінімум розрахований як $(k, l) \in W_i$, і

$$W_i = \left\{(k,l) \mid 1 \le K \le i, l = 1,\ldots,\left[T_i / T_k\right]\right\}$$

Вживання узагальненої теорії планерування розглянемо на наступному прикладі. Хай дано чотири завдання – дві періодичні (t1,t4) і дві аперіодичні (t2,t3), при цьому завдання 3 ініціюється перериванням, мінімальний час між якими – 200мс.

Завдання 1,2,4 для спільної роботи використовують якийсь ресурс, доступ до якого синхронізується через семафор. Завданням призначаються пріоритети відповідно до алгоритму монотонних частот і обліку особливості виконання завдання 3.

Таблиця 11.1

Набор завдань з коефіцієнтом використання рівним 1

Завдання	Пр	Тс	Тп	Тс/Тп	Коефіцієнт використання	Межа
1	2	20	100	0,2	0,2	1.000
2	3	15	150	0,1	0,3	0.828
3	1	4	200	0,02	0,32	0.779
4	4	30	300	0,1	0,42	0,756

Не дивлячись на те, що повний коефіцієнт використання нижче верхньої межі (0,42 < 0,69), необхідно перевірити можливість диспетчеризуємості завдань, оскільки в призначенні пріоритетів були відхилення від частотно-монотонних пріоритетів, та при розрахунку коефіцієнту використання не враховувались умови витіснення та блокування .

Завдання 3 – завдання по перериванню має найвищий пріоритет, коефіцієнт використання 0,02, витіснити її жодна з інших завдань не може, тому вона встигатиме завершити свого виконання до критичного терміну.

Для останніх завдань необхідно проаналізувати наступні ситуації:

1. витіснення пріоритетнішими завданнями з періодом меншим, ніж в тієї, що розглядається;
2. виконання аналізованого завдання;
3. витіснення пріоритетнішими завданнями з періодом більшим, ніж в тієї, що розглядається;
4. блокування завданнями з нижчим пріоритетом.

Аналіз для завдання 1:

1. відсутні завдання з вищим пріоритетом і періодом меншим, ніж 100, які могли б витіснити завдання 1;
2. час виконання 20, коефіцієнт використання 0,2;
3. завдання 3 (пріоритет вищий, період більший) може витіснити задачу 1. Коефіцієнт використання на періоді за рахунок витіснення дорівнює 0,04 (Тс3 / Тп1).

4. потенційно завдання 1 можуть заблокувати завдання 2,4, так як. вони спільно використовують один ресурс. Але у гіршому разі це буде лише одна з більшим часом виконання, тобто завдання 4. Коефіцієнт використання з врахуванням блокування складає 0,03 (Тс4/ Тп1).
5. сумарний коефіцієнт використання складає 0,54 (0,04+0,2+0,3), тобто менше верхнього кордону, а отже, завдання 1 – диспетчеризуемо.

Аналіз для задачи2:

1. єдине завдання, період якого менший, ніж в тієї, що розглядається, а пріоритет більший, це завдання 1. Коефіцієнт використання за рахунок витіснення дорівнюватиме 0,2.
2. час виконання 15, коефіцієнт використання 0,1;
3. завдання, яке може витіснити задачу 2 і в якого період більше і пріоритет вищий, це завдання 3. Коефіцієнт використання за рахунок витіснення на періоді Тп2 дорівнюватиме 0,03 (Тс3/ Тп2);
4. блокувати завдання при доступі до ресурсу може лише завдання 4 (пріоритет нижчий). Коефіцієнт використання в цьому випадку буде 0,2 (Тс4/ Тп2.);
5. спільний коефіцієнт використання у гіршому разі складе 0,53, т.ч. завдання 2 задовольняє граничним умовам.

Аналіз для завдання 4:

1. в даного завдання найбільший період і найменший пріоритет, тому її може витіснити будь-яке завдання . Коефіцієнт використання за рахунок витіснення складе 0,32 (0,2+0,1+0,02);
2. час виконання 30, коефіцієнт використання 0,1;
3. немає завдань, які могли б витіснити і, в яких вищі пріоритети та більші періоди;
4. відсутні завдання з нижчим пріоритетом, які могли б заблокувати .
5. коефіцієнт використання у гіршому разі складає 0,42, завдання 3 задовольняє тимчасовим обмеженням.

Висновок - всі завдання є такими, що диспетчеризуються.

