УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Кафедра спеціалізованих комп'ютерних систем

МЕТОДИЧНІ ВКАЗІВКИ
до практичних занять та самостійної роботи

з дисципліни
«ОПЕРАЦІЙНІ СИСТЕМИ»

Харків – 2021

Методичні вказівки розглянуто і рекомендовано до друку на засіданні кафедри спеціалізованих комп'ютерних систем 24 лютого 2020 р., протокол № 9.

У методичних вказівках розглянуто інструментальні засоби Windows для відстежування системних процесів, засоби керування пам'яттю та розподіл пам'яті в системі для вирішення індивідуальних завдань практичних занять. Тематика занять пов'язана з майбутньою професійною діяльністю студентів і має прикладний характер.

Призначено для студентів факультету ІКСТ зі спеціальності 123 «Комп'ютерна інженерія» першого освітнього рівня (бакалавр) усіх форм навчання.

Укладач

доц. Є. П. Павленко

Рецензент

доц. С. І. Доценко

ЗМІСТ

ВСТУП

Методичні вказівки призначаються для закріплення лекційного матеріалу, конкретизації здобутих знань в процесі самостійного вивчення певних розділів курсу.

Основна особливість операційних систем (ОС) – забезпечення обробки завдань протягом заданих інтервалів часу, які не можна перевищувати. Потік завдань не є планомірним і не регулюється оператором, тобто завдання надходять у непередбачувані моменти часу і без будь-якої черги. В ОС відсутні накладні витрати процесорного часу на етап ініціювання – завантаження програми, виділення ресурсів, оскільки набір завдань фіксований і вся інформація про завдання відома до надходження запиту.

Особливо важливе значення при побудові ОС мають модулі, які дозволяють більш ефективно використовувати ресурси обчислювальної системи. Принцип модульності відбиває технологічні та експлуатаційні властивості ОС. Найбільший ефект досягається при поширенні принципу модульності на ОС, прикладні програми та апаратуру.

Побудова віртуальних ресурсів, їх розподіл і використання є майже у кожній ОС. Це дозволяє подати структуру системи у вигляді певного набору планувальників процесів і розподільників ресурсів і використовувати єдину централізовану схему розподілу ресурсів.

ПРАКТИЧНЕ ЗАНЯТТЯ 1
Засоби Windows для відстежування системних процесів

1 Мета заняття: ознайомитися з видами інструментальних засобів Windows та основними прийомами роботи з командним рядком на прикладі системних процесів та процесів, які працюють у режимі користувача.

2 Завдання та порядок виконання

2.1 Вивчити теоретичний матеріал.

2.2 Підготувати відповіді на контрольні запитання.

2.3 Виконати перегляд інформації про процеси за допомогою диспетчера завдань.

Контрольні питання

1 Що таке процес операційної системи? Як процеси взаємодіють між собою?

2 Для чого призначене ядро ОС?

3 Що таке переносимість ОС?

3 Зміст звіту

3.1 Назва заняття, визначення мети.

3.2 Стислий зміст теоретичного матеріалу та відповіді на контрольні питання.

3.3 Результати перегляду інформації про процеси ОС та результати роботи з системним монітором.

3.4 Висновки з роботи.

4 Навчальний матеріал

Сучасні ОС для ПК реалізують і мультипрограмний, і багатозадачний режими.

У мультитермінальних ОС з однією обчислювальною системою одночасно можуть працювати кілька користувачів, кожен зі свого терміналу, при цьому у користувача виникає ілюзія, що у нього є власна обчислювальна система. Для організації мультитермінального доступу необхідний мультипрограмний режим роботи обчислювальної системи.

Частина модулів, які повинні постійно перебувати в оперативній пам'яті для більш ефективної організації обчислювального процесу, називається ядром ОС. При формуванні складу ядра враховують дві суперечливі вимоги:

- до складу ядра повинні увійти найбільш часто використовувані системні модулі;
- кількість модулів повинна бути такою, щоб обсяг пам'яті, який займає ядро, не був занадто великим.

До складу ядра входять, зазвичай, такі модулі: модулі з управління системою переривань; засоби для переведення програм зі стану виконання у стан очікування, готовності і назад; засоби для розподілу основних ресурсів: оперативної пам'яті і процесорного часу.

Транзитні програмні модулі завантажуються у пам'ять тільки за необхідності і у випадку відсутності вільного дискового простору можуть бути заміщені іншими транзитними модулями.

ОС є засобом розподілу ресурсів і організована за певними правилами управління процесами, тому приховує від користувача і його додатків реальні апаратні та інші ресурси, замінюючи їх абстракціями. Користувач бачить віртуальну машину як якийсь пристрій, здатний сприймати його програми і команди.

Користувача не цікавить реальна конфігурація обчислювальної системи і способи управління її компонентами. Він оперує з тими ресурсами, які йому надані у рамках віртуальної машини.

Віртуальна машина, яка надається користувачеві, відтворює архітектуру реальної машини, але архітектурні елементи у такому поданні мають нові або поліпшені характеристики, часто спрощують роботу з системою. Характеристики можуть бути довільними, але зазвичай користувачі хочуть бачити ідеальну за своїми архітектурними характеристиками машину.

Інформація, що зберігається або передається на віртуальні пристрої, не обмежена припустимими розмірами. Доступ до такої інформації здійснюється на основі або послідовного, або прямого способу доступу у термінах відповідної системи управління файлами.

Передбачено розширення інформаційних структур даних, які зберігаються на віртуальних пристроях.

Ступінь наближення до «ідеальної» віртуальної машини може бути більшим або меншим у кожному конкретному випадку. Чим більше віртуальна машина, яка реалізується засобами ОС на базі конкретної апаратури, наближена до ідеальної за характеристиками машини, чим більше її архітектурно-логічні характеристики відмінні від реально існуючих, тим більше ступінь віртуальності в отриманій користувачем машині.

Оперативна пам'ять – це найважливіший ресурс будь-якої обчислювальної системи, оскільки без неї, як і без центрального процесора, неможливе виконання жодної програми. Пам'ять є ресурсом. Способи поділу пам'яті і часу центрального процесора сильно впливають на швидкість виконання окремих обчислень і на загальну ефективність обчислювальної системи.

ОС виконує такі основні функції, пов'язані з управлінням завданнями:

- створення і видалення завдань;
- планування процесів і диспетчеризація завдань;
- синхронізація завдань, забезпечення їх засобами комунікації.

Система управління завданнями забезпечує проходження їх через комп'ютер. Залежно від стану процесу йому повинен бути наданий той чи інший ресурс. Створення та видалення завдань проводиться за відповідними запитами від користувачів або самих завдань.

Основним підходом до організації того або іншого методу управління процесами є організація черг процесів і ресурсів.

На розподіл ресурсів впливають конкретні потреби тих завдань, які повинні виконуватися паралельно.

Завдання динамічного планування, тобто найбільш ефективного розподілу ресурсів, які виникають практично при кожній події, називаються диспетчеризацією. Планування здійснюється рідше, ніж завдання поточного розподілу ресурсів між процесами і потоками, які вже виконуються. Різниця між довгостроковим і короткостроковим плануванням полягає у частоті запуску.

Довгостроковий планувальник вирішує, який з процесів, що знаходяться у вхідній черзі, повинен бути переведений в чергу готових до виконання процесів у випадку звільнення ресурсів пам'яті. У черзі готових до виконання процесів мають перебувати у рівній пропорції процеси, орієнтовані на введення/виведення, і процеси, орієнтовані на роботу центрального процесора.

Короткостроковий планувальник вирішує, яка з задач, що знаходяться у черзі готових до виконання, має бути передана на виконання. У більшості сучасних ОС довгостроковий планувальник відсутній.

Диспетчеризація пов'язана з поняттям задачі. Якщо операційна система не підтримує механізму потоків, то поняття завдання можна замінити на поняття процесу.

Відомо багато правил, відповідно до яких формується черга готових до виконання завдань. Є два великі класи дисциплін обслуговування:

- безпріоритетне;
- пріоритетне.

