**3.1. Стандарты семейства UNIX**

Причиной появления стандартов на операционную систему UNIX стало то, что она была перенесена на многие аппаратные платформы. Ее первые версии работали на аппаратуре PDP, но в 1976 и 1978 годах система была перенесена на Interdata и VAX. С 1977 по 1981 годы оформились две конкурирующие ветви: UNIX AT&T и BSD. Наверное, цели разработки стандартов были разными. Одна из них – узаконить главенство своей версии, а другая – обеспечить переносимость системы и прикладных программ между различными аппаратными платформами. В связи с этим говорят и о мобильности программ. Такие свойства имеют отношение как к исходным текстам программ, так и исполнимым программам.

Дальнейший материал приводится в хронологическом порядке появления стандартов.

**Стандарты языка программирования C**

Этот стандарт не относится непосредственно к UNIX. Но поскольку C является базовым как для этого семейства, так и других ОС, упомянем о стандарте этого языка программирования. Начало ему было положено выходом в 1978 году первой редакции книги Б.Кернигана и Д.Ритчи. Этот стандарт часто называют K&R. Программисты, авторы этого труда, работали над UNIX вместе с Кеном Томпсоном. При этом первый из них предложил название системы, а второй изобрел этот язык программирования. Соответствующий текст доступен в Интернете [45].

Однако промышленный стандарт языка программирования С был выпущен в 1989 году ANSI и имел имя X3. 159 – 1989. Вот что написано про этот стандарт [46]:

"Стандарт был принят для улучшения переносимости написанных на языке Си программ между различными типами ОС. Таким образом, в стандарт, кроме синтаксиса и семантики языка Си, вошли рекомендации по содержанию стандартной библиотеки. О наличии поддержки стандарта ANSI C говорит предопределенное символьное имя \_STDC".

В 1988 году на основе этого стандарта языка программирования была выпущена вторая редакция книги Кернигана и Ритчи о С. Заметим, что фирмы, производящие программные продукты для разработки программ на языке С, могут формировать свой состав библиотек и даже несколько расширять состав других средств языка.

**System V Interface Definition (SVID)**

Другое направление развития стандартов UNIX связано с тем, что не только энтузиасты задумывались о создании "эталонов". Основные разработчики системы с появлением многих "вариантов" решили издавать собственные документы. Так появляются стандарты, выпускаемые USG, организацией, разрабатывающей документацию версий UNIX AT&T с того момента, когда для создания операционной системы была образована эта дочерняя компания. Первый документ появился в 1984 году на основе SVR2. Он имел название SVID (System V Interface Definition). Четырехтомное описание было выпущено после выхода в свет версии SVR4. Эти стандарты дополнялись набором тестовых программ SVVS (System V Verification Suite). Основное назначение этих средств состояло в том, чтобы разработчики имели возможность судить, может ли их система претендовать на имя System V [14].

Отметим, что положение дел со стандартом SVID в чем-то сходно со стандартом языка программирования С. Изданная авторами этого языка программирования книга является одним из эталонов, но не единственным. Выпущенный позже стандарт С является результатом коллективного труда, прошел этап обсуждения широкой общественности и, видимо, может претендовать на ведущую роль в списке стандартов. Так и SVVS является набором тестов, позволяющих судить, достойна ли система соответствовать имени System V, только одной из версий UNIX. При этом не учитывается весь опыт разработки операционных систем от разных производителей.

**Комитеты POSIX**

Работа по оформлению стандартов UNIX началась группой энтузиастов в 1980 году. Была сформулирована цель – формально определить услуги, которые операционные системы обеспечивают приложениям. Такой стандарт программного интерфейса стал основой документа POSIX (Portable Operating System Interface for Computing Environment – переносимый интерфейс операционной системы для вычислительной среды) [14]. Первая рабочая группа POSIX была образована в 1985 году на основе UNIX-ориентированного комитета по стандартизации /usr/group, также называемой UniForum [47]. Название POSIX было предложено родоначальником GNU Ричардом Столмэном.

Ранние версии POSIX определяют множество системных сервисов, необходимых для функционирования прикладных программ, которые описаны в рамках интерфейса, специфицированного для языка С (интерфейс системных вызовов). Заложенные в нем идеи использовались комитетом ANSI (American National Standards Institute) при создании стандарта языка C, упомянутого ранее. Исходный состав функций, закладываемый в первые версии, опирался на UNIX AT&T (версия SVR4 [48]). Но в дальнейшем происходит отрыв спецификаций стандартов POSIX от этой конкретной ОС. Подход к организации системы на основе множества базовых системных функций был применен не только в UNIX (например, WinAPI фирмы Microsoft).

В 1988 году был опубликован стандарт 1003.1 – 1988, определяющий API (Application Programming Interface, программный интерфейс приложений). Через два года был принят новый вариант стандарта IEEE 1003.1 – 1990. В нем были определены общие правила программного интерфейса, как для системных вызовов, так и для библиотечных функций. Далее утверждаются дополнения к нему, определяющие сервисы для систем реального времени, "нитей" POSIX и др. Важным является стандарт POSIX 1003.2 – 1992 – определение командного интерпретатора и утилит.

Имеется перевод [1] этих двух групп документов, которые получили такие названия: POSIX.1 (интерфейс прикладных программ) и POSIX.2 (командный интерпретатор и утилиты – интерфейс пользователя). В упомянутом переводе содержатся три главы: основные понятия, системные услуги и утилиты. Глава "Системные услуги" разделена на несколько частей, каждая из которых группирует сходные по функциям услуги. Например, в одном из разделов "Базовый ввод/вывод" седьмая часть, посвященная операциям с каталогами, описывает три функции ( opendir, readdir и closedir ). Они определяются в четырех пунктах: "Синтаксис", "Описание", "Возвращаемое значение" и "Ошибки".

Для тех, кто знаком с алгоритмическим языком программирования С, приведем пример фрагментов описания. Фактически такое описание дает представление о том, как специфицируется "Интерфейс системных вызовов". В пункте "Синтаксис" про функцию readdir приведены такие строки:

#include <sys/types.h>

#include <dirent.h>

struct dirent \*readdir(DIR \*dirp);

Второй пункт ("Описание") содержит следующий текст:

"Типы и структуры данных, используемые в определениях с каталогами, определяются в файле dirent.h. Внутренний состав каталогов определяется реализацией. При чтении с помощью функции readdir формируется объект типа struct dirent, содержащий в качестве поля символьный массив d\_name, в котором находится завершаемое символом NUL локальное имя файла.

Readdir читает текущий элемент каталога и устанавливает указатель позиции на следующий элемент. Открытый каталог задается указателем dirp. Элемент, содержащий пустые имена, пропускается".

А вот что приводится в пункте "Возвращаемое значение":

" Readdir при успешном завершении возвращает указатель на объект типа struct dirent, содержащий прочитанный элемент каталога. Прочитанный элемент может заноситься в статическую память и перекрывается очередным таким вызовом, примененным к тому же открытому каталогу. Вызов readdir для различных открытых каталогов не перекрывает читаемую информацию. В случае ошибки или достижении конца файла возвращается нулевой указатель".

В пункте "Ошибки стандарта" указано следующее:

" Readdir и closedir обнаруживают ошибку. [EBADF] Dirp не являются указателем на открытый каталог".

Этот пример показывает, как описываются представляемые приложением услуги. Требования к операционной системе (реализации) заключается в том, что она "…должна поддерживать все обязательные служебные программы, функции, заголовочные файлы с обеспечением специфицированного в стандарте поведения. Константа \_POSIX\_VERSION имеет значение 200112L [49]".

В мире компьютерных технологий существует такое словосочетание: "программирование POSIX". Этому можно научиться, используя различные руководства по системному программированию UNIX и операционным системам (например, [5]). Есть отдельная книга с таким названием [3]. Заметим, что в предисловии к этой книге сказано, что она описывает ". . . стандарт втройне . . ", так как она опирается на последнюю версию POSIX 2003 года, в основе которой три стандарта: IEEE Std 1003.1, технический стандарт Open Group и ISO/IEC 9945.

Как же проверить соответствие конкретной системы стандарту POSIX? Формализация такого вопроса не так проста, как кажется на первый взгляд. В современных версиях предлагается 4 вида соответствия (четыре семантических значения слова "соответствие": полное, международное, национальное, расширенное).

В рассматриваемых документах приводятся списки двух видов интерфейсных средств: обязательные (по возможности предполагается его компактность) и факультативные. Последние должны либо обрабатываться предписанным образом, либо возвращать фиксированное значение кода ENOSYS, означающего, что функция не реализована.

Отметим, что набор документов POSIX изменяется уже много лет. Но разработчики новых версий всегда стараются максимально сохранить преемственность с предыдущими версиями, В более свежих редакциях может появиться что-то новое. Например, в документе 2004 года были объединены четыре части [50]:

* Base Definitions volume (XBD) – определение терминов, концепций и интерфейсов, общих для всех томов данного стандарта;
* System Interfaces volume (XSH) – интерфейсы системного уровня и их привязка к языку Си, где описываются обязательные интерфейсы между прикладными программами и операционной системой, в частности – спецификации системных вызовов;
* Shell and Utilities volume (XCU) – определение стандартных интерфейсов командного интерпретатора (т.н. POSIX-shell), а также базовой функциональности Unix-утилит;
* Rationale (Informative) volume (XRAT) – дополнительная, в том числе историческая, информация о стандарте.

