

Графические Системы. Часть II

Лекция №1

Ресурсы Открытых Систем

Ресурсы Открытых Систем

Современный пользователь, находясь в среде Открытой Системы имеет дело не с отдельным вычислителем (ЭВМ) – он имеет дело с распределенной вычислительной средой, предоставляющей пользователю ряд **РЕСУРСОВ** (не путать с ресурсами в X11!!):

Вычислительный ресурс: это, по сути, вычислительные мощности Открытой Системы, которые могут быть использованы USER'ом распределенной вычислительной системы. Еще проще – это микропроцессоры.

Системный ресурс: это необходимый пользователю ресурс, который обеспечивается внешними устройствами – внешняя память, диски (различные технические решения для создания файлохранилищ), модемы, принтеры и т.д. и т.п.

Графический ресурс: – пока – различные графические возможности участников Открытых систем, необходимые для решения пользовательских задач, например, воспроизведение объекта проектирования (CAD/CAM/CAE), работа различных пользователей в среде одного и того же приложения, снабженного GUI и пр.

Возникает проблема: осуществление доступа к этим ресурсам каждого пользователя Открытой Системы и создание системы управления этими ресурсами



Система управления Ресурсами Открытых Систем

Вычислительный ресурс: Можно сказать, что основная задача пользователя в Открытых Системах – найти эффективный и максимально отвязанный от конкретного вычислителя способ управления вычислительными ресурсами.

Такой способ существует и был найден достаточно давно. Речь идет о командных процессорах (уровень командных файлов) - **SHELL**

Командный процессор нужен для интерпретации команд операционной системы, запуска программ, выполнения скриптов, заданий и некоторых других задач, обеспечивающий интерфейс для взаимодействия пользователя с функциями системы.

Первый в мире командный процессор был написан в 1976 году Стивеном Борном из Bell Labs и входил в **UNIX Version 7**; **Bourne shell** часто называют **sh** по имени исполняемого файла.

Через некоторое время в Беркли Биллом Джоем был написан **C-shell (csh)** со встроенным скриптовым языком, впоследствии он же написал редактор **vi**, был одним из авторов первой версии **BSD UNIX** и входил в число основателей **Sun Microsystems**.

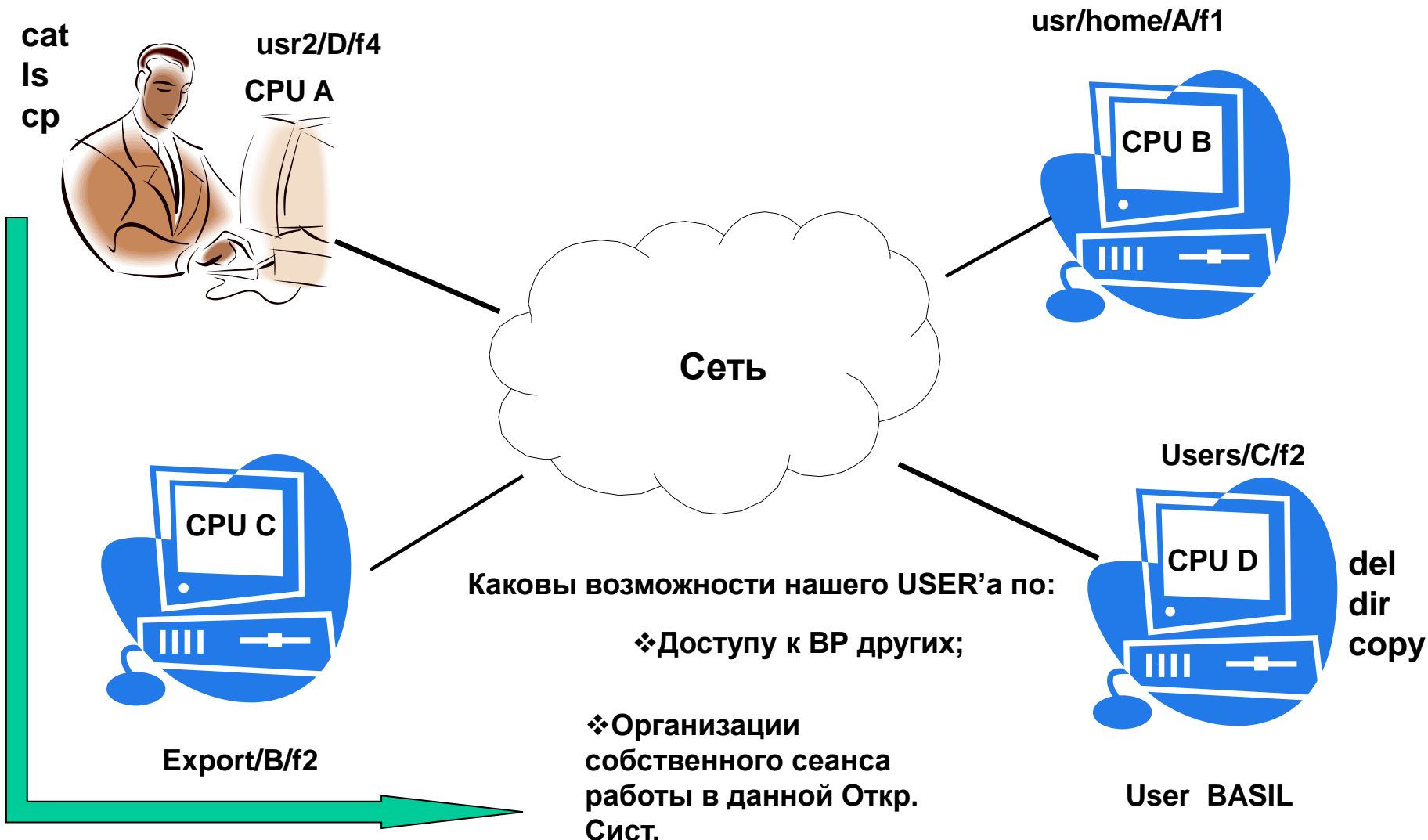
В конце 1970-х годов был написан **tcsh** - расширение **csh**, совместимое с ним по командам и синтаксису языка скриптов.