При безпріоритетному обслуговуванні вибір задачі проводиться у довільному порядку без урахування її важливості і часу обслуговування.

При реалізації пріоритетних дисциплін обслуговування окремим завданням надається переважне право на виконання.

Найпростішою у реалізації є дисципліна FCFS (first come – first served), завдання обслуговуються за чергою, тобто в порядку їх появи. Завдання, які були припинені для очікування будь-якого ресурсу, після переходу у стан готовності стають до цієї черги перед завданнями, які ще не виконувалися.

Утворюються дві черги:

- нові завдання;
- завдання, що раніше виконувалися, але потрапили до стану очікування.

Дисципліна FCFS реалізує стратегію обслуговування «по можливості закінчувати обчислення за їх появою». Ця дисципліна не вимагає зовнішнього втручання у хід обчислень і перерозподілу процесорного часу. За класом диспетчеризації (витісняючі/не витісняючі) дисципліна FCFS відноситься до витісняючих.

Одним з унікальних атрибутів, які стосуються процесу і не відображаються більшістю інструментальних засобів, є ідентифікатор батьківського (parent or creator process ID). Це значення можна одержати за допомогою Системного монітора (Performance Monitor) або програмним засобом, шляхом запиту Creating Process ID. Дерево процесів може показати такий засіб, як tlist.exe (зі складу призначених для Windows засобів налагодження Debugging Tools), який використовується з ключем /t.

Щоб переглянути список служб, запущених всередині процесу з вікна командного рядка, можна також скористатися інструментальним засобом tlist.exe з комплекту Debugging Tools for Windows, або засобом Tasklist, який поставляється з Windows. Для перегляду служб з Tlist можна використовувати такий синтаксис:

tlist /s

А для перегляду з tasklist такий:

tasklist /svc

Щоб показати взаємини кожного процесу з його батьківськими і дочірніми процесами, застосовуються відступи. Процеси, батьки яких припинили своє існування, вирівняні по лівому краю (як Explorer.exe у попередньому прикладі), оскільки, навіть за наявності прабатьківського процесу, засобів виявлення зв'язку з ним просто не існує. Windows зберігає тільки ідентифікатор процесу-батька і не дає посилань на батька цього батька [1].

Щоб продемонструвати той факт, що Windows не відслідковує більше одного ідентифікатора батьківського процесу, виконайте такі дії:

1 Відкрийте вікно командного рядка.

2 Наберіть title Parent, щоб змінити заголовок вікна на «Parent» (батьківський).

3 Наберіть start cmd (що призведе до запуску другого вікна командного рядка).

4 Наберіть у другому вікні командного рядка title Child, щоб змінити заголовок вікна на «Child» (дочірній).

5 Відкрийте Диспетчер завдань.

6 Наберіть у другому вікні командного рядка mspaint (команду, яка запускає Microsoft Paint).

7 Знову зверніться до другого вікна командного рядка і наберіть exit. (Зауважте, що Paint залишається у робочому стані.) Перейдіть до Диспетчера завдань.

8 Натисніть на вкладці додатка.

9 Натисніть правою кнопкою миші на завданні Parent і оберіть пункт Перейти до процесу.

10 Натисніть правою кнопкою миші на процесі cmd.exe і оберіть пункт Завершити дерево процесів.

11 У вікні підтвердження Диспетчера завдань клацніть на кнопці Завершити дерево процесів.

Перше вікно командного рядка зникне, але, як і раніше, можна буде спостерігати вікно програми Paint, оскільки воно було нащадком у другому поколінні завершеного процесу командного рядка. Оскільки проміжний процес (батьківський щодо Paint) був завершений, зв'язок між батьківським процесом і його нащадком у другому поколінні був втрачений.

Убудований до Windows Диспетчер завдань надає короткий список процесів, які перебігають у системі. Запустити цей диспетчер можна одним з чотирьох способів: 1) натисканням комбінації клавіш Ctrl + Shift + Esc; 2) натисканням правої кнопки миші на панелі завдань з наступним вибором пункту «Запустити диспетчер завдань»; 3) натисканням клавіш Ctrl + Alt + Delete з подальшим натисканням на кнопці запустити Диспетчер завдань; 4) запуском програми Taskmgr.exe.

Щоб побачити перелік процесів, потрібно після запуску Диспетчера завдань натиснути на вкладку «Процеси». Зверніть увагу на те, що процеси ідентифікуються за іменами тих образів, екземплярами яких вони є. На відміну від деяких об'єктів Windows, процесам можна давати глобальні імена. Щоб подивитися додаткову інформацію, виберіть у меню «Вид» пункт «Показати стовпці» і відзначити ті стовпці, які потрібно додати [2].

На вкладці «Процеси» Диспетчера завдань показується список процесів, а про вміст вкладки «Програми» з такою ж часткою очевидності судити не можна. Там показується список видимих вікон верхнього рівня на всіх Робочих столах інтерактивного віконного терміналу, до якого ви під’єднані.

За замовчуванням є тільки один інтерактивний Робочий стіл, але додаток може створити і більше, якщо скористається функцією Windows CreateDesktop, як це зроблено в засобі Sysinternals Desktops. У стовпці «Стан» показується, чи знаходиться потік, який є власником вікна, у стані очікування повідомлення від цього вікна. Стан «Працює» означає, що потік очікує надходження до цього вікна якогось введення, а режим «Не відповідає» означає, що потік не чекає надходження введення до вікна (наприклад, потік може бути запущений або ж

перебувати в очікуванні введення-виведення, або в очікуванні зміни стану якогось об'єкта синхронізації Windows).

На вкладці «Додатки» можна зіставити завдання тому процесові, який є власником потоку, який у свою чергу є власником вікна завдання. Для цього потрібно натиснути правою кнопкою миші на імені завдання і вибрати пункт «Перейти до процесу», як показано в попередньому експерименті із засобом tlist.

Щоб подивитися, скільки часу ваша система працює у режимі ядра порівняно з роботою у режимі користувача, можна скористатися Системним монітором (Performance Monitor). Виконайте такі дії:

1 Запустіть Системний монітор (Performance Monitor), відкривши меню Пуск (Start) і виберіть пункт Панель керування-адміністрування-Системний монітор (All Programs-Administrative Tools-Performance Monitor). На розташованому зліва деревоподібному списку інструментів Продуктивність (Performance) виберіть пункти Засоби спостереження (Monitoring Tools) – Системний монітор (Performance Monitor).

2 Натисніть на кнопку додавання (+), яка знаходиться на панелі інструментів (рисунок 1.1).

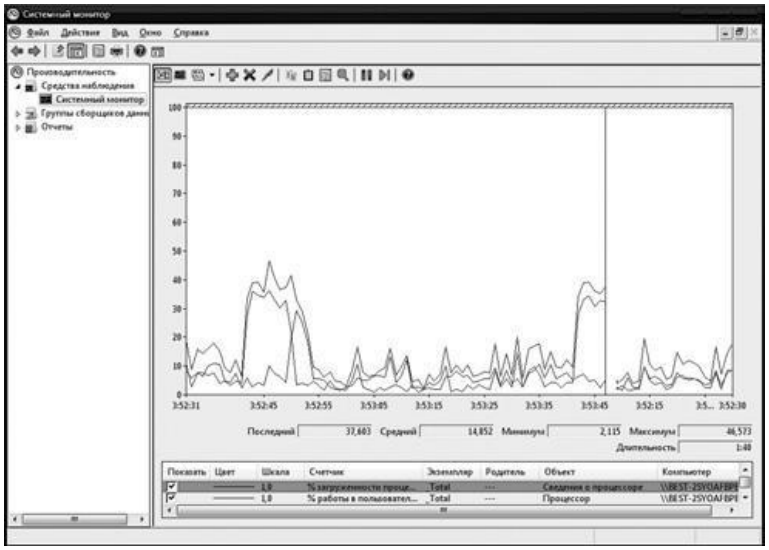

Рисунок 1.1 – Системний монітор

3 Розкрийте розділ лічильників Процесор (Processor), натисніть на пункті % роботи у привілейованому режимі

(% Privileged Time counter) і, утримуючи в натиснутому стані клавішу Ctrl, натисніть на пункті % роботи у режимі користувача (% User Time).