Как и первые редакции, документ в своей основной части описывает группы представляемых услуг. Каждый элемент там описан в следующих пунктах: NAME (Имя), SINOPSIS (Синтаксис), DISCRIPTION (Описание), RETURN VALUE (Возвращаемое значение), ERRORS (Ошибки) и в заключении EXAMPLE (Примеры).

Современные версии стандарта определяют требования как к операционной системе, так и к прикладным программам. Приведем пример [51].

Функция readdir() должна возвращать указатель на структуру, относящуюся к очередному элементу каталога. Возвращаются ли элементы каталога с именами "точка" и "точка-точка", стандартом не специфицировано. В этом примере возможно четыре исхода, а требование к прикладной программе состоит в том, что она должна быть рассчитана на любой из них.

И в заключение приведем отрывок из курса лекций Сухомлинова ("ВВЕДЕНИЕ В АНАЛИЗ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ", Сухомлинов В.А. Часть V. Методология и система стандартов POSIX OSE), посвященным области применимости стандартов [52]:

"Область применимости стандартов POSIX OSE (Open System Environment) – обеспечение следующих возможностей (называемых еще свойствами открытости) для разрабатываемых информационных систем:

* Переносимость приложений на уровне исходных текстов (Application Portability at the Source Code Level), т.е. предоставление возможности переноса программ и данных, представленных на исходных текстах языков программирования, с одной платформы на другую.
* Системная интероперабельность (System Interoperability), т.е. поддержка взаимосвязанности между системами.
* Переносимость пользователей (User Portability), т.е. обеспечение возможности для пользователей работать на различных платформах без переобучения.
* Адаптируемость к новым стандартам (Accommodation of Standards), связанным с достижением целей открытости систем.
* Адаптируемость к новым информационным технологиям (Accommodation of new System Technology) на основе универсальности классификационной структуры сервисов и независимости модели от механизмов реализации.
* Масштабируемость прикладных платформ (Application Platform Scalability), отражающая возможность переноса и повторного использования прикладного программного обеспечения применительно к разным типам и конфигурациям прикладных платформ.
* Масштабируемость распределенных систем (Distributed System Scalability), отражающая возможность функционирования прикладного программного обеспечения независимо от развития топологии и ресурсов распределенных систем.
* Прозрачность реализаций (Implementation Transparency), т.е. сокрытие от пользователей за интерфейсами систем особенностей их реализации.
* Системность и точность спецификаций функциональных требований пользователей (User Functional Requirements), что обеспечивает полноту и ясность определения потребностей пользователей, в том числе в определении состава применяемых стандартов."

Это позволяет решать следующие задачи:

* интеграция информационных систем из компонент различных изготовителей;
* эффективность реализаций и разработок, благодаря точности спецификаций и соответствию стандартным решениям, отражающим передовой научно-технический уровень;
* эффективность переноса прикладного программного обеспечения, благодаря использованию стандартизованных интерфейсов и прозрачности механизмов реализации сервисов систем.

Также в стандартах формально определяются такие важные понятия операционных систем: пользователь; файл; процесс; терминал; хост; узел сети; время; языково-культурная среда. Там не приводятся формулировки такого определения, но вводятся применяемые к ним операции и присущие им атрибуты.

Всего в списке стандартов POSIX более трех десятков элементов. Их имена традиционно начинаются буквой "Р", после которой расположено четырехзначное число с дополнительными символами. Существуют также групповые имена стандартов POSIX1, POSIX2 и т.д. Например, POSIX1 связан со стандартами на базовые интерфейсы ОС (Р1003.1х, где вместо х либо пусто, либо символы от a до g; таким образом, в этой группе 7 документов), а POSIX3 – методы тестирования (два документа – Р2003 и Р2003n).

**X/Open, OSF и Open Group**

Основанная в 1984 году рядом компьютерных фирм организация X/Open своей основной задачей ставила согласование и утверждение для разных версий UNIX стандартов общего программного интерфейса и программной среды для приложений. В 1988 году появился документ XPG3 (X/Open Portability Guide3). Он включал POSIX 1003.1 – 1988 и стандарт на графическую систему X Window System (MTI). Этот документ был развит, включая последние идеи UNIX версии BSD и System V. Он получил название XPG4.2 [14].

X/Open объединила свои усилия с Open Software Foundation, создав The Open Group, которая до настоящего времени работает над идеологией открытых систем. С момента создания ей принадлежит торговая марка UNIX. Фирма активно работает (вместе с IEEE, ISO и IEC) над последними стандартами операционных систем.

Заметим, что OSF была образована рядом организаций в ответ на объединение Sun Microsystems и AT&T для разработки операционной системы, претендующей на универсальность. Сегодня организация The Open Group является главным держателем стандартов UNIX. Она размещает на своем сайте много самой разнообразной информации, начиная от истории описания "What is UNIX" ("Что такое UNIX") и заканчивая серией стандартов Single Unix, Unix95, Unix98, Unix03, ISO/IEC 9945:2003, а также UNIX System API Table.

В этот неправительственный консорциум входят около 200 членов. В их числе правительственные организации, учебные заведения и представители бизнеса разных стран. Приведем (в алфавитном порядке) нескольких представителей компьютерного бизнеса членов The Open Group:

|  |  |  |
| --- | --- | --- |
| AT&T | Hewlett-Packard | IBM |
| Intel | NEC Corporation | Oracle |
| Red Hat (R), Inc. | Sun Microsystems | SUSE LINUX Products GmbH |

В заключение этого раздела заметим, что существует еще много менее значимых стандартов на программное обеспечение. Один из них, Linux Standard Base (LSB), создавался Free Standards Group. Но последняя объединилась с Open Source Development Labs (OSDL) и образовала новую организацию The Linux Foundation. Назовем и еще один документ – лицензию BSD (программная лицензия университета Беркли, применяемая для распространения UNIX-подобных операционных систем BSD).

**3.2. Лицензии на программное обеспечение и документацию**

С появлением Linux и подобных ей систем распространяемые свободно программы стали вытеснять коммерческие продукты. Для разных платформ существует много бесплатных программ семейства UNIX/Linux. Большая часть из них разрабатывалась в соответствии со специальной лицензией GPL (General Public License), которая была издана в рамках проекта GNU, начатого в 1984 году Ричардом Столменом (Richard Stollman) [16].

Информацию о Ричарде Столмене как одном из плеяды "выдающихся программистов мира" можно посмотреть в Интернете [53]. Ричард Столмен окончил Гарвардский университет по специальности "физика". Затем поступил на работу в Массачусетский технологический институт, где участвовал в нескольких проектах по разработке программного обеспечения. К примеру, он написал включенный во многие версии UNIX текстовый редактор emacs. С 1984 года он работает над проектом, первоначальной целью которого было создание на основе идей UNIX свободно распространяемой (бесплатной) операционной системы. Для ее разработки были нужны другие программные средства, например, транслятор с одного из языков программирования и редактор текстов. Но они также должны были быть бесплатными, иначе их авторы могут заявить свои права на часть созданной операционной системы.

Мысли Столмена были перенаправлены на создание новых методов разработки программного обеспечения. Для этого была создана лицензия GPL, в рамках которой разрабатываются свободно распространяемые программы. Для развития такого направления основывается FSF (Free Software Foundation), который возглавил Столмен. Его идеи заключаются в том, что программы обязательно должны иметь открытые исходные тексты. Любой программист может воспользоваться фрагментом чужой программы, но открыв исходный текст, созданный им самим. Кроме изменений, связанных с возможностью использовать чужие фрагменты, такой метод разработки программ улучшает и процедуру тестирования программ.

Удачные алгоритмы применяются многими программистами и подвергаются неоднократной и разнообразной проверке. Вообще Столмен сравнивал такой способ разработки программ с обменом кулинарными рецептами. Заметим, что разрабатываемые в соответствии с GPL программы не обязательно должны быть бесплатными. Можно включить программы других авторов как часть своего продукта и продавать последний. Конечно, при этом надо указать всех авторов всех частей проекта.

Что же такое свобода программного обеспечения по Столмену [8]?

1. Разрешается запускать программу и использовать ее по назначению в любых целях.
2. Разрешается изучать устройство программы, то, как она создана. При этом можно и даже необходимо использовать ее свободно предоставляемые исходники.
3. Разрешается копировать программу в любых количествах и распространять бесплатно всем, кому она нужна.
4. Разрешается изменять код программы, изменять ее в соответствии со своими представлениями и распространять как на коммерческой основе, так и на некоммерческой (платно или бесплатно).

Приведем и еще одну интерпретацию четырех пунктов "свободы" для разработчиков программ по Столмену [54]. Разработка свободно распространяемого ПО была очень важным шагом, но еще большей заслугой Р. Столмена следует признать создание "Стандартной Общественной Лицензии GNU" (GNU General Public License, или GPL). На русский язык это название разные авторы переводят по-разному: "Универсальная общественная лицензия", "Обобщенная Публичная Лицензия" и т.п. Но считается, что юридическую силу имеет только английский вариант этой лицензии. Основная идея GPL состоит в том, что пользователь должен обладать следующими четырьмя правами (или четырьмя свободами):

* правом запускать программу для любых целей (свобода 0);
* правом изучать устройство программы и приспосабливать ее к своим потребностям (свобода 1), что предполагает доступ к исходному коду программы;
* правом распространять программу, имея возможность помочь другим (свобода 2);
* правом улучшать программу и публиковать улучшения в пользу всего сообщества (свобода 3), что тоже предполагает доступ к исходному коду программы.

