Міністерство освіти і науки України
Донбаська державна машинобудівна академія

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт

з дисципліни

«СИСТЕМНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ»

(для студентів спеціальності 123“Комп’ютерна інженерія»)

Освітній рівень - бакалавр

Краматорськ 2020

Методичні вказівки до лабораторних робіт з дисципліни «Системне програмне забезпечення» для студентів галузі знань 12 «Інформаційні технології» спеціальності 123 «Комп'ютерна інженерія» (практикум з моделювання) / Укл.: Д.А. Зайцев, С.М.Вороной, Т.Р.Шмельова, О.А.Костіков. – Краматорськ : ДДМА. – 2020. – 42 с.

Представлені індивідуальні завдання для проведення лабораторних і практичних занять з дисципліни «Системне програмне забезпечення». Кожна тема починається коротким викладенням теоретичного матеріалу, потім йде загальний опис завдання, приклад його виконання, контрольні питання; в додатках наведені варіанти завдань.

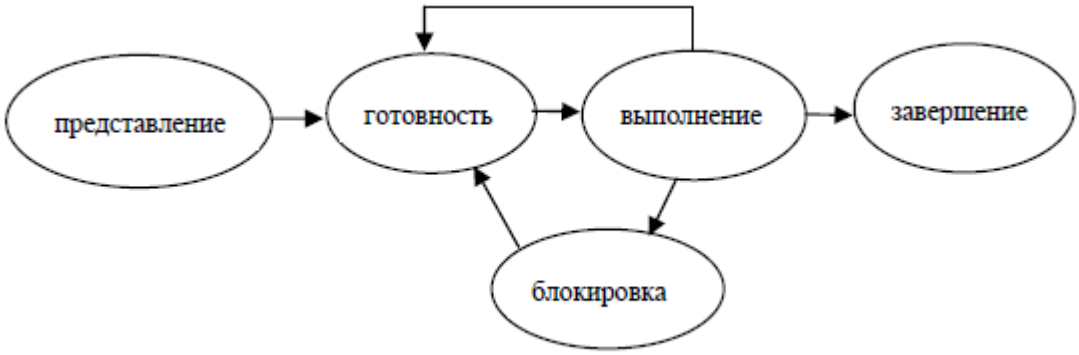
ЗМІСТ

РОЗДІЛ 1. УПРАВЛІННЯ ПРОЦЕСАМИ

Ключем до розуміння функціонування засобів управління процесорами (процесами) сучасних ОС є діаграма станів процесу. Модулі ОС забезпечують зміну станів процесів відповідно до діаграми, перемикання контексту процесора для запуску поточного процеса, виклик підсистеми введення-виведення для ініціювання операцій на зовнішніх пристроях. Основними станами процесу є:

- уявлення;
- готовність
- виконання;
- блокування;
- завершення.

У стані уявлення запущеному процесу виділяються необхідні ресурси; в стані готовності процес володіє всіма необхідними для виконання ресурсами за винятком центрального процесора (ЦП); в стані виконання програмний код процесу фактично виконується на процесорі; в стані блокування процес очікує події (наприклад, завершення введення-виведення); в стані завершення процес звільняє ресурси ОС. Абстрактна діаграма станів має наступний вигляд:



При проектуванні ОС абстрактна діаграма доповнюється безліччю станів, що відображають різні причини блокувань і етапи виділення (звільнення) ресурсів.

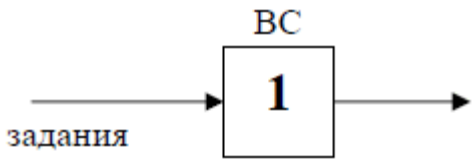
Заняття 1.1. Мультипрограмування

Мета заняття: засвоїти основи організації мультипрорамного режиму роботи операційних систем, оцінити переваги мультипрограмування.

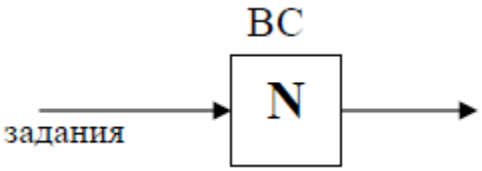
Короткий виклад теоретичного матеріалу

Перші обчислювальні системи (ВС) працювали в однопрограмному режимі. ОС починала виконання нового завдання лише в разі повного

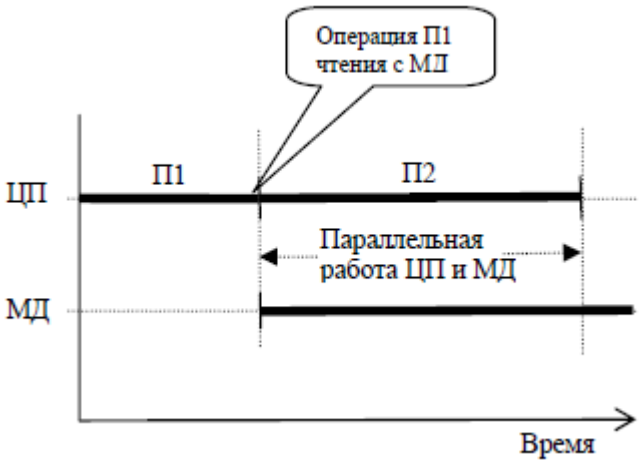
завершення поточного завдання. Схематично це можна представити таким чином



Це призводило до неефективного використання ресурсів обчислювальної системи в зв'язку з простоєм обладнання. Тільки одне з численних пристроїв (процесор, магнітний диск, принтер, магнітна стрічка) працювало в конкретний момент часу. Мультипрограмні ОС дозволяють завантажити безліч пристроїв обчислювальної системи за рахунок одночасного виконання декількох завдань, що призводить до скорочення сумарного часу виконання суміші завдань. Схематично ОС, що виконує N завдань, можна представити таким чином:



Число N називають коефіцієнтом мультипрограмування. Скорочення сумарного часу виконання досягається за рахунок суміщення роботи процесора з роботою зовнішніх пристроїв. У той час як Процес 2 виконується на процессорі, Процесс1 може обслуговуватися на магнітному диску:

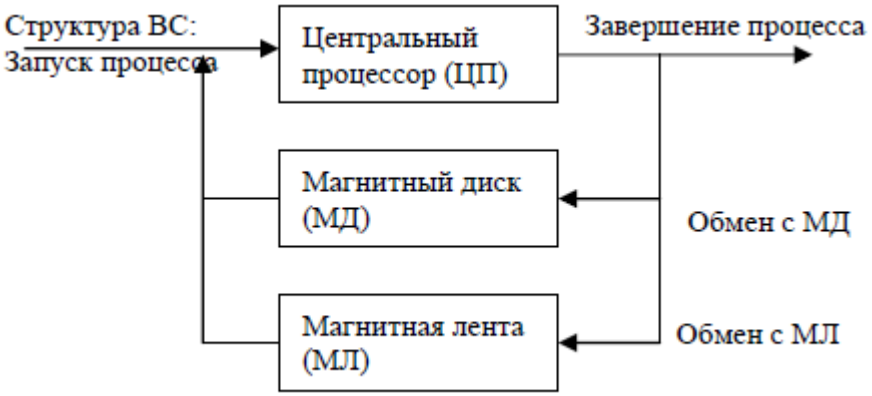


Слід зазначити, що мультипрограмний режим роботи вимагає відповідної апаратної підтримки. Обладнання повинно забезпечувати автономну роботу зовнішніх пристроїв, керованих своїм контролером

(каналом), а також систему переривань для сигналізації зовнішніх пристроїв про завершення операції.

Завдання

Виконати ручне трасування роботи засобів управління процесами мультипрограммної ОС. оцінити ефективность роботи мультипрограммної ОС



.

Порядок виконання

1. Виконати ручну трасування виконання зазначеної суміші процесів:
а) в однопрограмній обчислювальній системі;
б) в мультипрограмній обчислювальної системі.
2. Заповнити трасувальні таблиці
3. Виконати аналіз ефективності мультипрограмного режиму:
   3.1. Оцінити прискорення виконання суміші процесів
   3.2. Розрахувати і порівняти коефіцієнти завантаження пристроїв
4. Сформулювати переваги мультипрограмного режиму

Приклад виконання

Характеристики виконуваних процесів

Характеристики виконуваних процесів

номер	час надходження	Послідовність дій
1	0	ЦП (40) -МД (100) -ЦП (20)
2	10	ЦП (20) -МД (100) -ЦП (80)
3	20	Ш1 (60) -МД (100) -ЦП (10)

Приклад трасування

час	Черга готових процесів	процес - сор	Черга до МД	МД
0		П1 (40)		
10	П2 (20)	Ш (ЗО)		

20	ПЗ (20), П2 (20)	П1 (20)		
40	ПЗ (20)	П2 (20)		ПЦ100)
60		ПЗ (60)	112 (100)	П1 (80)
120			ПЗ (ЮО). 2 (100)	П1 (20)
140		ПЦ20)	ПЗ (ЮО)	П2 (100)
160			ПЗ (ЮО)	П2 (80)
240		П2 (80)		ПЗ (ЮО)
320				ПЗ (20)
340		ПЗ (Ю)		
350				

Прискорення виконання суміші процесів

Час виконання суміші в однопрограмні режимі:

Т1 = (40 + 100 + 20) + (20 + 100 + 80) + (60 + 100 + 10) = 530

Час виконання суміші в мультипрограммном режимі: Тм = 350 прискорення А = Т1 / Тм = 530/350 = 1,51

Коефіцієнти завантаження пристроїв К = Тустр / Тобщ

однепрограмний	мультипрограмний
Кпроц = 230/530 = 0,45 Кмд = 300 1 530 = 0.57	Кпроц = 230. 350 = 0.66 Кмд = 300 350 = 0.86

Варіанти завдань - Додаток 1.1.

Контрольні питання

1. У чому полягає мультипрограмний режим роботи ОС?

2. За рахунок чого скорочується час виконання суміші процесів в мультипрограммном режимі?

3. Які вимоги до апаратних засобів комп'ютера для організації мультипрограмному режиму?

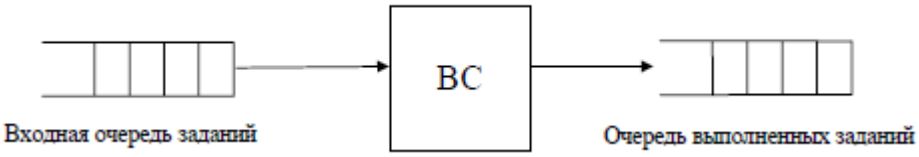
4. Які накладні витрати організації мультипрограмному режиму?

Заняття 1.2. Циклічне квантування часу

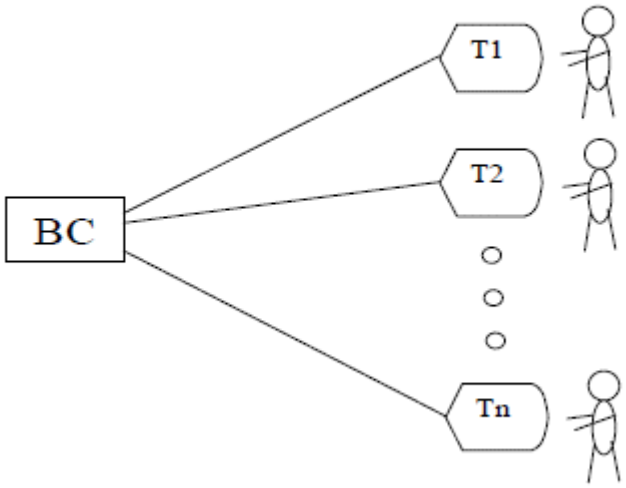
Мета заняття: засвоїти основи організації режиму поділу часу роботи операційних систем, оцінити переваги режиму поділу часу

Короткий виклад теоретичного матеріалу

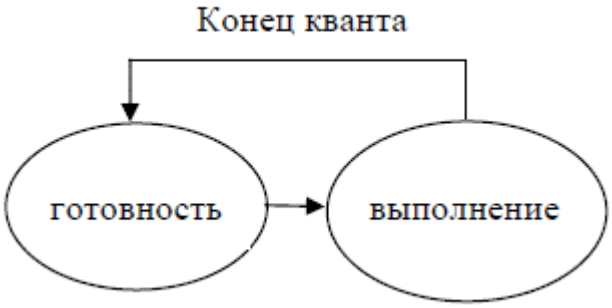
Перші мультипрограмні ОС функціонували в пакетному режимі, в якому користувач відокремлювався від процесу виконання його завдань. попередньо підготовлені завдання організовувалися в черзі на перфокартах або магнітному диску. Потім суміш завдань подавалася на вхід обчислювальної системи:



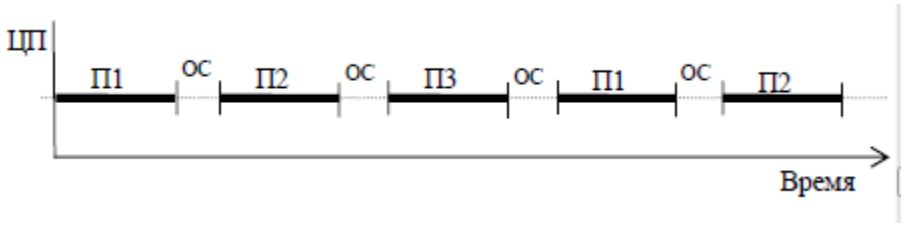
Необхідність взаємодії користувача з запущеними програмами, автоматизація процесів підготовки завдань за допомогою текстових редакторів, а також поява дисплеїв з електронно-променевою трубкою зумовили появу ВС з діалоговим режимом роботи. В цьому режимі декілька користувачів одночасно взаємодіють з ВС за допомогою терміналів: Для забезпечення роботи декількох користувачів в діалоговому режимі особливо важливим стає час реакції ОС на команди користувача. При використанні класичного мультипрограмування один з процессів може захопити процесор на невизначений час і, таким чином, блокувати роботу інших користувачів. Ефективний діалоговий режим можливий при гарантованому часу реакції ОС, який не перевищує декілька секунд.