Позднее появились **Bourne again shell (bash)**, разработанный в рамках проекта **GNU**, и **Korn shell (ksh)** Дэвида Корна.



Система управления Ресурсами Открытых Систем

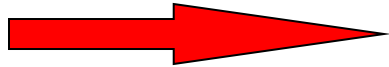
Вычислительный ресурс: Как осуществить доступ к Вычислительному ресурсу в распределенной гетерогенной вычислительной среде?



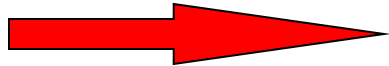
Существует два способа, появившиеся хронологически в следующей последовательности

Система управления Ресурсами Открытых Систем

Вычислительный ресурс



Берклевские утилиты сетевой коммуникации



Средства, входящие в состав ТСП/IP - протокола

(Transmission Control Protocol/Internet Protocol – протокол управления передачей пакетов данных и протокол межсетевых соединений)

Система управления Ресурсами Открытых Систем

Вычислительный ресурс

Берклевские утилиты сетевой коммуникации

В 70-е годы, когда в рамках проектов ARPA и DARPA активно велись работы по созданию протоколов межсетевых соединений, большинство университетских факультетов компьютерных наук использовали версию операционной системы UNIX, разработанную в программном отделении Берклиевского Университета в Калифорнии, называемую BSD (Berkeley Software Distribution) UNIX. Помимо стандартных прикладных программ TCP/IP (в рамках проекта совместно с агентством DARPA - Defence Advanced Research Projects Agency), Беркли предлагало набор утилит для работы с сетью, которые напоминали средства UNIX, используемые на одной машине. Главное преимущество утилит Беркли заключалось в их сходстве со стандартным UNIXом.

Эти утилиты получили название «r-утилит». Буква «r» в их названии означает «remote» (удаленный), так как они предназначены для выполнения задач удаленного доступа. Основными из этих утилит являются следующие:

Система управления Ресурсами Открытых Систем

Вычислительный ресурс

Берклевские утилиты сетевой коммуникации

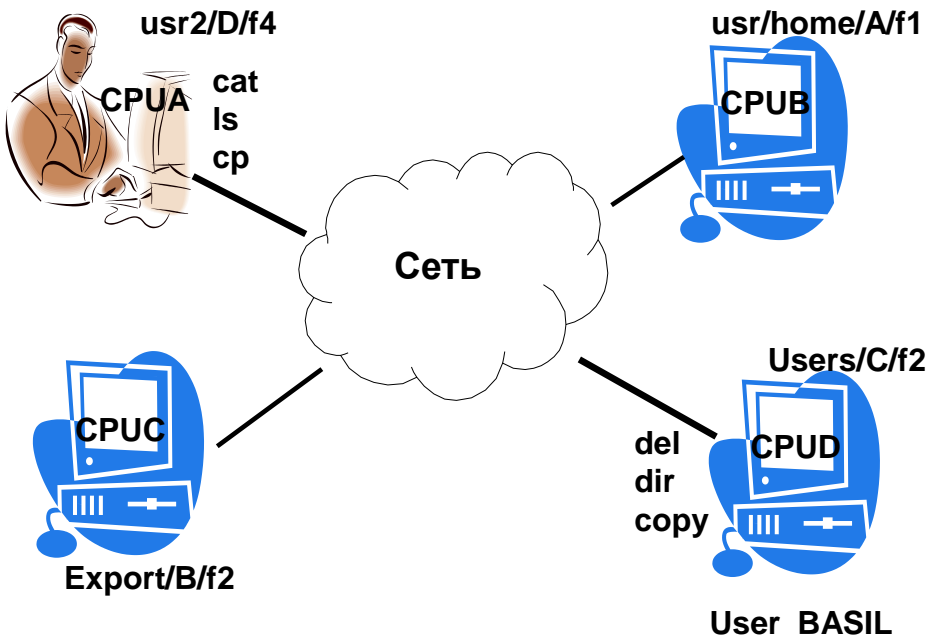
rlogin

Команда **rlogin** позволяет «войти» в сеанс работы на удаленном компьютере. По сути, она обеспечивает доступ в режиме удаленного терминала к удаленному компьютеру через сетевое соединение.

Формат команды:

```
$>rlogin hostname
```

где *hostname* это логическое имя удаленной машины.(Отметим, что ВСЕГДА вместо логических номеров машины вы можете использовать IP-адрес.)



Например, для нашего случая, пользователь машины CPU A хочет получить доступ к ВР машины CPU D, т.е. войти в систему, как пользователь этой машины:

```
CPUA $> rlogin CPUD BASIL
```

Если на CPUD зарегистрирован пользователь BASIL , то отзыв вернется на CPUA и ее пользователю будет отдано управление ресурсами пользователя BASIL на CPUD. (Это справедливо, если имеется «прозрачный», т.е. не требующий пароля, доступ к машине CPUD с данной локальной машины. В противном случае придется вводить пароль входа на машину CPUD. После получения приглашения, работа осуществляется на удаленной машине.)

Система управления Ресурсами Открытых Систем

Вычислительный ресурс

Берклевские утилиты сетевой коммуникации

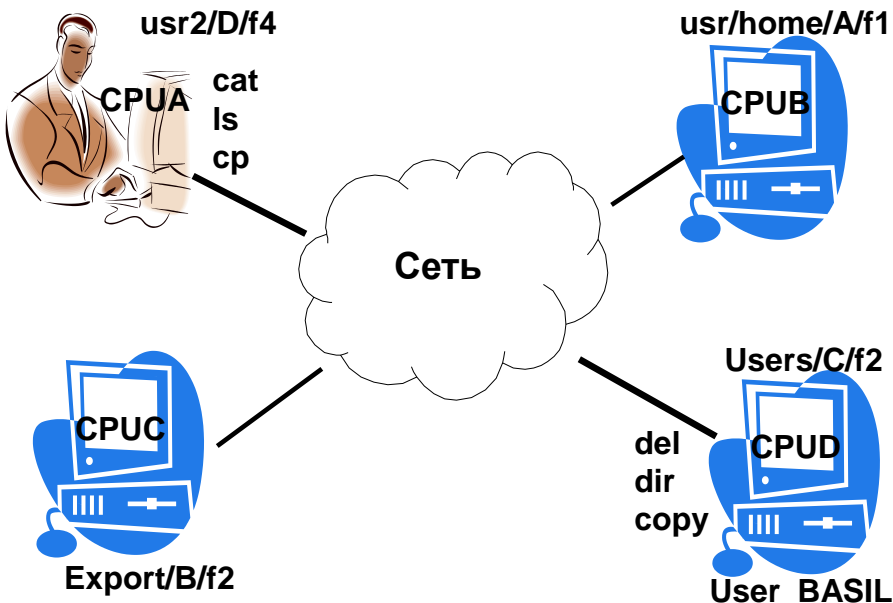
rsh

Предоставляет возможность выполнять команды на удаленной машине через сеть.

