Графические Системы. Часть I

Лекция № 11

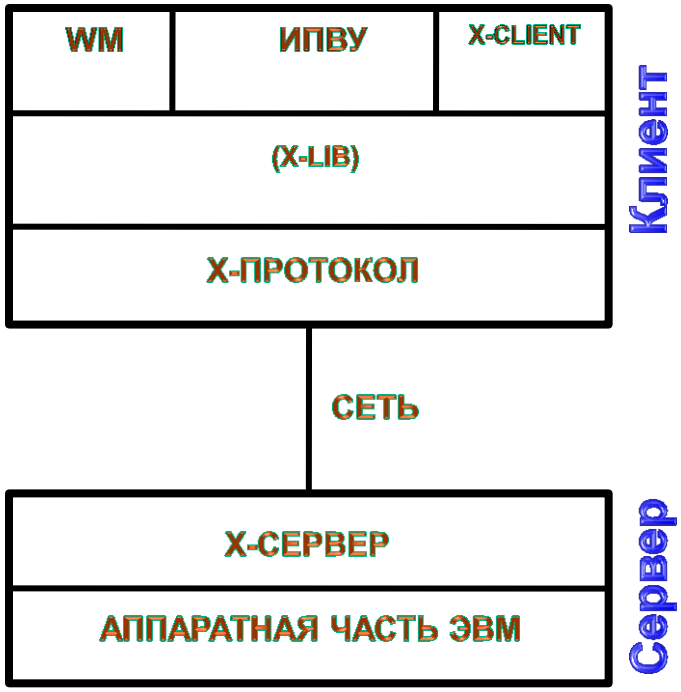
Развитие и применение

основных концепций создания GUI для различных архитектур Открытых Систем

Развитие основных концепций создания GUI X Window System.

X Window System — оконная система, обеспечивающая стандартные инструменты и протоколы для построения графического интерфейса пользователя.

X Window System обеспечивает базовые функции графической среды: отрисовку и перемещение окон на экране, взаимодействие с мышью и клавиатурой.



X Window System не определяет деталей интерфейса пользователя — этим занимаются менеджеры окон, которых разработано множество.

В **X Window System** предусмотрена *сетевая прозрачность*: графические приложения могут выполняться на другой машине в сети, а их интерфейс при этом будет передаваться по сети и отображаться на локальной машине пользователя (в случае если это разрешено в настройках).

В контексте **X Window System** термины «клиент» и «сервер» имеют непривычное для многих пользователей значение: **«сервер»** означает локальный дисплей пользователя (*дисплейный сервер*), а **«клиент»** — программу, которая этот дисплей использует (она может выполняться на удалённом компьютере).

В настоящее время выпущена версия **X11R7.5**, базирующаяся на **X.Org Server** (*"X.Org Foundation Open Source Public Implementation of X11"*) — свободной реализации сервера **X Window System** с открытым кодом. Текущая стабильная версия 1.16.2 является частью **X11R7.7**; выпущена в ноябре 2014.

Клиент – серверная архитектура

Как правило компьютеры и программы, входящие в состав информационной системы, не являются равноправными. Некоторые из них владеют ресурсами (файловая система, процессор, принтер, база данных и т.д.), другие имеют возможность обращаться к этим ресурсам. Компьютер (или программу), управляющий ресурсом, называют сервером этого ресурса (файл-сервер, сервер базы данных, вычислительный сервер...). Клиент и сервер какого-либо ресурса могут находится как на одном компьютере, так и на различных компьютерах, связанных сетью.

Клиент-сервер (англ. Client-server) — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

Физически клиент и сервер — это программное обеспечение.

Обычно они взаимодействуют через компьютерную сеть посредством сетевых протоколов и находятся на разных вычислительных машинах, но могут выполняться также и на одной машине.

Программы — сервера, ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями, просмотр web-страниц во всемирной паутине).

Основы программирования в системе X Window System.
РЕСУРСЫ X-Клиентов

Одна из базовых концепций Window System,
придающая системе бОльшую гибкость, это концепция РЕСУРСОВ

Основной тезис - Не следует заводить пару widget'ов "красная кнопка" и "зеленая кнопка", если можно завести один widget "кнопка" с параметром "цвет" (тем более, в дальнейшем разработчику понадобится еще и "желтая кнопка"). Поэтому widget должен быть достаточно универсальным: у него должно быть много параметров, списки которых отличаются от widget'а к widget'у , и, в зависимости от параметров, объект - widget должен РИСОВАТЬСЯ и реагировать на события по-разному.