Для забезпечення діалогової роботи був запропонований режим розподілу часу, що забезпечує спільну роботу декількох користувачів. Основою для забезпечення режиму поділу часу є циклічне квантування часу центрального процесора. Кожному процесу при отриманні процесора виділяється певний інтервал часу - квант, протягом якого він виконується на процесорі; потім процес повертається в кінець черги готових процесів. Таким чином, процесор циклічно перемикається між готовими процесами і кожен з користувачів вважає, що ОС працює тільки з ним одним. Далі представлений фрагмент діаграми станів процесу, який забезпечує циклічне квантування:



Слід зазначити, що організація квантування тягне за собою деяке зниження продуктивності ВС. Перемикання між процесами вимагає певного часу, протягом якого працюють програми ОС, і являє собою накладні витрати ресурсів ОС при організації режиму поділу часу. при реалізації циклічного квантування постає питання про вибір оптимального розміру кванта. Великий розмір кванта знижує час реакції ОС, малий - збільшує накладні витрати через часте перемикання. Реальні ОС застосовують адаптивні механізми вибору розміру кванта.



Квантування трьох процесів

Для реалізації квантування часу використовується апаратний таймер. Таймер встановлюється на продовжність кванту при виділенні процесора процесу; переривання таймеру після закінчення часу кванта ініціює перемикання.

Завдання

Виконати ручне трасування роботи засобів управління процесами. Заповнити трасувальну таблицю.

Характеристики ОС: циклічне квантування часу, мультипрограмування

Порядок виконання

1. Виконати ручне трасування виконання зазначеної суміші процесів
2. Заповнити трасувальні таблиці
3. Виконати аналіз ефективності режиму поділу часу:
   а) при миттєвому перемиканні процесів;
   б) при часу перемикання рівному 1
4. Сформулювати переваги і недоліки режиму поділу часу

Приклад виконання

Розмір кванта - 10

Характеристики виконуваних процесів

Номер	Время поступления	Последовательность действий
1	0	ЦП(40)-МД(100)-ЦП(20)
2	10	ЦП(20)-МД(100)-ЦП(80)
3	20	ЦП(60)-МД(100)-ЦП(10)

Приклад заповнення трасування таблиці

Время	Очередь готовых процессов	Процессор	Очередь к МД	МД
0		П1(40)		
10	П1(30)	П2(20)		
20	П2(10), П1(30)	П3(60)		
30	П3(50),П2(10)	П1(30)		
40	П1(20),П3(50)	П2(10)		
50	П1(20)	П3(50)		П2(100)
60	П3(40)	П1(20)		П2(90)
70	П1(10)	П3(40)		П2(80)
80	П3(40)	П1(10)		П2(70)
90		П3(40)	П1(100)	П2(60)
100	…	…	…	…

Варіанти завдань - Додаток 1.2.

Контрольні питання

1. У чому полягає режим поділу часу?
2. Як впливає режим поділу часу на продуктивність ВС?

Заняття 1.3. Пріоритетні дисципліни

Мета заняття: засвоїти основи організації пріоритетних дисциплін роботи операційних систем, оцінити переваги пріоритетних дисциплін

Короткий виклад теоретичного матеріалу

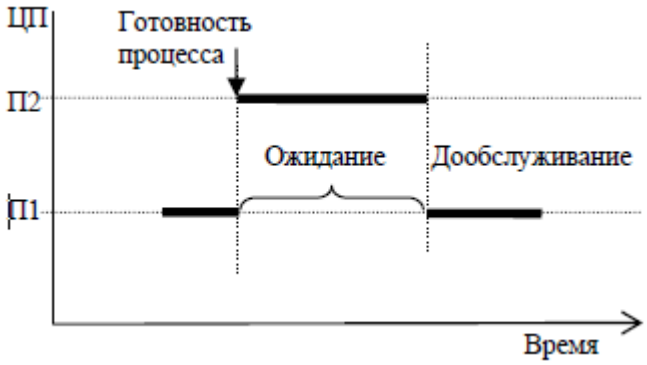
Безліч процесів, виконуваних в середовищі ОС, можна розділити на системні і прикладні. Системні процеси є частиною операційної системи і забезпечують роботу ВС в цілому. Таким чином, у багатьох випадках потрібно забезпечити переважне виділення їм ресурсів, особливо, центрального процесора.

Крім того, прикладні процеси також слід розрізняти на основі кращого виділення ресурсів, особливо при використанні ВС в технологічному управлінні. З огляду на жорсткі обмеження на час реакції і пов'язаних з цим особливостей побудови, операційні системи, призначені для технологічного управління, виділяють в спеціальний клас, іменований операційні системи реального часу (ОСРВ).

Пріоритетні дисципліни планування процесів були запропоновані як для оптимізації виконання суміші завдань, так і для забезпечення оперативності виконання системних процесів і процесів реального часу.

Рівень пріоритету в багатьох випадках може бути представлений цілим числом. Розрізняють два типи пріорітетів процесів: відносні і абсолютні. Відносні пріоритети використовуються тільки при постановці процесів в чергу, таким чином, поява процесу з більш високим відносним пріоритетом не перериває виконання на процесорі поточного процесу.Абсолютнний пріоритет порівнюється з пріоритетом поточного процесу і, в разі перевищення, перериває виконання поточного процесу до завершення його кванта:

Відносний пріоритет



Абсолютний пріоритет

Таким чином, абсолютні пріоритети використовуються для миттєвого перемикання на новий процес, і застосовується для процесів реального часу і найбільш важливих системних процесів, наприклад, самої підсистеми управління процесами.

Розрізняють статичні і динамічні пріоритети. Статичні пріоритети призначаються вручну адміністратором ОС, або користувачем в певних відведених йому адміністратором ОС межах і не змінюються під час виконання процесів. динамічні пріоритети призначаються і змінюються ОС під час виконання процесів з метою оптимізації функціонування ВС.

Наприклад, ОС може підвищити пріоритет процесу, який часто виконує введення / виведення і знизити пріоритет процесу, який тривало займає процессор, для підвищення завантаження пристроїв ВС.

Статичні абсолютні пріоритети забезпечують мінімальний час реакції ОС РВ в технологічному управлінні. Слід зазначити, що застосування пріоритетних дисциплін не завжди оптимально з точки зору крітерія забезпечення максимального завантаження пристроїв ВС. Для ОС РВ на перше місце висуваються критерії мінімального часу реакції, що виправдано, так як втрати через неоперативне технологічне управління можуть значно перевищувати втрати через простоювання пристроїв ВС.

Завдання

Виконати ручне трасування роботи засобів управління процесами. Заповнити трасувальну таблицю.

Характеристики ОС: пріоритетна дисципліна, мультипрограмування

Порядок виконання

1. Виконати ручну трасування виконання зазначеної суміші процесів
2. Заповнити трасувальні таблиці
3. Виконати аналіз ефективності пріоритетною дисципліни

РОЗДІЛ 2. УПРАВЛІННЯ ПАМ'ЯТТЮ

Оперативна пам'ять (ОП) є другим після ЦП ресурсом за своєю значимістю для виконання процесу. Безліч способів управління пам'яттю конкретних ОС можна класифікувати за такими основними ознаками:

- повне або часткове розміщення процесу в ОП;
- чіткий або недоладне виділення оперативної пам'яті;
- виділення ділянок пам'яті фіксованого або змінної довжини.

Багато складних методів управління ОП, обумовлені історично високою вартістю і обмеженнями обсягу фізичної пам'яті, наприклад, оверлейні структури, практично не використовуються в даний час у зв'язку з розвитком концепції віртуальної пам'яті. Стандартом де-факто для сучасних ОС є віртуальна сегментно-сторінкова пам'ять.

Заняття 2.1. Динамічні розділи

Мета заняття: засвоїти основи організації управління пам'яті динамічними розділами, оцінити ефективність управління пам'яті динамічними розділами

Короткий виклад теоретичного матеріалу

Управління оперативною пам'яттю (ОП) динамічними розділами застосовувалося в ОС ранніх поколінь; однак і в даний час цей спосіб управління пам'яттю залишається актуальним, так як використовується для управлінням пулом пам'яті бази даних ОС, в якому розміщуються керуючі блоки ОС: блок управління процесом, блок управління пристроєм, блок управління файлом і інші.

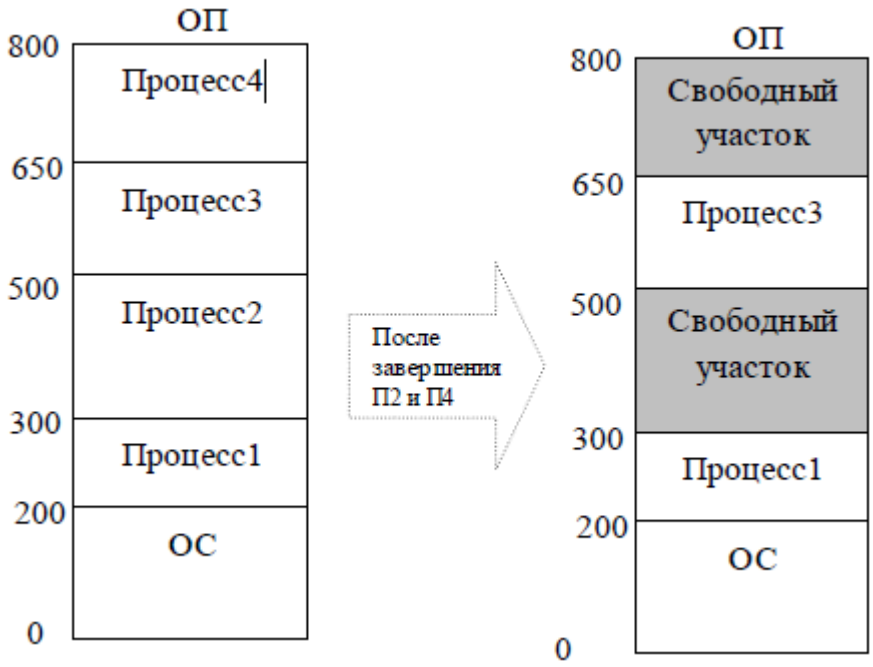
У ранніх поколіннях ОС оперативна пам'ять рас розглядалася як одновимірний масив байтів (слів). Для виділення процесу ЦП вимагалося виконання двох умов: повне розміщення процесу (виконуванового файлу) в оперативній пам'яті; зв’язне виділення пам'яті процесу у вигляді однієї безперервної ділянки.

В цьому випадку поточний розподіл пам'яті повністю описується масивом меж (початкова і кінцева адреси або початкова адреса і довжина) пам'яті процесів. Аналогічний описувач можна використовувати і для решти вільних ділянок ОП:

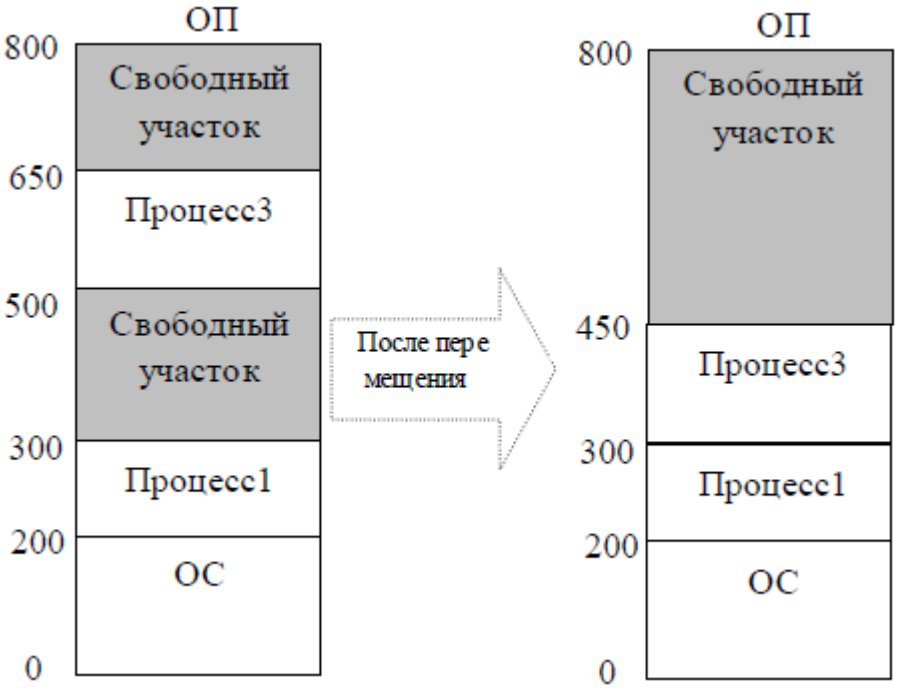
Память процессов		
Процесс	Начало	Длина
П1	200	100
П3	500	150

Свободная память	
Начало	Длина
300	200
650	150

Спочатку завантажені процеси займають одну безперервну ділянку пам'яті слідом за кодом ОС. Однак з огляду на випадковий час їх завершення, вільна і зайняті ділянки стають розкиданими в масиві байтів ОП:



Таку ситуацію називають фрагментацією пам'яті. При виборі необхідної ділянки пам'яті для нового процесу ОС може застосовувати критерії першої відповідної ділянки або найбільш підходящої ділянки. Однак можлива ситуація, при якій загальний обсяг вільної пам'яті задовольняє вимогам процесу, але складений сумою розмірів несуміжних вільних ділянок. В цьому випадку був запропонований спеціальний метод реорганізації ОП, званий переміщенням. Запускається спеціальний модуль управління пам'яті ОС, який зрушує (копіює) всі процеси в ОП для усунення фрагментації і створення однієї безперервної ділянки вільної ОП:



Слід зазначити, що переміщення неминуче тягне за собою накладні витрати ОС у вигляді додаткового часу копіювання процесів в ОП. Хоча архітектура багатьох процесорів пропонує спеціальні операції швидкого копіювання блоків пам'яті, ці витрати можуть бути істотними для загальної продуктивності ВС.