rcp

Команда **rcp** копирует файлы в или из других систем, которые поддерживаются BSD-сетью. Доступ к файлам на удаленной системе контролируются тем же способом, что и при использовании команды **rlogin**. Использовать **rcp** можно в том случае, если у пользователя есть разрешение использовать **rlogin** для связи с удаленной машиной без подтверждения пароля.

Утилита **rcp** наиболее популярная, используемая до сих пор, утилита, позволяющая локальному пользователю осуществлять обмен файлами между другими подсистемами (машинами) в сети.



Например, для нашего случая, пользователь машины CPU A может, «ничего не делая», заставить обменяться данными машины CPUB на CPUC (т.е. копировать файл f1 с машины CPUB на CPUC):

```
CPUA $> rcp CPUB: usr/home/A/f1 CPUC:Export/B
```


Система управления Ресурсами Открытых Систем

Вычислительный ресурс

Берклевские утилиты сетевой коммуникации

rwho

Запускает на удаленной системе демона (daemon) для получения информации о пользователях в удаленной сети;

Если сеть имеет широковещательные возможности, то можно использовать команду **rwho** для определения, кто из удаленных пользователей сделал **login** и на каких машинах локальной сети или локальной подсети, если система использует подсети. Для использования команды необходимо ввести **rwho**. На экран выведется результат выполнения команды.

ruptime

Выводит список компьютеров в удаленной сети и информацию о них, например число пользователей и время, прошедшее с момента последней перезагрузки системы;

Команда **ruptime** используется для проверки статуса других машин в локальной сети или в локальной подсети, если сеть разделена на подсети. Для использования **ruptime** необходимо ввести **ruptime**. Результат будет выведен на экран.

rdist

Команда позволяет на заданных машинах хранить идентичные копии файлов. По умолчанию, **rdist** просматривает только те файлы, версия которых на удаленных машинах более старая, чем на локальной машине. Это делается сравнением последнего времени модификации и размера файла на локальной машине и на удаленных.

Система управления Ресурсами Открытых Систем

Вычислительный ресурс

Берклевские утилиты сетевой коммуникации

Все это замечательно, однако использование такого набора утилит «поверх» ОС (UNIX), обеспечивающих сетевые коммуникации, обладает существенным недостатком. Каждое новое действие по управлению ресурсами в Открытых Системах требует пересмотра состава набора Berkeley-УТИЛИТ.

ВЫХОД:

Разработка нового концептуального подхода –

Стека протоколов TCP/IP (англ. *Transmission Control Protocol/Internet Protocol*) — собирательное название для сетевых протоколов разных уровней, используемых в сетях. Слово «стек» (англ. *stack*, стопка) подразумевает, что протокол TCP работает поверх IP.

В **модели OSI*** данный стек занимает (реализует) все уровни и делится сам на 4 уровня: прикладной, транспортный, межсетевой, уровень доступа к сети (в OSI это уровни — физический, канальный и частично сетевой). На стеке протоколов TCP/IP построено всё взаимодействие пользователей в сети, от программной оболочки до канального уровня модели OSI. По сути это база, на которой завязано всё взаимодействие. При этом стек является независимым от физической среды передачи данных.

(*)Сетевая модель OSI (*базовая эталонная модель взаимодействия открытых систем, (Open Systems Interconnection Basic Reference Model)*) — абстрактная сетевая модель для коммуникаций и разработки сетевых протоколов. Представляет уровневый подход к сети. Каждый уровень обслуживает свою часть процесса взаимодействия. Благодаря такой структуре совместная работа сетевого оборудования и программного обеспечения становится гораздо проще и прозрачнее. В настоящее время основным используемым семейством протоколов является TCP/IP, разработка которого не была связана с моделью OSI.