4 Натисність на кнопку Додати (Add), а потім на кнопку OK.

5 Відкрийте вікно командного рядка і проведіть безпосереднє сканування свого диска C у мережі, набравши команду dir \\% computername% \ c $ / s.

6 Після закінчення роботи закрийте вікно інструментального засобу.

Все це можна швидко переглянути за допомогою Диспетчера завдань (Task Manager). Натисність на вкладці Продуктивність (Performance), а потім оберіть у меню Вид (View) пункт Висновок часу ядра (Show Kernel Times). На графіку завантаженості центрального процесора зеленим кольором буде показана його загальна завантаженість, а червоним – завантаженість у режимі ядра.

Щоб побачити, скільки часу у режимі ядра і в призначеному для користувача режимі використовує сам Системний монітор (Performance Monitor), запустіть його ще раз, але при цьому додайте окремі лічильники процесу % роботи у режимі користувача (% User Time) і % роботи у привілейованому режимі (% Privileged Time) для кожного процесу у системі:

1 Якщо Системний монітор (Performance Monitor) не запущено, запустіть його знову. (Якщо він вже запущений, почніть роботу з пустого відображення, натиснувши в області графіків правою кнопкою миші і обравши пункт Видалити всі лічильники (Remove All Counters.)

2 Натисніть на кнопці додавання (+), яка знаходиться на панелі інструментів.

3 У доступній області лічильників розкрийте розділ Процес (Process).

4 Виберіть лічильники % роботи у режимі користувача (% User Time) і % роботи у привілейованому режимі (% Privileged Time).

5 Виберіть кілька процесів в області Примірники обраного об'єкта (Instance) (наприклад, mmc, csrss і Idle).

6 Натисність на кнопці Додати (Add), а потім на кнопці OK.

7 Інтенсивно порухайте мишею у різні боки.

8 Виберіть на панелі інструментів пункт Виділити (Highlight) або натисніть клавіші Ctrl + H, щоб увімкнути режим виділення. Поточний обраний лічильник буде позначено чорним кольором.

9 Прокрутіть список лічильників вниз для визначення процесів, чиї потоки були запущені: при переміщенні покажчика миші, і зверніть увагу на те, в якому режимі вони були запущені: у режимі користувача або у режимі ядра.

Ви маєте побачити (знайшовши в стовпці Примірник (Instance) процес mmc), що графік часу виконання процесу, який належить Системному моніторові, у режимі ядра і у режимі користувача при переміщенні миші пішов вгору, оскільки у ньому виконується прикладний код у режимі користувача, і викликаються Windows-функції, які запускаються у режимі ядра. Зверніть також увагу на активність потоку, який належить процесові csrss і виконується у режимі ядра при переміщенні миші.

Ця активність виникає завдяки тому, що цьому процесові належить вихідний потік введення тієї підсистеми Windows, виконуваної у режимі ядра, яка обробляє введення з клавіатури і з миші. І нарешті, процес Idle, який, як можна помітити, витрачає майже 100 % свого часу на роботу у режимі ядра, насправді процесом не є, це помилковий процес, який використовується для підрахунку холостих циклів центрального процесора [3].

ПРАКТИЧНЕ ЗАНЯТТЯ 2
Засоби керування пам’яттю

1 Мета заняття: вивчити функції для роботи з віртуальною та реальною пам'яттю при програмуванні у 32-бітному режимі.

2 Завдання та порядок виконання

2.1 Вивчити теоретичний матеріал.

2.2 Підготувати відповіді на контрольні запитання.

2.3 Скласти програму для формування системної інформації про віртуальну пам'ять: системну інформацію стосовно пам'яті визначити на початку програми. Виділити віртуальну пам'ять і визначити її статус. Виділити фізичну пам'ять і визначити її статус, тобто зміни у системній інформації до та після виконання. Послідовно вивільнити виділену пам'ять, Визначити максимальний розмір пам'яті, яка може бути виділена для віртуальної та фізичної пам'яті.

Контрольні питання

1 Що таке віртуальний адресний простір?

2 Яка функція використовується для визначення розміру сторінки?

3 Який прапор вказує на захищену сторінку?

4 Які особливості має зарезервована сторінка пам’яті?

3 Зміст звіту

3.1 Назва заняття, визначення мети.

3.2 Стислий зміст теоретичного матеріалу та відповіді на контрольні питання.

3.3 Опис виконання завдань щодо віртуальної та фізичної пам’яті.

4 Навчальний матеріал

Менеджер пам'яті – частина операційної системи, що відповідає за управління пам'яттю. Основні методи розподілу пам'яті:

- без використання зовнішньої пам'яті;
- з використанням зовнішньої пам'яті.

При розподілі пам'яті з фіксованими розділами пам'ять поділяється на кілька розділів (можливо, не рівних). Процеси можуть бути різними, тому кожному розділу необхідний різний розмір пам'яті. Системи можуть мати:

- загальну чергу до всіх розділів;
- до кожного розділу окрему чергу.

Основні способи використання диска:

- свопінг (підкачка) – процес цілком завантажується в пам'ять для роботи;

- віртуальна пам'ять – процес може бути частково завантажений в пам'ять для роботи.

Свопер – планувальник, що керує переміщенням даних між пам'яттю і диском.

При управлінні пам'яттю за допомогою бітових масивів вся пам'ять розбивається на блоки (наприклад, по 32 біт), масив містить 1 або 0 (зайнятий або незайнятий).

Щоб процесові у 32 кбіт зайняти пам'ять, потрібно набрати послідовність з 1000 вільних блоків.

Управління пам'яттю за допомогою зв'язкових списків. Цей спосіб відстежує списки зайнятих (між процесами) і вільних (процеси) фрагментів пам'яті. Запис у списку вказує:

- на зайнятий (Р) або незайнятий (Н) фрагмент;

- адресу початку фрагмента;

- довжину фрагмента.

Алгоритми виділення блоку пам'яті:

- першу підходящу ділянку;

- наступну підходящу ділянку, старт не з початку списку, а з того місця, на якому зупинився останнього разу;

- найкращу ділянку (повільніше, але краще використовує пам'ять);

- найбільш невідповідну ділянку, розрахунок робиться на те, що програма займе найбільшу ділянку, а зайві будуть відокремлені в нову ділянку, і вона буде досить великою для іншої програми.

Сторінки – це частини, на які розбивається простір віртуальних адрес. Сторінкові блоки – одиниці фізичної пам'яті.

Сторінки завжди мають фіксований розмір. Передача даних між ОП і диском завжди відбувається в сторінках.

Сторінкове переривання відбувається, якщо процес звернувся до сторінки, яка не завантажена в ОП. Процесор передається іншому процесу і паралельно сторінка завантажується в пам'ять.

Таблиця сторінок використовується для зберігання відповідності адрес віртуальної сторінки і сторінкового блоку.

Типові записи у таблиці сторінок:

- присутність/відсутність – завантажена або не завантажена в пам'ять;
- захист – види доступу, наприклад, читання/запис;
- зміна – чи змінилася сторінка, якщо так, то при вивантаженні записується на диск, якщо ні, просто знищується;
- звернення – чи було звернення до сторінки, якщо немає, то це найкращий кандидат на звільнення пам'яті.

Інформація про адресу сторінки, якщо вона зберігається на диску, в таблиці не розміщується.

Для прискорення доступу до сторінок в диспетчері пам'яті створюють буфер швидкого перетворення адреси, в якому зберігається інформація про найбільш часто використовувані сторінки.

Сторінкова організація пам'яті використовується і в UNIX, і у Windows.