Мультипрограмний режим робить актуальними аппаратні засоби захисту пам'яті для запобігання (помилкової або навмисної) зміни кодів одного процесу деяким іншим процесом. Зауважимо, що засоби захисту пам'яті вкрай необхідні і однопрограмним ОС для захисту розділу пам'яті самої ОС від втручання виконуваного процесу. У більшості ОС з керуванням пам'яттю динамічними розділами ця проблема вирішується шляхом використання апаратних регістрів меж поточного процесу. При зверненні до адреси пам'яті поза межами генерується апаратне переривання по захисту пам'яті, яке потім обробляється ОС.

Завдання

Виконати ручне трасування засобів управління оперативною пам'яттю. Заповнити трасувальну таблицю. Оцінити ефективність управління пам'яттю.

Характеристики ОС: динамічні розділи, мультипрограмування, пріоритетна дисципліна.

Порядок виконання

1. Виконати ручне трасування роботи засобів управління ОП
2. Заповнити трасувальні таблиці
3. Виконати аналіз накладних витрат на переміщення процесів
4. Сформулювати переваги і недоліки управління пам'яті динамічними розділами.

Приклад виконання Розмір ОП - 70

Характеристики виконуваних процесів

Номер	Час надходження	Час виконання	Розмір ВОП	пріоритет
1	0	30	30	.
2	20	40	20	1
3	30	50	40	3

Приклад заповнення трасування таблиці

Час	Стан ОП			Примітка
0	процес	П1 (акт.)	Вільний.	
	початок	0	30	

	довжина	30		40		
20	процес	П1 (акт.)	П2	Вільний.		
	початок	0	30	50		
	довжина	30	20	20		

час	стан	ОП			Примітка
30	процес	своє · го	П2	Вільний.	фрагментація
	початок	0	30	50	
	довжина	30	20	20	
30	процес	П2		Вільний.	переміщення
	початок	0		20	
	довжина	20		50	
30	процес	П2	П3 (акт.)	Вільний.	
	початок	0	20	60	
	довжина	20	40	10	
80	.				

Варіанти завдань - Додаток 1.4.

Контрольні питання

1. Яким чином описується розподіл пам'яті при використанні динамічних розділів?
2. Чому виникає фрагментація пам'яті при використанні динамічних розділів?
3. Яким чином ліквідується фрагментація пам'яті?
4. Які апаратні засоби застосовуються для захисту оперативної пам'яті?

Заняття 2.2. свопінг процесів

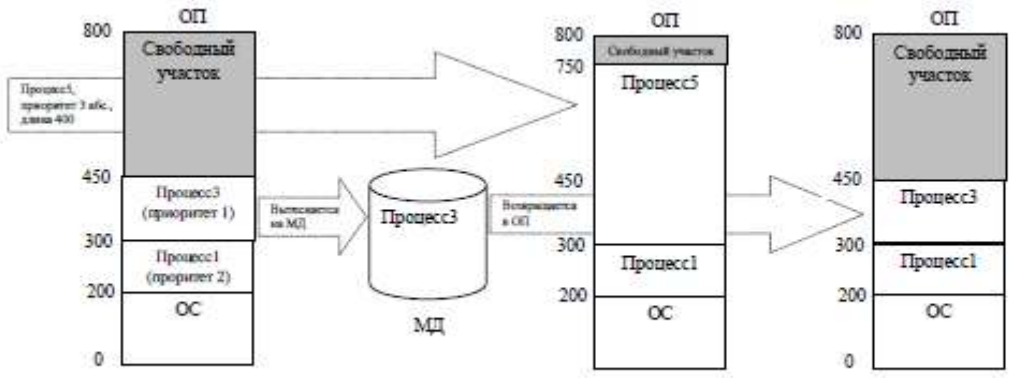
Мета заняття: засвоїти основи організації свопинга процесів, оцінити ефективність застосування свопинга процесів

Короткий виклад теоретичного матеріалу

Методи свопінгу (заміщення) широко використовуються в управлінні ОП сучасних ОС. Піддаватися свопінгу (витіснятися) можуть як процеси цілком, так і окремі ділянки пам'яті процесів. Сучасні ОС застосовують свопинг сторінок і сегментів процесів, попередні покоління ОС заміняли процеси цілком, або організовували спеціальний набір ділянок, які заміщуються, (оверлеї) всередині процесу.

Вивчимо найбільш простий різновид: свопінг процесів при управлінні пам'яттю динамічними розділами. Під час тривалої операції введення-виведення на повільному пристрої процесор може виконати кілька інших процесів, але оперативна пам’ять зайнята. Аналогічна ситуація виникає при появі нового процесу з абсолютним пріоритетом. Хоча процес може захопити процесор, йому недостатньо місця в оперативної пам'яті. Запропоновано просте рішення: знаходиться процес з найменшим пріоритетом серед завантажених в ОП; ділянка пам'яті процесу цілком копіюється на швидкий зовнішній пристрій (магнітний диск) і звільняється; високопріоритетний процес завантажується в пам'ять і виконується. При створенні сприятливих умов (відношення пріоритетів) витіснений процес повертається в ОП і продовжує виконання:

Свопінг неминуче пов'язаний з накладними витратами на виконання операцій вивантаження і завантаження процесів, але в ОС РВ такі накладні



витрати виправдані. Крім того, в діалогової або пакетної ОС свопінг може підвищувать загальну ефективність роботи ВС. Особливо, в поєднанні з динамічними пріоритетами, коли витісняється «поганий» процес з рівнем пріоритету, який став нижче деякої межі, а під час перебування в витісненому стані пріоритет процесу поступово підвищується ОС.

Відзначимо, що для забезпечення свопінгу на час виконання тривалої операції введення-виведення (принтер, магнітна стрічка) потрібна організація буферів в пам’яті ОС для проміжного зберігання даних процесу.

Завдання

Виконати ручне трасування роботи засобів свопінгу процесів. Заповнити трасувальну таблицю.

Характеристики ОС: свопінг процесів, динамічні розділи, мультипрограмування, пріоритетна дисципліна.

Порядок виконання

1. Виконати ручне трасування роботи засобів управління ОП і свопінга.
2. Заповнити трасувальні таблиці:

а) без урахування часу завантаження / розвантаження;

б) вказати коригування при обліку часу завантаження / розвантаження

3. Виконати аналіз накладних витрат на свопінг процесів.
4. Сформулювати переваги і недоліки свопінгу процесів

Приклад виконання

Розмір ОП - 70. Час завантаження / розвантаження процесу - 10.

Характеристики виконуваних процесів

номер	час надходження	час виконання	Розмір ВОП	пріоритет
1	0	100	50	1
2	10	100	40	3 абс.
3	20	100	20	2

Приклад заповнення трасування таблиці

час	стан ОП					стан ВП	Примітка
0	процес	П1 (акт.)	Вільн.				
	початок	0	30				
	довжина	50	40				
10	процес	Розв.				П1 (90)	вивантажити каш
	початок	0					
	довжина	70					
10	процес	Ш (акт.)	Вільн.			ПЦ90)	
	початок	0	40				
	довжина	40	30				
20	процес	П2 (акт.)	ПЗ	Вільн.		ПЦ90)	
	початок	0	40	60			
	довжина	40	20	10			
ПО	процес	Розв.	ПЗ	Вільн.		ПЦ90)	завершення П2
	початок	0	40	60			
	довжина	40	20	10			
ПО	процес	ПЗ (акт.)	П1				підкачка П1
	початок	0	20				
	довжина	20	50				
210							

Варіанти завдань - Додаток 1.5.

Контрольні питання

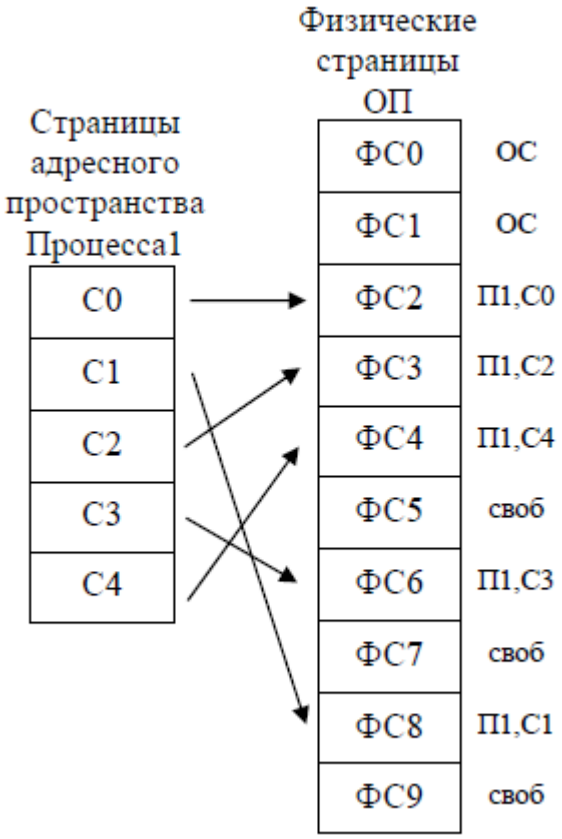
1. У чому полягає свопинг процесів?
2. Для чого застосовують свопинг процесів?
3. В яких випадках процес витісняється у зовнішню пам'ять?
4. Як впливає свопинг на продуктивність ВС?

Заняття 2.3. Сторінкова пам'ять

Мета заняття: засвоїти основи організації віртуальної сторінкової пам'яті, оцінити ефективність сторінкового управління пам'яттю.

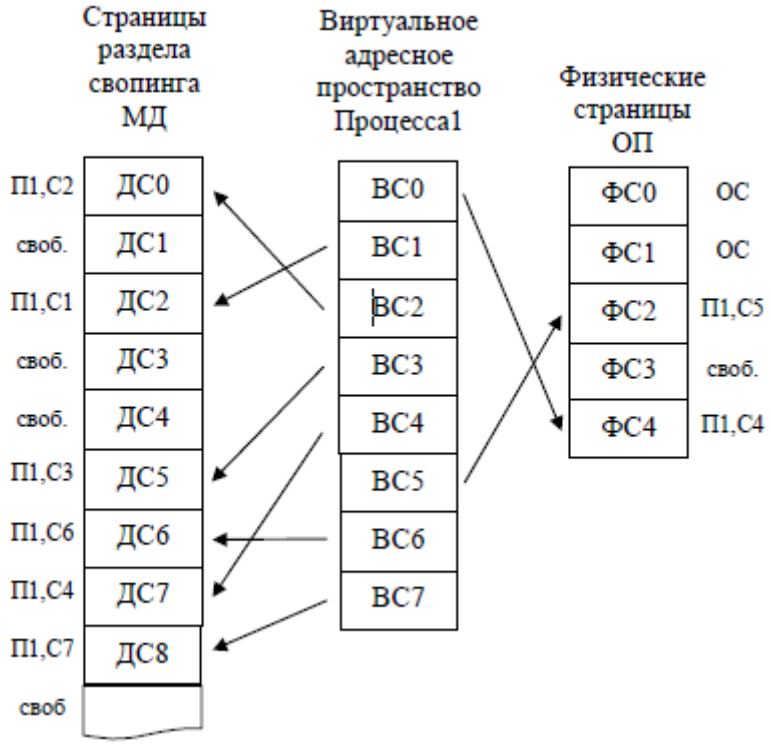
Короткий виклад теоретичного матеріалу

Сторінкова пам'ять заснована на незв'язному виділенні ОП процесу. Оперативна пам'ять комп'ютера розбивається на ділянки фіксованої довжини, які називаються сторінками. Адресний простір процесу також розбивається на сторінки тієї ж самої довжини. Кожна з сторінок процесу може бути розміщена в довільній сторінці оперативної пам'яті:



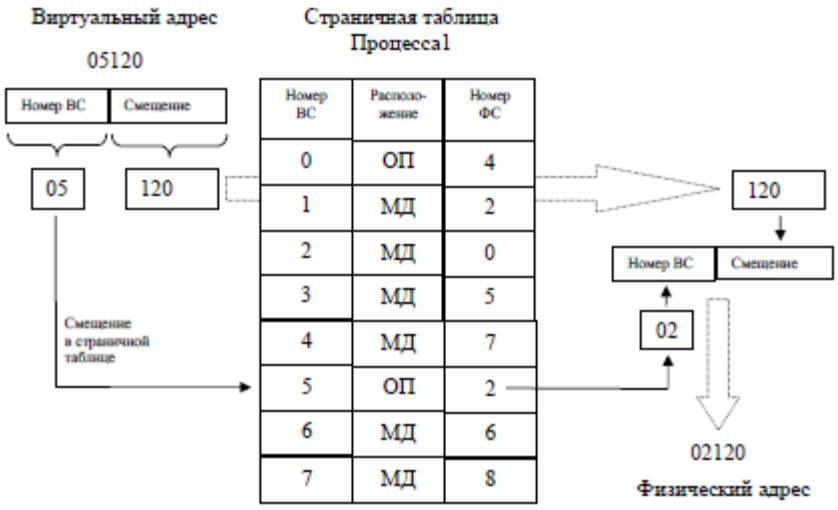
Крім того, застосовується концепція віртуальної пам’яті. Віртуальна (моделюєма ОС) пам'ять процесу може перевищувати розміри фізичної ОП і

обмежена лише сумою обсягів ОП і зовнішньої (дискової) пам'яті комп'ютера:



Для забезпечення віртуальної пам'яті було вирішено відмовитися від принципу повного розміщення процесу в ОП. В ОП завантажується лише початкова сторінка (декілька початкових сторінок) процесу; інші сторінки підкачуються в міру звернення до них в процесі свопінгу сторінок. Свопінг забезпечує заміщення сторінок процесів, якщо вільна сторінка ОП відсутня.