Система управления Ресурсами Открытых Систем

Вычислительный ресурс

Стек протоколов TCP/IP

Для гетерогенной распределенной вычислительной среды – ОС - и ее участников всегда существовала потребность эффективно реализовать следующие стандартные сетевые действия, которые с точки зрения сложности можно соотнести с тремя уровнями:

НИЗШИЙ УРОВЕНЬ: «старт удаленного файла», т.е. осуществление доступа локального пользователя к данным независимо от их расположения в Откр. Сист.

СРЕДНИЙ УРОВЕНЬ: «старт удаленного терминала», т.е. получение у локального пользователя promter'а удаленной машины (получение доступа к управлению всеми Вычислительными Ресурсами удаленной машины в качестве ее пользователя).

ВЫСШИЙ УРОВЕНЬ: «старт удаленного приложения», т.е. получение для локального пользователя не только доступа к ВР удаленной машины, но и получение для него возможности работать с удаленным приложением, графические настройки которого в общем случае могут не совпадать с возможностями графики локальной машины. Это – самый сложный случай, т.к. удаленные приложения на своей машине могут быть настроены на особенный набор ресурсов, которые не обязаны присутствовать на локальной, вызывающей машине.

Система управления Ресурсами Открытых Систем **Вычислительный ресурс**

Можно сказать, что для решения всех этих проблем и был разработан TSP/IP

Рассмотрим, каким образом можно осуществить доступ к ВР удаленной машины в рамках Откр. Сист. через старт удаленного терминала с помощью протоколов TSP/IP.

Если в Откр. Сист. Где-то есть ВР, то им можно управлять с локальной машины с помощью команд [удаленной] машины – владельца этого ВР (возможно – принципиально иной ОС), если все «участники» Открытой системы «воспринимают» TSP/IP.

Единственное, что для этого нужно сделать – это стартовать сессию **telnet**, т.е. реализовать «старт удаленного терминала».



Система управления Ресурсами Открытых Систем

Вычислительный ресурс

Старт удаленного терминала через TCP/IP

Команда **telnet** позволяет «войти» в терминальный сеанс работы с удаленным компьютером – «стартовать удаленный терминал».

Формат команды: `telnet [remote_host]`

где **remote_host** представляет собой адрес удаленной машины (логическое имя или IP-адрес).

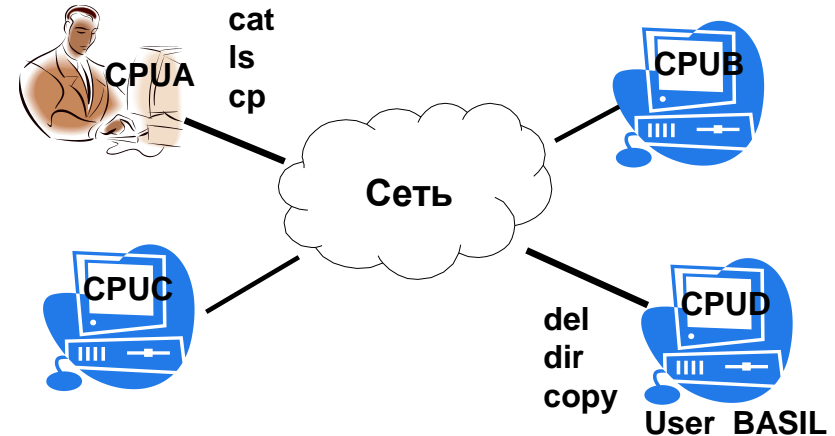
Пример: `CPUA$telnet CPUD`

В ответ вы получите обычное приглашение с удаленной машины.

`CPUD%>`

Вы вводите свое имя (BASIL) и пароль и оказываетесь в режиме терминальной работы с машиной CPUD. Для разрыва связи и возвращения на локальную машину используйте команду **exit**.

```
login>: BASIL
Password: *****
%>dir .....
.....
%>exit
telnet>exit
CPUA$
```