Такие (поименованные) параметры или атрибуты widget'ов называются ресурсами. Ресурсами widget'ов могут быть, например, цвет фона его окна, шрифт выводимого текста, цвет границы окна; механизм ресурсов может быть использован для определения модели поведения widget'а в зависимости от происходящих событий.

Главная проблема - как задавать и изменять ресурсы widget'ов? Не умозрительно, а руками, в программе? Различные параметры widget'а разбросаны по нескольким (иногда десяткам) вложенных друг в друга структур. Не помнить же наизусть названия всех этих структур?

Разработчики X Window предложили способ разрешения данной проблемы.
Это Механизм конфигурирования ресурсов.

Основы программирования в системе X Window System.

РЕСУРСЫ X-Клиентов

Ресурсы разных типов можно конвертировать друг в друга. В частности, почти все типы допускают конвертирование в **вид текстовой строки и обратно.**

Например, тип "Цвет" естественно задавать либо в виде трех чисел (формат RGB), либо в виде названия этого цвета. Нетрудно сформировать (и это сделано в **X Window**) таблицу цветов: название и числа RGB. Пользователю удобнее работать с этими названиями, куда более содержательными, чем три числа.

Если ресурсы почти всех типов можно конвертировать из вида текстовой строки, значит, можно завести текстовый файл со значениями каких-либо ресурсов. Файлы эти можно менять, не затрагивая исходный код приложения. Будем называть их **конфигурационными файлами**.

Конфигурационные файлы состоят из директив задания значения ресурсов. При инициализации (**старте X-сессии**) эти директивы подгружаются в небольшую базу данных, поддерживаемую **XToolkit**. **Rdb** – это ресурсная база данных, используемая **X-Сервером** для предоставления окнам **X-Клиентов** различных характеристик для своего воспроизведения.

Далее, при создании каждого **widget'a**, наступает фаза переустановки значений ресурсов в соответствии с конфигурационными файлами. На этой фазе **XToolkit** опрашивает свою базу по каждому ресурсу в отдельности и из всех директив, которые могут относиться к данному ресурсу, выбирает максимально "специальную" директиву, наиболее подробно соответствующую этому ресурсу. Правила выбора этой "самой специальной" директивы достаточно сложны, но интуитивно понятны.

Основы программирования в системе X Window System.
РЕСУРСЫ X-Клиентов

Формат каждой директивы довольно прост:

путь_к_ресурсу: значение ресурса

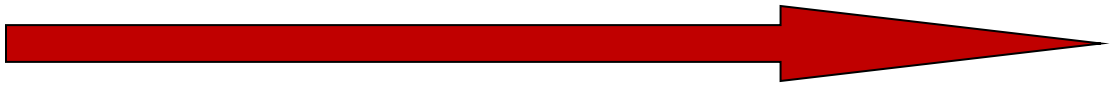
Что же такое путь к ресурсу? Ресурс - это часть widget'а. Widget, как правило, входит в состав другого widget'а. Тот в свою очередь... Но рано или поздно эта цепочка заканчивается головным widget'ом приложения, т.е X-Клиента. Остается теперь вспомнить, что у каждого ресурса и у каждого widget'а есть целых два имени: имя объекта и имя класса. Путь к ресурсу состоит из последовательности этих имен, разделенных точками.

Иными словами, в X Window файл ресурсов есть обычный текстовый файл, каждая строка которого задает тот или иной параметр (ресурс) программы. (При этом предполагается, что программу "населяют" именованные объекты – widget'ы, вернее их совокупности, связанные в некоторую иерархию - Widget-tree).

Общий вид строки следующий:

```
<имя программы>.<подобъект1>.<подобъект2>. . .
                                <подобъектN>.<имя ресурса>:  <значение ресурса>
```

Подобная строка задает значение ресурса для подобъектов иерархии объектов программы. Например, запись



Основы программирования в системе X Window System.

РЕСУРСЫ X-Клиентов

```
myprog.dialogwnd.background: Red
```

говорит, что в программе **myprog** у объекта- **widget'а** с именем **dialogWindow** параметр **background** (цвет фона) имеет значение **red** (красный цвет).