Публичная лицензия первой версии была выпущена в 1989 году. Через пару лет вышла ее вторая версия, а третья была написана в 2005 году, но окончательный вариант был принят в 2007 году. Эти лицензии обозначаются так: GPL vX (где Х может быть 1, 2 или 3). Из-за ограниченности размера книги приводим только название частей второй версии GPL:

1. Определения.
2. Право на копирование и распространение.
3. Изменение программы.
4. Требование предоставления исходного кода.
5. Прекращение действия лицензии при нарушении ее условий.
6. Акты, означающие принятие лицензии.
7. Запрещение дополнительных ограничений при дальнейшем распространении.
8. Внешние ограничения не снимают обязательства выполнять условия лицензии.
9. Возможность географических ограничений.
10. Будущие версии GNU GPL.
11. Запросы на исключения из правил.
12. Отказ от предоставления гарантий.
13. Отказ от ответственности.

Иногда, в противовес правам на интеллектуальную собственность (в том числе и на программы), именуемым сopyright, программы, распространяемые в соответствии с лицензией, разработанной Столменом, связывают с термином copyleft (копилефт-лицензии).

Также сегодня, в противовес чисто коммерческому направлению разработки и распространения программного обеспечения, существует и другое направление – "открытые исходники" (Оpen Source). Его определение сформулировал Брюс Перенс (Bruce Perens) в 1997 году. С изменениями оно было опубликовано на сайте [54]. В Интернете об этом движении много самой разнообразной информации. Дадим только одну ссылку [55], содержащую адреса этой тематики в Рунете.

Отметим, что Оpen Source не эквивалентен GNU или FSF. Яркие последователи каждого из них часто высказывают свое несогласие между собой. Сам же разработчик Linux (Торвальс) старается держаться "поодаль" от перечисленных и других подобных движений. Эти два термина отличаются расстановкой приоритетов. Сторонники open source делают упор на эффективность открытых исходников как метода разработки, модернизации и сопровождении программ. Сторонники free software считают, что именно права на свободное распространение, модификацию и изучение программ является достоинством свободного ПО.

Linux – один из самых ярких представителей программного продукта, реализованного по методу открытых исходников. Но в этой разработке есть и нечто большее. Об этом ярко написал Эрик С. Рэймонд в статье "Базар и Собор" (The Cathedral and the Bazaar) Русский перевод можно найти в [56]. Там в противовес централизованному методу разработки программ предлагается другой метод – параллельный. При его использовании, разрабатывая программу, надо публиковать ее исходный текст с ранних стадий. Тогда создаются условия участия в проекте, например, на уровне обсуждения идей или частичной отладки, многих программистов. Об этом можно прочитать и статью Безрукова [57, 58].

Оpen Source имеет как много сторонников, так и противников. Его сторонники собираются на различные мероприятия, обсуждают свои проблемы в открытой печати и Интернете. Среди противников, что естественно, находим, прежде всего, представителей компьютерного бизнеса. Глава Microsoft неоднократно высказывался об Оpen Source. Например, в интернете есть публикация "Гейтс о бесплатном ПО" [59].

Исходные тексты своих программ публикуют и самые мощные представители компьютерного бизнеса. Это сделали, к примеру, Sun и даже Microsoft. Правда, последнюю фирму вряд ли можно "заподозрить" в приверженности к Оpen Source. Просто они оказались вынужденными передать исходные тексты своих программ, например, операционной системы Windows, под давлением [60].

Заметим, что параллельно с выпуском GPL v2 был разработан и в 1991 году оформлен документ, названный GNU Lesser General Public License (англ. "Стандартная общественная лицензия ограниченного применения GNU", сокращенно – GNU LGPL). Она была основана на GNU Library General Public License (англ. "Стандартная общественная лицензия GNU для библиотек"). Эти лицензии действуют на свободное программное обеспечение и одобрены Фондом свободного программного обеспечения. Их цель – достигнуть компромисса между GPL и простыми разрешительными лицензиями (например, BSD License, MIT License, Mozilla Public License). LGPL была написана в 1991 году, а затем обновлена в 1999 и 2007 годах Ричардом Столлмэном и Эбеном Могленом. На странице "Лицензии открытого ПО" Википедии приведен список из более 50 элементов. Естественно, это создает определенные трудности.

В семейства GNU есть еще одна лицензия. Ее имя FDL, а с описанием можно познакомиться на http://ru.wikipedia.org/wiki/GNU\_Free\_Documentation\_License – "Свободная лицензия GNU на документацию". Может рассматриваться как дополнение к основной лицензии GPL.

Эта копилефт-лицензия первоначально разрабатывалась для пользовательских руководств, учебников и документации, сопровождающей программы для компьютеров. Как и основная лицензия GNU (GPL), предполагает возможность воспроизведениия, распространения и изменения исходных документов (в том числе и в коммерческих целях). При этом обязательно указывать авторов первоисточника. Заметим, что последний может содержать неизменяемые разделы.

**4.1. Основные понятия, связанные с интерфейсом операционных систем**

В области информационных технологий имеется несколько фундаментальных понятий. Одно из них – "интерфейс". Отметим, что оно может трактоваться с различных точек зрения. В предыдущей главе описано понятие "Интерфейс системных вызовов". Если искать такой термин в "Словарях" Yandex, то будет получено более десятка определений термина, большая часть которых дана в сочетаниях с другими терминами, например: "Интерфейс передачи данных", "Программный интерфейс", "Прикладной интерфейс". В словаре "Естественные науки" на ГЛОССАРИЙ.RU дается следующее определение фундаментальному понятию.

Интерфейс в широком смысле – определенная стандартами граница между взаимодействующими независимыми объектами. Интерфейс задает параметры, процедуры и характеристики взаимодействия объектов.

В "Издательском словаре-справочнике" [61] есть такое определение основному термину "интерфейс". Это:

1. Система связей и взаимодействия устройств компьютера.
2. Средства взаимодействия пользователей с операционной системой компьютера, или пользовательской программой. Различают графический интерфейс пользователя (взаимодействие с компьютером организуется с помощью пиктограмм, меню, диалоговых окон и пр.) и интеллектуальный интерфейс (средства взаимодействия пользователя с компьютером на естественном языке пользователя).

Как видим, здесь этот термин имеет два значения. Но мы кратко остановимся на втором – "интерфейс пользователя". На уже упомянутом нами источнике ГЛОССАРИЙ.RU он определяется так: "Интерфейс пользователя – это элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением, в том числе:

* средства отображения информации, отображаемая информация, форматы и коды;
* командные режимы, язык пользователь-интерфейс;
* устройства и технологии ввода данных;
* диалоги, взаимодействие и транзакции между пользователем и компьютером;
* обратная связь с пользователем;
* поддержка принятия решений в конкретной предметной области;
* порядок использования программы и документация на нее".

По мере развития вычислительной техники методы и средства взаимодействия пользователя с операционной системой менялись. Широкое распространение цифровых вычислительных машин привело к режиму общения между человеком и ЭВМ на специальном языке. Сначала, в период пакетной обработки заданий, это реализовалось с применением специальных носителей информации (например, перфокарт, на которые наносились задания для компьютера). Но в дальнейшем, с широким распространением терминалов и клавиатуры, основным стал командный режим работы пользователя, при котором взаимодействие строилось на основе системы встроенных команд. В свободной энциклопедии "Википедия" он определен так.

Интерфейс командной строки (англ. Command line interface, CLI) – разновидность текстового интерфейса (CUI) между человеком и компьютером, в котором инструкции компьютеру даются в основном путем ввода с клавиатуры текстовых строк (команд), в UNIX-системах возможно применение мыши. Также известен под названием "консоль".

Приведем приблизительный фрагмент экрана, который появляется в режиме командной строки (рис. 4.1).



**Рис. 4.1.**

Слева в строке появляется приглашение ( [asplinux@asplinuxlive ~] ), после него можно набрать команду, результаты которой выводятся далее. Приведем пример выполнения команды date в системе Linux (рис. 4.2).



**Рис. 4.2.**

Первые операционные системы фирмы Microsoft для персональных компьютеров IBM PC (они назывались MS DOS) также поддерживали командный режим, схожий с другими системами. Строка, в которой набирались команды, была схожей с приведенными выше. Сегодня командный режим операционных систем обеспечивается эмуляторами cmd.exe (для 32-х разрядного режима) или command.com (для 16-х разрядного режима). В графическом режиме семейства UNIX/Linux командная строка эмулируется программой Терминал (xterm).