Після запуску процесу він займає певну пам'ять, на диску відразу йому виділяється такий же простір. Тому файл підкачки повинен бути не менше пам'яті, а в разі нестачі пам'яті – навіть більше. Як тільки процес завершиться, він звільнить пам'ять і місце на диску.

На диску завжди є дублікат сторінки, яка знаходиться в пам'яті.

Якщо використовується динамічна область свопінгу, то передбачається не виділяти для сторінок місце на диску, а виділяти його тільки при вивантаженні сторінки, і як тільки сторінка повернеться в пам'ять, звільняти місце на диску.

Цей механізм складніше, тому що процеси не прив'язані до якогось простору на диску і потрібно зберігати інформацію (карту диска) про місцезнаходження на диску кожної сторінки.

Планування процесів включає до себе вирішення таких завдань:

1) визначення моменту часу для зміни виконуваного процесу;
2) вибір процесу на виконання з черги готових процесів;
3) перемикання контекстів «старого» і «нового» процесів.

Перші два завдання вирішуються програмними засобами, а останнє в значній мірі апаратно.

Існує багато різних алгоритмів планування процесів, які по-різному вирішують перераховані вище завдання, що переслідують різні цілі і забезпечують різну якість мультипрограмування. Серед цих алгоритмів дві групи найбільш часто зустрічаються: алгоритми, засновані на квантуванні, і алгоритми, засновані на пріоритетах.

Якщо алгоритм заснований на квантуванні, то зміна активного процесу відбувається [4]:

- якщо процес завершився і залишив систему;
- виникла помилка;
- процес перейшов в стан очікування;
- вичерпаний квант процесорного часу, відведений даному процесу.

Процес, який вичерпав свій квант, переводиться в стан готовності і очікує, коли йому буде надано новий квант процесорного часу, а на виконання відповідно до певного правила вибирається новий процес з черги готових. Таким чином, жоден процес не займає процесор надовго, тому квантування широко використовується в системах поділу часу. Граф станів процесу відповідає алгоритму планування, заснованому на квантуванні.

Віртуальний адресний простір складає 4 Гб. Окремий процес має свій адресний простір, недоступний іншим процесам. У Windows пам'ять операційної системи закрито від інших процесів, це допомагає запобігати збоям (таблиця 2.1).

Таблиця 2.1 – Адресний простір для Windows

Діапазон адрес	Розмір (байт)	Призначення
00000000-00000FFF	4096	Для MSDOS
00001000-003FFFFF	1Гб – 4096 байт	Для r/w, краще не використовувати
00400000-7FFFFFFF	1 Гб	Для процесів користувача
80000000-BFFFFFFF	1 Гб	DLL Win32 (Kernel32.dll, user32.dll, gdi32.dll), file mapping, доступні усім програмам виконання (r/w)
C0000000-FFFFFFFF	1 Гб	Ядро ОС, VXD драйвери (r/w)

Резервування пам'яті починається з виділення регіону пам'яті – визначення потрібного адресного простору. Для цього використовується функція VirtualAlloc, яка дозволяє виділяти віртуальну та фізичну пам'ять. Фізична пам'ять, виділена цією функцією, обнуляється.

lpAddress – визначає початкову адресу виділеної області. Адреса віртуальної пам'яті вирівнюється на границю 64 Кб. Якщо пам'ять вже зв’язується (виділяється фізична пам'ять), адреса вирівнюється на границі сторінки.

Для визначення розміру сторінки можна використовувати функцію GetSystemInfo. Якщо параметр = NULL, система визначає можливість виділення регіону.

DwSize – розмір регіону у байтах. Якщо lpAddress= NULL, це значення буде округлене до границі сторінки у бік збільшення.

FlAllocationType – тип операції виділення. Можна вказати прапори або їх комбінації (таблиця 2.2).

Таблиця 2.2 – Прапори та їх призначення

Прапор	Призначення
1	2
MEM_COMMIT	Виділяє фізичну пам'ять або сторінковий файл на диску. Повторне виділення тієї ж області не приводить до помилки
MEM_RESERVE	Резервує віртуальний адресний простір вказаного розміру. Зарезервований діапазон не може використовуватися іншими функціями, наприклад LocalAlloc. Виділений регіон може бути пов'язаний з фізичною пам'яттю
MEM_TOP_DOWN	Виділяється пам'ять у верхній області адресного простору
FlProtect	Тип захисту. Для фізичної пам'яті можуть бути задані прапори PAGE_GUARD і PAGE_NOCACHE разом з прапорами, зазначеними нижче
PAGE_READONLY	Сторінка тільки для читання. Використовується для фізичної пам'яті. Якщо ОС розрізняє варіант для виконання, виключення для файлів виконання
PAGE_READWRITE	r/w
PAGE_EXECUTE	Виконання

Продовження таблиці 2.2

1	2
PAGE_EXECUTE_READ	r/e
PAGE_EXECUTE_READWRITE	r/w/e
PAGE_GUARD	Захищена сторінка. Будь-який доступ до сторінки – помилка
PAGE_NOACCESS	Вимкнено доступ до сторінки, відображеної у фізичній пам'яті
PAGE_NOCACHE	Не кешується зміст сторінки. Використовується у драйверах

Значення, які повертаються:

Без помилки – адреса пам'яті. Помилка – NULL. (GetLastError).

Таким чином, VirtualAlloc може виконати такі функції:

1 Зв’язати регіон сторінки з фізичною пам’яттю, який виділений при попередньому виклику функції VirtualAlloc.

2 Зарезервувати регіон з вільних сторінок.

3 Зарезервувати і зв’язати.

Можна виділити великий регіон, а потім пов’язати з фізичною пам’яттю тільки невеликий блок регіону. Зв’язок з фізичною пам’яттю виконується тільки для тієї ділянки, для якої це необхідно.

Сторінки можуть бути в одному з трьох станів.

1 Вільна – сторінка не зарезервована і не зв’язана. Такі сторінки функція VirtualAlloc може зарезервувати або зарезервувати і зв’язати.

2 Зарезервована – не може бути використана іншими функціями виділення пам'яті, але не має відповідної фізичної адреси. Функція VirtualAlloc може її зв’язати з фізичною пам’яттю, але не може повторно зарезервувати. Можна звільнити зарезервовані сторінки функцією VirtualFree.

3 Зв’язана – фізична пам'ять виділена і встановлені атрибути доступу. Система фактично ініціалізує і завантажує виділені сторінки при першій спробі доступу до заданої пам'яті. Коли завершується процес, ОС звільняє усі закріплені сторінки. Функція VirtualAlloc може повторно зв’язати вже зв’язані

сторінки. Функція VirtualFree може звільнити фізичну пам'ять, пов’язану зі сторінками.

Якщо lpAddress!=NULL, функція використовує параметри lpAddress і dwSize для обчислення кількості потрібних сторінок. Якщо задана кількість сторінок не може бути виділена, функція завершується з помилкою.

ПРАКТИЧНЕ ЗАНЯТТЯ 3
Вивчення призначення і функціональних можливостей ОС Linux

1 Мета заняття: ознайомлення з галуззю застосування, основними командами та методами розробки програм за допомогою операційної системи Linux.

2 Завдання та порядок виконання

2.1 Вивчити теоретичний матеріал.

2.2 Підготувати відповіді на контрольні питання.

2.3 За допомогою команд Linux виконати такі дії.

2.3.1 Переглянути вміст поточного каталогу.

2.3.2 Створити текстовий файл і ввести кілька рядків.

2.3.3 Переглянути вміст файлу. Порівняти різні команди перегляду.

2.3.4 Створити тимчасовий каталог.

2.3.5 Виконати копіювання, перейменування і переміщення файлу.

2.3.6 Видалити файл і каталог.

2.3.7 Виконати злиття двох текстових файлів.

2.3.8 Вивести вміст каталогу в різних форматах. Порівняти команди ls і dir і їх опції. Виконати сортування файлів у каталозі за різними ознаками.

2.3.9 Визначити права доступу до файлів і каталогів.

