1. **Тема 12. Системная шина D-Bus**

Рассмотренные в предыдущих темах средства пакета IPC являются достаточ- но эффективными для большинства системных приложений, в которых требуется *сложная логика* и имеется *опасность блокировок*.

С развитием сложности и многоплановости асинхронно взаимодейтсвующих приложений стали возникать проблемы, вызванные следующими двумя причинами:

* + - *возрастание логической сложности* программных систем требующей сложной алгоритмической реализации управляющих подсистем, что снижает

их качество при использовании низкоуровневых средств программирования, которым является IPC;

* + - *снижаются темпы* разработки и отладки больших программных систем. Примером таких приложений является *графическая подсистема ОС*.

**Проблематика равития** графических подсистем ОС, которые построены на основе множества асинхронно взаимодействующих процессов, потребовало созда- ния специальных высокоуровневых средств синхронизации процессов, вызванных все возрастающими требованиями к интерактивным системам.

Чтобы разобраться в этой проблематике, в подразделах 1.1 — 1.4 рассматри- ваются различные аспекты использования *X Window System*, которая и до насто- ящего времени является основой графических подсистем UNIX-подобных ОС.

**Конкретизацию решения** указанной проблематики рассмотрим на примере шины *D-Bus*, которая является современным подходом, обеспечивающим *высоко- уровневое асинхронное взаимодействие* процессов.

**D-Bus -** система межпроцессного взаимодействия, которая позволяет приложениям в операционной системе общаться друг с другом.

**D-Bus** является частью проекта *freedesktop.org*, обладает высокой скоростью рабо- ты и не зависит от рабочей среды ОС.

**D-Bus** работает на POSIX-совместимых ОС и имеет вариант реализации для *MS Windows*.

**D-Bus** состоит из трех частей:

* + - *демона*, реализующего саму шину;
    - *низкоуровневого API*, на основе библиотеки *libdbus*;
    - высокоуровневой библиотеки для фреймворков: *Qt, Java, Glib, C#, Python, Ruby и библиотека для C++*.

**Указанный круг вопросов** составляет содержание теоретической части дан- ного пособия.

## Графические среды ОС

В отличие от **MS Windows**, имеющей *встроенное в ядро ОС* графическое ПО, без которого она не может работать, Linux и другие UNIX-подобные ОС, для этих целей, используют набор приложений, взаимодействующих на основе модели

«клиент/сервер».

**Базовая часть** графического ПО Linux называется *X Window System* или просто - *X Window*.

**Начиная с 1988 года**, этот стандарт поддерживался консорциумом *X*, создан- ным с целью унификации графического интерфейса для ОС UNIX.

**В 1997 году**, консорциум *X* был преобразован в *X Open Group*, в чем можно убедиться на сайте [*http://www.x.org*.](http://www.x.org/)

**X Window** - сложная система, подробно описаная большим количеством пер- воисточников. Основой ее является *программа X-сервер*, которая через драйверы устройств взимодействует с видеокартой ЭВМ, клавиатурой, мышью и монитором компьютера.

**Именно** Х-сервер:

* + - *устанавливает и переключает* графические режимы видеокарты ЭВМ;
    - *рисует* элементы изображений;
    - *определяет* координаты мыши;
    - *формирует* программные прерывания при нажатии кнопок мыши и клавиа- туры.

**Все остальные программы**, включая менеджер окон, взаимодействуют с Х- сервером по особому протоколу, который называется ***X-протокол***, или протокол сетевой связи (*X Network Protocol*).

Для написания программ, поддерживающих Х-протокол, имеется *базовая библиотека X-lib*. На основе этой библиотеки пишутся дополнительные графи- ческие библиотеки более высокого уровня, например, *GTk+, Qt, Motif и другие*.

**Обычно**, менеджеры окон, рабочие столы пользователей и сложные графические приложения пишутся с использованием этих библиотек.

**Общая архитектура** X Window может быть представлена рисунком 1.1, где Х-сервер обрабатывает *4 типа сообщений*:

* + - *Запрос* – клиент требует нарисовать что-либо в окне или запрашивает у сер- вера информацию;
    - *Ответ* – сервер отвечает на запрос;
    - *Событие* – сервер сообщает клиенту о событии, например, о нажатии клави- ши пользователем;
    - *Ошибка* – сервер сообщает об ошибке в запросе клиента.