При цьому виникає проблема відображення адрес. При кожному зверненні до ОП потрібно обчислити фізичну адресу пам'яті за заданою віртуальною адресою. Зазначена проблема вирішується за допомогою набору спеціальних сторінкових таблиць, що забезпечують відображення:



Безліч вільних сторінок оперативної і зовнішньої пам'яті описуються аналогічними таблицями:

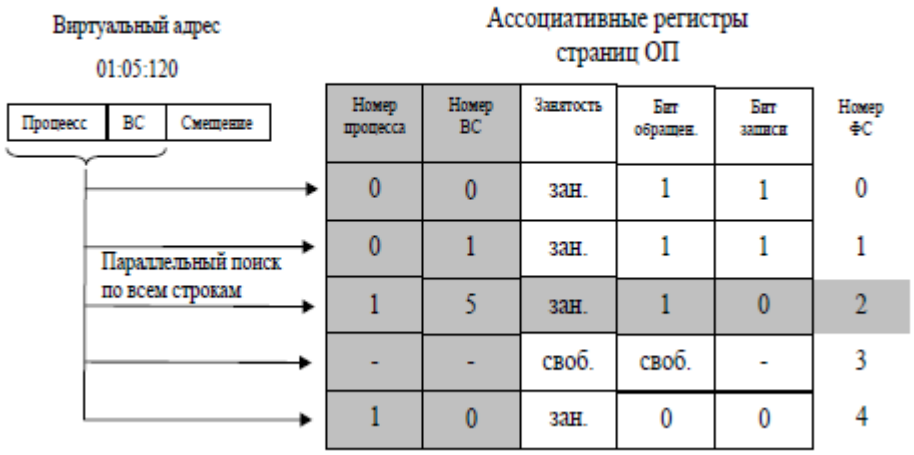
Таблиця сторінок МД

Номер страницы	Занятость	Номер процесса	Номер ВС
0	зан.	1	4
1	своб.	-	-
2	зан.	1	1
3	своб.	-	-
4	своб.	-	-
5	зан.	1	3
6	зан.	1	6
7	зан.	1	4
8	зан.	1	7

Таблица страниц ОП

Номер страницы	Занятость	Номер процесса	Номер ВС
0	зан.	0	0
1	зан.	0	1
2	зан.	1	4
3	своб.	-	-
4	зан.	1	0

При реалізації свопінгу постає питання вибору сторінки процесу для витіснення в разі, якщо всі фізичні сторінки ОП зайняті. Найбільш поширеним є алгоритм LRU витіснення сторінки, яка найдовше не використовувалася. Безпосередня реалізація алгоритму вимагає зберігання значення таймера останнього звернення до сторінці, проте практично застосовуються більш прості модифікації, які використовують лише кілька бітів інформації. Слід зазначити, що сторінкова пам'ять може бути реалізована ОС тільки в разі спеціальної архітектури ВС і, особливо, ЦП. Для швидкого апаратного відображення адрес використовуються сторінкові регістри, які забезпечують паралельний асоціативний пошук необхідної сторінки і подальше обчислення фізичної адреси:



Свопінг сторінок ініціюється сторінковим перериванням, яке генерується процесором в тому випадку, якщо потрібна сторінка відсутня в ОП. Для економної реалізації алгоритму LRU, як правило, застосовують пару біт сторінкових регістрів, які модифікуються автоматично: біт звернення A і біт записи W. Біти AW всіх сторінок періодично скидаються; потім біти деяких сторінок встановлюються процесором під час виконання процесів. В першу чергу витісняється сторінка зі значеннями бітів 00, якщо такі сторінки відсутні, то з 01, інакше будь-яка зі сторінок зі значеннями бітів 11. Операція витіснення сторінки з бітами 00 не вимагає її записи в зовнішню пам'ять і є найбільш швидкою.

При реалізації сторінкової пам'яті важливим є також питання вибору розміру сторінки. Мала довжина сторінки призводить до збільшення частоти свопінгу, велика - до збільшення відсотка коду, що зберігається в ОП, але не використовується. Реальна довжина сторінок коливається від 4 до 16 Кбайт.

Сучасні ОС використовують більш складну сегментно-сторінкову організацію пам'яті. Процес розбивають на декілька логічних частин змінної довжини, іменованих сегментами. Кожен із сегментів розбивається на сторінки фіксованої довжини. Використовуються дворівневі таблиці сегментів і їх сторінок.

Завдання

Виконати ручне трасування засобів управління віртуальною пам'яттю. Заповнити трасувальну таблицю. Оцінити ефективність управління пам'яттю.

Характеристики ОС: сторінкова пам'ять, LRU, мультипрограмування.

Порядок виконання

1. Виконати ручне трасування роботи сторінкової віртуальної пам'яті.
2. Заповнити трасувальні таблиці.
3. Виконати аналіз ефективності свопинга сторінок.

4. Сформулювати переваги і недоліки сторінкового управління пам'яттю.

Приклад виконання

Кількість сторінок ОП – 3
Кількість сторінок ВНП - 11

Характеристики виконуваних процесів

номер		час надходження	Кількість сторінок	Послеловательность дій
1		0	2	0 (30) -1 (20) -0 (20)
2		20	2	0 (20) -1 (20) -0 (30)
3		30	3	0 (20) -1 (20) -2 (20) -0 (20)

Приклад заповнення трасування таблиці

t	сторінки ОП			сторінки ВНП			Пр.
0	ФС	процес	ВС	ФС	процес	ВС	
	0	ПІ	0 (30) а	0			
	1			1			
	т			...	...	...	
20	ФС	процес	ВС	ФС	процес	ВС	
	0	ПІ	0 (20)	0			
	1	П2	0 (20) а	1			
	"*						
30	ФС	процес	ВС	ФС	процес	ВС	
	0	ПІ	0 (20)	0			
	1	П2	0 (10)	1			
	2	ПЗ	0 (20) а				
50	ФС	процес	ВС	ФС	процес	ВС	А
	0			0	ПІ	0 (20)	
	1	П2	0 (10)	1			
	2	ПЗ	0 (20) а	. . .	• • •	• • •	
50	ФС	процес	ВС	ФС	процес	ВС	В
	0	ПЗ	1 (20) а	0	ПІ	0 (20)	
	1	П2	0 (10)	1			
	7	ПЗ	0	. . .	• • •	• • •	
70	ФС	процес	ВС	ФС	процес	ВС	З
	0	ПЗ	1	0	ПІ	0 (20)	
	1			1	П2	0 (10)	
	7	ПЗ	0	">	• • •	• • •	

70	ФС	процес	ВС	ФС	процес	ВС	D
	0	ПЗ	1	0	ПІ	0 (20)	
	1	ПЗ	2 (20) a	1	П2	0 (10)	
	2	ПЗ	0	?	•••	•••	
90							

Позначення приміток:

A - Запит ПЗ ВС1, вивантаження П1 ВС0;
B - Завантаження ПЗ ВС1;
C Запит ПЗ ВС2, вивантаження П2 ВС;
D - Завантаження ПЗ ВС2

Варіанти завдань - Додаток 1.6.

Контрольні питання

1. У чому полягає концепція віртуальної пам'яті?
2. Що таке сторінка ОП?
3. Де можуть розміщуватися сторінки процесу?
4. Яким чином виконується відображення сторінок?
5. Яким чином ОС враховує вільні і зайняті сторінки?
6. Для чого використовується свопинг сторінок?
7. Яким чином ОС вибирає сторінку для витіснення?
8. Які вимоги до апаратних засобів ЗС для організації сторінкової пам'яті?
9. Що таке сегмент?
10. Які алгоритми витіснення сторінок застосовуються в ОС?
11. У чому полягає ідея LRU алгоритму?
12. Які вимоги до апаратних засобів ВС для реалізації LRU алгоритму витіснення сторінок?
13. Наведіть приклад сегментної таблиці.
14. Наведіть приклад сторінкової таблиці.

РОЗДІЛ 3. УПРАВЛІННЯ ПРИСТРОЯМИ

Фактичне виконання операцій введення-виведення для зовнішніх пристроїв реалізується спеціальними модулями ОС, які назвали драйверами. Зважаючи на істотні відмінності зовнішніх пристроїв, кожен з типів зовнішніх пристроїв, підключених до ВС, має свій власний драйвер в складі ОС. Крім того, значно відрізняються драйвери пристроїв з блоковим (МД) або байтовим (клавіатура) введенням-виведенням. У своїй роботі драйвери використовують фізичні регістри контролерів зовнішніх пристроїв або канальні програми (при підключенні пристроїв за допомогою каналів), а також апаратні преривання зовнішніх пристроїв. Драйвери, як правило, мають кілька точок входу і одна з

них - по апаратному перериванню. Докладне вивчення організації драйверів виходить за рамки цього посібника.

У базі даних ОС кожен пристрій представлено спеціальним блоком управління, в якому зберігається його опис. Крім того, створюються блоки управління контроллером (каналом), які містять покажчики на блоки управління підключених до них пристроїв і робочі дані поточної операції.

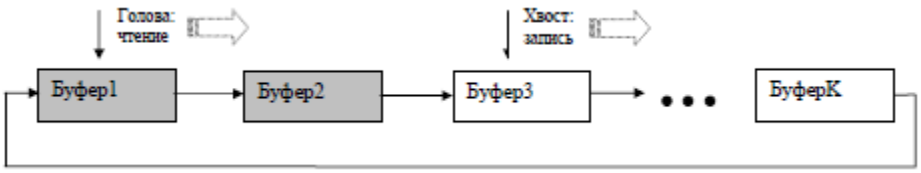
У цьому розділі вивчаються найбільш розповсюджені методи, використовувані більшістю модулів управління пристроями, такі як буферизація, кешування, планування операцій, а також особливості організації взаємодії ВС в мережі.

Заняття 3.1. Циклічна буферизация

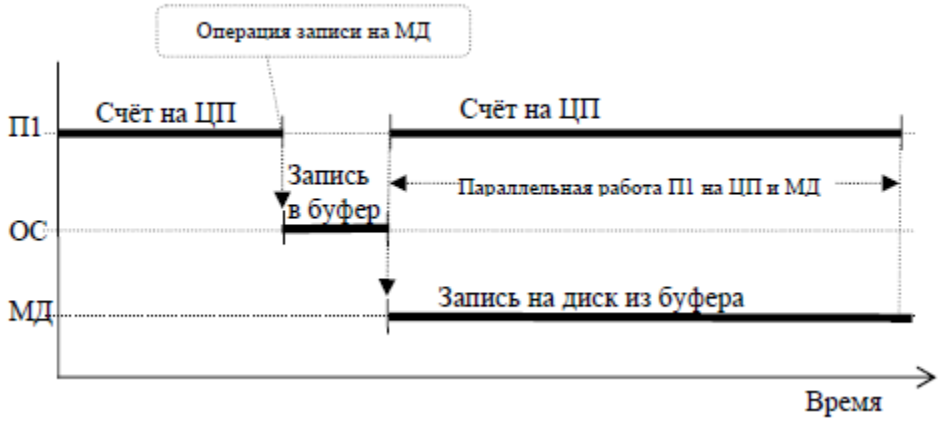
Мета заняття: засвоїти основи буферизації введення-виведення, оцінити ефективність застосування буферизації.

Короткий виклад теоретичного матеріалу

Буферизація є стандартним засобом сучасних ОС для узгодження швидкостей роботи пристроїв ВС. Буфер являє собою ділянку пам'яті для проміжного зберігання даних. Розрізняють буфери постійної і змінної довжини. Окремі буфери об'єднуються в зв'язкові списки. найбільш поширеною є *циклічна буферизация:*



Узгодження швидкостей забезпечується, наприклад, при виведенні на магнітний диск, за рахунок того, що після швидкого копіювання ОС виведених даних в буфер процес вважає операцію завершеною і продовжує своє виконання. Таким чином, фактичние виведення даних на пристрій виконується паралельно з роботою процесу:



Досить великий розмір буферу дозволяє значно прискорити виконання процесу. У тому випадку, якщо записані в буфер дані зчитуються до їх фактичної записи на диск, ОС забезпечує їх швидкий витяг з буфера. Слід зазначити, що інтенсивне введення-виведення може привести до заповнення буферів і неминучого очікування їх звільнення, але в середньому буферизація істотно підвищує ефективність роботи ВС. Для оцінки ефективності можна використовувати порівняння часів роботи процесу без буферизації і з використанням буферизації.

До недоліків буферизації можна віднести можливість втрати даних при краху ОС; в сучасних надійних ОС при використанні безперебійного живлення ВС ймовірність такої ситуації зведена до мінімуму. Проте, більшість сучасних ОС передбачалоють операцію примусового запису вмісту буферів.