Отметим, что для компьютеров с операционной системой MS DOS удачным дополнением реализации такого интерфейса пользователя стала легендарная программа Norton Commander. Она минимизировала действия по набору текста в командной строке, позволяя оперировать, прежде всего, выбором подходящей команды из меню. В этой программе также активно используются функциональные клавиши компьютера. Википедия эту систему описывает следующим образом:

"Norton Commander (NC) – популярный файловый менеджер для DOS, первоначально разработанный американским программистом John Socha (некоторые дополнительные компоненты были полностью или частично написаны другими людьми: Linda Dudinyak – Commander Mail, вьюеры; Peter Bradeen – Commander Mail; Keith Ermel, Brian Yoder – вьюеры). Программа была выпущена компанией Peter Norton Computing (глава – Питер Нортон), которая позже была приобретена корпорацией Symantec".

Приведем пример снимка экрана этого файлового менеджера (рис. 4.3).



**Рис. 4.3.**  Легендарный файловый менеджер Norton Commander

Популярность программы была настолько велика, что появились многочисленные клоны, которые более или менее точно копировали нортоновский интерфейс. К примеру, DOS Navigator, визуально схожий с Norton Commander-ом, предоставлял даже большие возможности. Для операционной системы Microsoft Windows появились Volkov Commander, FAR Manager, Total Commander и другие аналогичные программы. Впоследствии клоны появились и на других операционных системах: BSD, GNU/Linux – Midnight Commander, Krusader.

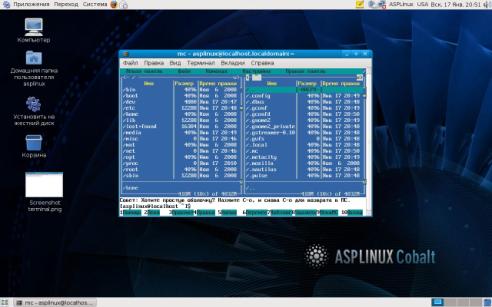
Norton Commander не только спровоцировал целую серию собственных клонов и реплик, но и внес в русский язык пару новых слов – "нортон" и "коммандер" стали в жаргоне пользователей ПК синонимами словосочетания "файловый менеджер".

Введенная программой парадигма работы с файлами (2 одинаковые панели, между которыми происходят операции; большинство команд выполняется по "горячим клавишам") до сих пор применяется в подавляющем большинстве файловых менеджеров.

Norton Commander также стал персонажем серий притч и анекдотов. Первая серия была написана Александром Голубевым, несколько последующих выпускались и дополнялись различными авторами, имена которых постепенно были утеряны, после чего эти рассказики перешли в состояние фольклора.

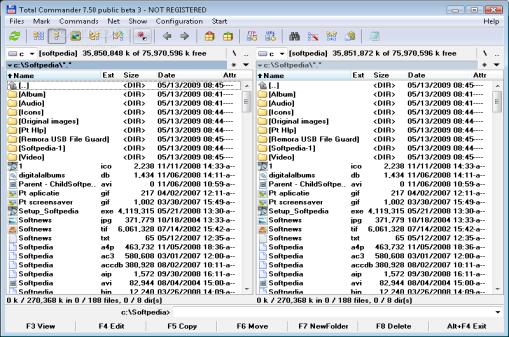
Также имеется музыкальная группа Nord'n'Commander.

В разных версиях Linux используется аналог такой программы, называемой Midnight Commander. Приведем ее вид (рис. 4.4), когда она вызвана в режиме эмуляции командной строки.



**увеличить изображение**  
**Рис. 4.4.**  Программа Midnight Commander, выполненная в Терминале

Но идея разделений окна на две части, в которых представлено содержимое каталогов, осталась привлекательной и при появлении операционной системы только с графическим интерфейсом – Windows 95. Аналоги Norton Commander для этой и последующих версий многочисленны. В интегрированной графической среде UNIX аналогом NC является GNOME Commander. Приводим вид (рис. 4.5) файлового менеджера Total Commander (ранее известного как Windows Commander) операционной системы Windows XP [62].



**увеличить изображение**  
**Рис. 4.5.**  Файловый менеджер Total Commander

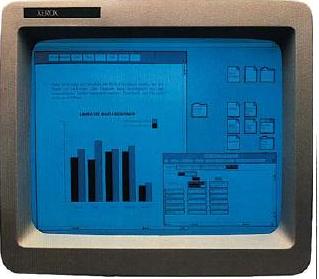
Но сегодня командный режим уходит в прошлое, уступая место другим. Кроме командного, определяются еще два современных вида интерфейса: WIMP и SILK.

WIMP-интерфейс (Window – окно, Image – образ, Menu – меню, Pointer – указатель). Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не с помощью команд, а с помощью графических образов – меню, окон, других элементов. Хотя и в этом интерфейсе подаются команды машине, но это делается "опосредованно", через графические образы. Этот вид интерфейса реализован на двух уровнях технологий: простой графический интерфейс и "чистый" WIMP-интерфейс.

SILK-интерфейс (Speech – речь, Image – образ, Language – язык, Knowledge – знание). Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный "разговор" человека и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса наиболее требователен к аппаратным ресурсам компьютера, и поэтому его применяют в основном для военных целей.

Долгое время возможности компьютеров, их технические характеристики предписывали пользователям работу в командном режиме как в основном. Первые персональные компьютеры также использовали его. Но в последние годы такой режим вытеснен другим – графическим. Он потребовал от компьютера больших ресурсов, но привнес новое – удобство, разнообразный дизайн, многозадачность (правда последняя может быть реализована и в командном режиме). Для обозначения графического режима используют аббревиатуру GUI (Graphics User Interface), что дословно переводят как "графический интерфейс пользователя", но часто при переводе заменяют на "многооконный графический интерфейс".

Первое появление графического интерфейса (рис. 4.6) следует связывать с фирмой XEROX. В ее лаборатории PARC (Palo Alto Reseach Center) в 1973 году создавался компьютер Alto. Последний был оснащен мышью и хорошим монитором. Считают, что этот компьютер обладал GUI, но широкого распространения не получил. Xerox все-таки решается вдохнуть жизнь в экспериментальный Alto, выпустив на рынок его полноценного коммерческого преемника – компьютер Star.

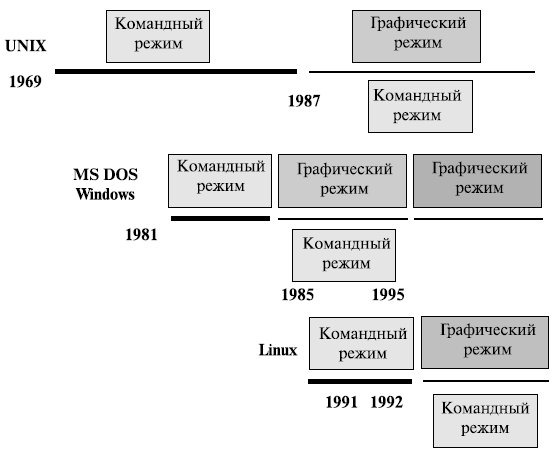


**Рис. 4.6.**  Первый графический интерфейс от фирмы Xerox

Приведем высказывание из статьи Олега Свиргстина [63]: "Alto был первым в мире компьютером, на котором были практически реализованы метафора "рабочего стола" и графический пользовательский интерфейс, прежде существовавшие только в теоретических разработках".

Для операционных систем семейства UNIX, как и многих других, долгое время командный режим работы был основным. Пожалуй, сегодня он используется в основном для администрирования, его потеснил режим графического интерфейса. Фирма Microsoft более 10 лет (с 1981 года) обеспечивала персональным компьютерам IBM PC только командный режим, в то время как у соперников уже в 1984 году был реализован GUI. Правда, эта компания стремилась реализовать последний режим работы, что и было достигнуто в середине 90-х.

Приведем рисунок, иллюстрирующий этапы работы операционных систем Microsoft и UNIX в командном и графическом режимах. Из него видно, что для операционных систем UNIX/Linux до настоящего времени графический режим является надстройкой над командным, а для Windows – командный режим как основной прекратил существование в 1995 году (рис. 4.7).



**Рис. 4.7.**  Командный и графический интерфейс семейства UNIX/Linux и Windows

Отметим, что операционная система MS DOS последние свои годы снабжалась надстройкой, обеспечивающей пользователями GUI. Названия этих графических оболочек были Windows1, Windows2, Windows3.

Из других графических интерфейсов назовем OPENSTEP, реализованный на компьютерах фирмы NeXT. Его создавал Стивен Джобс, основатель фирмы Apple, в период, когда он покинул ее и пытался завоевать мир новой разработкой. Этот интерфейс в дальнейшем был перенесен и на другие компьютерные платформы (рис. 4.8).

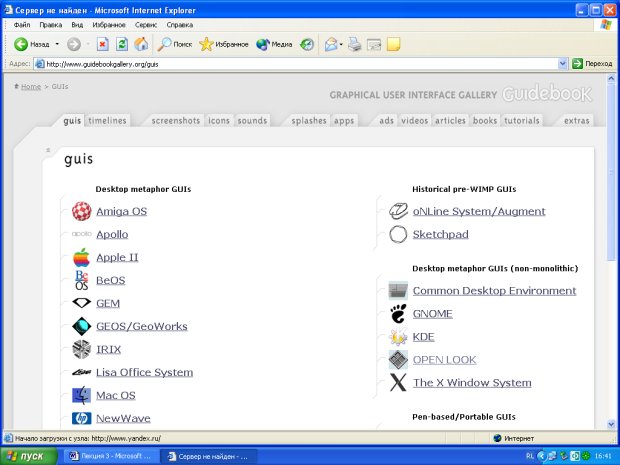
Обратите внимание на его отличия от того, что в это время предлагала фирма Microsoft со своей Windows95 (пример рабочего стола приведен в главе 2, в части, посвященной операционным системам этой фирмы).