2.3.10 Змінити права доступу. Дослідити різні комбінації прав r, w, x. Створити каталог, перегляд вмісту якого буде заборонений, але можна буде переглядати вміст файлів у ньому.

Контрольні питання

1 Назвіть основні характеристики сімейства операційних систем UNIX.

2 Які переваги має ОС Linux?

3 Чи підтримує ОС Linux множинні віртуальні консолі?

4 Назвіть найбільш популярні версії ОС Linux.

5 У чому суть режиму розподілу часу в ОС Linux?

3 Зміст звіту

3.1 Назва заняття, визначення мети.

3.2 Стислий зміст теоретичного матеріалу та відповіді на контрольні запитання.

3.3 Призначення основних команд операційної системи, приклади їх роботи.

3.4 Висновки з заняття.

4 Навчальний матеріал

Linux – вільно розповсюджувана версія UNIX, спочатку була розроблена на початку 90-х рр. Лінусом Торвальдсом (Linus Torvalds) в Університеті Гельсінкі (Фінляндія). Linux був створений за допомогою багатьох UNIX-програмістів і ентузіастів з Internet, тих, хто має достатньо практики і здатності розвивати систему. Ядро Linux не використовує коди AT&T або якого-небудь іншого приватного джерела, і більшість програм Linux розроблені в рамках проекту GNU з Free Software Foundation в Cambridge, Massachusetts. В цю ОС внесли свій доробок також програмісти всього світу [5].

Більшість вільно розповсюджуваних по мережі Internet програм для UNIX можна відкомпілювати для Linux практично без особливих змін. Крім того, всі вихідні тексти для Linux, включаючи ядро, драйвери пристроїв, бібліотеки, призначені для користувача програми та інструментальні засоби, поширюються вільно.

Переваги ОС Linux:

- надійна, стійка робота, майже без зависань;

- не схильна до вірусів;

- ефективно управляє багатозадачністю і пріоритетами, фонові завдання не заважають інтерактивній роботі;

- дозволяє легко інтегрувати комп'ютер у локальні і глобальні мережі, в тому числі в Internet;

- працює з мережами на базі Novell і MS Windows;

- забезпечує використання величезного числа різноманітних програмних пакетів, накопичених у світі Unix і вільно розповсюджуваних разом з вихідними текстами;

- надає багатий набір інструментальних засобів для розробки прикладних програм будь-якого ступеня складності, включаючи системи класу клієнт-сервер, об'єктно-орієнтовані, з багатовіконним текстовим і графічним інтерфейсом, придатних для роботи як в Linux, так і в інших ОС;

- дає всім охочим можливість спробувати свої сили в розробці, організувати спілкування і спільну роботу через Internet з будь-якими з розробників ОС Linux і зробити свій внесок, ставши співавтором системи.

Багато провайдерів Internet застосовують Linux як основну операційну систему. Багато організацій використовували Linux для роботи з графікою.

Особливості Linux як ОС:

- економне завантаження: Linux зчитує з диска тільки ті частини програми, які дійсно використовуються для виконання;

- віртуальна пам'ять зі сторінковою організацією (на диск з пам'яті витісняється не весь неактивний процес, а тільки потрібна сторінка);

- збій програми не може викликати зависання системи;

- динамічно завантажувані колективні бібліотеки;

- підтримка національних алфавітів і угод, можливість додавати нові;

- множинні віртуальні консолі: на одному дисплеї кілька одночасних незалежних сеансів роботи, що перемикаються з клавіатури;

- підтримка ряду поширених файлових систем (MINIX, Xenix, файлові системи System V);

- наявність власної передової файлової системи обсягом до 4 Тб і з іменами файлів до 255 символів;

- прозорий доступ до розділів DOS (або OS / 2 FAT): розділ DOS виглядає як частина файлової системи Linux.

Деякі компанії пропонують свіжі рішення в побудові ОС Linux, наприклад компанії Red Hat і Caldera.

Компанія Red Hat випустила один з найбільш популярних пакетів Linux, що не заважає їй робити і комерційні програми. Поряд з цим Red Hat випускає систему, іменовану Applixware, і включає текстовий редактор, програму електронних таблиць, графічну презентаційну програму, електронну пошту і різні засоби розробки.

Продукт компанії Caldera – Caldera Open LinuxBase – недорога операційна система, аналогічна UNIX, на основі ядра Linux 2.0 і власного пакета Open Linux. Вона включає графічний користувальницький інтерфейс, що забезпечує керування системою і мережевими ресурсами, в тому числі взаємодією клієнт / сервер через Internet [6].

Ca1dera пропонує також WordPerfect компанії Corel для Linux і офісний пакет із засобами Internet, що містить повний набір офісних програм.

Великої популярності набув дистрибутив Debian. За своєю філософією він найбільш близький до проекту Linux. Технічно відрізняється тим, що має логічну продуману процедуру установлення нових компонент, що не вимагає переформатування системи.

Комп'ютер під управлінням Linux може одночасно виконувати кілька завдань (програм). Кожне завдання виконується в попередньо виділеній безперервній області пам'яті, що називається розділом.

Процесор може одночасно виконувати тільки одну задачу. Але задачі потребують не тільки в процесорі, наприклад, при операціях введення/виведення вона не потрібна. Тому, якщо деяка задача знаходиться в стані очікування і процесор нею не задіяний, можна відкрити доступ до процесора для інших задач.

Робота в режимі поділу часу створює у кожного користувача ілюзію монопольної роботи на машині.

Суть режиму поділу часу полягає в наступному: кожній програмі, готовій до виконання, планується для виконання на процесорі фіксований інтервал часу. Якщо програму не виконано

до кінця протягом цього інтервалу, її виконання примусово переривається, і вона поміщається у хвіст черги готових до виконання програм. З початку цієї черги вибирається наступна програма, якій знову планується фіксований інтервал часу.

Таке циклічне мультиплексування процесора гарантує, що всі програми будуть обслуговуватися «справедливо», що ніхто не зможе монополізувати процесор.

Файли в Linux відіграють ключову роль, що не завжди справедливо для інших операційних систем. Файли не тільки зберігають інформацію користувачів, але і забезпечують доступ до периферійних пристроїв комп'ютера, включаючи диски, принтери, термінали, мережеві адаптери і навіть пам'ять. Для додатків Linux доступ до дискового файла не відрізняється від доступу до принтера.

У Linux весь доступний користувачам файловий простір об'єднано в єдине дерево каталогів, коренем якого є каталог "/". Таким чином, повне ім'я будь-якого файла починається з "/" і не містить ідентифікатора пристрою (дискового накопичувача або віддаленого комп'ютера в мережі), на якому він фактично зберігається.

Ім’я файла є атрибутом файлової системи, а не набору деяких даних на диску, який не має імені як такого. Кожен файл має пов'язані з ним метадані (що зберігаються в індексних дескрипторах – inode), які містять всі характеристики файла, і дозволяє операційній системі виконувати операції, замовлені програмою: відкрити файл, прочитати або записати дані, створити або видалити файл. Зокрема, метадані містять покажчики на дискові блоки зберігання даних файла. Файл у файловій системі є покажчиком на його метадані, в той час як метадані не містять покажчика на ім'я файла.

У Linux існують різні типи файлів, що розрізняються за функціональним призначенням і діями операційної системи при виконанні тих чи інших операцій над файлами.

Звичайний файл являє собою найбільш загальний тип файлів, що містить дані в деякому форматі. Для операційної системи такі файли являють собою просто послідовність байтів. Вся інтерпретація вмісту файла здійснюється прикладною

програмою, що обробляє файл. До цих файлів відносяться текстові файли, бінарні дані, виконувані програми.

Каталог – це суперфайл, що містить посилання на файли, об'єднані за певною ознакою, а також покажчики на додаткову інформацію – метадані, що дозволяють операційній системі виконувати операції над файлами. Каталоги визначають положення файла в дереві файлової системи, оскільки сам файл не містить інформації про своє місцезнаходження.

