**1. Вступ.**

QNX - це зареєстрована торгова марка фірми Quantum Software Systems, Canada. Фірма заснована в 1980 році. У той же самий час QNX - назва інтегрованої операційної системи, призначеної для підтримки роботи ЛВС у реальному масштабі часу і розробленою фірмою Quantum. В даний час QNX знаходить усе найбільш широкий попит на ринках Європи, Канади і США завдяки своїм унікальним властивостям. QNX - це багатозадачна багатокористувацька - операційна система, що працює на РС-совместительных комп'ютерах.

QNX - система реального часу, у якій реалізована концепція зв'язку між задачами на основі повідомлень, що посилаються від однієї задачі до іншої, причому задачі ці можуть знаходитися як на тому самому вузлі ЛВС, так і на різних. Реальний час і концепція зв'язку між задачами у виді повідомлень впливають на розроблювальне для QNX програмне забезпечення і на програміста, що прагне з максимальною вигодою використовувати переваги системи. У такий спосіб можна створювати загальні прикладні програми, що придатні для виконання в будь-який операционой середовищу, але в той же час можна створювати і такі програми, для яких QNX буде найбільш придатним операційним середовищем. Закінчуючи огляд ОС QNX, варто додати, що названа ОС може працювати в режимі захисту пам'яті, включає такі стандарти як POSIX і IEEE, а розроблювачі продовжують поповнювати її новими продуктами і властивостями. QNX може в даний час функціонувати на машинах із процесорами від і8088 до і80486, включаючи PS/2. QNX підтримує 255 вузлів мережі (процесорів), що можуть спільно використовувати програми, файли і периферійні пристрої. QNX у середньому виконує операції в 20 разів швидше, ніж UNIX, забезпечуючи цілком роботу мережі, при цьому вона вимагає 140КБ оперативної пам'яті. QNX мається эмулятор РС-DOS.

**2. Функціональні можливості QNX.**

Застосовувані в даний час традиційні операційні системи відомі як "монолітні". На відміну від них, у QNX усі функції "обертаються" навколо ядра, що відповідає за створення нових задач і їхнє переключення, організацію послідовної комунікації периферійних вузлів, забезпечення інтерфейсу дисків і їхньої файлової системи, а також забезпечення комунікаційних мереж. Образно таку систему можна представити у виді "колеса", у центрі якого знаходиться + ядро ОС. Програми зв'язуються з ядром за допомогою системних викликів, заснованих на перериваннях і спеціальному механізмі зв'язку. Так, якщо задача (програма) формує виклик до операційної системи, те остання виконує викликану функцію у власному адресному просторі, представляючи із себе своєрідну велику бібліотеку підпрограм, що використовується спільно виконуваними програмами (задачами).

У "монолітних" операційних системах функції введення/висновку файлів на те чи інший пристрій повинні виконуватися ядром. Тому, щоб модифікувати ці функції, треба модифікувати саму операційну систему. А оскільки "монолітні" системи компонуються з урахуванням безлічі внутрішньо властивих їм зв'язків між компонентами, те всякі зміни в системі можуть бути просто небезпечні. Тому твердо можна сказати, що QNX є функціональною альтернативою монолітним системам, тому що вона має властивість "передачі повідомлень". У QNX, тому, більшість адміністративних задач працюють як окремі задачі, а не знаходяться постійно в пам'яті, у ядрі. Так, одна з задач може відповідати за операції на пристроях з послідовним уведенням/висновком, інша - за системні файлові функції, третя - за створення і диспетчирование задач і розподіл пам'яті, четверта - за роботу мережі (мережні комунікації). Тоді основною задачею ядра стає надання засобів зв'язку програм із задачами- адміністраторами й один з одним за допомогою повідомлень.

Повідомлення - це послідовність байтів довільної довжини (0-65535 байтів) довільного формату. Протокол обміну повідомленнями виглядає в такий спосіб. Наприклад, задача блокується для чекання повідомлення. Інша ж задача, посилає першої повідомлення і при цьому блокується сама, очікуючи відповіді. Перша задача розблокується, обробляє повідомлення і відповідає, разблокируя при цьому другу задачу.

**3. Структура QNX.**

QNX - базується на концепції передачі повідомлень. Ядро системи забезпечує передачу цих повідомлень, а також їхню диспетчеризацію. Крім того, ядро керує тимчасовими перериваннями. Виконання інших функцій забезпечується задач-адміністраторами, сім з який є основними.

1) Адміністратор задач (task) відповідає за розподілення пам'яті, запуск і закінчення задач у системі. Програма, що бажає створити задачу, посилає повідомлення Адміністратору задач і блокується для чекання відповіді. Якщо нова задача повинна виконуватися одночасно з її задачею, що породжує, Адміністратор задач створює її, і, відповідаючи, видає задачі, що породжує, ідентифікатор (id) створеної задачі. У противному випадку ніякого повідомлення не посилається доти, поки нова задача не закінчиться сама по собі. У цьому випадку у відповіді Адміністратора задач будуть міститися кінцеві характеристики задачі, що закінчилася.

2) Адміністратор периферійних пристроїв (dev) керує всією периферією: консоллю, терміналами, модемами, принтерами, віртуальними терміналами (вікнами). Він містить драйвери для цих пристроїв. Адміністратор периферійних пристроїв відповідає також за такі допоміжні функції, як висновок луни на екран, стирання і відновлення рядків і т.д. Програми бажаючі одержати інформацію з термінала, повинні зв'язатися з Адміністратором периферійних пристроїв. Зв'язок потрібна й у тому випадку, коли програми побажають установити термінал у визначений стан (режим). Адміністратор периферійних пристроїв відповідає також за асинхронне використання задачами терміналів і інших пристроїв.