Список рекомендованої літератури

1. Кайхан Эрджиес. Distributed real time systems. Розподілені системи реального часу. Теорія і практика / К.: ДМК Прес, 2019.- 382 с.
2. Jack Ganssle and Michael Barr. Embedded systems dictionary. - CMP Books, 2003. - 293 p.
3. Dimosthenis Kyriazis, Theodora Varvarigou, Kleopatra Konstanteli. Achieving Real-Time in Distributed Computing. - IGI Global, 2011. - 452 p.
4. Комп'ютерні системи реального часу. – навч. посіб. для здобувачів ступеня магістра за освітньою програмою "Системне програмування та спеціалізовані комп'ютерні системи" спеціальності 123 «Комп'ютерна інженерія»/ В. Г. Зайцев, Є. І. Цибаєв - Київ: КПІ ім. Ігоря Сікорського, 2019. - 162 с.
5. Кобзар О.В. Мусієнко М.В. Топологія побудови операційної системи реального часу автоматизованої системи управління військово-морськими силами збройних сил України // Проблеми створення, випробування, застосування та експлуатації складних інформаційних систем. – Випуск № 20. (Лютий) 2022.- С. 28-41.
6. Ерджиєс К. Розподілені системи реального часу. Теорія і практика. – Видавництво Springer, 2020. – 382 с.
7. Buttazzo, G., Lipari, G., Abeni, L., Caccamo, M. Soft Real-Time Systems: Predictability vs. Efficiency. –Springer, 2005. – 258 с.
8. Аналіз операційних систем реального часу. https://jrnl.nau.edu.ua/index.php/PIU/article/view/6980
9. Rajib Mall. Real-Time Systems: Theory and Practice. - IGI Global, 2006. – 242 p. https://acadndtechy.files.wordpress.com/2015/01/real-time-systems-rajib-mall-pearson-education-india-2007.pdf
10. Комп'ютерні системи реального часу. https://ela.kpi.ua/bitstream/123456789/29604/1/Kompyuterni_systemy_realnoho_chasu.pdf
11. RTOS: Операційна система реального часу. https://www.hwlibre.com/uk/rts/
12. RE-198 Операційні системи реального часу та 32 бітні МК. https://my.kpi.ua/site/view?id=529

13. Операційні системи реального часу, їх особливості. http://um.co.ua/6/6-6/6-64372.html
14. Програмування систем реального часу, як навчальна дисципліна. https://ua.kursoviks.com.ua/kompyuterni/programuvannya-sistem-realnogo-chasu
15. Різниця між розподілом часу та операційною системою реального часу. https://uk.gadget-info.com/difference-between-time-sharing
16. Що таке операційні системи реального часу. https://gazette.com.ua/it/shcho-take-operatsijni-sistemi-realnogo-chasu.html
17. Особливості використання операційних систем реального часу. https://openarchive.nure.ua/handle/document/13854
18. Основні вимоги до систем реального часу. https://studfile.net/preview/5471446/page:2/
19. Порівняльний аналіз операційних систем реального часу для автопілота БПЛА. https://conf.ztu.edu.ua/wp-content/uploads/2018/05/138-1.pdf
20. Топологія побудови операційної системи реального часу автоматизованої системи управління військово-морськими силами збройних сил України. http://znp.zvir.zt.ua/article/view/253056
21. Використання операційних систем реального часу при проектування та розробці системи збору даних акселерометрії. http://elartu.tntu.edu.ua/handle/lib/28836
22. Операційна система реального часу QNX V 4.2. http://fuck-hack.com/operatsijna-systema-realnogo-chasu-qnx-v4-2-2/
23. Комп'ютерні операційні системи: сімейство ОС для комп'ютерів. https://www.techlila.com/uk/computer-operating-systems/
24. Дослідження часових характеристик затримки переривань в системах реального часу. http://eadnurt.diit.edu.ua/handle/123456789/14485

НАВЧАЛЬНО-МЕТОДИЧНЕ ВИДАННЯ

КОНСПЕКТ ЛЕКЦІЙ
З ДИСЦИПЛІНИ «ПРОЕКТУВАННЯ СИСТЕМ РЕАЛЬНОГО ЧАСУ»

для студентів напряму підготовки 6.050102 «Комп'ютерна інженерія»
та спеціальності 123 Комп'ютерна інженерія
усіх форм навчання

Комп'ютерний набір і верстка:

Шамаєв Віталій Віталійович

Укладачі:

Ковальов Сергій Олександрович,

Шамаєв Віталій Віталійович

Конспект лекцій з дисципліни «Проектування систем реального часу» розроблено за участю ***О.Г. Шевченко***. Автори щиро вдячні їй за співпрацю та надані матеріали.

ДВНЗ «Донецький національний технічний університет»

85300, м. Покровськ, пл. Шибанкова, 2.