По суті каталог являє собою таблицю, кожен запис якої відповідає деякому файлу. Перше поле кожного запису містить покажчик на метадані (номер inode), а друге визначає ім'я файла.

Спеціальний файл пристрою забезпечує доступ до фізичного пристрою. У Linux розрізняють символьні (character) і блокові (block) файли пристроїв. Доступ до пристроїв здійснюється шляхом відкриття, читання і запису в спеціальний файл пристрою.

Символьні файли пристроїв використовуються для небуферизованого обміну даними з пристроєм, на противагу цьому блокові файли дозволяють проводити обмін даними у вигляді пакетів фіксованої довжини - блоків.

FIFO, або іменований канал – це файл, який використовується для зв'язку між процесами.

Особливим типом файла є символічний зв'язок, що дозволяє побічно адресувати файл. Символічний зв'язок адресує файл, який у свою чергу посилається на інший файл. У результаті останній файл адресується символічним зв'язком побічно.

Символічний зв'язок є особливим типом файла (про це свідчить символ T в першій позиції виведення ls (l)), і операційна система працює з таким файлом не так, як зі звичайним. Наприклад, при виведенні на екран вмісту файла symfirst з'являться дані файла /home/andrei/first.

Сокети призначені для взаємодії між процесами. Інтерфейс сокетів часто використовується для доступу до мережі TCP/IP. У системах гілки BSD UNIX на базі сокетів реалізована система міжпроцесної взаємодії, за допомогою якої працюють багато системних сервісів, наприклад, система друку.

Загальний вигляд командного рядка Linux:

команда опції аргументи

Команда – це ім'я програми, яка повинна бути виконана.

Опції визначають, як виконується команда.

Аргументи – дані, з якими повинна працювати команда (зазвичай це ім'я файла або каталога).

Наявність командного імені обов'язкова, опції або аргументи можуть бути відсутні. Компоненти командного рядка відокремлюються як мінімум одним пропуском.

Наведемо призначення деяких команд Linux.

Команда виведення поточної дати і часу

date

Команда, що виводить список користувачів, які працюють в даний момент

who

Команда, що виводить ім'я поточного каталога

pwd

Команда, що виводить вміст каталога

ls [опція] [<ім'я кат.>]

Приклад: ls -l /usr/stud

Якщо ім'я каталога не вказується, буде виведено вміст поточного каталога.

Перелік опцій команди ls:

-l виведення вмісту каталога в довгому форматі: вказуються права доступу, кількість посилань на файл, імена власника та групи, розмір в байтах і час останньої модифікації;

-a виведення списку всіх файлів, у тому числі і тих, імена яких починаються з точки;

-r зміна порядку сортування на зворотний;

-t виведення імен файлів у порядку часу створення (спочатку йдуть найсвіжіші файли);

-S сортування. Порядок сортування вказується другою опцією.

Наприклад ls -S -S - сортування за розміром файла.

Команда переходу до іншого каталога

cd <ім'я каталога>

Приклад: cd /usr/stud

Якщо ім'я каталога опущено, проводиться перехід у домашній каталог користувача.

Команда створення каталога всередині поточного

mkdir <ім'я каталога>

Команда видалення каталога

rmdir <ім'я кат.>

Каталог, що видаляється, повинен бути порожній. Цей каталог не може бути поточним. Для видалення поточного каталога необхідно попередньо перейти в інший каталог.

Команда видалення файла

rm [опція] <ім'я файла>

Приклад: rm lab.c

Якщо вказана опція -i, перед видаленням кожного файла буде запитуватися підтвердження видалення.

rm -i lab * – видалити всі файли, що починаються на lab, з підтвердженням видалення.

Команда перейменування або переміщення файла

mv <ім'я файла 1> <ім'я файла 2> - перейменування файла

Приклад: mv lab.c lb.c

mv <ім'я файла> <ім'я кат.> - переміщення файла в інший каталог

Приклад: mv lab.c /udd/sks31s

Копіювання файла

cp <ім'я файла> <ім'я файла 2> - копіювання файла в інший файл

cp <ім'я файла> <ім'я кат.> - копіювання файла в інший каталог.

Приклад: cp lab.c /usr/serg

ПРАКТИЧНЕ ЗАНЯТТЯ 4
Вивчення особливостей ОС Linux

1 Мета заняття: ознайомлення з можливостями управління процесами в операційній системі Linux. Ознайомлення з особливостями роботи в мережі в ОС Linux.

2 Завдання та порядок виконання

2.1 Вивчити теоретичний матеріал.

2.2 Підготувати відповіді на контрольні запитання.

2.3 Виконати перенаправлення введення/виведення. Здійснити фільтрацію виведення команд.

2.4 Вивести відомості про процеси, запущені в даний момент. Використовувати довгий і короткий формати виведення. Отримати список процесів у розрізі користувачів; терміналів.

2.5 Припинити виконання деякого процесу.

2.6 Скласти свій скрипт ініціалізації, змінивши запрошення системи, домашній каталог і список каталогів для пошуку.

2.7 Переглянути конфігураційні файли TCP/IP.

2.8 Переглянути таблицю активних з’єднань Internet, активних доменних сокетів.

Контрольні питання

1 Що таке процес та які він має атрибути?
2 Які особливості мають системні процеси?
3 Для чого використовуються сигнали?
4 Які особливості блокових пристроїв у Linux?

3 Зміст звіту

3.1 Назва заняття, визначення мети.
3.2 Стислий зміст теоретичного матеріалу та відповіді на контрольні питання.
3.3 Результати виконання завдання: призначення розглянутих в лабораторній роботі команд операційної системи, приклади їх роботи.
3.4 Висновки з заняття.

4 Навчальний матеріал

Процес – це оточення завдання або середовище виконання завдання, що включає ресурси пам'яті, можливість доступу до пристроїв введення/виведення і різних системних ресурсів (наприклад, послуг ядра).

Для того щоб деяка програма могла бути запущена на виконання, операційна система повинна створити процес. Можна сказати, що процес – це програма в стадії її виконання.

Процес складається з інструкцій, які виконуються процесором, даних та інформації про виконувану задачу (програми), таких як розміщена пам'ять, відкриті файли і статус процесу [7].

Програма може створити при її виконанні більш одного процесу. Найпростіші програми, наприклад, who або cat, породжують тільки один процес. Складні завдання породжують в системі декілька процесів, що одночасно виконуються.

При виконанні процесу управління ніколи не може бути передано командам іншого процесу. Процес зчитує і записує інформацію в розділ даних, але йому недоступні розділи даних інших процесів.

Однак процеси мають можливість обмінюватися один з одним даними за допомогою системи взаємодії між процесами. В UNIX існує набір засобів взаємодії між процесами, таких як сигнали, канали, колективна пам’ять, семафори, повідомлення, але в іншому процеси ізольовані один від одного.

Атрибути процесу:

1 PID – ідентифікатор процесу. Він дозволяє ядру системи розрізняти процеси. Коли створюється наступний процес, ядро присвоює йому наступний вільний ідентифікатор (за зростанням). Коли процес завершує свою роботу, ядро вивільняє присвоєний йому ідентифікатор;

2 PPID – ідентифікатор процесу, який породив даний процес;

3 NI – відносний пріоритет процесу. Він враховується при визначенні черговості запуску. Відносний пріоритет не змінюється системою протягом усього життя процесу в системі;

4 PRI – поточний динамічний пріоритет процесу. Він визначає фактичний розподіл процесорних ресурсів і залежить від декількох факторів, зокрема, від значення NI. Динамічний пріоритет постійно оновлюється ядром;

5 TTY – номер керуючого терміналу процесу. (Відсутність керуючого терміналу фіксується знаком «?»);

6 UID – ідентифікатор користувача-власника процесу.

Системні процеси є частиною ядра і завжди розташовані в оперативній пам'яті. Системні процеси не мають відповідних їм програм у вигляді виконуваних файлів і запускаються особливим чином при ініціалізації ядра системи. Їх інструкції і дані знаходяться в ядрі системи, таким чином вони можуть викликати функції і звертатися до даних, недоступних для інших процесів.