3) Адміністратор файлової системи (fsys) відповідає за створення і роботу файлової системи. Програми, що бажають відкрити, закрити, чи читати писати файли, посилають повідомлення Адміністратору файлової системи.

4) Адміністратор неодруженої роботи (idle) системи "усмоктує" незайнятий час у системі. Він також виступає за замовчуванням як власник задач, своїх батьків, що пережили. Сам він ніколи не вмирає.

5) Мережний Адміністратор (net) забезпечує комунікації в мережі. Звичайно він викликається програмами й іншими задачами- адміністраторами у випадках, коли потрібен зв'язок з іншими вузлами (процесами).

6) Адміністратор-таймер (timer) дозволяє керувати програмами й асинхронними подіями, критичними вчасно, а також здійснювати повторний запуск програм.

7) Адміністратор черг (queue). Веде черга повідомлень між задачами як на одному вузлі, так і у всій мережі.

У QNX мається кілька інших задач-адміністраторів, що відповідають за доступ до різних ресурсів системи.

**4. Особливості виконання задач у QNX.**

QNX відрізняється тим, що деяка задача може бути частиною операційної системи і її чи середовища може бути "віртуальною" задачею. Остання має більшість атрибутів локальної задачі, у той час як виконується на інших процесорах (в інших вузлах мережі). Кожна задача, у тому числі і та, котра внутрішньо зв'язана з операційною системою, одержує унікальний ідентифікатор (tid). Щоб передати повідомлення іншій задачі, треба знати неї идентификатор. tid - це два байти, де в нижньому поміщений фактичний номер задачі (індекс, під яким вона поміщена в таблицю задач), а у верхньому - її послідовний номер і віртуальний прапор. Завдяки послідовному номеру, фактичний номер стає унікальним при вході в таблицю задач, якщо для двох різних задач використовується той самий вхід у таблиці. Для "дізнавання" імен (tid) задач у QNX існує два способи: порти і сервери імен.

а) порти - своєрідні крапки рандеву для задач, що позначаються невеликим цілим числом без знака. Типова конфігурація QNX має 28 портів у дійсному режимі і 40 - у захищеному. 16 з них резервуються самою операційною системою. Так, якщо задача - адміністратор бажає ідентифікувати себе для інших програм, вона виконує операцію приєднання до заздалегідь визначеного порту. Програми ж, від'єднуючи від даного порту, одержують tid задач-адміністратора, видаваною операційною системою. (В інших ОС порти називаються ще семафорами). Порти використовуються в QNX і як механізми переривання.

б) сервери імен - це задачі, що створюють список зареєстрованих імен програм і ідентифікаторів - tіd's - зв'язаних з іменами цих програм. Зареєстроване ім'я- це група до 8 символів, що задача за допомогою сервера може помістити в список. Будь-яка програма може довідатися це ім'я шляхом звертання до сервера імен; далі можна посилати повідомлення на це ім'я.

У мережній конфігурації розрізняються локальні і глобальні сервери імен. Глобальний сервер - задача зі своїм списком імен, що діють на одному з вузлів мережі. Зокрема, серед імен може бути ім'я принтера, розміщеного на іншому вузлі мережі. Програми можуть зажадати ім'я цього принтера і потім посилати туди свої файли для висновку, постачивши ці файли різними параметрами. Завдяки "локальній" реєстрації виключається конфлікт імен у мережі. QNX включає спеціальну задачу, що дозволяє створювати розподілену базу даных імен задач.

**5. Віртуальні задачі і віртуальні ланцюги.**

Завдяки такій властивості QNX, як можливість обміну посланнями між задачами і вузлами мережі, програми не піклуються про конкретне розміщення ресурсів у мережі. Ця властивість додає системі незвичайну гнучкість. Так, вузли можуть довільно додаватися і вилучатися із системи, не торкаючись системні програми. QNX здобуває цю конфігураційну незалежність завдяки застосуванню концепції "віртуальних" задач. У віртуальних задач безпосередній код і дані, будучи на одному з вилучених вузлів, виникають і поводяться так, ніби вони були локальними задачами якогось вузла з всіма атрибутами і привілеями. Програма, що посилає повідомлення в мережі, ніколи не посилає його точно. Спочатку вона відкриває "віртуальний ланцюжок". Віртуальний ланцюжок включає усі віртуальні задачі, зв'язані між собою. На обох кінцях такого зв'язку маються буфери, що дозволяють зберігати найбільше послання з тих, котрі ланцюжок може нести в даному сеансі зв'язку. Мережний адміністратор поміщає в ці буфери всі повідомлення для з'єднаних задач. Віртуальна задача, таким чином, займає усього лише простір, необхідний для буфера і входу в таблиці задач. Щоб відкрити зв'язок, необхідно знати ідентифікатор вузла і задачі, з яким установлюється зв'язок. Для цього необхідно знать ідентифікатор задачі- адміністратора, відповідального за дану чи функцію глобальне ім'я сервера. Не розкриваючи тут докладно механізм обміну посланнями, додамо лише, що наша задача може узагалі виконуватися на іншому вузлі, де, допустимо, мається більш зроблений процесор.

**6. Файлова система QNX**

Файлова система в QNX, можна сказати, UNIX-подібна в тім, що має деревоподібну структуру каталогів, пдібні угоди по найменуванню файлів і забезпеченню безпеки даних (і т.д.). І взагалі, файлове середовище, з погляду системного програміста, схожа на таку в UNIX. Однак варто виділити внутрішня відмінність (і перевага) файлової системи QNX від UNIX. Це її міцність (живучість), зменшену фрагментарність файлів і збільшену швидкість роботи. Самі файли в QNX організовані за принципом списку ділянок з дублюванням покажчиків. Це дає перевагу в тім, що, якщо вхід у каталог загублений чи ушкоджений, то файл усе рівно може бути відновлений шляхом відшукання хоча б одного з його ділянок. QNX може поділяти той самий диск з іншими операційними системами. На відміну від UNIX, QNX не зв'язує ім'я файлу з його фізичною інформацією. Дані у файлах зберігаються в каталогах поряд з іменами файлів.