Система управления Ресурсами Открытых Систем

Вычислительный ресурс

Старт удаленного терминала через TCP/IP

Имя удаленной машины в командной строке **telnet'a** является необязательным параметром. Введя команду **telnet** без параметров вы немедленно получите приглашение **telnet>** и теперь, пользуясь командами самого **telnet'a** вы можете установить соединение с удаленной машиной или настроить параметры **telnet'a**. Наиболее полезной командой **telnet'a** для начинающих пользователей является команда «?», выдающая краткую справку по командам **telnet'a**. Пример работы с командами **telnet'a**:

```
% telnet
telnet>          ?
Commands may be abbreviated. Commands are:
close            close current connection
display          display operating parameters
mode             try to enter line-by-line or character-at-a-time mode
open             connect to a site
quit             exit telnet
send             transmit special characters ('send ?' for more) s
set              set operating parameters ('set ?' for more)
unset           unset operating parameters ('unset ?' for more)
status          print status information
toggle          toggle operating parameters ('toggle ?' for more)
slc             change state of special charaters ('slc ?' for more)
z              suspend telnet
!              invoke a subshell
?              print help information
telnet>        quit
%
```

Система управления Ресурсами Открытых Систем

Системные ресурсы

Здесь для нас самое главное – это дисковое пространство, т.е. ответ на вопрос – где и как хранить информационные данные в Открытой системе и получать доступ к ним.

В настоящее время существует одно универсальное средство – система NFS. С точки зрения NFS для пользователя Откр. Сист. нет разницы между «своими» (локальными) и «чужими» (на удаленной машине) файлами

Network File System (NFS) — протокол сетевого доступа к файловым системам, первоначально разработан Sun Microsystems в 1984 году. Основан на протоколе вызова удалённых процедур (ONC RPC, Open Network Computing Remote Procedure Call). Позволяет подключать (монтировать) удалённые файловые системы через сеть.

NFS абстрагирована от типов файловых систем как сервера, так и клиента, существует множество реализаций NFS-серверов и клиентов для различных операционных систем и аппаратных архитектур. В настоящее время (2007) используется наиболее зрелая версия NFS v.4, поддерживающая различные средства аутентификации (в частности, Kerberos и LIPKEY с использованием протокола RPCSEC_GSS) и списки контроля доступа (как POSIX, так и Windows-типов).

rNFS (параллельный NFS) — последняя версия стандарта NFS, включающая в себя распараллеленную реализацию общего доступа к файлам, которая увеличивает скорость передачи данных пропорционально размерам системы.

Управление Системным Ресурсом Откр. Сист. – типичная практическая задача системных программистов и администраторов сетей.

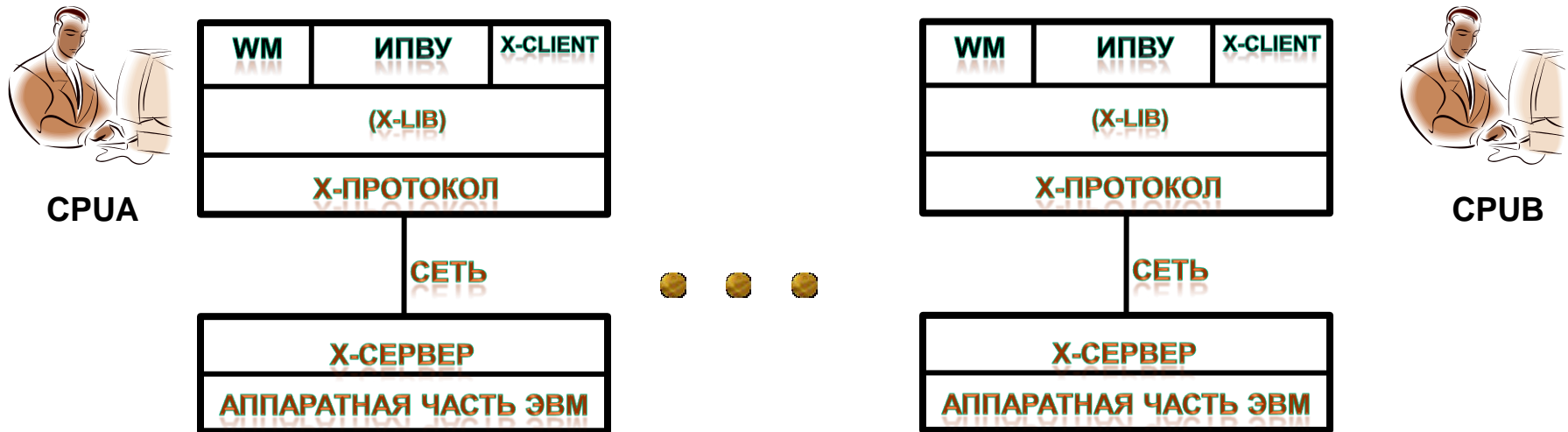
И, наконец, самое главное для нас



Система управления Ресурсами Открытых Систем

Графические ресурсы

Вернемся в связи с постановкой задачи управления Граф. Ресурсами в Откр. Сист. к стандарту X11