По адресу http://www.guidebookgallery.org/guis можно познакомиться с "галереей" графических интерфейсов пользователей на разных компьютерных платформах. Приведем два снимка экрана, на которых представлен перечень всех элементов галереи (рис. 4.9).



**Рис. 4.8.**  Графический интерфейс OPENSTEP Jan 1997 платформы

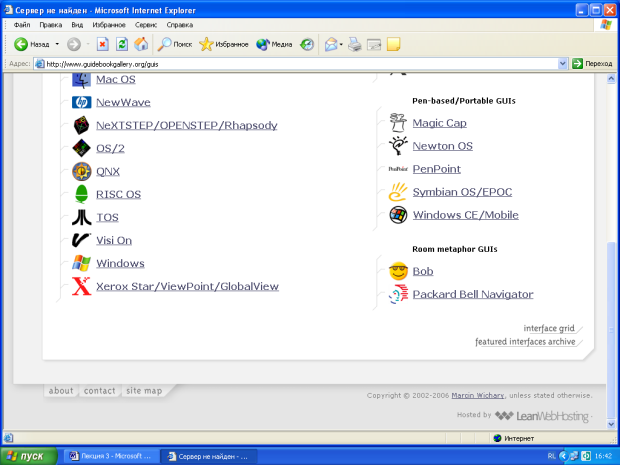
Отдельно остановимся на списке из 5 элементов Desktop metaphor GUI (non monolitic). Они содержат ссылки на описание систем, обеспечивающих графический интерфейс пользователя UNIX. Здесь коротко упомянем только два, остальные подробно рассматриваются дальше.



**Рис. 4.9а.**  Галерея графических интерфейсов на разной аппаратуре (часть 1)

На этих рисунках обратите внимание на более чем десяток типов рабочих столов (от Amiga OS до Xerox Star/View Point/Global View). Хотя рабочий стол Windows занимает одно из мест, но на сегодняшний день многие производители приняли его стандарты. В этом же ряду упомянуты системы, активно влиявшие на развитие операционных систем, но сегодня уже не существующие. Среди них:

* OS/2 от IBM, долгое время являвшаяся конкурентом Windows;
* BeOS, созданная корпорацией Ве Inc и обладавшая в момент своего выпуска многими пионерскими новинками. Это работа на 64-разрядной аппаратуре, удобный интерфейс пользователя и многое другое.



**Рис. 4.9б.**  Галерея графических интерфейсов на разной аппаратуре (часть 2)

OPEN LOOK представляет собой спецификацию графического интерфейса пользователя рабочих станций UNIX. Была создана в конце 1980-х годов Sun Microsystems и AT&T при участии Xerox. Эта спецификация была основной для операционной системы на ранних стадиях реализации графического интерфейса. Впоследствии утеряла свое значение в связи с появлением графического интерфейса Motif от OSF. Common Desktop Environment (CDE) – среда рабочего стола, основанная на системе Motif. Она была создана The Open Group вместе с рядом фирм: Hewlett-Packard, IBM, Novell. Некоторое время она была промышленным стандартом для UNIX-систем.

Режим GUI используется в разных операционных системах. Многие его разработчики пытались найти свой, наиболее привлекательный для пользователей "стиль". Со временем они вынуждены были оглядываться на то, что делают другие фирмы, или даже объединяться для стандартизации составляющих графического интерфейса. Современное представление о графическом интерфейсе, на наш взгляд, объединяет все лучшее от разных производителей. Думается, поиски в этом направлении продолжатся и далее, хотя часто говорят о том, что с первых шагов становления графического интерфейса ничего принципиально не изменилось – все его основные элементы остаются прежними (рабочий стол, меню, иконки).

Пожалуй, следует отметить и еще одну тенденцию: последние варианты реализации графического интерфейса построены с "оглядкой" на то, что реализовано в операционных системах Windows. Это объясняется большой их долей (около 90%) на рынке персональных компьютеров.

Как уже говорилось выше, для систем UNIX долгое время – с начала 70-х годов и, пожалуй, до конца 80-х – единственным режимом был командный режим работы. Сегодня он уступил свое место графическому. В семействе операционных систем UNIX (напомним, работающих на разных аппаратных платформах) графический интерфейс пользователя поддерживается системой X Window System. Основной сайт с информацией о ней имеет адрес http://www.x.org. Последняя версия, представленная там, имеет имя X11R7.5.

**4.2. Графический интерфейс пользователя в семействе UNIX/Linux**

**4.2.1. К истории X Window system**

X Window system появилась в результате объединения усилий двух исследовательских групп MIT: группа, ответственная за сетевую программу (проект "Афина" – Project Athena) и Лаборатория информатики (Laboratory for Computer Science). До десятой версии X Window этот проект реализовали три программиста: Роберт Шейфлер (Robert Sheifler), Джим Геттис (Jim Gettys) и Рон Ньюмен (Ron Newman). Двое из них работали в MIT, а третий в DEC [16, 17].

Первоначально разрабатываемая в MIT (Массачусетском технологическом институте) система X Window стала распространяться свободно. Было создано несколько версий, и последняя из них, успешно используемая до настоящего времени, имела номер, присвоенный при создании и равный 11. Чаще других применяется версия 11, имеющая номер реализации 6. Поэтому на компьютерах с установленной системой Window часто встречаются каталоги, в названии которых есть символы X11R6 или X11.

В дальнейшем разработкой средств, обеспечивающих GUI для операционной системы UNIX, в режиме жесткой конкуренции занимались многие крупные компьютерные фирмы. При этом некоторые из них объединялись для совместных действий и даже создания стандартов.

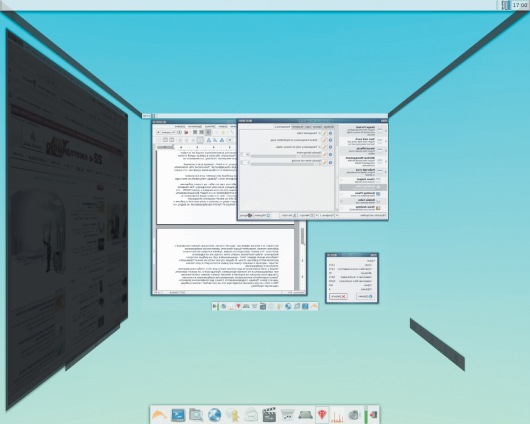
В 1987 году ряд фирм решили создать единый стандарт оконного интерфейса для UNIX и для этого основали X Consortium ("Консорциум X"). В нем приняли участие IBM, DEC, HP и другие компании. Этот проект возник в противовес объединению AT&T и Sun. С 1997 X Consortium преобразовалась в "Открытую группу Х" (X for the Open Group) [16]. Информацию о деятельности этой организации (ее современное имя X.Org Foundation) можно получить в Интернете [64].

В статье [65] приведены примеры четырех исторически появлявшихся видов графического интерфейса XWindows (OpenLook, Motif, KDE и трехмерный графический интерфейс). Там о них говорится следующее:

"Эволюция пользовательских интерфейсов, построенных на основе X Window, наглядно доказывает преимущество выбранного разработчиками системы подхода. Свобода в определении политик и простота использования механизмов позволили X Window пройти эволюционным путем от внешне примитивного вида OpenLook к де-факто стандартному экранному представлению примитивов пользовательского интерфейса Motif, к гибко настраиваемому современному виду KDE и, наконец, к прообразам трехмерного интерфейса".

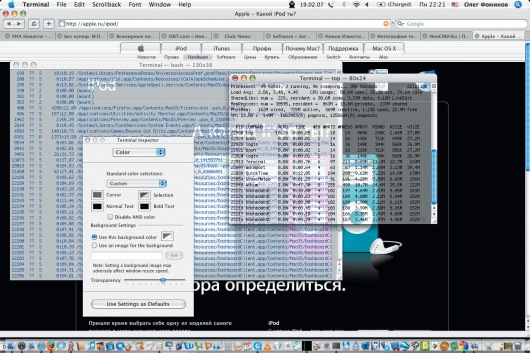
Заметим, что трехмерный графический интерфейс появился сравнительно недавно. Но самые последние версии популярных операционных систем реализуют его. Это относится к разновидностям Linux, Mac OS и версий Microsoft начиная с Vista [66].

Не претендуя на полноту охвата вопроса, отметим, что трехмерные рабочие столы могут быть построены на разных эффектах. Одним из первых была реализована метафора рабочей комнаты со шкафчиками, ящичками и тому подобными элементами, которые открывались, выдвигались и т.д. Другой подход, видимо, основан на объемной фигуре, которую можно поворачивать и изменять в размерах. И еще одна идея 3D Desktop предполагает использование прозрачных окон, за которыми можно увидеть информацию расположенных за ними окон. С одной из наиболее распространенных версий Linux Mandriva сегодня поставляется Metisse (рис. 4.10). Последний основан на эффекте перспективы.