Для підтримки авторизованого доступу до інформації QNX має номера груп доступу, як і в інших операційних системах. Ці числа наступні : [ група, член-в-групі ]. Користувач номер групи, що має, 255, є суперкористувачем, що має необмежений доступ до файлової системи. Користувач, що має номер члена групи, рівний 255, є лідером у цій групі і має всі привілеї по роботі з файлами в даній групі. QNX підтримує також механізм закриття записів, наявний у UNIX System V.

Важливої властивістю QNX є те, що в її складі може поставлятися Адміністратор файлової системи MS/DOS, що, будучи запущений як звичайна задача, забезпечує безпосередній доступ до гнучких і твердих дисків системи. Файли на цих дисках можуть редагуватися, виводитися на печатку, компілюватися і т.п., як якби вони знаходилися в стандартній файловій системі MS/DOS.

Ще однією ключовою особливістю QNX, що пояснює гнучкість і ефективність системи, є розподілена система бібліотек. Їхні два типи: бібліотеки, що завантажуються разом з ядром, і бібліотеки монтуємі. Останні можуть створюватися користувачем. Бібліотеки, що завантажуються разом з ядром, виконують по більшій частині системні функції (аналіз імен файлів, забезпечення введення/виводу, форматування повідомлень і т.д.). Важливою функцією поділюваних бібліотек є забезпечення незалежності системи у випадку заміни устаткування.

**7. Засоби програмування.**

QNX написана мовою високого рівня "С". У системі також мається компілятор BASIC. Обидві мови використовують той самий генератор кодів і мають однаковий формат об'єктних файлів. Функціональна бібліотека включає процедури вводу/виводу на рівні пристроїв і файлів, математичних функцій, графічні, прямого вводу/виводу на консоль, а також пакет рутин, що аналізує термінали. Фірма Quantum ассемблер, линкер і два отладчика, один із яких виконує налагодження вихідних текстів, а іншої - системних програм.

**8. QNX і локальні мережі.**

Локальні обчислювальні мережі (ЛВС) обіцяють революціонізувати шляху використання комп'ютерів. Завдяки мережі, від 10 до 100 разів обробки даних, що збільшує потужність, персональними комп'ютерами, можна створювати "установи майбутнього". В міру удосконалювання технології мережі стають усе більш доступними, отже можна одержати доступ до глобальної бази знань. Звичайно, ключовим питанням доступу й обчислень залишаються питання швидкої комутації. ОС QNX, маючи швидкість переключення задач 7200 задач/сек при частоті 16 Мгц, гарантує саме сприятливе середовище для зв'язку задач, що виконуються на різних машинах. QNX бачить "світ" як збори задач, що виконуються на одному чи декількох вузлах (процесорах) і имеющих доступ до ресурсів усієї системи. У мережі QNX немає обмежень на використання ресурсів будь-якою задачею. Не існує також обмежень на вибір задачею процесора, на якому вона буде виконуватися. Це означає, що, наприклад, програма може виводити інформацію на будь-який принтер, приєднаний до будь-якої машини мережі, а також звернутися до будь-якого файлу на будь-якому дисководу. Для розходження вузлів мережі користувачами і програмами, вузлам призначаються номери. Користувач, працюючи на одному з вузлів мережі, за замовчуванням буде використовувати русурсы цього вузла, однак при застосуванні в командах "переадресації", користувач здійснює доступ до ресурсів інших вузлів.

# 9.Що таке QNX Windows

Якщо говорити коротко, ***QNX Windows*** це ***інтелектуальний графічний сервер,*** однак щоб зрозуміти зміст цього вираження необхідні досить ясні пояснення, тому що дана система має мало загального з загальновідомими системами начебто MS Windows чи X Window System.

Як ні дивно, її розроблювачі (канадська фірма Basis Computer Systems Inc.) зуміла запропонувати щось несхоже ні на що інше. Розроблялася ця система спеціально для ОС QNX з розрахунком на додатки працюючі в реальному масштабі часу і з використанням усіх переваг цієї ОС. Права на систему належать фірмі QNX Software Systems Ltd. (розроблювач OC QNX), оскільки розробка велася по її замовленню. Робота ця починалася ще коли існувала тільки серія 2.x OC QNX, призначена для процесорів і8086-80286.

Перша версія системи була випущена в 1990 році. Згодом QNX Windows була перенесена в 32-х розрядну версію QNX (серія 4.x), однак варто відзначити той факт, що відповідна версія даної системи здатна працювати навіть на IBM PC/XT-86 (при цьому забезпечується багатокористувацький доступ, що витісняє багатозадачність, прозорі мережні комунікації і багато чого іншого).

Тепер самий час розглянути систему QNX Windows з погляду зовнішнього інтерфейсу й архітектурних концепцій. Зовнішній інтерфейс цієї системи сполучимо зі специфікаціями графічного інтерфейсу користувача OPEN LOOK, що були розроблені експертами фірми Sun Microsystems, за замовленням корпорації AT&T. Оскільки в нашій країні з ним знайомо лише незначна кількість фахівців, яким довелось попрацювати з технікою і програмним забезпеченням фірми Sun, має сенс зупинитися на цьому питанні докладніше.