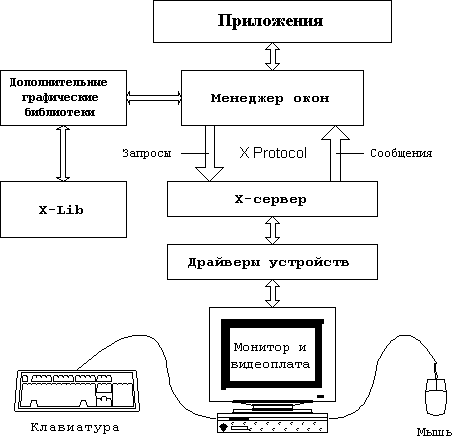


Рисунок 1.1 - Общая архитектура графической системы X Window

**Замечание**

Дистрибутив Х-сервера содержит набор шрифтов, настроечные файлы и ряд приложе- ний, которые можно найти в директории: */etc/X11/app-defaults*

На рисунке 2.2 представлен пример одного из таких приложений, которое поставляется вместе с дистрибутивом X-сервера. Это - *xcalc* (*калькулятор*), работающий на всех системах X Window System, включая запуск его в сети.

**Замечание**

Приложение, показанное на рисунке 1.2, запущено в среде рабочего стола *Xfce4*, поэ- тому оно имеет окно с титульной надписью, которое отображает менеджер окон. Х-сер- вер обычно отображает приложения в верхнем левом углу дисплея (*display*).

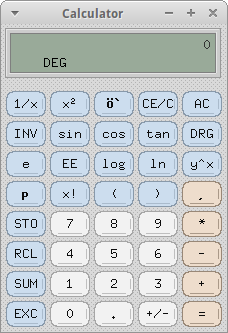


Рисунок 1.2 — Калькулятор системы X Window System

## Рабочий стол пользователя

**Писать и настраивать** приложения для Х-сервера довольно сложно, поэто- му все современные ОС, использующие графический интерфейс с пользователем, имеют специальное приложение, которое называется *оконным менеджером*.

**Оконный менеджер** - это приложение, которое отвечает за размещение, де- корирование окон и за взаимодействие между окнами:

* + - изменение размеров,
    - максимизация,
    - свертывание,
    - закрытие.

Оконный менеджер взаимодействует с графическим сервером *X11*, который, в

свою очередь, занимается взаимодействием c видеокартой и устройствами ввода/ вывода: *клавиатура, мышь и монитор*.

**Именно оконный менеджер** обеспечивает функциональность графической среды пользователя.

**Чтобы максимально** обеспечить удобство работы пользователя, в каждой ОС, разрабатывается набор приложений, который называются *окружение рабочего стола* или просто — *рабочий стол*.

**Учебная система** ОС УПК АСУ, созданная на основе ОС ArchLinux, исполь- зует рабочий стол:

* + - *Xfce* — как окружение рабочего стола;
    - *xfwm4* — как оконный менеджер.

### Замечание

**Абревиатура Xfce** появилась как акроним toolkit's: *XForm Common Environment*.

**Ее автор**, *Оливер Фордан*, начал проект в 1996 году.

**Целью проекта** было создание лёгкого настольного рабочего окружения для различных UNIX-подобных систем, способного быстро загружать и выполнять приложения, сохра- няя ресурсы системы.

**Со временем**, основа была много раз переписана, а название осталось.

**В настоящее время**, Xfce включает следующие приложения:

* + - *thunar* — файловый менеджер;
    - *xfwm* — менеджер окон;
    - *xfce4-panel* — панель задач;
    - *xfdesktop* — менеджер рабочего стола;
    - *xfce4-session* — менеджер сеансов;
    - *xfce4-settings* — диспетчер настроек;
    - *xfconf* — система хранения настроек, работающая через D-Bus;
    - *xfce4-appfinder* — поиск приложений;
    - *xfce4-terminal* — эмулятор терминала;
    - *xfce4-power-manager* — менеджер питания.

**Имеется также**, набор приложений, которые традиционно устанавливаются, по умол-

чанию: *mousepad, orage, parole, ristretto, xburn* и другие.