Для узгодження швидкостей і прискорення роботи зовнішніх пристроїв ВС використовують також кешування. В цьому випадку блоки даних розміщуються в проміжному високошвидкісному сховищі (кеші) незв'язно; забезпечується швидкий асоціативний пошук за ключевою інформацією, наприклад, номерам логічних (фізичних) блоків зовнішніх пристроїв.

Завдання

Виконати ручну трасування роботи засобів управління зовнішніми пристроями. Заповнити трасувальну таблицю.

Характеристики ОС: циклічна буферизация, мультипрограмування.

Порядок виконання

1. Виконати ручне трасування роботи засобів буферізації
2. Заповнити трасувальні таблиці
3. Виконати аналіз ефективності буферизації
4. Сформулювати переваги і недоліки буферизації

Приклад виконання

Розмір буфера - 3 блоку.

Час виведення одного блоку на пристрій – 10.

Послідовність запису блоків процесом (ім'я блоку - затримка) A-5-B-5-C-5-D-5-E-30-F-5-G

Приклад заповнення трасувальної таблиці

Время	Состояние процесса	Состояние буфера			Состояние устройства
		0	1	2	
0	Запись A	A (голова)	(хвост)		Вывод A (10)
5	Запись B	A (голова)	B	(хвост)	Вывод A (5)
10	Запись C	(хвост)	B (голова)	C	Вывод B (10)
15	Запись D	D	B (гол. и хв.)	C	Вывод B (5)
20	Запись E	D	E	C (гол. и хв.)	Вывод C (10)
30	Счёт	D (голова)	E	(хвост)	Вывод D (10)
40	Счёт		E (голова)	(хвост)	Вывод E (10)
50	…	…	…	…	…

Варіанти завдань - Додаток 1.7.

Контрольні питання

1. Для чого застосовують буферизацию?
2. У чому полягає циклічна буферизация?
3. За рахунок чого забезпечується узгодження швидкостей роботи пристроїв при буферизації?
4. У чому полягають відмінності буферизації і кешування?

Заняття 3.2. Планування дискових операцій

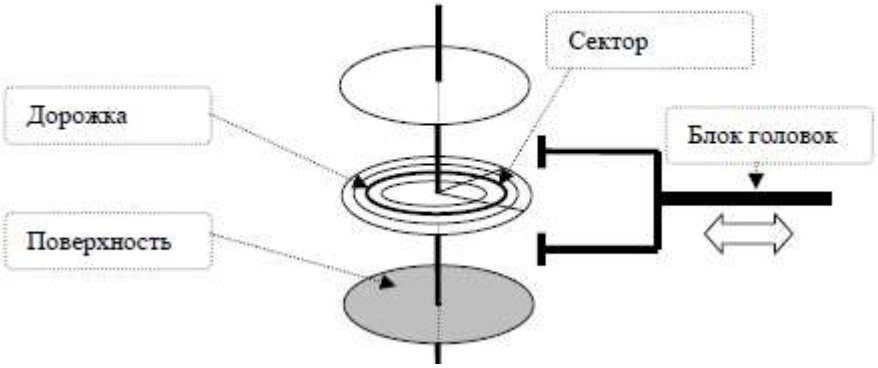
Мета заняття: засвоїти основи планування дискових операцій, оцінити ефективність застосування планування дискових операцій.

Короткий виклад теоретичного матеріалу

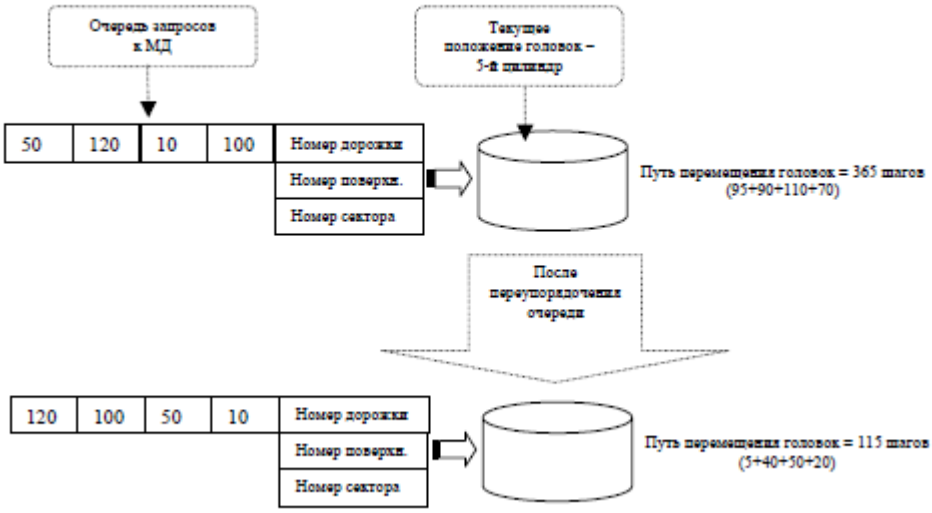
Планування є найбільш складною операцією ОС. Більшість ОС обмежуються реалізацією лише операцій виділення, звільнення і відстеження своїх ресурсів. Однак, використання операцій планування може істотно підвищити продуктивність ВС. Вивчимо засоби планування ОС на прикладі планування дискових операцій.

Магнітний диск уявляє собою складний електронно-механічний пристрій. На загальній осі зібрані кілька дисків з магнітними поверхнями; як правило, зовнішні поверхні нижнього і верхнього дисків не використовуються. Поверхні дисків розбиті на доріжки в формі концентричних кіл. Доступ до даних доріжок забезпечує блок магнітних головок, переміщуємий між

доріжками кроковим двигуном. Кожна доріжка розбита на сектори рівної довжини; сектор є мінімальною адресуємою одиницею магнітного диска і містить блок даних. Безліч рівновіддалених від осі доріжок всіх поверхонь називають циліндром. Таким чином, фізична адреса сектора (блоку даних) складається з номеру циліндра, номеру поверхні і номеру сектора:



Швидке обертання диска (1000 об / с) забезпечує порівняно високу швидкість читання-запису для всіх секторів поточного циліндру. Операція переміщення блоку головок до іншого циліндру вимагає включення крокового двигуна і займає порівняно більше (10 - 20 разів) час. Запити читання-запису магнітного диска розміщуються ОС в чергу. При відсутності планування вони виконуються в порядку надходження. Зауважимо, що такий спосіб реалізації запитів в більшості випадків є неефективним, оскільки призводить до частих переміщень блоку головок. Упорядкування черги запитами за критерієм мінімізації сумарного шляху переміщення головок дозволяє мінімізувати загальний час виконання всіх запитів. З огляду на різницю часів читання-запису і переміщення головок, можна прийти до висновку, що підвищення продуктивності є істотним:



Зауважимо, що можливо також додаткове планування номерів секторів поточного циліндру, але воно рідко використовується зважаючи на порівняно

високу швидкість обертання диску. Недоліком описаного простого підходу до планування є можливе збільшення часу введення-виведення окремих процесів. Таке збільшення небажано для процесів реального часу. Таким чином, при використанні пріоритетних дисциплін потрібні застосування більш складних критеріїв планування.

Завдання

Виконати ручне трасування роботи планувальника черги запитів до магнітного диску. Заповнити трасувальну таблицю.

Характеристики ОС: планування дискових операцій, мультипрограмування

Порядок виконання

1. Виконати ручне трасування роботи засобів планування дискових операцій.
2. Заповнити трасувальні таблиці.
3. Виконати аналіз ефективності планування дискових операцій.
4. Сформулювати переваги і недоліки планування дискових операцій.

Приклад виконання

Час переміщення головки на 1 циліндр - 1.

Час запису - 10. Послідовність операцій введення / виводу

Время	Очередь к МД	Операция МД
0		Движение к 100 (100)
20	150	Движение к 100 (80)
40	150, 120	Движение к 100 (60)
60	150, 120, 110	Движение к 100 (40)
80	200, 150, 120, 110	Движение к 100 (20)
100	10, 200, 150, 110	Запись 100 (10)
110	10, 200, 150	Движение к 110 (10)
120	10,200, 160, 150	Запись 110 (10)
130	…	…

Варіанти завдань - Додаток 1.8.

Контрольні питання

1. Для чого необхідне планування операцій МД?
2. З чого складається фізичну адресу даних на МД?
3. Яким чином виконується переупорядочение черзі до МД?
4. Як впливає планування операцій МД на продуктивність ВС?

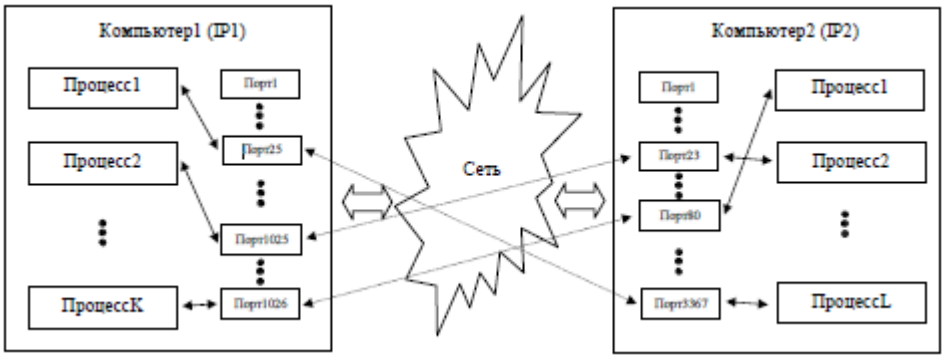
Заняття 3.3. Взаємодія комп'ютерів в мережі.

Мета заняття: засвоїти основи організації взаємодії комп'ютерів в мережі на основі технології клієнт-сервер.

Короткий виклад теоретичного матеріалу

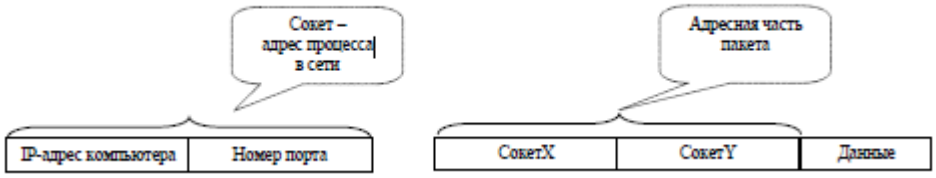
В даний час домінуючим сімейством мережевих протоколів є TCP / IP; стек протоколів TCP / IP реалізований у всіх сучасних ОС. Передача інформації в мережі заснована на взаємодії прикладних процесів на основі технології клієнт-сервер. Клієнт ініціює взаємодію і звертається із запитом на обслуговування до деякого сервера; сервер виконує запит і передає результати клієнту. Клієнтами і серверами є процеси операційної системи.

Таким чином, виникає задача адресації процесів в мережі. У мережі TCP / IP адресою процесу є виділений йому сокет (гніздо). Сокет складається з IP-адреси комп'ютера і номера порту. Порт є абстрактним об'єктом і служить для організації взаємодії. У сімействі протоколів TCP / IP порти з номерами 0-1024 зарезервовані і служать для адресації стандартних прикладних серверів, наприклад, порт 25 - сервер електронної пошти, протокол SMTP; порт 23 - служба віддаленого управління, протокол емуляції терміналу telnet; порт 80 - сервер передачі гіпертекстової (WWW) інформації, протокол HTTP. При запуску сервер займає відповідний порт і «слухає» його. Відповідному клієнту виділяється порт з випадковим номером, що перевищує 1024:

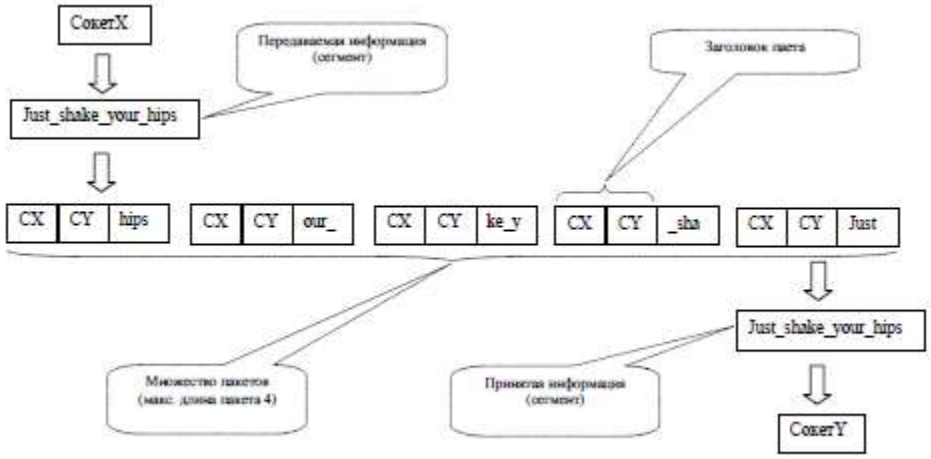


Адресну частину пакетів, переданих в мережі, можна уявити як пару: сокет відправника, сокет одержувача. Таким чином, забезпечується повна

адресація мережевих процесів, що забезпечує доставку і обробку інформації.



Крім того, мережа обмежує максимальну довжину пакету. Тому передана інформація розбивається на безліч пакетів, а після отримання збирається з прийнятих пакетів. Більш складні аспекти фрагментації, пов'язані з можливою зміною порядку доставки пакетів не розглядаються в даній роботі.