Вместо имен объектов- **widget'ов** могут указываться их классы. Обычно класс имеет то же самое имя, что и **widget**, но начинается с заглавной буквы, например,

```
Myprog.dialogwnd.Background: Red
```

Часть **widget'ов** или классов в левой части строки, задающей ресурс, может заменяться символом **'*'**, например, строка

```
myprog*background: Red
```

указывает, что для всех объектов программы **myprog** ресурс **background** имеет значение **Red**.

Связка с помощью символа **'.'** имеет бОльший приоритет, чем связка с помощью **'*'**. Так, если в файле, задающем ресурсы, есть две строки

```
myprog*background: Red
myprog.dialogwnd.background: Green
```

то все объекты программы будут иметь ресурс **background** равный **Red**, кроме объекта **dialogwnd**, для которого этот параметр - **Green**.

Основы программирования в системе X Window System.

Создание базы данных ресурсов X-Клиентов

Загрузка ресурсов производится специальными инициализационными процедурами, которые конструируют базу данных ресурсов X-сервера для воспроизведения Window-tree X-Клиента из различных ресурсных файлов, опций командной строки вызова и других источников.

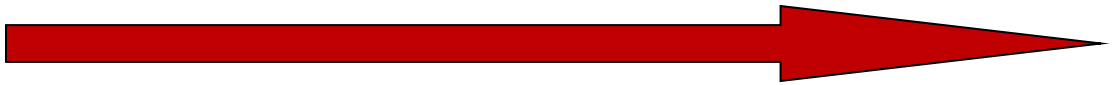
Для доступа к данным X Window предоставляет набор функций, которые совокупно называются *менеджер ресурсов* (Resource Manager), и специальную программу xrdb, которая позволяет считать любой ресурсный файл и включить его в общую таблицу ресурсов сервера. Последняя называется базой данных ресурсов сервера (resource data base), и представляет собой область памяти, ассоциированную со свойством (property) XA_RESOURCE_MANAGER корневого окна экрана дисплея.

Основы программирования в системе X Window System.
Создание базы данных ресурсов X-Клиентов

Рассмотрим последовательность шагов по поиску и загрузке ресурсов в rdb X-сервера. Эти ресурсы необходимы X-Клиентам для воспроизведения себя на экране дисплея.

Шаг № 1. Загрузка ресурсов из командной строки старта **X-Клиента**.
В **X– Window System** поддерживается стандартное множество опций – конфигурационных ресурсов **X-Клиента**, комбинации которых могут указываться в командной строке при запуске программы **X-Клиента**.
(Было предметом изучения в Лабораторной работе №1)

Эти ресурсы перечислены в таблице:



Основы программирования в системе X Window System.

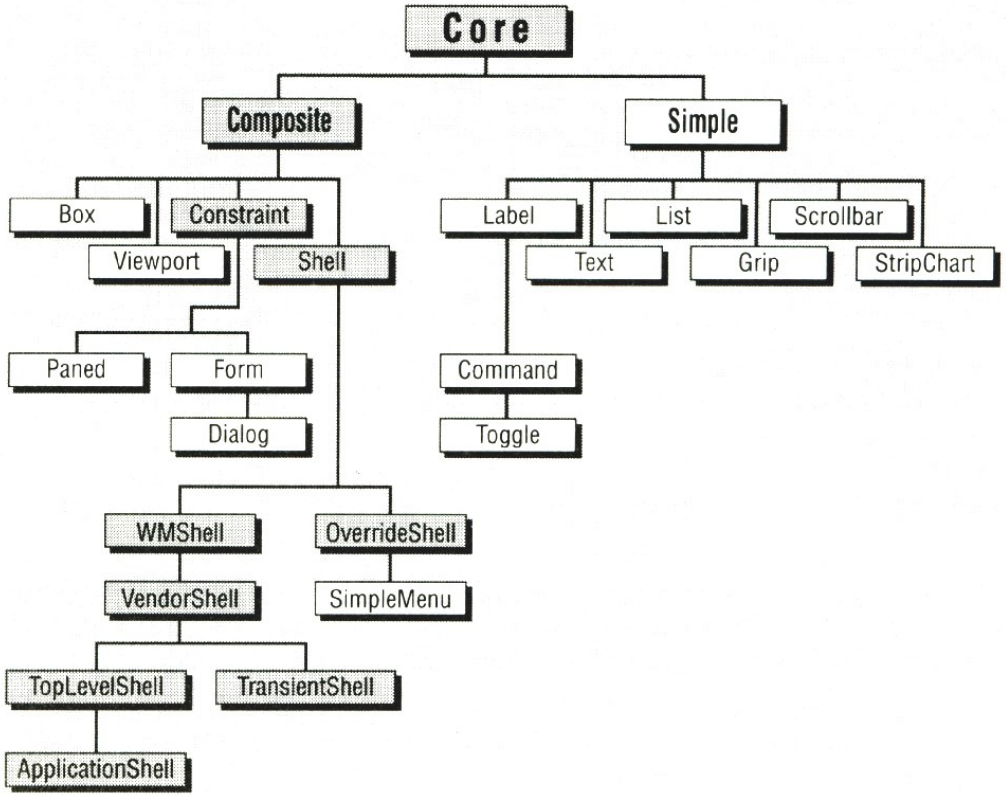
Создание базы данных ресурсов X-Клиентов. Шаг № 1