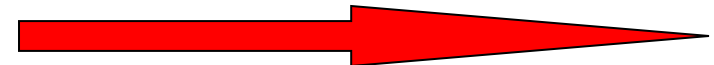


Вспомним, что основное преимущество представленной конфигурации заключается в том, что протокол X Window умеет "ездить" по сети: например по TCP/IP. Поэтому X-сервер может крутиться на одной машине, а X-клиент - на другой. То есть, картинка рисуется на одной машине, а программа, которая ее обеспечивает - крутится на другой. X-сервер способен обслуживать сразу много клиентов, причем всех - одновременно.

До сих пор мы обсуждали запуск локальных X-клиентов на собственной, локальной машине.

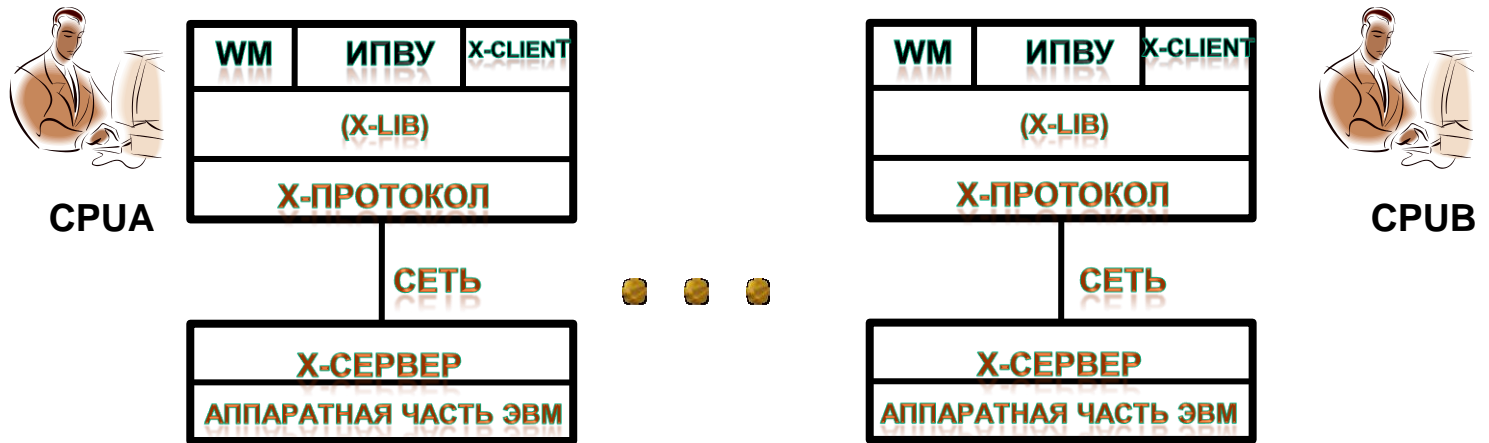
Реально в Открытых системах возникает задача запуска локального приложения (X-клиента) на удаленной машине и запуск удаленного X-Клиента на локальной машине.

Это и является первой темой этого раздела курса и содержанием первой лабораторной работы