Системними процесами є: shed (диспетчер свопінгу), vhand (диспетчер сторінкового заміщення), bdflush (диспетчер буферного кешу) і kmadaemon (диспетчер пам'яті ядра). До системних процесів слід віднести ink, який є прабатьком всіх інших процесів у Linux. Хоча ink не є частиною ядра, і його запуск відбувається з файла (/etc/init), його робота життєво важлива для функціонування всієї системи в цілому.

Демони – це неінтерактивні процеси, які запускаються шляхом завантаження в пам'ять відповідних їм виконуваних

файлів і виконуються у фоновому режимі. Зазвичай демони запускаються при ініціалізації системи (але після ініціалізації ядра) і забезпечують роботу різних підсистем UNIX: системи термінального доступу, системи друку, системи мережевого доступу і мережевих послуг. Демони не пов'язані з жодним сеансом роботи користувача і не можуть керувати безпосередньо користувачем. Велику частину часу демони очікують, поки той чи інший процес запросить певну послугу, наприклад, доступ до файлового архіву або друк документа.

До прикладних процесів належать решта процесів, що виконуються в системі. Як правило, це процеси, породжені в рамках сеансу роботи користувача. Найважливішим для користувача процесом є командний інтерпретатор shell, який забезпечує роботу в Linux. Він запускається відразу ж після реєстрації в системі, а завершення роботи shell призводить до відмови у доступі до системи.

Призначені для користувача процеси можуть виконуватися як в інтерактивному, так і фоновому режимі, але у будь-якому випадку час їхнього життя (і виконання) обмежений сеансом роботи користувача. При виході з системи всі призначені для користувача процеси будуть знищені.

Сигнали є способом передачі від одного процесу іншому або від ядра операційної системи будь-якому процесу повідомлення про виникнення певної події. Сигнали можна розглядати як форму взаємодії між процесами. Сигнали нагадують програмні переривання – засіб, за допомогою якого нормальне виконання процесу може бути перервано. Наприклад, якщо процес робить ділення на 0, ядро посилає йому сигнал SIGFPE, а при натисканні клавіш переривання, зазвичай <Del> або <Ctrl> + <C>, поточному процесу надсилається сигнал SIGINT.

Для відправлення сигналу служить команда kill:

```
kill sig_no pid
```

де sig_no – номер або символічна назва сигналу, а pid – ідентифікатор процесу, якому надсилається сигнал. Адміністратор системи може посилати сигнали будь-яким

процесам, звичайний же користувач може посилати сигнали тільки процесам, власником яких він є.

При отриманні сигналу процес має три варіанти дій для вибору.

1 Він може ігнорувати сигнал. Не слід ігнорувати сигнали, викликані апаратною частиною, наприклад, при діленні на 0 або посиланням на неприпустимі області пам'яті, так як подальші результати щодо даного процесу непередбачувані.

2 Процес може зажадати дії за замовчуванням. Зазвичай це зводиться до завершення виконання процесу.

3 Процес може перехопити сигнал і самостійно обробити його. Наприклад, перехоплення сигналу SIGINT дозволить процесу видалити створені ним тимчасові файли. Слід мати на увазі, що сигнали SIGKILL і SIGSTOP не можна ані перехопити, ані ігнорувати.

За замовчуванням команда kill посилає сигнал з номером 15 – SIGTERM, дія за замовчуванням для якого – завершення виконання процесу, який отримав сигнал.

Іноді процес продовжує існувати і після відправлення сигналу SIGTERM. У цьому випадку можна застосувати більш жорсткий засіб – послати процесу сигнал SIGKILL з номером 9, оскільки цей сигнал не можна ані перехопити, ані ігнорувати.

```
$ kill -9 pid
```

Однак можливі ситуації, коли процес не зникає і в цьому випадку. Це може статися для таких процесів:

1 Процеси-зомбі. Фактично процесу як такого не існує, залишився лише запис у системній таблиці процесів, тому видалити його можна тільки перезапуском операційної системи. Зомбі в невеликих кількостях не становлять небезпеки, проте якщо їх багато, це може привести до переповнення таблиці процесів.

2 Процеси, які очікують недоступні ресурси , наприклад, записують дані у файл файлової системи віддаленого комп'ютера, що відключився від мережі. Цю ситуацію можна подолати, пославши процесу сигнал SIGINT або SIGQUIT.

3 Процеси, які очікують завершення операції з пристроєм, наприклад, зчитування з флешки.

Linux «ізолює» додатки (а значить, і користувача) від апаратної частини обчислювальної системи. Наприклад, в імені файла відсутній покажчик диска, на якому цей файл розташований, а більша частина взаємодії з периферійними пристроями не відрізняється від операцій зі звичайними файлами.

Linux надає єдиний інтерфейс різних пристроїв системи у вигляді спеціальних файлів пристроїв. Спеціальний файл пристрою пов’язує прикладний додаток з драйвером пристрою. Кожен спеціальний файл відповідає будь-якому фізичному пристрою (наприклад, диску, принтеру або терміналу) або псевдопристрою (наприклад, мережевому інтерфейсу, пустому пристрою, сокету або пам'яті). Вся робота програми з пристроєм відбувається через спеціальний файл, а відповідний йому драйвер забезпечує виконання операцій введення/виведення відповідно до конкретного протоколу обміну даними з пристроєм.

Обмін даними з блочними пристроями відбувається великими фрагментами фіксованої довжини, що називаються блоками. При цьому ядро операційної системи забезпечує необхідну буферизацію.

В основному драйвери блочних пристроїв використовуються файловою підсистемою і підсистемою управління пам'яттю. Наприклад, свопінг характеризується обміном даними з вінчестером, розмір яких зазвичай дорівнює розміру сторінки, що становить 4 або 8 Кб. Типовими представниками блочних пристроїв є жорсткий та гнучкий диски.

Символьні пристрої – це пристрої, драйвери яких забезпечують власну буферизацію та побайтну передачу даних. Як приклад пристроїв з символьним інтерфейсом можна навести термінали, мишу, клавіатуру і локальні принтери. Символьні пристрої, як правило, передають невеликі обсяги даних.

У полі розміру файла (п'ята колонка виведення команди ls) у спеціальних файлів пристроїв виводяться два числа. Це так звані старше (major) і молодше (minor) числа. Часто драйвер обслуговує більше одного пристрою. При цьому старше число вказує ядру на конкретний драйвер (наприклад, драйвер

псевдотерміналу), а молодше передається драйверу і вказує на конкретний пристрій (наприклад, конкретний псевдотермінал).

Команда, що виводить список процесів у системі.

Формат команди:

ps [-Опції] [-t tlist] [-p plist] [-u ulist]

Опції:

-e видає інформацію про всі процеси, запущені на даний момент;

-d видається інформація про частину процесів;

-a виводиться інформація про найбільш часто запитувані процеси, тобто виводяться процеси, які не асоційовані з терміналом;

-f по кожному процесу видаються значення UID, PID, PPID, STIME (час створення процесу (може відрізнятися від часу запуску команди)), C (частка виділеного планувальником процесорного часу), TTY, TIME (сумарний час виконання процесу процесором), COMMAND (ім'я команди, що відповідає процесу);

-l видається лістинг у довгому форматі. Крім уже зауважених значень, видається F (статус процесу), S (стан процесу), значення пріоритетів PRI і NI, адреса процесу в пам'яті ADDR, розмір (у блоках) пам'яті, наданої процесу SZ, і деяка інша інформація. У колонці F (статус процесу) можуть бути такі 16-кові символи: 01 – системний процес, завжди в основній пам'яті, 10 – процес в основній пам'яті, 19 – процес в основній пам'яті, і блокований до завершення деякої події, 40 – йде сигнал до віддаленої системи, 80 – процес в черзі на введення/виведення.