Опции	Имя ресурса	Тип	Описание и примеры
-bg -background	*background	String	Цвет фона -bg blue
-bd -bordercolor	*borderColor	String	Цвет границы -bd black
-bw -borderwidth	.borderWidth	Integer	Ширина края окна в пикселах -bw 2
-display	.display	String	Сетевой ресурс для связи с сервером – дисплей -d CPUB:0.0
-fg -foreground	*foreground	String	Цвет «передника» -fg red
-fn -font	*font	String	Имя шрифта -fn 9x15
-geometry	.geometry	String	Размер и положение окна: -geometry 171x252
-iconic	.iconic	None	Иконизация окна – если значение «on»
-name	.name	String	Имя X-Клиента: -name VASJA
-rv -reverse	*reverseVideo	None	Инверсия видимого изображения – если значение «on»
+rv	*reverseVideo	None	Возврат к нормальному изображению
-title	.title	String	Заголовок окна: -title WINVASJA
-xrm	Значение аргумента	String	Ресурс и его значение задаются аргументом опции -xrm “*height:100”

Основы программирования в системе X Window System.

Создание базы данных ресурсов X-Клиентов

В данной таблице точка, стоящая перед именем ресурса, означает, что опции могут быть использованы для установки ресурса только в widget'е, принадлежащем классу TopLevelShell или его подклассу (т.е. только для корневых окон приложений). В свою очередь, «*» означает, что ресурс может быть установлен для произвольного widget'а X-Клиента.



Данный способ задания ресурсов – самый приоритетный с точки зрения управления процессом их загрузки в rdb.

Шаг № 2. Загружается файл, на который указывает системная переменная среды **XENVIRONMENT** (если таковая вообще задана). Данная переменная содержит полный путь к файлу, включая и его имя. Если же переменная **XENVIRONMENT** не установлена, то **rm** будет пытаться подгрузить файл «**.Xdefaults-<hostname>**», который находится в домашней (**home**) директории пользователя. Здесь **<hostname>** - это имя компьютера, на котором происходит запуск **X-Клиента.**

Основы программирования в системе X Window System.

Создание базы данных ресурсов X-Клиентов

Шаг № 3. Если «корневое» окно (**root window**) экрана имеет ресурсы, загруженные в «свойство» **XA_RESOURCE_MANAGER (RESOURCE_MANAGER property)** программой **xrdb**, то они также добавляются в базу данных ресурсов **X-Клиента**. Если же «корневое» окно не имеет такого свойства, **rm** будет пытаться подгрузить данные, находящиеся в файле «**.Xdefaults**», который находится в домашней (**home**) директории пользователя.

Шаг № 4. Различные варианты загрузки ресурсных файлов класса **X-Клиента**. Это может быть осуществлено через переменные окружения **XAPPLRESDIR, APPLRESDIR** или **XUSERFILESEARCHPATH** и др., которые позволяют задать путь к файлу ресурсов, например «**<$ XAPPLRESDIR/<classname>**». Здесь **<classname>** - имя класса **X-Клиента**. Если переменные не установлены, **rm** будет пытаться подгрузить данные, находящиеся в файле «**.Xdefaults**», который находится в домашней (**home**) директории пользователя.

Шаг № 5. Rm осуществляет поиск файла:

«usr/lib/X11/app-defaults/<classname>»

Если этот файл существует то он загружается в базу данных ресурсов.