## 10.Користувальницький інтерфейс OPEN LOOK

Насамперед слід уточнити, що ми будемо розуміти під інтерфейсом. Зверніть увагу, що OPEN LOOK це саме ***специфікація***, а не конкретна реалізація інтерфейсу. Розробка велася без оглядки на який-небудь з існуючих інтерфейсів і без орієнтації на яке-небудь конкретне середовище виконання, правда до деякої міри використовувалися результати дослідницьких робіт, що проводилися компанією Rank Xerox. Специфікації включають три книги, що визначають зовнішній вигляд графічних елементів інтерфейсу, рекомендації зі стилю додатків і процедуру сертифікації додатків на відповідність специфікаціям. Усі книги опубліковані видавництвом Addison-Wesley Inc., російських перекладів не існує. Офіційна публікація специфікацій відбулася в 1990 році. Потім з'явилося кілька реалізацій, в основному для середовища X Window. Найбільш відомі серед них системи Open Windows і X View, розроблені фірмою Sun, а також бібліотека виджетов AT&T. Реалізації класифікуються на 3 рівні, по повноті. В даний час інтерфейс OPEN LOOK є основним конкурентом інтерфейсу OSF/Motif, широко використовуваного в системі X Window, що більш знаком російським фахівцям, через того що основними джерелами натхнення для його творців були MS Windows і Presentations Manager.

Суперечки про те, який інтерфейс краще, напевно не закінчаться ніколи, оскільки в цьому питанні занадто багато чого залежить від особистого смаку і задач, який потрібно вирішувати. На погляд автора, OPEN LOOK відрізняється деякої зовнішній аскетичністю, наприклад рекомендується уникати застосування іконок як органи керування , і не зловживати візуальними ефектами (типу ' вікон, щовибухають,'). Якщо задуматися, то для багатьох додатків це непогані рекомендації, адже користувача цікавить не зовнішня краса програми, а задача яку він повинний вирішити. З іншого боку, розроблювачі OPEN LOOK пишуть про те, що цей інтерфейс призначений не для конкуренції, а для співіснування з іншими інтерфейсами. Одним словом, вибирайте те, що більше по смаку Вам (чи Вашим користувачам).

Основними принципами, покладеними в основу при проектуванні інтерфейсу OPEN LOOK є ***ефективність, несуперечність і простота***, тобто саме ті принципи, що необхідні для додатків, що працюють у реальному масштабі часу. Не випадкова більшість додатків створених з використанням цього інтерфейсу орієнтовані саме на дану область застосування.

Під ефективністю розуміється надання користувачу можливості швидко (з мінімумом проміжних кроків) виконувати бажані дії, при цьому повинні бути в однаковій мірі враховані інтереси як новачків, так і досвідчених користувачів. Під несуперечністю розуміється те, що в будь-якій частині додатка (і у всіх додатках) однорідні дії повинні виконуватися однаковим способом. Під простотою розуміється легка видимість інтерфейсу, тобто користувач не повинний подовгу блукати в лабіринтах меню і вікон у пошуках потрібної чи команди опції. Крім той інтерфейс не повинн фокусувати увагу користувача на собі самому, відволікаючи його від виконання основної чи задачі змушуючи задумуватися над змістом незвичайних ефектів і, найважливіше - він повинний мати чітко виражену систематичну концепцію взаємодії людини з програмою.

Звичайно, будь-який фахівець із графічних інтерфейсів скаже, що цим принципам повинна (і звичайно намагається) випливати будь-яка пристойна графічна система. Однак благими намірами вимощена дорога в пекло, і досить поексперементувати з деякою кількістю програм для MS Windows, щоб помітити як далеко більшості з них до відповідності оголошеним принципам. Проблема полягає в тім, що ці принципи є не більш ніж загальним місцем, якщо вони не супроводжуються конкретними рекомендаціями з прийняття тих чи інших рішень.

Кожний, хто хоча б раз приймав рішення, що стосуються інтерфейсу програми, знає як важко зупинитися на чому або одному. Достоїнство ж інтерфейсу OPEN LOOK полягає саме в тім, що він заснований на ретельно пророблених принципах побудови додатків і взаємодії користувача з ними, підкріплених самими детальними рекомендаціями щодо застосування тих чи інших графічних елементів у типових ситуаціях. У результаті програмою, сертифікованої на відповідність інтерфейсу OPEN LOOK, будь-який користувач, знайомий з ним, користається так само легко, як будь-який водій керує автомобілем будь-якої марки - він не повинний задумуватися де знаходиться чи кермо, чи яка з педалей включає гальмо.

Тепер розглянемо коротко основні елементи інтерфейсу OPEN LOOK. Очевидно, що головним елементом будь-якого графічного інтерфейсу є вікна (windows). У даному інтерфейсі визначено чотири типи вікон:

Базове вікно (Base window) - Вікно, що з'являється при запуску додатка й у який зосереджена велика частина функціональних можливостей і органів керування додатка. Тут же як правило максимальне використовується візуальний зворотний зв'язок.

Спливаюче вікно (Pop-up Window) - Вікно призначене для виконання якої або конкретної дії, чи вибору зміни параметрів і т.д. При необхідності частого використання може бути 'приколено' до екрана за допомогою 'шпильки'. Визначено чотири підтипи:

Command Window - Призначено для виконання команд із можливістю завдання параметрів (Find & Replace).

Properties Window - Призначено для установки чи зміни параметрів (властивостей) обраного об'єкта.

Help Window - Призначено для висновку довідкової інформації про об'єкт, над яким знаходиться покажчик миші.

Меню (Menu Window) - Вікно призначене для вибору одного з декількох пропонованих варіантів. Може мати смугу прокручування (scrollbar) якщо варіантів багато.

Самою незвичайною особливістю вікон в інтерфейсі OPEN LOOK є можливість визначити для вікна 'крайні зони' (Window Bars), тобто проміжки між границею вікна і його внутрішньою частиною.