Зауважимо, що фізично комп'ютер повинен мати хоча б один мережевий інтерфейс, з яким асоційована IP-адреса. Таким інтерфейсом може бути Ethernet адаптер при використанні локальної мережі, модем, підключений до інтерфейсу USB (RS232), при віддаленому доступі.

Завдання

Виконати ручну трасування процесів взаємодії комп'ютерів в мережі. Заповнити трасувальні таблицю.

Характеристики ОС: технологія «клієнт-сервер», мульти-програмування.

Порядок виконання

1. Виконати ручне трасування роботи мережевих засобів ОС.
2. Заповнити трасувальні таблиці.
3. Оцінити додатковий обсяг інформації заголовків пакетів.
4. Сформулювати основні принципи адресації процесів в мережі.

Приклад виконання.

Час виконання запиту сервером – 40.
Сокет сервера (8,21).
Час передачі пакета в мережі – 40.
Розмір пакета - 5.
Час формування пакета - 1.
Клієнт 1: сокет (5,6); команда «get f1» - прочитати файл f1.
Клієнт 2: сокет (2,3); команда «get f2» - прочитати файл f2.
F1: "Extremes meet"
F2: "Practice make perfect"

Структура трасувальної таблиці.

Час	Клієнт 1	Клієнт 2	Мережа			сервер
			Відправ.	Получ.	дані	
0	"Get ft"		(5-6)	(8,21)	"Get f '	
1			(5.6)	(8,21)	"1"	
20		"Get £ 2"	(2,3)	(8,21)	"Get f '	
21			(2,3)	(8,21)	* T *> »»	
40						Отримання "getf”
41						Отримання "get f1" і виконання запиту
60						Отримання “get f”
61						Отримання "get f2" і Виконання запиту
31			(8,21)	(5,6)	"Extre"	Початок передачі fl
32			(8,21)	(5-6)	"Mes m"	
33			—	—	—	

Варіанти завдань - Додаток 1.9.

Контрольні питання

1. З чого складається адреса процесу в мережі?
2. Яким чином розподіляються порти комп'ютера?
3. Що таке сокет?

4. У чому полягає технологія клієнт-сервер?
5. З чого складається пакет інформації, що передається в мережі?

РОЗДІЛ 4. УПРАВЛІННЯ ІНФОРМАЦІЄЮ

Підсистема управління пристроями (також як і підсистема управління ОП) є захищеною і прихованої від кінцевого користувача і прикладних процесів в сучасних ОС. Користувачеві надаються засоби управління інформацією ОС з одиницею зберігання - файл. Слід зазначити також тенденцію уявлення пристроїв у вигляді спеціальних файлів (наприклад, в ОС Unix) для забезпечення обмеженого (і безпечного) доступу користувачів.

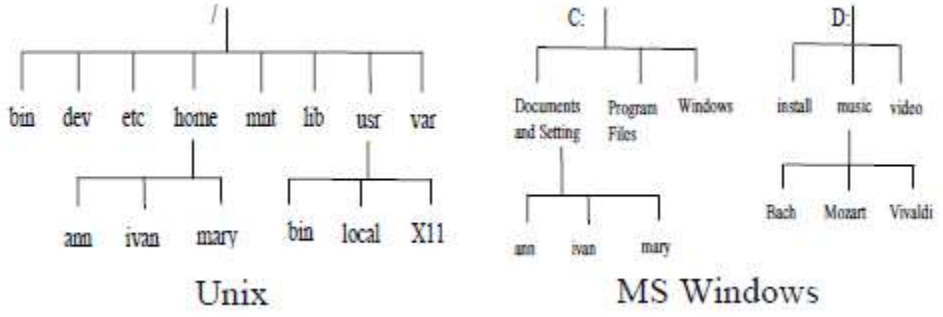
У базі даних ОС об'єкти файлової системи представлені блоками управління томом, каталогом, файлом, організовані в багатозв'язкові списки. Зауважимо, що ОС забезпечує «гарячий» захист інформації на фактично підключених пристроях. Безпека в випадку можливої несанкціонованої передачі пристрою (диска) зловмисникам може бути забезпечена тільки при використанні додаткових можливостей шифрування інформації, що зберігається.

Заняття 4.1. Файлова структура диска.

Мета заняття: засвоїти основи організації управління інформацією, виконати порівняльну оцінку різних способів організації файлової структури диска.

Короткий виклад теоретичного матеріалу

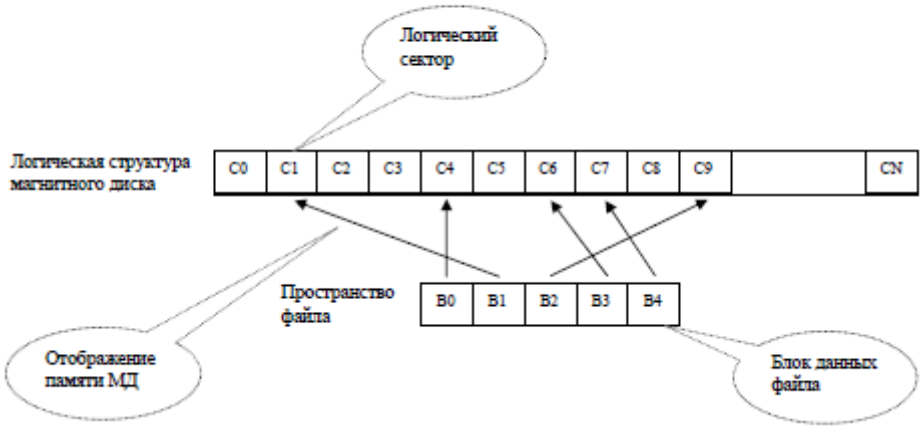
Деревоподібна (ієрархічна) файлова система, утворена такими елементами як том, каталог, файл, є стандартною для сучасних ОС. Файл являє собою пойменовану одиницю зберігання інформації; файли об'єднуються в каталоги, причому каталог може містити як файли, так і інші каталоги; том являє собою пристрій з файлової системою. Єдина різниця між родинами домінуючих в даний час ОС Unix і MS Windows полягає в представленні томів: Unix використовує загальну ієрархію всіх томів, MS Windows представляє файлову систему з розбивкою по томам (пристроїв):



Як правило, сучасні ОС реалізують такі основні файлові операції як читання-запис і позиціонування всередині файлу, забезпечуючи послідовний і

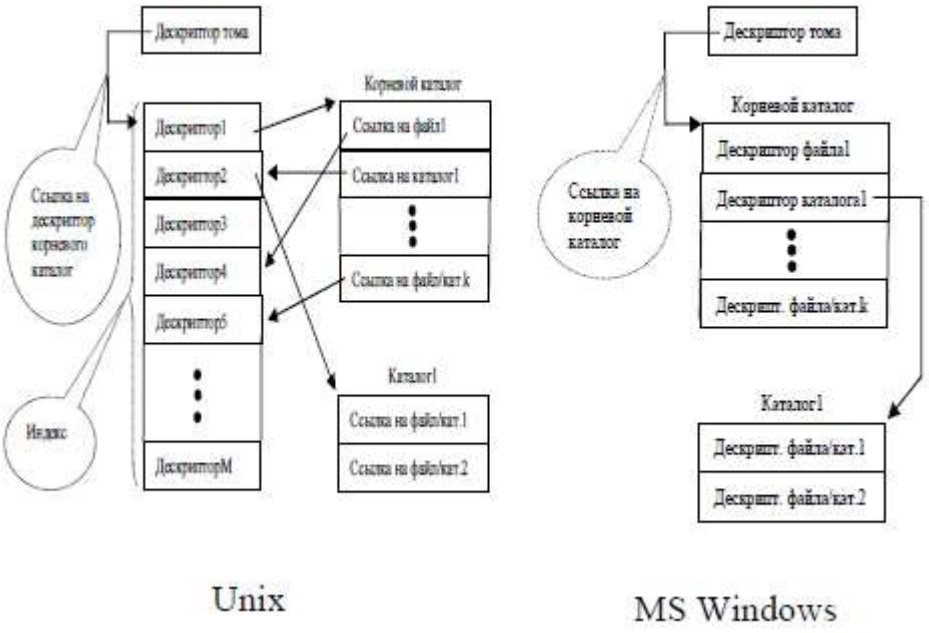
прямий доступ до інформації. Більш складні методи доступу реалізуються СУБД на основі зазначених операцій ОС.

Логічна структура диска може бути представлена одновимірним масивом секторів створеним занумерованими фізичними секторами всіх поверхонь, циліндрів і доріжок. Виникає проблема відображення файлової системи на логічну структуру диска для забезпечення доступу до файлів:



Для забезпечення відображення створюються спеціальні структури даних, що описують відображення і невидимі для кінцевого користувача. Спосіб організації таких даних прийнято називати файловою структурою. Файлова структура забезпечує вирішення двох основних завдань: знаходження логічних секторів заданого файлу / каталогу на диску; облік вільних / зайнятих секторів.

Більшість файлових структур використовує спеціальний блок - дескриптор для кожного зі своїх об’єктів: том, каталог, файл. Дескриптор файлу / каталогу містить ім'я, власника, права доступу, дати створення / коригування, довжину. Дескриптор тому містить також покажчик на дескриптор кореневого каталогу. Дескриптори файлу / каталогу мають фіксовану довжину. ОС використовують два основних підходу до розміщення дескрипторів: в ОС сімейства Unix дескриптори файлів / каталогів розміщуються в загальному одновимірному масиві - індексі, в цьому випадку каталог містить список покажчиків на дескриптори його файлів / каталогів; в ОС сімейства MS Windows дескриптори файлів / каталогів розміщуються всередині каталогу, що їх містить:



Unix MS Windows

При вирішенні задачі відображення для зменшення розміру структур даних застосовують кластеризацию. Кластер складається з декількох блоків (секторів) і є одиницею виділення простору диска. В теперішній час домінують два підходи до опису розміщення файлу на диску: використання загальної таблиці розміщення для всіх кластерів (в ОС сімейства MS Windows); використання масиву описувачів зв'язних ділянок в дескрипторі файлу (в ОС сімейства Unix).

Загальна таблиця розміщення містить по одному запису для кожного кластера. Запис дорівнює нулю, якщо відповідний кластер вільний. Якщо кластер належить до деякого файлу, то відповідний запис містить номер наступного кластера; запис останнього кластера файлу має спеціальне значення - ознаку кінця ланцюжка. (наприклад, всі двійкові одиниці).

Використання таблиці зв'язкових ділянок, описаних зазначенням початкового кластера ділянки і довжини в дескрипторі файлу, забезпечує більш швидке відображення. Однак суттєвим недоліком цього методу є фіксоване число ділянок в дескрипторі файлу. Розміщення великих фрагментованих файлів вирішується в ОС Unix за рахунок виділення додаткових дескрипторів.

Таблиця розміщення файлів (File Allocation Table)

0	1	2	3	4	5	6	7	8	9
0	0	3	4	8	0	-	0	9	6

Таблиця зв'язкових ділянок (в дескрипторі)

Нач.	2	8	6
Дл.	3	2	1

Використання єдиної таблиці розміщення забезпечує також рішення задачі обліку вільних / зайнятих кластерів: записи вільних кластерів мають нульове значення в таблиці. До додаткових способів обліку вільного / зайнятого простору можна віднести бітові карти і спеціальний (фіктивний) файл, який займає всі вільні кластери. У бітової карті кожен біт відповідає кластеру; значення 0 - кластер вільний, значення 1 - зайнятий. Застосування спеціального файлу забезпечує переваги швидкого пошуку найбільш відповідної ділянки при виділенні простору і сприяє зменшенню фрагментації.

Завдання.

Виконати ручне трасування роботи файлової системи. Заповнити трасувальні таблицю.

I. Характеристики ОС: незв’язне виділення дискового простору, таблиця зв'язаних ділянок в дескрипторі файлу, спеціальний файл обліку вільного простору, мультипрограмування.

II. Характеристики ОС: незв’язне виділення дискового простору, окрема таблиця розміщення файлів, облік вільного простору в таблиці розміщення, мультипрограмування.

Порядок виконання

1. Виконати ручне трасування роботи мережевих засобів ОС.
2. Заповнити трасувальні таблиці.
3. Оцінити додатковий обсяг інформації заголовків пакетів.
4. Сформулювати основні принципи адресації процесів в мережі.

Приклад виконання.

Послідовність операцій:
Створити файл 1. Записати 3 блоку в файл 1. Створити файл 2.
Записати 2 блоку в файл 2. Дописати 1 блок в файл 1.
Дописати 2 блоку в файл 2. Видалити файл 1.
Доступний простір диска - 10

I. Приклад заповнення трасувальної таблиці

операція	Дескриптори файлів		
	файл 0	файл 1	файл 2
	A, (0.10)		
Створити файл 1	A, (0.10)	F1	
Записати 3 блоку в файл 1	A. (3.7)	F 1. (0.3)	
Створити файл 2	A. (3.7)	F 1. (0.3)	F2
Записати 2 блоку в файл 2	| A, (5.5)	F 1. (0.3)	F2. (3.2)
Дописати 1 блок в файл 1	A. (6.4)	F1. (0.3). (5.1)	F2, (3.2)
...	...	...	...