## Различие графических сред ОС

**Имеются** операционные системы, которые не могут работать без графичес- кого окружения.

К таким ОС относятся: *MS Windows* и *MacOS*.

**Для UNIX и Linux**, графическое окружение ОС является необязательным и может быть отключено *с целью экономии ресурсов ЭВМ*.

**В целом**, оконные менеджеры могут работать:

* + - *как вместе* с окружением рабочего стола;
    - *так и отдельно* от него.

**Функционал**, предоставляемый оконными менеджерами, может достаточно сильно различаться. В частности, оконные менеджеры подразделяются на:

* + - *тайловые*, в которых окна не перекрываются;
    - *композитные*, позволяющие окнам перекрываться.

**Принципиально**, для использования графических приложений, не требуется наличие окружения рабочего стола. Например, оконный менеджер *openbox* прекрас- но работает без рабочего стола ***LXDE***, который его использует.

**Ряд рабочих столов** использует собственные менеджеры окон, например:

* + - *Xfce* — xfwm;
    - *KDE* — Kwin;
    - *Gnome* — Metacity.
    - Возможно также переключение рабочего стола на другие оконные менедже- ры, например, *Compiz*.

**Замечание** Переключение рабочего стола на другой менеджер окон может потребовать множества дополнительных настроек, что не всегда удобно.

Учитывая, что за последние тридцать с лишним лет X11 уже морально уста- рел, ведется разработка нового графического сервера, в рамках текущего проекта *Wayland*.

**Первый выпуск** этого сервера состоялся *в 2008 году*, но до сих пор эта рабо- та продолжается.

**Wayland** будет использовать *Weston* — в качестве композитного менеджера.

**Для Linux**, будут задействованы уже существующие в ядре технологии:

* + - *KMS* — Kernel mode-setting;
    - *DRM* — Direct Rendering Manadger;
    - *GEM* — Graphics Execution Manadger.

**Для обеспечения** работы с уже используемыми Х-приложениями, разработа-

на специальная программная прослойка *xwayland*, - *сервер доступа к Wayland*.

## X-сервер UNIX

**Важной особенностью** Х-сервера является возможность его работаты на сте- ке протоколов TCP/IP с программами, запущенными на удаленных компьютерах.

**Обычно**, для этих целей используется ассинхронная связь (*протокол UDP*), но возможна и *синхронная связь по протоколу TCP*, которая работает в 30 раз мед- леннее, чем асинхронная связь по UDP.

**Кроме того**, на компьютере *может быть запущено несколько Х-серверов*, которые выводят графическую информацию на разные дисплеи с разными номера- ми: *0, 1, 3* и так далее.

**X-сервер** запускается дисплейным менеджером, который предлагает клиенту ОС ввести *login* и *password* для открытия сессии и входа в нее:

* + - *до мая 2011 года*, ОС Ubuntu и ее клоны использовали в качестве дисплейно- го менеджера *gdm*;
    - *с мая 2011 года*, дисплейным менеджером становится *lightdm*.

**Замечание**

Хотя X-сервер способен работать в сети, по умолчанию он запускается с опцией *- nolisten tcp*, запрещающей ему прослушивать сеть.

Чтобы разрешить Х11 **работать в сети**, необходимо:

* + - *открыть* файл /etc/lightdm/lightdm.conf;
    - *добавить* в секцию **[***SeatDefaults***]** строку:

**xserver-allow-tcp=true**

* + - *создать* еще одну секцию с командой:

**[*XDMCPServer*] enabled=true**

* + - *перезапустить* оконный менеджер командой:

**sudo systemctl restart lightdm**

* + - *проверить* новый запуск сервера Х11 в диспетчере задач, как показано на рисун- ке 1.3.

**Диспечер задач** запускается из меню: «*Главное меню/Система, Диспетчер задач*». **Вызвав диспетчер задач**, необходимо потребовать, чтобы он *отображал все процессы*.