Шаг № 6. Если значения каких-либо ресурсов остались неустановленными, рассматриваются параметры, переданные с помощью определенных аргументов инициализационных процедур. Значения, переданные через параметры, представляются в виде массива указателей на строки. Каждая строка имеет вид:

«<описание ресурса>: <значение ресурса>»

Основы программирования в системе X Window System.
Изменение базы данных ресурсов X-Клиентов

Пользователь имеет возможность изменять значения ресурсов **X-Клиента** с помощью утилиты **xrdb** и ее опций, подгружая в ресурсную базу данных (**rdb**) содержимое ресурсных файлов, имеющих произвольные имена.

```
Xrdb  – load [<имя файла ресурсов>]
```

Означает, что значение ввода (ресурс, как строка или содержимое поименованного ресурсного файла) загружается как новое значение свойства **(property) XA_RESOURCE_MANAGER,** т.е. изменяет содержимое ресурсной базы данных.

```
xrdb  – merge [<имя файла ресурсов>]
```

Означает, что значение ввода (ресурс, как строка или содержимое поименованного ресурсного файла) загружается как дополнение к существующему значению свойства **(property) XA_RESOURCE_MANAGER,** т.е. дополняет содержимое ресурсной базы данных новыми значениями.

```
xrdb  – query [<имя файла ресурсов>]
```

Означает, что текущее значение свойства **(property) XA_RESOURCE_MANAGER,** т.е. содержимое ресурсной базы данных, выводится на стандартный выход (stdout).

Основы программирования в системе X Window System.

Дополнительные **X-утилиты** для опроса текущих значений ресурсов **X-клиентов** и другой информации об **X-сервере**

```
xdpyinfo  – display [host]:sever[.screen]
```

Отображает информацию об X-сервере;

```
xwininfo
```

Отображает информацию об окне. Окно может быть указано пользователем. Используйте в формате

```
xwininfo -help
```

```
xprop –[help]
```

Отображает информацию об окне или фонте. Окно может быть указано пользователем.

```
listres <имя виджета>
```

Отображает список ресурсов указанного виджета (класса). Например:

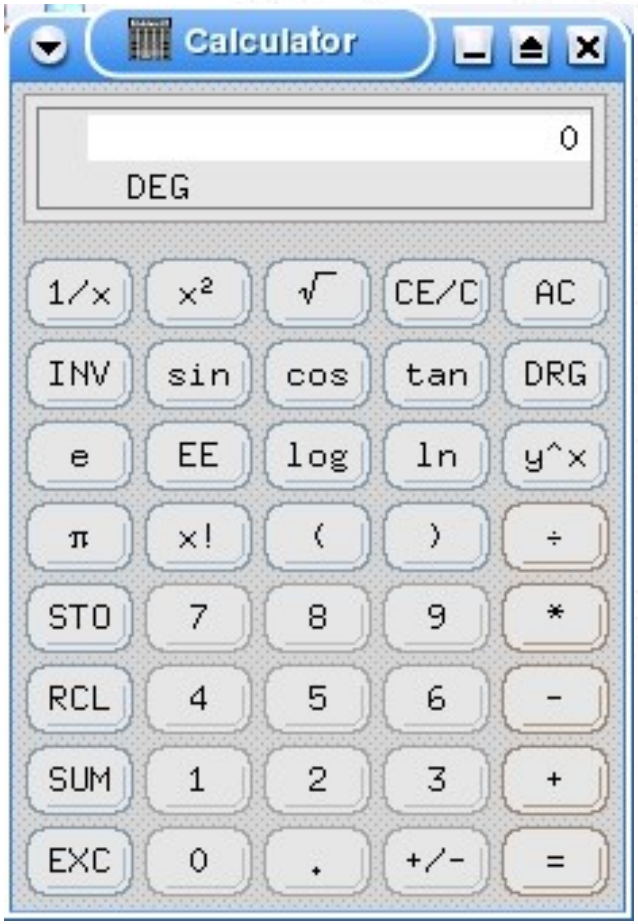
```
listres all
```

```
listres CORE
```

Основы программирования в системе X Window System.