Приклад такого проміжку - область заголовка вікна, але тут ці проміжки можуть знаходитися з будь-якої сторони вікна і мати будь-як ширину. Вони мають незалежну систему координат і можуть використовуватися для висновку так само як і внутрішня область вікна. Верхній (і іноді лівий) проміжки використовуються як область керування (див. далі), а правий і нижній проміжки - для зон скролінга. Нижній проміжок використовується ще і для висновку повідомлень (аналог Message Area в інтерфейсі OSF/Motif).

Внутрішня область вікна може бути поділена на 'кватирки' (Panes), кожна з який може використовуватися незалежно (і мати свої власні смуги прокручування). Усього можна визначити не більш 4-х кватирок (не більш 2-х в одному напрямку), причому їхній загальний зовнішній контур повинний збігатися з контуром вікна.

При запуску 'менеджера робочого столу', усі вікна забезпечуються декораціями (тривимірна рамка, куточки для зміни розмірів, заголовок, службова кнопка (window button) чи шпилька (pushpin)), іконками (для базових вікон) і меню вікна, що залежить від типу.

Для базового вікна будь-якого додатка через пункт Properties у меню можна задати початковий стан (іконка, стандартне, розкрите), початкове положення на екрані і розмір, а також колірну палітру для вікон і діалогів, що відносяться до даного додатка. Пункт меню базового вікна Quit надає стандартний (і звичайно єдиний) спосіб завершення додатка.

Самим характерним елементом спливаючого вікна служить шпилька, що є навряд чи не найвідомішим елементом інтерфейсу OPEN LOOK. При відкритті вікна вона звичайно 'лежить на боці' у верхньому лівому куті. Коли користувач виконує дію, зв'язана з вікном, воно звичайно закривається. Однак, якщо клацнути мишею по шпильці, вона 'устромляється' в екран і 'приколює' вікно до нього. Після цього вікно не закривається, поки користувач ще раз не натисне на шпильку, що дозволяє багаторазово виконувати часто використовувані дії. Особливо це зручно при використанні спливаючого меню - його можна приколоти в будь-якім місці екрана, що зручно користувачу. Помітьте, що ця красива ідея була згодом використана, у трохи іншій формі, в інтерфейсі OSF/Motif версії 1.2, за назвою Tear-Off Menu (там замість шпильки використовується 'лінія перфорації' і користувач може 'відірвати' меню від 'корінця' як листок із блокнота).

Меню в інтерфейсі OPEN LOOK визначені 3-х типів: Drop-Down, Pull-Right і Pop-Up. Перший тип є основним і використовується разом із кнопками - меню з'являється при натисканні правої кнопки миші на кнопці (звичайно під кнопкою). Другий тип є різновидом першого і служить для створення ієрархічних меню. Таке меню використовується тільки разом з одним із двох інших типів і з'являється при натисканні правої кнопки миші на елементі меню старшого рівня, c яким воно зв'язано (звичайно праворуч від нього - звідси і назва). Третій тип призначений для забезпечення швидкого доступу до часто використовуваних функцій, доступним через основне меню і з'являється при натисканні правої кнопки миші на будь-якій порожній ділянці вікна. Кожне меню повинне мати один елемент, що вибирається за замовчуванням. Для всіх типів меню (особливо приколених) додатка повинні забезпечувати візуальний зворотний зв'язок - якщо який-небудь елемент меню незастосуємо в даному контексті додатка - він повинний бути деактивований і навпаки.

Замість традиційної області меню (Menu Bar), базове вікно в інтерфейсі OPEN LOOK має область керування (Control Area), у якості якої рекомендується верхній проміжок вікна (Top Bar). У цій області прийнято розташовувати кнопки, c можуть бути асоційовані Drop-Down меню.

Рекомендується також склад і порядок розташування кнопок - File, View, Edit, Properties, причому для кнопок File і Edit, рекомендується ще і стандартний склад асоційованих меню. Склад інших меню залежить від додатка. Спеціальної області для введення команд (по типі Command Area у OSF/Motif) у базовому вікні не передбачається.

Кнопки (Buttons) в інтерфейсі OPEN LOOK мають овальну форму і завжди виводяться з 3D-ефектом, тобто вони 'випирають' над поверхнею, а при натисканні утоплюються. Одна з кнопок у вікні призначається 'кнопкою за замовчуванням' і буде виділятися ободком усередині контуру кнопки. У деяких реалізаціях ця кнопка буде натискатися при натисканні клавіші Enter. Усі кнопки класифікуються на 3 типи - кнопки меню (Menu Buttons), кнопки вікон (Window Buttons) і кнопки команд (Command Buttons). Кнопки першого типу позначаються значком трикутника, спрямованого вниз. При натисканні лівої кнопки миші на таку кнопку автоматично вибирається елемент за замовчуванням з асоційованого меню (інтерпретація AT&T) чи цей елемент виводиться на кнопку замість назви, щоб його можна було побачити (інтерпретація Sun Microsystems). При натисканні правої кнопки миші, що відповідає меню з'являється під кнопкою. Кнопки другого типу виділяються знаком (...). При натисканні на таку кнопку відкривається яке-небудь вікно. Кнопки останнього типу ніяк не виділяються. При натисканні на них просто виконується відповідне дія. Точно такая-жі класифікація застосовується стосовно елементів меню. Це відноситься і до виділення, але замість трикутника спрямованого вниз, застосовується трикутник, спрямований вправо.

Принципи взаємодії з користувачем в інтерфейсі орієнтовані в основному на використання миші, хоча можна обходитися і без неї. Позиціювання через клавіатуру не визначено, хоча і підтримується в тім чи іншому ступені більшістю реалізацій (у тому числі QNX Windows). Поняття фокуса введення стосовно кнопок також не визначено, тому, що OPEN LOOK використовує інший механізм для натискання кнопки через клавіатуру - мнемонічні комбінації. Для кожної кнопки можна визначити букву, що входить у її назву, у якості мнемонічної (за замовчуванням перша). Вона буде виділятися (кольором підкресленням) при відображенні кнопки. Одночасне натискання клавіші Alt і зазначеної букви буде еквівалентно натисканню лівої кнопки миші на цю кнопку.