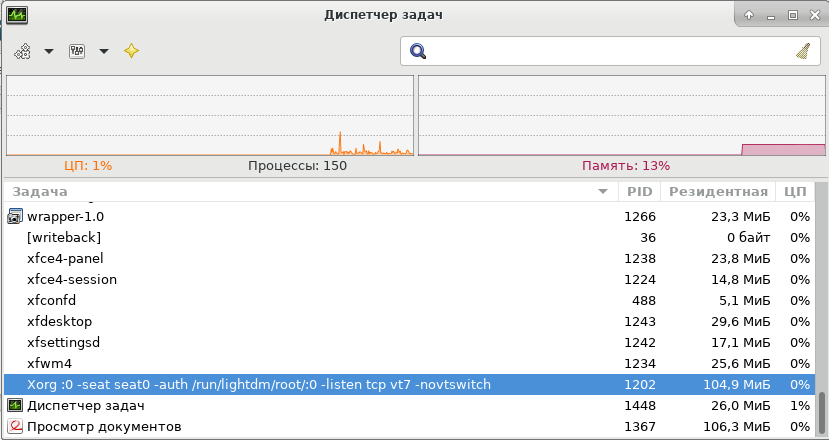


Рисунок 1.3 — Проверка запуска сервера Х11 в диспетчере задач

В используемом дистрибутиве ОС УПК АСУ, все необходимые настройки Х11 уже проведены, поэтому Х-сервер уже готов к работе в сети.

**Как отмечено выше**, на ЭВМ может быть запущено несколько Х-серверов, хотя обычно используется только один.

**Обычно**, для обеспечения сетевого соединения, *Х-сервер прослушивает не- который порт*.

**По умолчанию**, прослушивается порт с номером *6000*.

Все графические приложения ОС UNIX/Linux выводят изображение на Х- сервер, который определяется адресом:

**<IP-адрес Х-сервера>:<номер дисплея>.<номер экрана дисплея>**

**Чтобы** прикладные программы знали на каком компьютере и на каком дисплее находится Х-сервер, используется:

* + - *переменная окружения DISPLAY*;
    - *аргумент программы* с ключем: *-display*.

В таблице 1.1, приведены примеры некоторых возможных значений перемен- ной DISPLAY.

### Замечание

Таблица 1.1 приведена только для примера.

*Таблица 1.1 - Варианты задания переменной DISPLAY*

|  |  |  |
| --- | --- | --- |
| ***Значение переменной DISPLAY*** | ***Номер порта соединения*** | ***Пояснение*** |
| :0 | -- | Х-сервер на локальном компютере, дисплей №0, экран №0 |
| :1.0 | -- | Х-сервер на локальном компютере, дисплей №1, экран №0 |
| asu.tusur.ru:4 | 6004 | Х-сервер на компютере asu.tusur.ru, дисплей №4, экран №0, номер UDP-порта - 6004 |
| 192.168.1.17:2 | 6002 | Х-сервер на компютере 192.168.1.17, дисплей №2, экран №0, номер UDP-порта - 6002 |

Чтобы, после старта ОС УПК АСУ, проверить адрес вывода Х-сервера, следу- ет воспользоваться командой *echo*, как показано на рисунке 1.4.

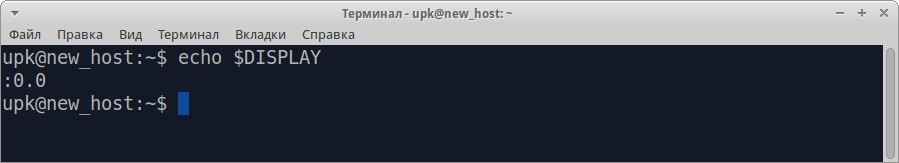


Рисунок 1.4 — Проверка текущего адреса вывода Х-сервера

**Замечание** X-сервер отображает не все программы, пытающиеся к нему подключится, а только те, которым разрешен доступ.

#### xhost.

**Чтобы разрешить** Х-серверу подключение программ, используется утилита

**Если** запустить ***xhost*** без параметров, как показано на рисунке 1.5, то мы по-

лучим сообщение, что только авторизованный пользователь может быть подклю- чен.