Итак: Во время работы программа **X-Клиент** создает сами объекты- **widget'ы** (экземпляры **widget**-классов). Они образуют совокупности, каждая из которых также представляет собой некоторую иерархию. Каждая такая иерархия называется деревом объектов – **widget tree.**

Widget- and window-tree X-Клиента Xcalc:

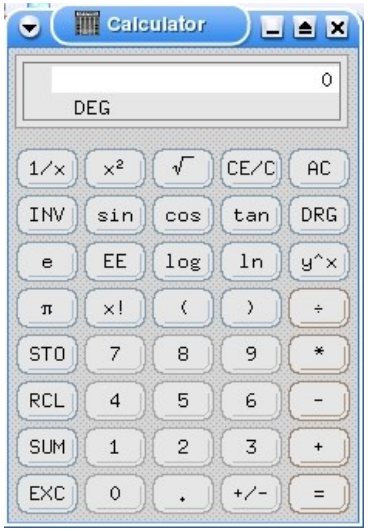



```
XCalc xcalc
        Form  ti  or  rpn        (the name depends on the mode)
          Form  bevel
            Form  screen
                Label   M        (the memory indicator on the screen)
                Toggle  LCD      (where the data is displayed)
                Label   INV      (the inverted indicator on the display)
                Label   DEG      (the degrees indicator on the display)
                Label   RAD      (the radians indicator on the display)
                Label   GRAD     (the gradians indicator on the display)
                Label   P        (the Parenthesis indicator on the display)
        Command  button1         (the actual calculator buttons)
        Command  button2         (buttons are numbered from right to left)
        Command  button3         (See the app-defaults file for associations
                                  and so on...)
        Command  button38        (between widget names and default labels)
        Command  button39
        Command  button40        (Only 39 buttons in HP mode)
```

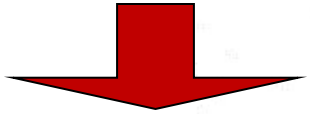
Основы программирования в системе X Window System.

Итак: Во время работы программа **X-Клиент** создает сами объекты- **widget'ы** (экземпляры **widget**-классов). Они образуют совокупности, каждая из которых также представляет собой некоторую иерархию. Каждая такая иерархия называется деревом объектов – **widget tree.**

Widget- and window-tree X-Клиента xcalc:



Ресурсы X-Клиента **Xcalc** формируются следующим образом (примеры строк ресурсного файла, типа **Appdefaults**):



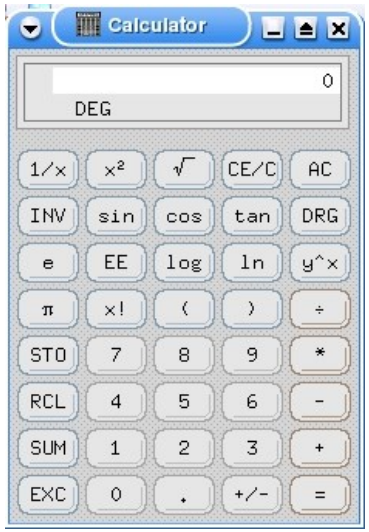
```
XCalc xcalc
Form  ti  or  rpn          (the name depends on the mode)
   Form  bevel
      Form  screen
         Label  M          (the memory indicator on the screen)
         Toggle LCD        (where the data is displayed)
         Label  INV        (the inverted indicator on the display)
         Label  DEG        (the degrees indicator on the display)
         Label  RAD        (the radians indicator on the display)
         Label  GRAD       (the gradians indicator on the display)
         Label  P          (the Parenthesis indicator on the display)
Command  button1           (the actual calculator buttons)
Command  button2           (buttons are numbered from right to left)
Command  button3           (See the app-defaults file for associations
                            and so on...)
Command  button38          (between widget names and default labels)
Command  button39
Command  button40          (Only 39 buttons in HP mode)
```

```
XCalc*bevel.screen.Label.horizDistance: 4

XCalc*ti.Geometry: 171x252

XCalc*ShapeStyle: rectangle

XCalc*ti.button1.label: 1/x

XCalc*ti.button6.label: INV
```

Основы программирования в системе X Window System.

Итак: Во время работы программа **X-Клиент** создает сами объекты- **widget'ы** (экземпляры **widget**-классов). Они образуют совокупности, каждая из которых также представляет собой некоторую иерархию. Каждая такая иерархия называется деревом объектов – **widget tree.**

Widget- and window-tree X-Клиента xcalc:



К примеру, **Constrain resources Form-widget'a** X-Клиента **Xcalc** формируются следующим образом:

```
XCalc xcalc
    Form  ti  or  rpn        (the name depends on the mode)
        Form  bevel
            Form  screen
                Label   M       (the memory indicator on the screen)
                Toggle  LCD     (where the data is displayed)
                Label   INV     (the inverted indicator on the display)
                Label   DEG     (the degrees indicator on the display)
                Label   RAD     (the radians indicator on the display)
                Label   GRAD    (the gradians indicator on the display)
                Label   P       (the Parenthesis indicator on the display)
    Command   button1   (the actual calculator buttons)
    Command   button2   (buttons are numbered from right to left)
    Command   button3   (See the app-defaults file for associations
                         and so on...)
    Command   button38  (between widget names and default labels)
    Command   button39
    Command   button40  (Only 39 buttons in HP mode)
```