C іншої сторони, OPEN LOOK дуже систематично використовує миша, причому всі 3 її кнопки. Вони навіть мають спеціальні назви - SELECT, ADJUST і MENU. Кнопка SELECT (ліва) служить для вибірки елементів у вікні, переключення фокуса введення між вікнами, натискання на кнопки. Кнопка ADJUST (середня) призначена для зміни поточної вибірки (наприклад, коли користувач повинний вибрати 3 файли, він повинний натиснути SELECT на першому і ADJUST на інших. Повторний SELECT скидає попередню вибірку, а повторний ADJUST виключає елемент із вибірки). Якщо використовується двукнопочна миша, натискання комбінації Shift-SELECT звичайно емулює ADJUST. Кнопка MENU (права) служить для відкриття меню всіх типів. Ті, хто знає інтерфейс OSF/Motif помітять, що основне розходження - у використанні середньої кнопки - там вона призначена для операцій у режимі Drаg'n'Drор (версія 1.2). В інтерфейсі OPEN LOOK для цієї мети застосовується кнопка SELECT, але варто помітити, що в цілому концепція режиму Drag and Drop тут продумана слабкіше і недостатньо систематизована, що порозумівається тим, що Motif 1.2 з'явився на 2 роки пізніше чим OPEN LOOK (у попередніх версіях цей режим узагалі був відсутній).

Специфікації OPEN LOOK визначають також операції з Clipboard і з файлами, тобто менеджер фалів є необхідним елементом для реалізації 2-го рівня. Крім того даються рекомендації з приводу вибору й оформлення заголовків і міток, розміщення елементів усередині вікна, вибору назв для пунктів меню, формату повідомлень про помилки, вибору колірної палітри і ще багато деталей, про які починаючі проектувальники звичайно навіть не думають. Обсяг реферату не дозволяє нам, на жаль, навіть коротко розглянути всі елементи інтерфейсу OPEN LOOK, тому розглянемо тепер більш докладно систему QNX Windows як приклад реалізації описаних концепцій (причому одну з перших).

## 11.Розробка додатків для QNX Windows

Незважаючи на класичний прикладний інтерфейс із мовою “С”, QNX Windows зсередини являє собою обьєктно-орієнтовану систему, як і у випадку з X Window. Тут також використовується поняття ресурсу, але в іншому змісті. Ресурси - це картинки (pictures), елементи (picture elements), екрани (screens), вікна (windows), кватирки (panes) і діалоги (dialogs). Кожен ресурс має *власника* (процес) і *ім'я* (символьний рядок). Підтримка ресурсів здійснюється відповідними менеджерами. Деякі з ресурсів мають ієрархічні взаємини:

1. сервер може керувати декількома екранами;
2. екран може містити кілька вікон;
3. вікна можуть містити кілька кватирок (і зв'язаних з ними картинок);
4. картинки можуть містити кілька елементів;

Додатки взаємодіють з QNX Windows створюючи ресурси і маніпулюючи ними. Наприклад, вікна можуть бути відкриті і закриті, картинки - створені, скопійовані і вилучені, елементи - намальовані, скопійовані, переміщені, вилучені і т.д. Любою ресурс може знаходитися на будь-якому вузлі мережі QNX, оскільки всі маніпуляції здійснюються через прозорий для мережі механізм передачі повідомлень.

Базовою основою роботи QNX Windows із графікою служить унікальна концепція ‘картинки’ (**picture**). Картинку можна уявити собі як лист папера з координатною сіткою, незалежної від пристрою відображення (ідея, що викликає асоціації з мовою PostScript). Координати виміряються в типсах (**tips**) - десятих частках пункту - типографської одиниці, рівного 1/72 дюйма. Початок координат знаходиться в лівому верхньому куті, розміри картинки обмежені координатами 65535. **Весь графічний висновок, якщо спеціально не зазначене інше, відбувається не у вікна, а в картинки** (вікно може і не існувати).З кожної з ‘кватирок’ вікна можна динамічно зв'язати будь-яку існуючу картинку, можна також перемінити картинку в будь-який момент.

Інша фундаментальна концепція полягає у визначенні поняття графічного елемента(**picture element**), інакше кажучи **об'єкта** маючого **тип**. Типи в основному відповідають стандартним елементам, визначеним в інтерфейсі OPEN LOOK, і, саме головне, **ці типи відомі менеджеру екрана**. Висновок у картинку здійснюється **сервером,** по запиті додатка, що повідомляє серверу тип елемента, якому необхідно створити, і його атрибути. **Усе, що з'являється на екрані є екземпляри елементів того чи іншого типу.** Картинка ж є не що інше, як упорядкований список елементів.Це відноситься не тільки до картинок, створеним програмою, але і до вікон, меню, іконкам і т.п.