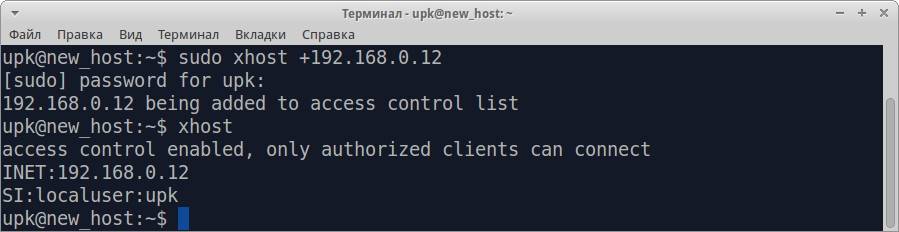


Рисунок 1.5 — Разрешение доступа программам, запущенным с хоста 192.168.0.12

**Если** нужно разрешить подключение с компьютера по извесному адресу, нап- ример, 192.168.0.12, то нужно использовать команду:

**sudo xhost +192.168.0.12**

где символы '+' и '-' означают разрешение или запрет подключения.

**После** выполнения этой команды, мы получим сообщение показанное на том же рисунке 1.5. Хорошо видно, что доступ с хоста 192.168.0.12 теперь разрешен.

**Замечание** Если вы запускаете множество программ, отображаемых на удаленном компьютере, сле- дует значение переменной *DISPLAY* прописать в файле *.bashtc*

## Архитектура шины D-Bus

**Системы IPC** для Unix/Linux обладают развитыми средствами IPC низкого уровня: сокеты, каналы, семафоры, сегменты разделяемой памяти, передача сооб- щений. Эти средства ОС были изучены на предыдущих занятиях.

**D-Bus** – еще одна система межпроцессного взаимодействия (Interprocess Communication или ***IPC***).

**Суть проблемы** состоит в том, что приложения одного рабочего стола долж- ны тесно взаимодействовать между собой. Для этой цели применялись разные разработки, например, *DCOP в KDE или CORBA/Bonobo в GNOME*.

**В марте 2000 года,** *freedesktop.org* **-** инициативная группа по стандартизации различных графических сред пользователя для операционных систем POSIX, взя- лась за разработку стандартов (рекомендаций) таких сред.

**В частности**, эта группа определяет такие вещи, как формат ярлыков, взаи- модействие элементов рабочего стола при перетаскивании его элементов и другие.

**Требовалось также** организовать обмен сообщениями между приложениями двух разных сред.

**Более того**, *freedesktop.org* участвует в разработке ПО, например, полностью разрабатывает *графическую пользовательскую среду Enlightenment*.

**Раньше**, эта группа называлась «*X Desktop Group*» и, до сих пор, использует аббревиатуру *XDG*.

**Для решения этих задач** и был создан проект *D-Bus*. Реализация оказалась удачной и было принято решение проект *KDE 4* перевести на использование *D-Bus*.

**Таким образом**, *D-Bus* - это система межпроцессного взаимодействия, пре- доставляющая приложениям несколько шин для передачи сообщений и обеспечива- ющая беспроблемную связь десктопных приложений. Причем, поддерживается не только широковещательная рассылка сообщений (сигналов), но и удаленный вызов методов.

### Основные понятия шины D-Bus:

* + - **Системная шина (***System bus***)** — данная шина создается при старте сервера D-Bus. К ней подключены системные сервисы: HAL, NetworkManager, bluez,

WPA Supplicant и другие. На этой шине работают сервисы, которые нельзя отнести к какой-то определенной пользовательской сессии, и которые относятся к системе в целом.

* + - **Сессионная шина (***Session bus***)** — данная шина создается на каждый вход (login) пользователя в систему. К данной шине подключаются приложения

пользователя, прошедшего процедуру ***login***. Также через нее проходит общение программ запущенных в данной рабочей сессии пользователя.

* + - **Имя на шине (***Bus name***)**. Каждая программа, подключенная к шине, получает свое уникальное имя, которое начинается с двоеточия (":") и

представлено двумя числами, разделенными точками. Например, «:1.5».

* + - **Имя сервиса (***Service name***)**. В *D-Bus* — приложение может взять несколько дополнительных имен, чтобы другие программы могли с ней связываться. Имя сервиса представляет собой набор из символов, разделенных точками

("."). Пример: сервис имеет реальное имя ":1.5", и символьное представление

- "org.freedesktop.NetworkManager". Мы можем соединяться как с ":1.5", так и с "org.freedesktop.NetworkManager". Это - один и тот же сервис. Нужно ли создавать символьное представление имени клиент решает самостоятельно.