**Рис. 4.10.**  Пример трехмерного графического интерфейса Metisse дистрибутива Linux Mandriva

Для операционных систем Mac OS 3D интерфейс реализуется в Aqua. Он основан на эффекте прозрачности (рис. 4.11).



**Рис. 4.11.**  Пример трехмерного графического интерфейса Aqua Mac OS

Еще один пример трехмерного интерфейса от Mac OS, при котором каждый пользователь работает на своей грани куба (рис. 4.12).



**Рис. 4.12.**  Пример трехмерного графического интерфейса Aqua Mac OS

Трехмерный интерфейс операционной системы Windows Vista получил название Aero. Он построен на эффекте 3D Flip.



**Рис. 4.13.**  Пример трехмерного графического интерфейса Aero Windows Vista

**4.2.2. Основные понятия системы X Window**

X Window system (или просто X Window, а теперь часто и Х) – графическая среда пользователя, поддерживающая одновременное выполнение многих программ в сети. В основе X Window – библиотека графических программ, используемых для создания GUI.

**ЗАМЕЧАНИЕ**. Отметим, что термину X Window дают разное определение. Поиск в Интернете позволяет получить их более десятка.

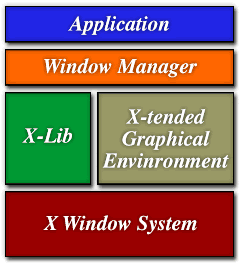
Достоинством системы X Window является ее мобильность (она не связана с конкретной операционной системой и не рассчитана на специфическое техническое обеспечение). Работа Х-системы основана на специфической модели клиент/сервер.

В традиционной модели "клиент-сервер" с пользователем взаимодействует клиентская часть. В системе же X Window с пользователем взаимодействует X-сервер. Он отвечает за вывод информации на экран пользователя и получение им команд. Такой сервер как бы "владеет" аппаратурой пользователя (называемой X-терминал) и представляет этот ресурс программам – клиентам. Именно они формируют изображение, выводимое на экране. При инициализации X Window system первым шагом будет загрузка X-сервера. Об этом можно узнать по появлению на сером экране в центре указателя мыши в виде крестика.

Но для окончательного вывода на экран сформированного программой клиентом изображения одного X-сервера мало. Для этого еще необходим менеджер окон.

Таким образом, система X Window представляет собой комплекс взаимодействующих компонент. Интересно, что существует несколько вариантов каждого элемента, из которых "собирается" конкретный экземпляр системы.

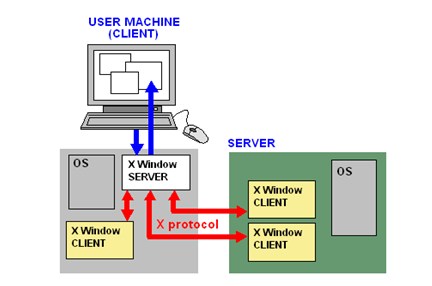
Следуя J.Bait [6], приведем схематическое изображение архитектуры графической системы (рис. 4.14).



**Рис. 4.14.**  Архитектура X Window

Это упрощенная схема. Обязательными являются еще такие два компонента. Взаимодействие между графическими библиотеками и X-сервером реализуется по протоколу TCP/IP. Также важным элементом рассматриваемой системы являются шрифты, поэтому в системе можно выделить и еще один элемент – сервер шрифтов.

Приведем схему, взятую с сайта http://www.answers.com/topic/x-window-system (рис. 4.15).



**Рис. 4.15.**  X Windows сервер выполняется на клиенте

Из этой схемы видно, что программы, выполняющие роль X Window SERVER и X Window CLIENT, могут располагаться как на одном компьютере, так и на разных. Каждая из них может работать под управлением своей операционной системы. Взаимодействие между Х-клиентом и Х-сервером реализуется по специальному протоколу (X protocol). В этой схеме не обозначены драйверы устройств, обеспечивающие работу конкретной аппаратуры и вместе с Х-сервером образующие Х-терминал. За вывод информации отвечает такой компонент, как менеджер окон, обеспечивающий, по инициативе прикладных программ, вывод на экран множества перекрывающихся окон, расположенных в нужном месте экрана и имеющих требуемый размер. Этот компонент изображен на USER MACHINE (CLIENT). Отметим, что общей идеологии X Window system не противоречит ситуация, при которой все компоненты располагаются на одном компьютере, что реализуется, к примеру, в Linux.

Таким образом, система X Window представляет комплекс взаимодействующих элементов, каждый из которых, в принципе, может быть заменен новым компонентом. Все это делает систему достаточно гибкой и легко модифицируемой.

Обратим ваше внимание, что графический режим в операционных системах семейства UNIX/Linux не является обязательным. Он вызывается из командной строки. Заметим, что таким же образом запускалась, например, графическая многооконная оболочка Windows3.X фирмы Microsoft. Из-за сложности процедуры запуска графический интерфейс активизируется целым набором действий. Для систем UNIX в таких случаях предусматривают создание специальных скриптов (сценариев). Долгое время традиционное название файла запуска было startx, а файл конфигурации параметров графического режима носил название XF86Config. Но сейчас это не является обязательным для всех систем.

**4.2.3. X Window в Linux**

Операционная система Linux в последние годы отходит от командного режима как основного и использует графический режим для разных действий: от работы пользователя с прикладными программами до настройки системы администратором. На первоначальном этапе своего развития Linux ориентировался на скромные ресурсы компьютера и, как следствие, командный режим как основной. Но современные версии этой операционной системы для реализации графического режима требуют больших ресурсов компьютера.

Долгое время в Linux использовалась версия X Window, ориентированная на IBM PC и названная XFree86. Она основывалась на стандарте X11R6, но имела ограничения на используемое оборудование. Как и многое в семействе UNIX/Linux, XFree86 постоянно развивается усилиями многих программистов в соответствии с принятыми стандартами. Последняя ее версия имеет номер 4.8.0 (декабрь 2008 года, http://xfree86.org/releases/rel480.html). Для этого графического режима написано много самого разнообразного программного обеспечения. Большая его часть распространяется свободно и бесплатно, но ничем не уступает своим коммерческим аналогам. Это – офисные и графические программы, системы для управления предприятием и средства разработки.

Но с 1999 года параллельно с XFree86 возникает XOrg, основанная The Open Group. Первое время она не была популярной и использовала основные технические достижения XFree86. Но в последние годы ситуация изменилась. В начале 2004 года представители X.Org и freedesktop.org основали фонд X.Org Foundation. The Open Group передала ему управление доменным именем x.org. Это стало коренным изменением в разработке X. В то время как распорядители X с 1988 года (включая предыдущую X.Org) были организациями поставщиков, X.Org Foundation был основан самими разработчиками программного обеспечения, и в нем применялась открытая модель разработки, опирающаяся на вклад извне.

Графический интерфейс семейства UNIX/Linux похож на интерфейс других систем, но имеет отличия. Он поддерживает метафору рабочего стола. Но в отличие от некоторых систем имеет нескольких рабочих столов, которые иногда называются еще и "рабочие места". Их количество можно изменять. Хотя графический UNIX зародился раньше, чем у других операционных систем, сейчас работа с использованием GUI аналогична у Linux и Windows. Пользователь работает с приложением в окне, имеющем прямоугольную форму. Последний содержит стандартные элементы – строка заголовка, главное меню, панели инструментов и т.д.

В X Windows управление окнами приложений, их элементами выполняет компонент, называемый "менеджер окон" (иногда используют название "оконные менеджеры" или "диспетчер окон"). Может быть задействовано несколько диспетчеров окон.

Но сегодня пользователи редко выбирают менеджеры окон. Им представляются интегрированные графические среды. Две наиболее распространенные из них – KDE и GNOME – будут коротко рассмотрены далее. Но сначала приведем список инструментов пользователя. На странице Википедии "Менеджер окон X Window System" приводятся такие списки. Интерфейс пользователя в UNIX-подобных системах:

1. среды рабочего стола: CDE, EDE, etoile, GNOME, JDS, KDE, LXDE, Mezzo, OpenWindows, ROX, Xfce, Xpde;
2. оконные менеджеры: AfterStep, Awesome, Blackbox, CTWM, dwm, Enlightenment, Fluxbox, FVWM, IceWM, JWM, Openbox, Sawfish, twm, Window Maker, wmii;
3. командные оболочки: ash, Bash, BusyBox, csh, dash, es shell, fish, ksh, psh, rc, rsh, Sash, Scsh, sh, tcsh, Thompson shell, zsh и прочие.

Приводятся три категории: среды рабочего стола, оконные менеджеры и командные оболочки. Последние обеспечивают режим командной строки. Как видим, их много. Название первой образовано от английского shell (оболочка). В разных вариантах Linux распространена оболочка, имя которой Bash образовано от Born again shell (разработана Born).

А теперь, как было сказано ранее, приведем короткую информацию о двух интегрированных графических средах KDE и GNOME.

**4.2.4. Интегрированная графическая среда KDE**

Часто графическую среду KDE называют наиболее распространенной. Проект был основан в октябре 1996 года студентом Маттиасом Эттрихом, а в июле 1998 года выпущена версия 1.0. Сокращение образовано от K Desktop Environment. Она строится на основе инструментария разработки пользовательского интерфейса с именем Qt. Интересной особенностью последнего является свойство кроссплатформенности. Хотя эта среда разрабатывается для UNIX-подобных систем, но возможен ее запуск и на других платформах, например, с использованием cygwin под Microsoft Windows.