QNX Windows використовує наступні типи елементів:

|  |  |
| --- | --- |
| **arc** | дуга еліпса |
| **button** | кнопка |
| **curve** | крива Безьє 3-го порядку |
| **group** | комбінований елемент визначений програмістом |
| **image** | довільний кольоровий растр, від 4 до 16 біт на піксель |
| **line** | лінія |
| **link** | посилання на інший елемент |
| **number** | число ( чиціле з крапкою, що плаває,) |
| **paragraph** | параграф тексту (з табуляцією, відступами і шрифтом) |
| **pixmap** | растр, у форматі залежному від пристрою висновку |
| **points** | полігон (можливо замнкутый) |
| **rectangle** | прямокутник |
| **symbol** | довільний растр, що містить до 16 бітових площин |
| **string** | простий неформатованный текст |
| **text** | форматованный текст (з довільним шрифтом) |

З будь-яким елементом може бути асоційована мітка (**label**), діалог чи довільні дані, визначені програмістом. У залежності від типу, елементи мають характерний набір атрибутів (координати, колір, колір тіла, товщина ліній, шрифт і т.п.). Елементи всіх типів можуть мати також **опції** (options) і **стану** (states), що визначають поводження елемента і спосіб його висновку. По поводженню при натисканні на нього мишею, елемент може бути 'обираним' (selectable), що редагується, ' щоповідомляє' (notify), блокованим, що інвертує чи стан запам'ятовує стан. Виводитися елемент може стандартно, безпосередньо у вікно (минаючи картинку), із затемненням, з рамкою, з 3D-ефектом, з тінню.

Описуються елементи з незалежним від пристрою способом, за допомогою спеціальних структур даних. Упорядкована сукупність таких описів, постачена заголовком, утворить стандартну картинку (елементи існують тільки усередині картинок). Така картинка може бути збережена у файлі формату PICT і прочитана з нього.

Як уже згадувалося вище, елементами можна маніпулювати. Основним засобом ідентифікації елементів служить ім'я елемента (звичайно називане **тегом**). При цьому можуть використовуватися шаблони, схожі на шаблони імен файлів, що дозволяє ідентифікувати для однієї операції групу елементів (чи усі). Крім того, для ідентифікації може бути використана будь-яка комбінація опцій, що обмежує список елементів до бажаного складу (наприклад, можна специфікувати усі 'обирані' елементи), послідовний номер елемента в картинці, координати, прямокутна область картинки і т.д.

Для того, щоб зробити що-небудь з картинкою, додаток спецефікує елементи і вказує операцію, яку потрібно до них застосувати. З іншого боку, елементами може маніпулювати користувач, наприклад редагуючи значення текстового чи рядка змінюючи значення числа. У цьому випадку власник відповідного ресурсу (звичайно це процес створивший його), одержить відповідне повідомлення, якщо він замовляв подібне повідомлення. Інший підхід полягає в тому, що нові значення елементів можна запросити тоді, коли вони реально знадобляться. Процес-власник може також передати свої повноваження іншому процесу.

Для мінімізації вихідного коду QNX Windows використовує поняття графічного контексту, у якому зберігаються поточні значення ресурсів і атрибутів елементів усіх типів. Графічний контекст теж може бути збережений у файлі і відновлений відтіля.

Розглянуті вище концепції дозволили розроблювачам QNX Windows застосувати трохи дуже красивих і ефективних рішень, деякі з який не мають аналогів.

По-перше, усі ресурси, створювані додатками (вікна, картинки, діалоги) створюються менеджером екрана і **зберігаються в його адресному просторі.** Це радикально зменшує вимоги програм до оперативної пам'яті і підвищує швидкість виконання графічних операцій. Саме в цьому головну відмінність графічних елементів QNX Windows від виджетов X Window, що хоча і маскують від додатка деталі своєї роботи, але зберігаються в адресному просторі додатка. При цьому сервер у X Window усього лише виконує X Protocol, що має досить низький рівень. Наслідком цього є не тільки високий внутрішній трафік але і той менш помітний факт, що нитка керування при модифікації вмісту екрана дуже часто передається додатку, що дуже небажано для додатків реального часу. У QNX Windows усі зовсім інакше, тому що додаток повинний тільки замовити високорівневу операцію серверу, а виконує він її без подальшого втручання. При цьому витрати пам'яті на сервері теж невеликі, оскільки поняття типу елемента дозволяє **відмовитися від збереження растрів**, застосовуючи замість них картинки (сервер знає як малювати елементи відомого йому типу). Це злегка нагадує ОС NextStep, де зображення зберігаються у форматі Display PostScript, але картинки QNX Windows вимагають на порядок менше пам'яті (пом’ятаєте, що для кольоровий NextStep потрібно 64Mb пам'яті?) тому що PostScript описує абстрактні команди малювання, а не об'єкти визначених типів. По-друге, можливість групової ідентифікації елементів знижує трафік повідомлень і дозволяє серверу оптимізувати виконання графічних операцій. Крім того, це сильно спрощує логіку і розмір прикладних програм, про що ше буде сказано далі.

По-третє, маючи у своєму розпорядженні всі картинки і вікна, сервер може сам відслідковувати перекриття вікон, обчислювати відсікання і виконувати перемальовування умісту вікон! Це означає, що програма в QNX Windows не зобов'язана обробляти повідомлення типу EXPOSE чи WM\_PAINT, як це приходиться робити в системах X Window чи MS Windows. Режим Backing Store, що з'явився в X Window System release 11, не йде ні в яке порівняння, оскільки там зберігаються растри, і у випадку недостачі пам'яті (що дуже ймовірно) сервер цей режим виключає (тобто програма не може і не повинна на нього покладатися). Звичайно, будь-яка проміжна ступінь вносить затримки, тому для програм, критично залежних від швидкості висновку на екран, передбачений режим прямого висновку елементів у вікно.

По-четверте, менеджер екрана може автоматично забезпечувати бажане **поводження** вікон і кватирок, що задається набором **опцій**, як і у випадку з елементами. Наприклад, можна спецмфікувати які події варто посилати власнику ресурсів, а які немає.

Більш того, деякі типові ситуації сервер може обробляти самостійно, наприклад видавати повідомлення при введенні користувачем значення, що виходить за границі зазначеного діапазону, автоматично виводити смуги прокручування при якщо картинка не міститься в кватирку цілком і підтримувати скролінг. Вікно може містити елементи зі стандартними тегами, при натисканні на який сервер буде виконувати стандартні дії. Для елементів можна задати їхнє поводження при проходженні над ними чи курсору при натисканні на них.