* + - **Путь к объекту (***Object path***)**. Путь к некоторому объекту внутри адресуемого сервиса: *"/org/freedesktop/NetworkManager/Device/eth0"*. Сервис

может обслуживать несколько объектов.

* + - **Интерфейс (***Interface***)** — каждый объект предоставляет доступ к некоторому набору методов и сигналов, каждый из которых называется интерфейсом. При этом, один объект может предоставлять несколько разных интерфейсов,

например: *"org.freedesktop.NetworkManager.Device.Wireless"*.

* + - **Метод (***Method***)** — некоторое действие, которое может производить объект в данной программе и на данном интерфейсе. Аналогично функциям на языке

***С***, метод может вернуть: *набор некоторых данных, код ошибки, либо может вообще не возвращать данных (void)*.

* + - **Сигнал (***Signal***)** — некоторое сообщение, которое распространяется среди всех программ, подписанных на этот сигнал на этом интерфейсе данного

объекта и данной программы. Сигналы могут содержать набор данных.

* + - **Сообщение (***Message***)** — каждая передача данных на шине представляется в виде сообщений. Они могут быть 4-х типов: *вызовы методов, сигналы, ре- зультаты методов, ошибки*.
    - **Прокси-объект (***Proxy object***)** - объект одного из языков программирования: *C++*, *Python*, *Java* и других, вызовы методов которых проецируются на вызо- вы методов шины ***D-Bus***.

**В общем случае**, взаимодействие между процессами, отображается схемой, показанной на рисунке 1.6. За более подробной информацией, следует обратиться к сайту: <http://www.freedesktop.org/wiki/Software/dbus/>

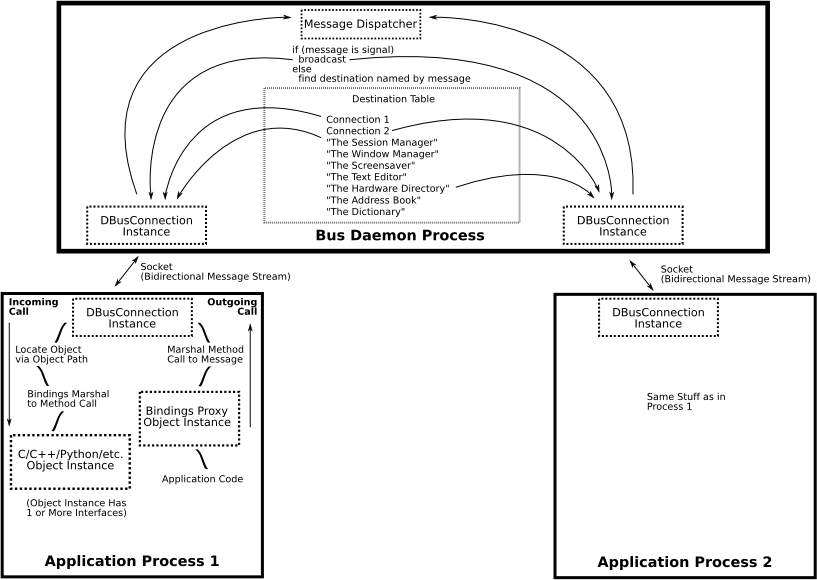


Рисунок 1.6 — Диаграмма взаимодействия процессов через шину D-Bus

**Взаимодействие** между приложениями происходит через серверное прило- жение *Bus Daemon Process*, которое реализовано в виде программы *dbus-daemon*.

**Процесс взаимодействия** обеспечивается специальным механизмом сокетов:

*Socket Bidirectional Message Stream*.

**Точками соединения** являются универсальные программные конструкции, которые обозначены как *DbusConnectionInstance*.

**Bus Daemon Process** обеспечивает диспетчеризацию именованных сообще- ний между приложениями, используя системные вызовы, реализованные в библио- теке *libdbus*.

**Взаимодействующие приложения** используют всего два типа вызовов:

*Incoming Call* и *Outgoing Call*.