KDE включает в себя набор тесно взаимосвязанных программ пользователя. В его рамках разрабатывается полнофункциональный офисный пакет KOffice, а также интегрированная среда разработки KDevelop.

Основной адрес в Интернете команды KDE – http://www.kde.org, а в России – http://kde.ru/. В 2010 году начат выпуск версии 4.0, содержащей следующие основные нововведения:

* переход на четвертую версию библиотеки элементов интерфейса Qt;
* новый стиль оформления – Oxygen;
* новый мультимедийный интерфейс API – Phonon;
* объединение Superkaramba, рабочего стола и панели Kicker в одно приложение – Plasma.

Эта версия обеспечивает новые технологии не только для UNIX, но и для Microsoft Windows и Mac OS X. Узнать компьютер, на котором работает KDE, можно по его талисману – дракончику Konqi (рис. 4.16). Обратите внимание, что на изображении Konqi можно увидеть другой символ, который часто появляется при работе в среде KDE.



**Рис. 4.16.**  Талисман KDE

Еще отметим, что по адресу http://www.kde.ru/wiki/HomePage расположены страницы русского проекта локализации KDE, где содержимое создается, изменяется, обсуждается и поддерживается пользователями, разработчиками и всеми остальными, кто как-либо причастен к этому проекту.

**4.2.5. Интегрированная графическая среда GNOME**

Название GNOME является акронимом от английского GNU Network Object Model Environment ("сетевая объектная среда GNU"). На русскоязычном сайте [67], посвященном этой интегрированной среде, дается такой ответ на вопрос "Что такое GNOME?": в рамках проекта GNOME создаются две вещи – рабочая среда GNOME, простая в использовании и привлекательная на вид среда рабочего стола; а также платформа разработки GNOME – расширяемая среда для создания приложений, тесно интегрируемых с рабочим столом.

Основной сайт проекта GNOME располагается по адресу http://www.gnome.org. Его история начинается с 1997 года и связана с именами Мигеля де Иказа и Федерико Мена. Основной целью было создать полностью свободную рабочую среду для операционной системы GNU/Linux [68], поскольку основной инструмент разработки Qt – другой интегрированной среды KDE – не был лицензирован на условиях GNU GPL. Отметим, что эти проблемы были ликвидированы в версии Qt 2.2 в 2000 году.

Среда рабочего стола GNOME была построена на основе GTK+, созданной при разработке мощного графического пакета GIMP. Кроме того, используется еще много различных технологий и библиотек. Описываемая интегрированная среда может быть запущена на большинстве UNIX-систем, адаптирована для работы под управление Solaris, а также через специальный порт может быть запущена под Windows.

Среди других особенностей интегрированной графической среды отметим java-апплеты – набор приложений, встраиваемых в панель рабочего стола (GNOME Panel) для выполнения различных функций (например, с именем "Часы" или "Расчистка рабочего стола"). Логотипом системы является следующее изображение (пятка Гнома).



**Рис. 4.17.**  Логотип GNOME

За локализацию среды GNOME отвечает проект перевода GNOME [3] (англ. GNOME Translation Project). Перевод пользовательского интерфейса и документации производится с помощью инструментария gettext.

Статистика [7] для GNOME 2.30:

* на 32 языков переведено более 90 % строк пользовательского интерфейса;
* еще на 33 языка переведено от 50 % до 90 % строк;
* на русский язык переведено 99 % строк пользовательского интерфейса и 46 % строк документации.

Последняя версия 2010 года имеет номер 2.30.

### Win32 API

Уже отмечалось, что в ОС Windows между приложением и совокупностью системных вызовов (системных сервисов в терминологии Microsoft) расположен дополнительный абстрактный слой - программный интерфейс Win32 API. За счет этого Win32-приложение может работать практически во всех версиях Windows (см. рис. 1.5), несмотря на то, что сами системные вызовы в различных версиях системы различны и не документированы.

Исчерпывающая информация по программному интерфейсу Win32 API содержится в справочной документации на Win32 API. Эту документацию можно просмотреть на сайте http://msdn.microsoft.com или на компакт-дисках MSDN (Microsoft Developer Network Library). MSDN является программой технической поддержки разработчиков.

Win32 API предоставляет всеобъемлющий интерфейс, позволяющий выполнить каждое действие несколькими способами и покрывающий все области, с которыми должна работать операционная система. Естественно, что этот интерфейс содержит вызовы для создания и управления процессов и потоков, управления файловым вводом-выводом, операций с окнами и графикой, безопасностью и т.д.

Если заглянуть в раздел MSDN \Platform SDK\ Win32\ Overview of the Win32 API, то можно увидеть, что Win32 API подразделяются на следующие группы.

* Base Services - базовые сервисы отвечают за обеспечение доступа к ресурсам компьютера и интерфейс для работы с памятью, файлами, устройствами, процессами и потоками.
* Common Control Library - библиотека общих элементов управления для разработки оконных интерфейсов.
* Graphics Device Interface - вывод графики на дисплей и другие устройства.
* Network Services - сетевые сервисы.
* User Interface - интерфейс пользователя.
* Windows Shell - функции для работы с оболочкой.
* Windows System Information - информация о конфигурации системы Windows.

В рамках данного курса нам потребуются главным образом функции, относящиеся к первому пункту списка и описанные в MSDN разделе \Platform SDK\Base Services, а также функции, описанные в разделе \Platform SDK\ Security.

### Компилятор Visual C++ и среда программирования для Windows

Предполагается, что читатель знаком с типичными приложениями Windows, такими, как Проводник (Windows Explorer), ориентируется в файловой системе компьютера и в состоянии найти любой файл, записанный на жестком диске.

Компилятор Windows Visual C++ удобно объединять со справочной системой MSDN Library, которая при этом вызывается через пункт меню "Справка" ("Help") в интегрированной среде Microsoft Visual C++, а также может использоваться автономно. Запуск установленной графической оболочки Microsoft Visual C++ осуществляется стандартными средствами системы.

Чтобы из текста программы на языке высокого уровня (файл с расширением "c" или "cpp") получить исполняемую программу в машинных кодах (файл с расширением "exe"), необходимо в графической оболочке Microsoft Visual Studio C++ создать *рабочий проект*, который представляет собой совокупность служебных файлов, необходимых для дальнейшей работы.

Программные примеры, иллюстрирующие данный курс, представляют собой Windows-приложения с текстовым интерфейсом (консольные). Разработка приложений с дружественным графическим интерфейсом сама по себе достаточно сложна и должна изучаться в рамках специальных учебных курсов. Поэтому, формируя проект, далее в графической оболочке Visual Studio в диалоговом окне "New" нужно выбрать Win32 Console Application в качестве типа приложения, а также дать проекту имя, указать каталог расположения файлов проекта и нажать кнопку "OK".

Затем при помощи пунктов меню и всплывающих окон графической оболочки нужно включить в проект файлы, содержащие программу, или ввести программу с клавиатуры с последующим ее сохранением в одном из файлов проекта. Через пункт меню "Build" можно выполнить компиляцию программы, создать исполняемый модуль, запустить программу на счет и, при необходимости, выполнить ее отладку. Существует большое количество разнообразных руководств по использованию Microsoft Visual Studio C++.

#### Прогон программы "Hello, world"

В качестве самостоятельного упражнения рекомендуется реализовать простейшую программу в интегрированной среде компилятора Visual C++, например, хрестоматийную программу "Hello, world", и ознакомиться со средствами разработки, отладки и контекстной помощи.

### Типы данных, используемые в Win32-приложениях

Мобильность программ и их независимость от конкретной платформы во многом обеспечивается введением новых стандартных типов данных - определенных на основе простых типов языка программирования Си. Имена стандартных типов данных состоят из символов верхнего регистра, для них не применяется оператор "\*".

Полный перечень используемых данных можно увидеть в разделе \Plarform SDK \Win32 API\ Reference \ Data Types. Ниже приведен список наиболее распространенных типов: симовольных, целых, булевских, указателей и описателей (handles). Символьные, целые и булевские типы соответствуют аналогичным типам большинства диалектов языка Си. Имена типов-указателей содержат префикс "P" или "LP". Описатели имеют отношение к ресурсам, загруженным в память.

Наиболее часто используются следующие типы данных:

* BOOL - булевская переменная (TRUE или FALSE );
* CHAR - 8- разрядный символ;
* DWORD - 32-разрядное беззнаковое целое;
* HANDLE - описатель объекта;
* INT - 32-разрядное целое;
* LPSTR - указатель на строку 8-разрядных символов, заканчивающуюся нулем;
* LPTSTR - LPWSTR в случае Unicode или LPSTR в случае ANSI;
* LPVOID - указатель на любой тип;
* LPWSTR - указатель на строку 16-разрядных (Unicode) символов, заканчивающуюся нулем;
* TCHAR - WCHAR, в случае Unicode или CHAR в случае ANSI;
* WCHAR - 16-разрядный Unicode символ;
* WORD - 16-разрядное беззнаковое целое.