II. Приклад заповнення трасувалюної таблиці

операція	Код операції
Створити файл 1	O1
Записати 3 блоку ь файл 1	O2
Створити файл 2	O3
Записати 2 блоки ь файл 2	O4
Дописати 1 блок в файл 1	O5

Код операції	Дескриптори файлів		
	файл 0	файл 1	файл 2
	A, (0,10)		
Ol	A. (0.10)	F1	
O2	A. (3.7)	F 1. (0.3)	
O3	A. (3.7)	F 1. (0.3)	F2
O4	A. (5.5)	F 1. (0.3)	F2. (3.2)
O5	A. (6.4)	F1. (0.3). (5.1)	F2. (3.2)

Код операції	Дескриптори файлів			Таблиця розміщення файлів							
	0	1	2	0	1	2	3	4	5	6	7
				0	0		0	0	0	0	0
Ol	F1										
O2	F1, 0			1	2	-	0	0	0	0	0
O3	F1, 0	F2		1	2	-	0	0	0	0	0

O4	F1, 0	F 2. 3		1	2	-	4	-	0	0	0
O5	F1, 0	F 2. 3		1	2	5	4	-	-	0	0
•••	•••	•••	•••								

Варіанти завдань - Додаток 1.10.

Контрольні питання

1. Які основні елементи файлової системи?
2. Для чого необхідна спеціальна файлова структура диска?
3. Яку інформацію містить дескриптор файлу (тому, каталогу)?
4. Як організовуються дескриптори для опису деревовидної файлової системи?
5. Що таке кластер?
6. Яким чином описується фактичне розміщення файлу на диску?
7. Які засоби використовуються для обліку вільного / зайнятого простору диска?
8. Яким чином забезпечується захист файлової системи?

РОЗДІЛ 5. ПЕРЕТВОРЕННЯ ГРАМАТИК

Мета: - закріпити поняття «еквівалентні граматики», «наведена КВ-граматика»;

- сформувати вміння і навички еквівалентних перетворень контекстно-вільних граматик.

Основи теорії

Визначення 5.1. КВ-граматика називається наведеною, якщо вона не має циклів, ε-правил та непотрібних символів.

Розглянемо основні алгоритми приведення КВ-граматик.

Перед усіма іншими дослідженнями і перетвореннями КВ-граматик виконується перевірка існування мови граматики.

Алгоритм 5.1. Перевірка існування мови граматики

Вхід: КВ-граматика $G = (V_T, V_N, P, S)$.

Вихід: висновок про існування або відсутність мови граматики.

Визначимо множину нетерміналів, що породжують термінальні рядки $N = \{Z \mid Z \in V_N, Z \Rightarrow^* x, x \in V_T^*\}$.

Крок 1. Покласти N0 = Ø.

Крок 2. Обчислити $N_i = N_{i-1} \cup \{A \mid (A \to \alpha) \in P \text{ і } \alpha \in (N_{i-1} \cup V_T)^*\}$.

Крок 3. Якщо $N_i \neq N_{i-1}$, то покласти i = i + 1 і перейти до пункту 2, інакше вважати $N = N_i$.

якщо $S \in N$, То видати повідомлення про те, що мова граматики існує, інакше повідомити про відсутність мови.

Приклад 5.1. дана граматика $G = (\{0,1\}, \{S, A, B\}, P, S)$, де множина правил P: 1) $S \to AB$; 2) $A \to 0A$; 3) $A \to 0$; 4) $B \to 1$. Побудуємо послідовність наближень множини N:

$N_0 = \varnothing$;

N_1 = {A, B};

N_2 = {S, A, B};

N_3 = {S, A, B}.

Оскільки N2 = N3, то N = {S, A, B}, отже, мова граматики існує, тому що початковий символ $S \in N$.

Визначення 5.2. Марними символами граматики називають:

а) нетермінали, що не породжують термінальних рядків, тобто множину символів

$$\{X \mid X \in V_N, \neg\exists(X \Rightarrow^* x), x \in V_T^*\};$$

б) недосяжні нетермінали, які породжують термінальні рядки, тобто безліч символів

$$\{X \mid X \in V_N, \neg\exists(S \Rightarrow^* \alpha X \beta), \exists(X \Rightarrow^* x); \alpha, \beta \in V^*; x \in V_T^*\};$$

в) недосяжні термінали, тобто безліч символів

$$\{X \mid X \in V_T, \neg\exists(S \Rightarrow^* \alpha X \beta); \alpha, \beta \in V^*\}.$$

Алгоритм 5.2. Усунення нетерміналів, що не породжують термінальних рядків

Вхід: КВ-граматика $G = (V_T, V_N, P, S)$.

Вихід: КВ-граматика $G' = (V_T, V_N', P', S)$, Така, що $L(G') = L(G)$ і для всіх $Z \in V_N'$ існують висновки $Z \Rightarrow^* x$, де $x \in V_T^*$.

Крок 1. Визначити безліч нетерміналів, що породжують термінальні рядки, за допомогою алгоритму 4.1.

Крок 2. Обчислити $V_N' = V_N \cap N$, $N_Б = V_N - V_N'$, $P' = P - P_Б$, де $P_Б \subseteq P$ - це безліч правил, що містять непотрібні нетермінали $X \in N_Б$.

Приклад 5.2. дана граматика $G = (\{a, b, c\}, \{S, A, B, C\}, P, S)$ з правилами P: 1) $S \to ab$; 2) $S \to AC$; 3) $A \to AB$; 4) $B \to b$; 5) $C \to cb$.

Перетворимо її в еквівалентну граматику G' за алгоритмом 4.2:

$N_0 = \varnothing$;

N_1 = {S, B, C};

N_2 = {S, B, C}.

Оскільки N1 = N2, то N = {S, B, C}. Після видалення непотрібних нетерміналів і правил виведення, отримаємо граматику $G' = (\{a, b, c\}, \{S, B, C\}, P', S)$ з правилами P': 1) $S \to ab$; 2) $B \to b$; 3) $C \to cb$.

Алгоритм 5.3. Усунення недосяжних символів

Вхід: КВ-граматика $G = (V_T, V_N, P, S)$.

Вихід: КВ-граматика $G' = (V'_T, V'_N, P', S)$, Така, що $L(G') = L(G)$ і для всіх $Z \in V'$ існує висновок $S \Rightarrow^* \alpha Z \beta$, де $\alpha, \beta \in (V')^*$.

Визначимо множину досяжних символів Z граматики G, тобто множину

$$W = \{Z \mid Z \in V, \exists (S \Rightarrow^* \alpha Z \beta); \alpha, \beta \in V^*\}.$$

Крок 1. Покласти $W_0 = S$.

Крок 2. Обчислити чергове наближення наступним чином:

$$W_i = W_{i-1} \cup \{X \mid X \in V, (A \to \alpha X \beta) \in P, A \in W_{i-1}; \alpha, \beta \in V^*\}.$$

Крок 3. Якщо $W_i \neq W_{i-1}$, то покласти i: = i + 1 і перейти до кроку 2, інакше вважати $W = W_i$.

Крок 4. Обчислити $V'_N = V_N \cap W, V'_T = V_T \cap W, V_Б = V - W, P' = P - P_Б$, де $P_Б \subseteq P$ - це безліч правил, що містять недосяжні символи $X \in V_Б$.

Приклад 5.3. дана граматика $G = (\{a, b, c\}, \{S, B, C\}, P, S)$ з правилами P': 1) $S \to ab$; 2) $B \to b$; 5) $C \to cb$.

Перетворимо її в еквівалентну граматику G' за алгоритмом 4.3:

W_0 = {S};

W_1 = {S, a, b};

W_2 = {S, a, b}.

Оскільки W1 = W2, то W = {S, a, b}. Множина недосяжних символів $V_Б = \{B, C, c\}$. Тоді після видалення недосяжних символів, отримаємо граматику $G' = (\{a, b\}, \{S\}, P, S)$ з правилом P': $S \to ab$.

Алгоритм 5.4. Усунення ε-правил

Вхід: КВ-граматика $G = (V_T, V_N, P, S)$.

Вихід: Еквівалентна КВ-граматика $G' = (V_T, V'_N, P', S')$ без ε-правил для всіх нетермінальних символів, крім початкового, який не повинен зустрічатися в правих частинах правил граматики.

Крок 1. У вихідній граматиці G знайти ε-породжуючі нетермінальні символи $A \in V_N$, Такі, що $A \Rightarrow^* \varepsilon$.

1.1 покласти $N_0 = \{A \mid (A \to \varepsilon) \in P\}$.

1.2 обчислити $N_i = N_{i-1} \cup \{B \mid (B \to \alpha) \in P, \alpha \in N_{i-1}^*\}$.

1.3 якщо $N_i \neq N_{i-1}$, То покласти i: = i + 1 і перейти до пункту 1.2, інакше вважати $N = N_i$.

Крок 2. З множини P правил вихідної граматики G перенести в множину P' всі правила, за винятком ε-правил, тобто $P' = P - \{(A \to \varepsilon) \in P$ для всіх $A \in V_N\}$.

Крок 3. Поповнити безліч P' правилами, які виходять з кожного правила цієї множини шляхом виключення всіляких комбінацій ε-породжуючих нетерміналів в правій частині. Отримані при цьому ε-правила в множині P' не вмикати.

Крок 4. Якщо $S \in N$, то $P' = P \cup \{S' \to \varepsilon, S' \to S\}, V_N' = V_N \cup S'$, де $V \cap \{S'\} = \varnothing$; інакше $V_N' = V_N, S' = S$.

Приклад 5.4. дана граматика $G = (\{0,1\}, \{S, A, B\}, P, S)$ з правилами P: 1) $S \to AB$; 2) $A \to 0A \mid \varepsilon$; 3) $B \to 1B \mid \varepsilon$. Перетворимо її в еквівалентну граматику за алгоритмом 4.4.

Крок 1. N0 = {A, B};

N_1 = {S, A, B};

N_2 = {S, A, B}.

Оскільки N1 = N2, то шукана множина побудована і N = {S, A, B}.

Крок 2, 3. Множина P': 1) $S \to AB \mid A \mid B$; 2) $A \to 0A \mid 0$; 3) $B \to 1B \mid 1$.

Крок 4. Оскільки $S \in N$, То введемо новий нетермінал C і поповнимо множину P' правилом виду $C \to S \mid \varepsilon$. Результуюча граматика матиме вигляд: $G' = (\{0,1\}, \{S, A, B, C\}, P, C)$ з правилами P': 1) $C \to S \mid \varepsilon$; 2) $S \to AB \mid A \mid B$; 3) $A \to 0A \mid 0$; 4) $B \to 1B \mid 1$.

Алгоритм 5.5. Усунення ланцюгових правил

Вхід: КВ-граматика $G = (V_T, V_N, P, S)$.

Вихід: Еквівалентна КВ-граматика $G' = (V_T, V_N', P', S')$ без ланцюгових правил, Тобто правил виду $A \to B$, де $A, B \in V_N$.

Крок 1. Для кожного нетерміналу A обчислити множину виведених з нього нетерміналів, тобтомножину $N^A = \{B \mid A \Rightarrow^* B,$ де $B \in V_N\}$.

1.1 покласти $N_0^A = \{A\}$.

1.2 обчислити

$$N_i^A = N_{i-1}^A \cup \{C \mid (B \to C) \in P, B \in N_{i-1}^A, C \in V_N\}.$$

1.3 якщо $N_i^A \neq N_{i-1}^A$, то покласти i: = i + 1 і перейти до пункту 1.2, інакше вважати $N^A = N_i^A$.

Крок 2. Побудувати множину P' так: якщо $(B \to \alpha) \in P$ не є ланцюговим правилом $(\alpha \notin V_N)$, То включити в P' правило $A \to \alpha$ для кожного A, Такого, що $B \in N^A$.

Приклад 5.5. граматика $G = (\{+, n\}, \{L, M, N\}, P, L)$ з правилами P: 1) $L \to M$; 2) $M \to N$; 3) $N \to N+ \mid n$. Перетворимо її в еквівалентну граматику G' за алгоритмом 4.5.

Крок 1. $N_0^L = \{L\}$;

$N_1^L = \{L, M\}$;

$N_2^L = \{L, M, N\}$;

$N_3^L = \{L, M, N\}$.

Оскільки $N_2^L = N_3^L$, то $N^L = \{L, M, N\}$.

$N_0^M = \{M\}$;

$N_1^M = \{M, N\}$;

$N_2^M = \{M, N\}$.

Оскільки $N_1^M = N_2^M$, то $N^M = \{M, N\}$.

$N_0^N = \{N\}$;

$N_1^N = \{N\}$.

Оскільки $N_1^N = N_0^N$, то $N^N = \{N\}$.

Крок 2. Перетворивши правила виведення граматики, отримаємо граматику $G' = (\{+, n\}, \{L, M, N\}, P', L)$ з правилами P':

1) $L \to N+ \mid n$; 2) $M \to N+ \mid n$; 3) $N \to N+ \mid n$.

Алгоритм 5.6. Усунення лівої факторизации правил

Вхід: КВ-граматика $G = (V_T, V_N, P, S)$.

Вихід: Еквівалентна КВ-граматика $G' = (V_T, V_N', P', S')$ без однакових префіксів в правих частинах правил, що визначають нетермінали.

Крок 1. Записати всі правила для нетермінала X, що мають однакові префікси $\alpha \in V^*$, У вигляді одного правила з альтернативами: $X \to \alpha\beta_1 \mid \alpha\beta_2 \mid \ldots \mid \alpha\beta_n$; $\beta_1, \beta_2, \ldots, \beta_n \in V^*$.

Крок 2. Винести за дужки вліво префікс α в кожному рядку-альтернативі: $X \to \alpha(\beta_1 \mid \beta_2 \mid \ldots \mid \beta_n)$.

Крок 3. Визначити новим нетерміналом Y вираз, що залишився в дужках: $X \to \alpha Y, Y \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$.

Крок 4. Поповнити безліч нетерміналів новим нетерміналом Y і замінити правила, які зазнали факторизации, новими правилами для X і Y.