Система управления Ресурсами Открытых Систем

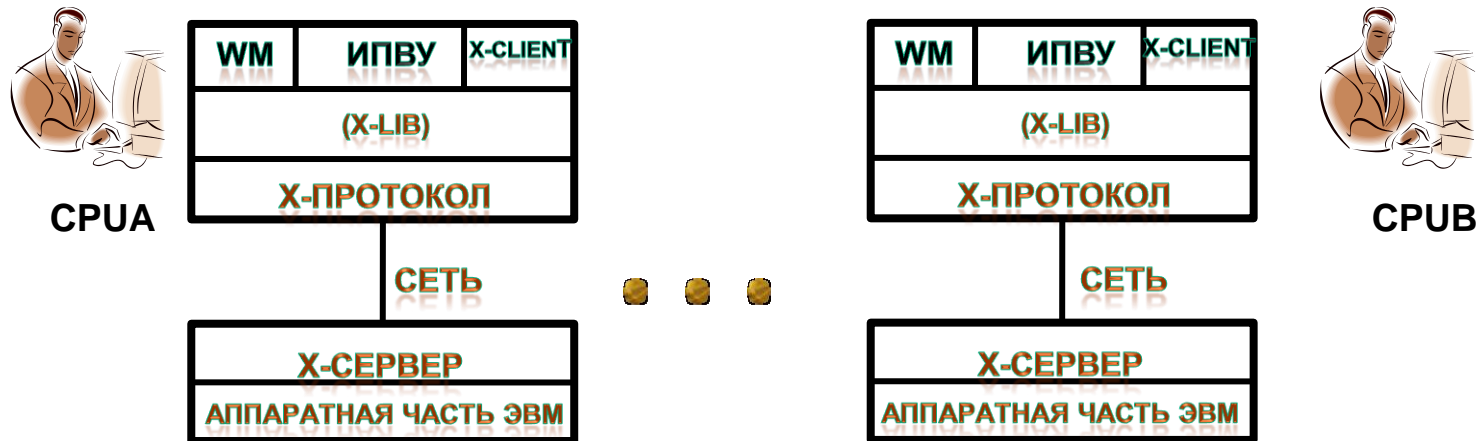
Графические ресурсы



Система управления Ресурсами Открытых Систем

Графические ресурсы

1) Запуск локального X-Клиента на удаленной машине



Вспомним, что все X-Клиенты унифицированы на сетевом уровне (X-протокол).

При запуске X-Клиента(CPUA) на машине CPUB его отображением должен заниматься X-Сервер CPUB.

Для реализации поставленной задачи необходимо задействовать специальный ресурс СЕТЕВОГО X-Протокола: **-display** или **(-d)**,

а также специальную X-Утилиту **xhost**, с помощью которой может быть изменен список машин (хостов), которым разрешено пользоваться X-Сервером локальной машины для отображения X-Клиентов.

X-сервер по умолчанию имеет возможность обрабатывать графические приложения, оформленные в соответствии со спецификацией X11, и запущенные на локальной машине, или графические приложения, запущенные на хостах, занесенных в список, размещенный в файле **/etc/Xn.hosts** (где **n** - номер дисплея сервера) локальной машины.


Система управления Ресурсами Открытых Систем

Графические ресурсы

1) Запуск локального X-Клиента на удаленной машине

Значение ресурса **-display** или **(-d)** строится по следующему закону:

-display [host]:server[.screen]



Имя/IP-адрес удаленной машины, на которой воспроизводится окно локального X-Клиента: CPUB

Запуск локального X-Клиента на удаленной машине:

CPUA\$>xlogo -d CPUB:0.0

(Не забыть про утилиту Xhost!!!)

№ X-Сервера, обрабатывающего воспроизведение окна локального X-Клиента: № X-Сервера CPUB; Т.к. мы не рассматриваем мультисерверные хосты, у нас № X-Сервера всегда = 0

№ Экрана дисплея машины, воспроизводящей окно локального X-Клиента: № Экрана CPUB; у нас № Экрана = 0, т.к. мы не рассматриваем общий случай, когда к одному Серверу м.б. подключено несколько экранов

Система управления Ресурсами Открытых Систем

Графические ресурсы

1) Запуск локального X-Клиента на удаленной машине

Значение ресурса **-display** или **(-d)** очень важно для работы в Открытых Системах и поэтому для его именованя, сохранения и передачи может быть использована специальная системная переменная

DISPLAY

Установка:

C-shell: %>setenv DISPLAY CPUB:0.0

Born-shell: \$> DISPLAY = CPUB:0.0
 \$> export DISPLAY

Система управления Ресурсами Открытых Систем

Графические ресурсы

2) Запуск удаленного X-Клиента на локальной машине

Производится в два этапа:

2.1 Разрешение доступа к локальному X-Серверу для воспроизведения удаленного X-Клиента:

CPUA \$>xhost + [Для всех доступных по сети машин] или

CPUA \$>xhost + CPUB [Для конкретной машины]

2.2 Собственно запуск удаленного X-Клиента:

```
CPUA$>telnet CPUB
```

```
CPUB%>login user1
```

```
CPUB%> Password user01
```

```
CPUB%>xlogo -d CPUA:0.0
```