Form-widget

Command-widget - потомок Form-widget

Constrain resource Form-widget'a, значением которого является другой widget – потомок Form-widget'a

```
XCalc*ti.button12.fromHoriz:    button11
XCalc*ti.button12.fromVert:     button7
```

Основы программирования в системе X Window System.
Widget'ы и их ресурсы. Формализация

Следует обратить особое внимание на то обстоятельство, что каждый ресурс имеет свой тип. Тип определяет, в скольких байтах оперативной памяти размещается ресурс и как трактовать их содержимое. **XToolkit** поддерживает около 40 встроенных типов ресурсов. Но X-программист может свободно завести новые типы.

Для задания значения ресурсов разработан механизм преобразования типов ресурсов.

По аналогии с **widget'ами** для каждого ресурса должны быть определены имя типа ресурса; имя класса ресурса; имя ресурса;

Дополнительно должны быть известны привязка к месту ресурса в структуре данных **widget'a** (или класса **widget'ов**), а также значение ресурса по умолчанию.

Таким образом можно подытожить :

В **X Window System**, в концепции ИПВУ (**XToolkit**), реализованы механизмы по работе с **widget'ами** :

- несколько "самых базовых" **widget'ов** с реализацией методов производных **widget'ов** с использованием методов базовых **widget'ов**;
- процедуры поддержки функционирования иерархии **widget'ов**;
- процедуры опроса и изменения ресурсов;
- механизмы управления потоком событий и реакции **widget'ов** на эти события;

Основы программирования в системе X Window System.
Widget'ы и их ресурсы. Преимущества

Введение понятия ресурсов снимает, как минимум, две проблемы.

Во-первых, это проблема адресации параметров (ресурсов) **widget'ов**, а значит доступа к ним, за счет механизма конфигурирования .

Во-вторых - унификация работы с ресурсами **X-сервера**; действительно, достаточно написать конвертор из типа ресурсов "текстовая строка" в тип ресурсов "шрифт", и работа со шрифтами станет гораздо удобнее.

При более внимательном рассмотрении оказывается, что введенная схема предлагает универсальное решение еще одной проблемы - конфигурирования приложений под конкретного пользователя.

По своему назначению ресурсы в **X-Window** очень похожи на то, что обозначается тем же термином "ресурсы" в **MS Windows**. Но вся идеология работы с ресурсами в **X** в корне отличается.

В частности, в **MS Windows** ресурсы являются частью бинарного исполняемого файла (например, **winword.exe**), и могут указываться или при компиляции программы, или меняться при помощи специальных редакторов, которые позволяют модифицировать бинарный файл (что, вообще говоря, является работой для квалифицированного программиста, и зачастую противоречит законам об авторских правах).

В **X-Window** же ресурсы существуют отдельно от исполняемого кода программы, в виде текстовых файлов, и могут свободно меняться или в этих файлах, или даже при помощи ключей в командной строке.

Таким образом, если в **MS Windows**, к примеру, перевод всех сообщений некоей программы на русский язык является "хакерской" задачей, то в **X-Window** подобное действие -- вполне стандартно, доступно любому пользователю и описано в документации.

Сервис – ориентированная архитектура

Как уже отмечалось – мы рассматриваем основные концепции построения GUI и стандарты на его построение в контексте распределенных вычислений и Открытых Систем.
Система становится открытой, когда возможен и происходит переход к коммуницированию отдельных участников распределенных вычислений (компьютеров) на основе общепринятых стандартов.
Клиент-серверная архитектура представляет собой концептуальную основу для построения ОС.

Дальнейшее развитие архитектур и принципов объединения ПТК в сети, а также появление и активное развитие WEB-технологий, как среды совместных распределенных вычислений и работы пользователей ОС, привели к следующему шагу в организации реальных архитектур объединения разнородных вычислителей и проектирования программных приложений для них.

Речь идет о т.н. сервис – ориентированных архитектурах (SOA - service-oriented architecture).