Якщо грамотно спроектувати інтерфейс, то в більшості простих випадків прикладній програмі залишиться тільки ініціалізувати його і закрити при виході, не вважаючи обробки декількох повідомлень, заради яких програма і написана. Саме завдяки таким можливостям QNX Windows можна називати інтелектуальним графічним сервером.

Ті, хто має досвід програмування в X Window, знають, що програми для цієї системи надзвичайно багатослівні. Величезна кількість функцій API робить його важкодоступним до огляду і ускладнює життя програмістам, через що для X Window була написана найбільша кількість генераторів коду. Мінімальна програма, що виводить у вікно напис hello world! має довжину близько 30 рядків, що ще дуже непогано в порівнянні з MS Windows. Розмір модуля, що виконується, виходить близько 800 Kb, у залежності від конкретної реалізації. У QNX Windows така програма виходить усього на двох рядків довше, ніж класичний варіант Керингана і Рітчі, якщо використовувати стандартний діалог, і ще на дві строчки довше, якщо використовувати звичайне вікно. Розмір модуля, що виконується, 7500 байт, при запуску програма займає 92 Kb (32-х розрядна версія**,** flat-модель).

#include <windows/Qwindows.h>

void main ( void )

{

GraphicsOpen( NULL ); // з'єднання із сервером

Tell( "Hello Program", "Hello, World!" );

GraphicsClose( 0 );

}

Однак, як справедливо замічено в книзі X Toolkit Рrоgrаммеr's Manual, програма hello world! є виродженим прикладом для X Window. Якщо наростити приведену програму до тих можливостей настроювання які надає версія для X Window, ми одержимо порівнянний розмір вихідного коду (однак розмір виконуючого модуля зміниться дуже незначно!!!).

Проте, розглянемо ще один приклад. У комплект постачання OSF/Motif 1.2 входить демонстраційна програма DNDDemo, що дає приклад використання протоколу Drag and Drop, що з'явився в цій версії інтерфейсу Motif. Розмір вихідного коду цієї програми близько 3000 рядків мовою “С”, розмір модуля, що виконується, (у версії X Window для QNX) - близько 900 Kb. Для порівняння автор даної статті написав програму для QNX Windows еквівалентну функціональність, що забезпечує. Програма створює вікно з двома кватирками, у нижній кватирці розміщені 16 кольорових прямокутників. Користувач може малювати прямокутники у верхній кватирці, використовуючи звичайний механізм 'гумової нитки'. Намальовані прямокутники можна чи копіювати переміщати мишею як у межах кватирки, так і у верхню кватирку іншого вікна, відкритого іншою копією програми. Колір будь-якого прямокутника можна змінити, 'перетягнувши' бажаний колір з палітри в нижній кватирці.

При цьому підтримуються візуальні ефекти Drag Over (зміна курсору), Drag Under (узяття прямокутника в рамку) і контекстно-залежна функція Help. Розмір вихідного коду - 300 (триста) рядків мовою C. Розмір модуля, що виконується, (32-х розрядна версія, компілятор Watcom C 9.51) - 15 (п'ятнадцять) Kb. Обсяг статті не дозволяє привести тут вихідний текст програми, але автор готовий надати його по запиті всім бажаючим.

## 12.Висновок

ОС QNX і система QNX Windows з успіхом застосовувалася при реалізації багатьох великих проектів, деякі з який самі по собі досить цікаві, щоб написати про їх окрему статтю. Як приклади можна привести застосування QNX для систем сортування вхідного транспорту і продажу квитків на терміналах залізничної магістралі між Францією і Великобританією, прокладеної в тунелі під протокою Ла-Манш. Проектна пропускна здатність цієї магістралі в годинник списів - один склад кожні 2.5 хвилини, швидкість проходження потягів - 100 миль у годину, що висуває підвищені вимоги до кваліфікації машиністів. Для рішення цієї проблеми був створений спеціальний тренажерний центр, що використовує технологію віртуальної реальності для 100% імітації реальної обстановки в кабіні локомотива. QNX Windows була використана для створення візуальної підсистеми, що забезпечує возпроизведение оцифрованих зображень навколишнього пейзажу з частотою 25 кадрів/сек. Система імітує також звукові і гравітаційні ефекти.

Інший недавній приклад - централізована система керування морськими нафтовими платформами, створена в Новій Зеландії, що дозволяє здійснювати контроль над процесом перекачування нафти на всіх установках у режимі on-line через дистанційні датчики.

Резюме всьому сказаному можна зробити наступне: вчитися не тільки на чужих помилках, але і на чужих досягненнях. Якщо Ви збираєтеся розробляти додаток реального часу - спробуйте QNX. Якщо ж Вам потрібний ще і графічний інтерфейс - спробуйте QNX Windows ще до того як наб'єте шишки.

### Міністерство освіти і науки України

#### Київський промислово-економічний коледж

**Національного Авіаційного Університету**

##### РЕФЕРАТ

###### З предмету:

**«Операційні системи»**

**На тему:**

**«ОС QNX»**

**Виконав:**

**студент гр. 242-ПОМ**

**Михайленко Андрій**

#### Київ-2002

**Зміст:**

**1. Вступ**

**2. Функціональні можливості QNX**

**3. Структура QNX**

**4. Особливості виконання задач у QNX**

**5. Віртуальні задачі і віртуальні ланцюги**

**6. Файлова система QNX**

**7. Засоби програмування**

**8. QNX і локальні мережі**

**9. Що таке QNX Windows**

**10. Користувальницький інтерфейс OPEN LOOK**

**11. Розробка додатків для QNX Windows**

**12. Висновок**

**Інформація взята з сайтів:**

***http//www qnxworld.main.ru***

***http//www.qnx.org.ru***