У колонці «S» (стан процесу) можуть бути такі символи: O – виконується процесором, S – знаходиться в стані сну, наприклад, очікує введення даних, R – готовий: стоїть у черзі на виконання, I – створюється, Z – процес завершено, але батьківський процес не чекає цього, X – процес очікує отримання більшого обсягу пам'яті;

-t опція вказується спільно зі списком терміналів tlist. Дозволяє отримати дані тільки про процеси, запущені з зазначених у списку терміналів;

-p вказується разом зі списком ідентифікаторів процесу PID. Виводить тільки дані про зазначені в списку процеси. Але треба знати ідентифікатори потрібних процесів;

-u вказується разом зі списком користувачів ulist. Виводить дані тільки про процеси, ініційовані зазначеними користувачами.

Опції можуть комбінуватися одна з одною. Приклад:

ps –df

UID	PID	PPID	STIME	TTY	TIME	COMMAND
root	1	0	14:35:50	?	0:15	/etc/init
root	2	0	14:35:50	?	0:00	vhand
root	3	0	14:35:50	?	0:02	bdflush
sks31	156	79	17:11:37	tty0i	0:25	pnc
sks31	270	268	17:39:47	tty0i	0:00	pr
sks32	257	94	17:32:49	tty9i	0:10	pnc

Перші три рядки – системні процеси. Знак «?» у графі «TTY» означає, що процес не був запущений яким-небудь терміналом.

ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ СТУДЕНТІВ

1 Написати програму, яка розташовує ярлик програми на робочому столі.

2 Написати програму, яка змінює колір стандартного курсора програми.

3 Написати програму, яка відображає стандартний курсор програми з рамкою навколо.

4 Написати програму, яка змінює розміри стандартного ярлика програми на робочому столі.

5 Написати програму, яка змінює стандартний ярлик програми на власний (створений особисто).

6 Написати програму, яка має одне вікно, розділене на логічні області, кожна з яких має свій курсор.

7 Побудувати зв’язаний список, що містить як дані матрицю чисел, яку вводить користувач з клавіатури.

8 Побудувати зв’язаний список, що містить як дані інформацію про файли і директорії у вказаній директорії. До інформації входить: ім'я файла, повний шлях до файла, дата останньої зміни, атрибути.

9 Побудувати зв’язаний список, що містить як дані інформацію про стан пам'яті: кількість зайнятої пам'яті, кількість вільної пам'яті.

10 Написати програму, яка породжує потік при натисканні будь-якої клавіші клавіатури. Кожному створеному таким чином потоку відповідає коло у вікні додатка, яке з'являється у звичайному місці і рухається у довільному порядку. Коли одне коло зіткнеться з іншим – утворюється ще одне коло.

11 Написати програму, яка містить два потоки, кожний з яких керує рухом однієї з двох куль. Перша куля рухається горизонтально, друга – вертикально. Швидкість куль різна. При досягненні границі вікна додатка куля змінює напрямок руху на протилежний. Реалізувати алгоритм руху куль без зіткнень.

12 Написати програму, яка запускає новий потік при натисканні лівої клавіші миші. Потік починає виводити числову послідовність, що збільшується, у поточну позицію курсора миші. При натисканні лівої клавіші програма видаляє потік, координати якого ближче до положення миші.

ПИТАННЯ ДЛЯ САМОСТІЙНОЇ ПІДГОТОВКИ ДО МОДУЛЬНОГО КОНТРОЛЮ

1 Які модулі входять до складу ядра ОС?

2 У чому полягає принцип нарощуваності ОС?

3 Які засоби мають бути на рівні С2 безпеки ОС?

4 Яка дисципліна диспетчеризації є пріоритетною?

5 Які функції виконує підсистема управління процесами ядра ОС?

6 Які модулі входять до складу ядра ОС?

7 Що необхідно для забезпечення двійкової сумісності ОС?

8 Які засоби мають бути на рівні С2 безпеки ОС?

9 Яка дисципліна диспетчеризації є безпріоритетною?

10 Наведіть сутність дисципліни обслуговування SRT (shortest remaining time)

11 Що необхідно для забезпечення двійкової сумісності ОС?

12 Які засоби мають бути на рівні С2 безпеки ОС?

13 Наведіть сутність дисципліни обслуговування RR (round robin).

14 У чому полягає принцип віртуалізації ОС?

15 Що необхідно для забезпечення двійкової сумісності ОС?

16 Які засоби мають бути на рівні С1 безпеки ОС?

17 Які функції виконує підсистема планування процесів ядра ОС?

18 У чому полягає принцип незалежності програм від зовнішніх пристроїв?

19 Що необхідно для забезпечення сумісності ОС на рівні вихідних текстів?

20 Яке завдання включає до себе планування процесів?

21 У чому полягає принцип сумісності ОС?

22 Що необхідно для забезпечення сумісності ОС на рівні вихідних текстів?

23 Яке завдання включає до себе планування процесів?

24 У чому полягає принцип відкритості ОС?

25 Яким чином організовується корпоративна багатозадачність?

26 Яке завдання включає до себе планування процесів?

27 Які функції виконує підсистема введення-виведення ядра ОС?

28 З якої причини відбувається зміна активного процесу?

29 Наведіть сутність дисципліни обслуговування FCFS (first come – first served).

30 Які функції виконує файлова підсистема ядра ОС?

31 Яким чином організовується витісняюча багатозадачність?

32 Наведіть сутність дисципліни обслуговування SJN (shortest job next).

33 Які функції виконує підсистема управління пам’яттю ядра ОС?

34 Яким чином організовується витісняюча багатозадачність?

35 З якої причини відбувається зміна активного процесу?

36 Від чого залежить віртуальний адресний простір?

37 Які задачі вирішує підсистема введення/виведення ОС?

38 Які задачі вирішує файлова підсистема ОС?

39 Які задачі вирішує підсистема управління пам’яттю ОС?

40 Які задачі вирішує підсистема управління процесами ОС?

41 Які задачі вирішує підсистема інтерфейсу користувача ОС?

42 Які задачі вирішує підсистема обліку користувачів ОС?

43 Які команди виконуються в режимі процесора, призначеному для користувача?

44 Які команди виконуються в привілейованому режимі процесора?

45 Які дії повинна першими виконати операційна система для запуску програми користувача?

46 Що є недоліком витісняючої багатозадачності?

47 Що є недоліком корпоративної багатозадачності?

48 Коли відбувається зміна активного процесу, якщо алгоритм базується на квантуванні?

СПИСОК ЛІТЕРАТУРИ

1 Шеховцов В. А. Операційні системи. Київ : Видавнича група BHV, 2005. 325 с.

2 Операційні системи: навч. посіб./ уклад. М. Ф. Бондаренко, О. Г.Качко. Харків : Компанія СМІТ, 2012. 417 с.

3 Габрусєв В. Ю., Лапінський В. В., Нестеренко О. В. Основи операційних систем. Ядро, процес, потік. Київ : Навчальна книга - Богдан, 2007. 96 с.

4 Єфименко В. В., Оніщенко С. М., Франчук В. М. Операційні системи. Лабораторний практикум: навч. посіб. Київ : НПУ імені М. П. Драгоманова, 2008. 124 с.

5 Інформатика. Комп’ютерна техніка. Комп’ютерні технології : навч. посіб./ уклад. В. А. Баженов та ін. Київ : Каравела, 2008. 640 с.

6 Грайворонський М. В., Новіков О. М. Безпека інформаційно-комунікаційних систем. Київ : Видавнича група BHV, 2009. 608 с

7 Нарожний В. В. Цифрові електронно-обчислювальні машини: конспект лекцій. Харків: УкрДАЗТ, 2010. 105 с.

МЕТОДИЧНІ ВКАЗІВКИ
до практичних занять та самостійної роботи

з дисципліни
«ОПЕРАЦІЙНІ СИСТЕМИ»

Відповідальний за випуск Павленко Є. П.

Редактор Решетилова В. В.

Підписано до друку 07.07.20 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 1,5. Тираж 5. Замовлення №

Видавець та виготовлювач Український державний університет
залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.