SOA в самом общем виде можно определить как модульный подход к разработке программного обеспечения, основанный на использовании распределённых, слабо связанных (англ. loose coupling) заменяемых компонентов, оснащённых стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам.

Программные комплексы, разработанные в соответствии с сервис-ориентированной архитектурой, обычно реализуются как набор веб-служб (веб-сервисов), взаимодействующих по определенному протоколу – чаще всего SOAP, но существуют и другие реализации (например, на базе jini, CORBA, на основе REST).

Сервис – ориентированная архитектура

SOAP (от англ. Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Первоначально SOAP предназначался в основном для реализации удалённого вызова процедур (RPC). Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур.

Очень важной особенностью SOA является то, что интерфейсы компонентов в сервис-ориентированной архитектуре инкапсулируют детали реализации (операционную систему, платформу, язык программирования) от остальных компонентов, таким образом обеспечивая комбинирование и многократное использование компонентов для построения сложных распределённых программных комплексов, обеспечивая *независимость от используемых платформ и инструментов разработки, способствуя масштабируемости и управляемости создаваемых систем* (важнейшие свойства ОС)

SOA не привязана к какой-то определённой технологии. Она может быть реализована с использованием широкого спектра технологий, включая такие технологии как REST, RPC, DCOM, CORBA или веб-сервисы. SOA может быть реализована, используя один из этих протоколов и, например, может использовать дополнительно механизм файловой системы для обмена данными.

Главное, что отличает SOA - это использование независимых сервисов с чётко определёнными интерфейсами, которые для выполнения своих задач могут быть вызваны неким стандартным способом, при условии, что сервисы заранее ничего не знают о приложении, которое их вызовет, а приложение не знает, каким образом сервисы выполняют свою задачу.

Сервис – ориентированная архитектура

Необходимо уточнить, что такое веб-сервисы?

Веб-служба, веб-сервис (англ. web service) — идентифицируемая веб-адресом программная система со стандартизированными интерфейсами.

Веб-службы могут взаимодействовать друг с другом и со сторонними приложениями посредством сообщений, основанных на определённых протоколах (SOAP, XML-RPC, REST и т. д.). Веб-служба является единицей модульности при использовании сервис-ориентированной архитектуры приложения.

Можно сказать, что веб-сервисами называют услуги, оказываемые в Интернете.

С другой стороны - веб-сервисы— это технология. И как и любая другая технология, они имеют довольно четко очерченную среду применения.

Если посмотреть на веб-сервисы в разрезе стека сетевых протоколов, то это, в классическом случае, не что иное, как еще одна надстройка поверх протокола HTTP.

По сути, веб-сервисы — это реализация абсолютно четких интерфейсов обмена данными между различными приложениями, которые написаны не только на разных языках, но и распределены на разных узлах сети.

Именно с появлением веб-сервисов развилась идея SOA — сервис-ориентированной архитектуры веб-приложений (Service Oriented Architecture).

Нас все эти особенности SOA интересуют с точки зрения построения GUI

Сервис – ориентированная архитектура

Применение сервис-ориентированной архитектуры для реализации WebGUI.

Сервис - как доступный по сети вызов приложения (on demand – по требованию, cloud, Remote control – удаленный доступ..). Серверы (тот, кто реализует сервис) доступны в идеале по их сетевому адресу, не зависят от платформы (особенностей реализации рабочего места) запрашивающего сервис и ориентированы на использование Web-протокола сетевого взаимодействия.

Запрос и предоставление сервиса через WebGUI подразумевает наличие "универсальных" базовых элементов интерфейса: поле ввода, кнопка, поле вывода, скроллер и т.д. (widsget'ами их не называют в SOA). Множество таких универсальных "базовых элементов интерфейса», его состав, не зависит от конкретной реализации оконечного оборудования рабочего места пользователя и от типа сервиса. Особенности каждого из сервисов (результаты применения сервиса) определяются не составом элементов GUI, а только спецификой (функционалом) каждого из сервисов.

Реализация универсальных" базовых элементов интерфейса (аналоги "виджетов") - на уровне дескрипций Web-протокола (тэги html, например).

Передаваемая информация между клиентской и серверной частью на стороне пользователя организована в виде контекстно-зависимых структур. Стандарты здесь - HTTP-протокол, а более широко и универсально - XML. Объекты, служащие для описания универсальных" базовых элементов интерфейса Web-протокола уже "встроены" в состав HTTP-протокола (XML) и могут компоноваться в виде описаний.