Крок 5. Повторити кроки 1-4 для всіх нетерміналів граматики, для яких це можливо і необхідно.

Приклад 5.6. Дана граматика $G = (\{k,l,m,n\}, \{S\}, P, S)$ з правилами P: 1) $S \to kSl$; 2) $S \to kSm$; 3) $S \to n$. Перетворимо її в еквівалентну граматику G' за алгоритмом 5.6:

Крок 1. $S \to kSl \mid kSm \mid n$.

Крок 2. $S \to kS(l \mid m) \mid n$.

Крок 3,4. Поповнивши безліч нетерміналів новим нетерміналом C і замінивши правила, які зазнали факторизации, отримаємо граматику $G' = (\{k,l,m,n\}, \{S, C\}, P', S)$ з правилами P': 1) $S \to kSC$; 2) $S \to n$; 3) $C \to l$; 4) $C \to m$.

Алгоритм 5.7. Усунення прямий лівою рекурсії

Вхід: КВ-граматика $G = (V_T, V_N, P, S)$.

Вихід: Еквівалентна КС-граматика $G' = (V_T, V_N', P', S')$ без прямої лівої рекурсії, тобто без правил виду $A \to A\alpha,\ A \in V_N,\ \alpha \in V^*$.

Крок 1. Вивести з граматики всі правила для рекурсивного нетермінала X:

$X \to X\alpha_1 \mid X\alpha_2 \mid \ldots \mid X\alpha_m \quad (X \in V_N; \alpha_1, \alpha_2, \ldots, \alpha_m \in V^*)$

$X \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n \quad (\beta_1, \beta_2, \ldots, \beta_n \in V^*)$.

Крок 2. Внести новий нетермінал Y так, щоб він описував будь-який «хвіст» рядка, що породжується рекурсивним нетерміналом X:

$Y \to \alpha_1 Y \mid \alpha_2 Y \mid \ldots \mid \alpha_m Y$

$Y \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m$.

Крок 3. Замінити в рекурсивному правилі для X праву частину, використовуючи новий нетермінал і все нерекурсивні правила для X так, щоб генерується язик не змінився:

$X \to \beta_1 Y \mid \beta_2 Y \mid \ldots \mid \beta_n Y$

$X \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

$Y \to \alpha_1 Y \mid \alpha_2 Y \mid \ldots \mid \alpha_m Y$

$Y \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m$.

Крок 4. Поповнити множину нетерміналів граматики новим нетерміналом Y. Поповнити безліч правил граматики правилами, отриманими на кроці 3.

Крок 5. Повторити дії кроків 1-4 для всіх рекурсивних нетерміналів граматики, після чого отримані безлічі нетерміналів і правил прийняти в якості V_N' і P'.

Приклад 4.7. дана граматика $G = (\{a, b, c, d, z\}, \{S, A, B, C\}, P, S)$ з правилами P: 1) $S \to Aa$; 2) $A \to Bb$; 3) $B \to Cc \mid d$; 4) $C \to Ccbz \mid dbz$. Після усунення прямої лівої рекурсії отримаємо еквівалентну граматику $G' = (\{a, b, c, d, z\}, \{S, A, B, C, Z\}, P', S)$ з правилами P':

1) $S \to Aa$; 2) $A \to Bb$; 3) $B \to Cc \mid d$; 4) $C \to dbzZ \mid dbz$; 5) $Z \to cbzZ \mid cbz$.

Постановка завдання до лабораторної роботи № 5

Розробити програмний засіб, що автоматизує процес еквівалентного перетворення КВ-граматик. Програмний засіб має виконувати наступні функції:

1) організація введення граматики і перевірка її на приналежність до класу КС-граматик;

2) перевірка існування мови КС-граматики;

3) реалізація еквівалентних перетворень граматики, спрямованих на видалення:

а) непотрібних символів;

б) недосяжних символів;

в) ε-правил;

г) ланцюгових правил;

д) лівої факторизации правил;

е) прямої лівою рекурсії.

Варіанти індивідуальних завдань представлені в таблиці 5.1.

Таблиця 5.1 - Варіанти індивідуальних завдань до лабораторної роботи № 5

Варіант	Контекстно-вільна граматика
1	*G*= ({S, A, B, D, E}, {a, b, c, e}, P, S), де P: 1) S→*AB* | *ε*; 2) A→*Aa* | *S* | *a*; 3) B→*bD* | *bS*| b; 4) D→*ccD*; 5) E→*eE* | *e*.
2	*G*= ({E, T, F, G, H}, {+, -, *, /, n, m, h}, P, E), де P: 1) E→*T*| E + T | ET |ε; 2) T→*F*| F * T | F / T |ε; 3) F→*G*| Fn | n; 4) G→*Gm*; 5) H→*Hh*| h.
3	*G*= ({S, R, T, X, Y}, {a, b, p, g, y}, P, S), де P: 1) $S \to R \mid T$; 2) $R \to pX \mid paR \mid paT \mid \varepsilon$ 3) $T \to Tg \mid g$; 4) $X \to aXb$; 5) $Y \to aYa \mid y$.

4	*G*= ({Q, A, B, C, D}, {a, b, c, d}, P, Q), де P: 1) $Q \to acA \mid acB \mid \varepsilon$; 2) $B \to A \mid Cb \mid \varepsilon$; 3) $A \to Aa \mid Ab \mid a$; 4) $C \to dCc$ 5)$D \to dc$
5	*G*= ({R, T, F, G, K}, {m, i, j, k, ^, ~,⊥}, P, R), де P: 1) R→*R*~ T⊥	*R*^ T⊥	ε; 2) T→*F*	*Fi*	*Fj*	*Gk*	ε; 3) G→*GkG*; 4) K→*Ki*	*Km*	*m*.
6	*G*= ({S, X, Y, Z, K}, {x, y, z, k, #, $}, P, S), де P: 1) S→*X*	*Y*	*Z*; 2) X→*x # X*	*x # Y*	ε; 3) Y→*Yy*$	Yz $	$	ε; 4) Z→*Zz*$; 5) K→*Kk*$	k $.
7	*G*= ({S, L, M, P, N}, {n, m, l, p, @,⊥}, V, S), де V: 1) S→@nL	@mM	P; 2) L→*M*	*Ll*⊥	*Lm*⊥	ε; 3) M→*L*	*Mm*	*mm*; 4) N→*pN*@	@; 5) P→*nmP*.
8	*G*= ({X, Y, Z, K, L}, {a, b, l, =, <,>,∧, ∨, ¬}, V, X), де V: 1) X→*Y*	*Y*= Y	Y <Y	Y> Y	K; 2) Y→*Y*∧*Z*	*Y*∨ *Z*	ε; 3) Z→ ¬ *a* | ¬ *b*	ε; 4) K→ ¬ *K*; 5) L→*l*	*a*	*b*.
9	*G*= ({Q, A, B, C, D}, {0, 1, -}, P, Q), де P: 1) Q→01A	01B	A; 2) A→0B1	B	1 |ε; 3) B→*BA*0	B1	C |ε; 4) C→0C11; 5) D→- D1	-0	-1.
10	*G*= ({R, T, U, W, V}, {0, 1, +, -, *, /}, P, R), де P: 1) R→*T*1T	T1U	W |ε; 2) T→*U*	*T*01	T10 |ε; 3) U→+ U	+0	+1 4) W→*W*-W	W + W; 5) V→* 0	/ 1.

Продовження таблиці 5.1 - Варіанти індивідуальних завдань до лабораторної роботи № 5

Варіант	Контекстно-вільна граматика
11	*G*= ({S, R, T, F, E}, {a, b, k, {, [,},],⊥}, P, S), де P: 1) S→{R	[R; 2) R→*Ra*}	Ra]	a	T	F |ε; 3) F→{F}	bb; 4) T→[T]; 5) E→*k*⊥.
12	*G*= ({Y, K, M, L, S}, {a, b, *, /, ^}, P, Y), де P: 1) Y→*KS*	*KM*; 2) K→*K**	K /	S; 3) S→*Sa*/	Sb /	ε; 4) M→* M *; 5) L→*L* ^	^ *a*.

ДОДАТКИ

Додаток 1. Варіанти завдань

Д.1.1. Мультипрограмування

Структура процесів:
З1: С1ЦП-D1МД-С2ЦП
З2: С3ЦП-L1МЛ-С4ЦП
З3: С5ЦП-D2МД-С6ЦП-D3МД-С7ЦП
З4: С3ЦП-L2МЛ-С4ЦП-D2МД-С1ЦП

Ci = (((n + i) mod 5) +1) * 10
Di = (((n + i) mod 4) +1) * 100
Li = (((n + i) mod 3) +1) * 1000
n - номер студента в журналі
Варіант №29.
Таблиця даних 1.

№	З	D	L
1	10	300	1000
2	20	400	2000
3	30	100	
4	40		
5	50		
6	10		
n	20		

Структура завдання:
З1: 10ЦП-300МД-20ЦП
З2: 30ЦП-1000мл-40ЦП
З3: 50ЦП-400МД-10ЦП-100МД-20ЦП
З4: 30ЦП-2000мл-40ЦП-400МД-10ЦП

Д.1.2. Циклічне квантування часу

Варіанти - Додаток 1.1.
Розмір кванта:

dt = ((n mod 2) + 1) * 10

Час перемикання:
tos = ((n mod 4) + 2)

Д.1.3. Пріоритетні дисципліни

Варіанти - Додаток 1.1.

Пріоритети завдань Пі:

Пі = ((n + i) mod 3) + 1

Пріоритет 3 - абсолютний.

Д.1.4. динамічні розділи

Розмір ОП: V = ((n mod 4) + 7) * 10

Характеристики виконуваних процесів

номер (I)	час надходження (Ti) '	час виконання	Розмір в ОП	пріоритет
1	0	(((N + i) mod 4) +2) * 10	(((N + i) mod 3) + 3) * 10	((N + i) mod 3) + l
2	t.-i + (((n + i) mod2) + l) * 10	...	. . .	…
3	…	...	…	…
4	...	…	…	

Д.1.5. свопінг процесів

Варіанти - Додаток 1.4.
Пріоритет 3 - абсолютний.
Час завантаження / розвантаження: tio = ((n mod 2) + 1) * 10

Д.1.6. сторінкова пам'ять

Кількість сторінок ОП: NO = ((n mod 3) + 3
Кількість сторінок ВНП: NE = ((n mod 4) + 5

Характеристики виконуваних процесів

номер (I)	Час надходження (ti)	К-ть сторінок	Послідовність дій
1	0	.	0 (C1) -1 (C2) -0 (C3)
2	Ti-1 + (((n + i) mod 2) +1) * 10	2	0 (C4) -1 (C5) -0 (C6)
3	...	3	0 (C7) -1 (C1) -2 (C2) -0 (C3)

4	…		0 (C6) -1 (C4) -0 (C2)

Сі з Додатка 1.1.

Д.1.7. циклічна буферизация.

Розмір буфера (блоків): BS = (n mod 3) + 2
Час виведення одного блоку на пристрій: TIO = ((n mod 2) + + 1) * 10
Послідовність запису блоків процесом (ім'я блоку - затримка) A-t1-B-t2-C-t3-D-t4-E-t5-F-t6-G ti = ((n mod 4) + 1) * 5

Д.1.8. Планування дискових операцій.

Час переміщення головки на 1 циліндр: TC = (n mod 2) +1
Час запису: TIO = ((n mod 3) + 1) * 5
Послідовність операцій введення / виводу

номер (i)	1	2	3	4	5	6	7	8
час (tO	0	ti + ((n + i) mod 3) +1) * 10						
Цилиндр	((П + i) * 100 + i * 10 + (п + i)) mod 200	…						

Д.1.9. Взаємодія комп'ютерів в мережі

Час виконання запиту сервером: TSer = ((n mod 3) + +2) * 10
Сокет сервера (n, 21)
Час передачі пакета в мережі: TNet = ((n mod 4) +1) * 10
Розмір пакета: Psize = (n mod 3) + 4
Час формування пакета: Pio = (n mod 2) + 2
Клієнт 1: сокет (n + 1,2 * n); команда «get f1» - прочитати файл f1
Клієнт 2: сокет (n + 2,3 * n); команда «get f2» - прочитати файл f2
F1: "Nothing succeeds like success"
F2: "The bait hides the hook"

Д.1.10. Файлова структура диска

Послідовність операцій:
Створити файл 1. Записати ((n mod 2) +2) блоку в файл 1.

Створити файл 2. Записати ((n mod 2) +1) блоку в файл 2.

Дописати ((n mod 2) +1) блоку в файл 1. дописати ((n mod 3) +1) блоку в файл 2. Видалити файл 1.

Доступний простір диска: V = (n mod 3) + 10

СПИСОК ЛІТЕРАТУРИ

1. Irv Englander. The Architecture of Computer Hardware, Systems Software, and Networking: An Information Technology Approach. – Wiley, 2014. – 699p.

2. William Stallings. Operating Systems: Internals and Design Principles. – Pearson, 2018. – 2480p.

3. Бондаренко М.Ф., Качко О.Г. Операційні системи. – Х.: СМІТ, 2008. – 432 с.

4. Молчанов А. Ю.Системное программное обеспечение: Учебник для вузов. 3-е изд. — СПб.: Питер, 2010. — 400 с.

5. Д.А. Зайцев, С.М. Вороной, Т.Р. Шмелёва.Трассировка процессов функционирования сетевых операционных систем: Методические указания к лабораторным и практическим занятиям. - Донецьк: ІПШІ «Наука і освіта», 2007. - 75 с.