Остальные типы данных будут изучаться по мере необходимости. Некоторые Win32 приложения могут быть выполнены в среде более ранних версий ОС Windows, в том числе и 16-разрядных. Вследствие этого имена некоторых типов отражают систему адресации ОС MS-DOS, например, LP (long pointer) означает "длинный" указатель, а на самом деле - это обычный указатель.

В качестве самостоятельного упражнения рекомендуется ознакомиться с данными различных типов в справочной системе MSDN.

### Unicode

В ОС Windows в качестве внутреннего формата для хранения и обработки текстовых строк используется Unicode. В Unicode каждый символ представляется 16-битным (двухбайтовым) кодом, что позволяет поддерживать разные языки и системы письменности (такие, как китайские и японские иероглифы).

Стандарт Unicode поддерживается консорциумом, в который входят такие компании, как Apple, Compaq, Hewlett-Packard, IBM, Microsoft и многие другие; подробная информация об этом имеется на сайте www.unicode.org. В справочнике MSDN соответствующие сведения хранятся в разделе \Visual Studio documentation\ Visual C++ Programmer's Guide \ Adding Program Functionality \ Overviews \ Unicode Programming.

Для совместимости со стандартами языков программирования и предыдущими версиями Windows в системе наряду с 16-битными (двухбайтовыми) символами активно используются и 8-битные (однобайтовые) ANSI символы. Так, многие Win32-функции , принимающие строковые параметры, существуют в двух версиях: для Unicode и для ANSI. Обычно при вызове ANSI-версии Win32-функции входные строковые параметры перед обработкой системой преобразуются в Unicode. В связи с этим перед разработчиками стоит задача написания приложений, способных работать с обеими кодировками.

#### Разработка приложений с использованием Unicode

Чтобы снизить зависимость приложения от используемой кодировки, целесообразно в программе определить два макроса -UNICODE и \_UNICODE. Также рекомендуется использовать новый набор данных и функций, обрабатывающих строки и описанных в стандартных заголовочных файлах.

Обычно, имена Unicode данных и функций содержат префикс "wc" (от wide character), "w". Например, WCHAR - Unicode символ,wcscmp - функция сравнения Unicode строк. Можно также поставить префикс "L" перед текстовой строкой, например, L"Текстовая строка" - строка в формате Unicode.

Чтобы реализовать возможности компиляции двойного назначения, нужно включить в состав программы заголовочный файл tchar.h. Он состоит из макросов, которые ссылаются на Unicode данные и функции, если в программе определен макросUNICODE, и на ANSI - в противном случае. Так, для объявления символьного массива универсального назначения применяется тип TCHAR, который транслируется в WCHAR, если UNICODE определен, и в CHAR, если не определен. Аналогичным образом макросы с префиксом "l" переопределяют строковые функции (lstrlen вместо strlen и т.д.), а для определения символьных и строковых литералов применяется макрос \_TEXT (или просто \_T). Более подробно этот материал описан в [4].

#### Прогон программы вывода строки в формате Unicode

В качестве упражнения рекомендуется реализовать программу вывода строки "Hello, world".

#define UNICODE

#ifdef UNICODE

#define \_UNICODE

#endif

#include <windows.h>

#include <tchar.h>

#include <stdio.h>

void main() {

PTCHAR TextString = \_T("Hello, world");

\_tprintf(\_T("String - %s\n"), TextString);

}

Необходимо убедиться, что программа одинаково работает в случае применения и отключения Unicode.

#### Прогон программы записи в файл в Unicode и обычном формате

Приведенная программа может вывести строку "Hello, world" в файл MyFile.txt в обычном формате и в формате Unicode.

#define UNICODE

#ifdef UNICODE

#define \_UNICODE

#endif

#include <windows.h>

#include <tchar.h>

#include <stdio.h>

void main() {

HANDLE hFile;

PTCHAR FileName = \_T("MyFile.txt");

PTCHAR TextString = \_T("Hello, world.");

DWORD iWrite, StringLength = lstrlen(TextString);

\_tprintf(\_T("There are %ld symbols in text string %s\n"), StringLength, TextString);

hFile = CreateFile(FileName, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL | 0, NULL);

iWrite = StringLength;

#ifdef UNICODE

iWrite = 2\*StringLength;

#endif

WriteFile(hFile, TextString, iWrite, &iWrite, NULL);

printf("%d bytes are written to file\n", iWrite);

CloseHandle(hFile);

}

Рекомендуется оба варианта получившегося файла просмотреть с помощью блокнота Nоtepad.

### Обработка ошибок

Профессиональная разработка программных приложений предполагает включение кода для корректного восстановления после потенциальных ошибок. В случае возникновения ошибки система может справиться с ней самостоятельно, но в экстремальных ситуациях может потребоваться вмешательство пользователя.

Вызываемая Win32-функция может возвратить значение, свидетельствующее об ошибке (например, NULL для функций типа HANDLE или ноль для функций типа BOOL). В таких случаях можно попытаться выявить тип ошибки при помощи функции GetLastError - она возвращает код последней ошибки, который хранится в локальной памяти потока, вызвавшего ошибку. Коды ошибок (а их более 10 тысяч), представляющие собой 32-битные числа, находятся в заголовочном файле WinError.h.

Если приложение содержит функции, к которым обращаются другие программы, то желательно, чтобы эти функции синтезировали код ошибки в случае возникновения ошибочных ситуаций, то есть вели себя подобно функциям Win32 API. Это можно сделать при помощи функции SetLastError.

Для преобразования кода ошибки в ее содержательное описание предназначена Win32-функция FormatMessage. Получить описание ошибки по ее коду можно также с помощью утилиты errlook.exe, поставляемой в составе Visual Studio. Аналогичная информация содержится в справочной системе MSDN.

В тех случаях, когда об ошибке необходимо оповестить пользователя, можно использовать звуковой сигнал (функция MessageBeep). Для обработки ошибок также активно применяется структурная обработка исключений (Structured Exception Handling, см., [4], [10]).

#### Прогон программы, синтезирующей информацию об ошибке, которая имитирует отсутствие нужного файла

#include <windows.h>

#include <stdio.h>

void GetError() {

DWORD dw = GetLastError();

printf("GetLastError returned %u\n", dw);

}

void SetError() {

DWORD dw = ERROR\_FILE\_NOT\_FOUND;

SetLastError (dw);

}

void main() {

SetError();

GetError();

}

Рекомендуется реализовать данную программу и сверить номер выдаваемой ошибки с соответствующим перечнем в MSDN или файле заголовка WinError.h.

### Инструментальные средства изучения системы

В рамках данного курса изучение внутреннего строения ОС Windows осуществляется, главным образом, путем разработки небольших программ, ориентированных на взаимодействие с различными компонентами системы. Вместе с тем, многие детали реализации рассматриваемой операционной системы можно выявить с помощью разнообразных инструментальных средств, некоторые (depends и errlook) уже упоминались в предыдущих разделах. Ниже приведен приблизительный перечень полезных ресурсов и утилит.

#### Штатные и встроенные средства

Большое количество полезных инструментов поставляется вместе с системой. Это, во-первых, штатные утилиты, такие, как диспетчер задач, редактор реестра, разнообразные средства настройки и администрирования, информативные панели. Очень много полезной информации можно получить путем интерпретации показаний многочисленных счетчиков производительности, предназначенных для мониторинга системы. Счетчики производительности, а их более сотни для различных объектов, доступны через оснастку "Производительность" административной панели управления, а также через API системы.

Большое количество полезных утилит входит в состав Windows Support Tools, для их установки надо запустить Setup из папки \Support \ Tools в дистрибутиве системы.

#### Утилиты и программные средства, входящие в состав Platform SDK

В состав Microsoft Platform SDK входит более 100 полезных утилит, находящихся после установки пакета в каталоге Program Files\Microsoft SDK\Bin. Их использование регламентируется встроенными подсказками, а также прилагаемой к Platform SDK гипертекстовой системой контекстной помощи. Кроме того, в состав пакета входит большое количество библиотек, заголовочных файлов, примеров программного кода и полезной документации.

#### Утилиты, поставляемые в составе Resourse Kit (ресурсы Windows)

В комплект входит большое число утилит. Их состав частично пересекается с утилитами, входящими в комплект Microsoft Platform SDK.

#### Утилиты с сайта sysinternals.com

На данном сайте имеется много полезных свободно распространяемых утилит, большинство из которых написано М.Руссиновичем.

Помимо перечисленных разработчиками активно используются утилиты, входящие в состав пакетов Visual Studio, DDK (Device Driver Kit), и разнообразные средства отладки. Вместе с тем, обилие инструментальных возможностей никоим образом не исключает необходимости разработки программ для всестороннего изучения ОС. Практическое применение API системы позволяет лучше изучить ее особенности, а также дает возможность создавать гибкие приложения, которые соответствуют сложным сценариям и требованиям, предъявляемым к современному программному обеспечению.

### Заключение

В данной лекции рассмотрены вопросы, важные с точки зрения практического освоения ОС Windows и разработки Win32-приложений. Основным источником сведений об API системы является справочная система MSDN. Разработчику приложений необходимо владеть средствами разработки и отладки, знать основные типы данных и форматы хранения текстовых строк, а также правильно обрабатывать ошибки. Дополнительным источником сведений о системе являются разнообразные инструментальные средства.