## Бибиотека libdbus

**Низкоуровневое** взаимодействие приложений через шину D-Bus обеспечива- ется с помощью библиотеки *libdbus*.

**В настоящее время**, проект D-Bus реализован как *версия 1.х*.

**Поэтому**, для его идентификации используется суффикс -1.

**Например**, для ОС УПК АСУ:

* + - */etc/dbus-1* — директория с файлами конфигурации D-Bus;
    - */lib/libdbus-1.so* — ссылка на библиотеку *libdbus-1.so.3.14.8*

**Простейший пример** приложений, взаимодействующих с помощью библио- теки *libdbus*, рассмотрим для случая ПО *Skype*, показанный на рисунке 1.7:

* + - запущено приложение Skype, которое регистрируется на шине с дополни- тельным именем *com.Skype.API*;
    - другое приложение, «Клиент автоматизации Skype», посылает первому сооб- щение, вызывая его метод *Invoke()*.

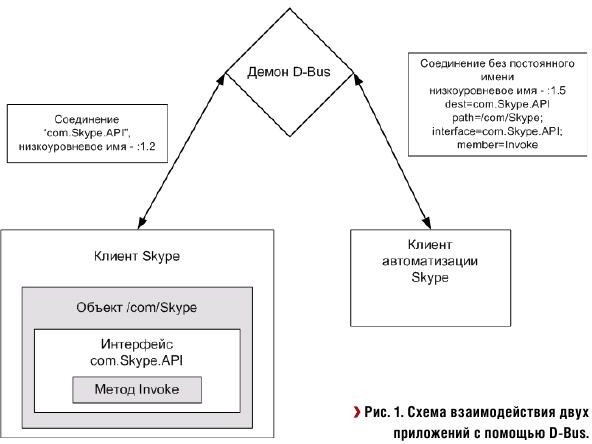


Рисунок 1.7 — Пример взаимодействия ПО Skype

На листинге 1.1, показана возможная реализация приложения «*Клиент авто- матизации Skype*».

**Замечание** Объекты ПО D-Bus не следует рассматривать в терминах ООП, поскольку понятие клас- са здесь не используется. Тем не менее, все объекты имеют имена и доступны, при ус- ловии ограничений доступа, из среды приложений ОС.

## Проекции ПО D-Bus на языки программирования

Приведенный выше пример использует низкоуровневый доступ к D-Bus че- рез библиотеку *libdbus*.

**Для эффективного использования** возможностей шины применяются *про- екции libdbus*, реализованные на различные языки.

**Широко известны** следующие проекции:

* + *GLib API* — библиотеки проекта GNU, которые можно найти по адресу: <https://developer.gnome.org/gio/stable/gdbus-convenience.html>;
  + *Python API* — который широко используется во всех дистрибутивах Linux: <http://dbus.freedesktop.org/doc/dbus-python/doc/tutorial.html>;
  + *Qt API* — которая составляет постоянную альтернативу библиотекам Glib API: <http://doc.qt.io/qt-5/qtdbus-index.html>.

**Для доступа** к шине *D-Bus* из языка *shell* используется набор утилит, которые следует внимательно изучить по руководству ***man***:

*dbus-cleanup-sockets* - используется для очистки директорий от остатков сокетов;

*dbus-daemon* - является демоном шины сообщений D-BUS;

*dbus-launch* - используется для запуска **dbus-daemon** из скрипта командной оболочки. Как правило, вызывается из скриптов, регистрирующих вход пользователей в систему;

*dbus-monitor* - используется для мониторинга сообщений, поступающих через шину сообщений D-BUS;

*dbus-send* - используется для отправки сообщения в шину сообщений D-BUS;

*dbus-uuidgen* - используется для создания или чтения универсального уникально- го идентификатора;

*libdbus-1.*

*{so,a}*

- содержит функции API, используемые демоном сообщений D- BUS. D-BUS является первой библиотекой, в которой предложены средства обмена сообщениями вида 1:1 между двумя любыми приложениями; *dbus-daemon* является приложением, использующим эту библиотеку для реализации демона шины сообщений;

*qdbus* - коммуникационный интерфейс для основанных на Qt API прило- жениях.

**В общем случае**, для работы с D-Bus необходимо хорошо знать приложение, с которым осуществляется взаимодействие, или написать такое приложение самому.

**Для практических целей**, очень удобна утилита *qdbus*, напимер, запущенная без параметров, она выведет все доступные имена D-Bus, как показано на рисунке 1.3.

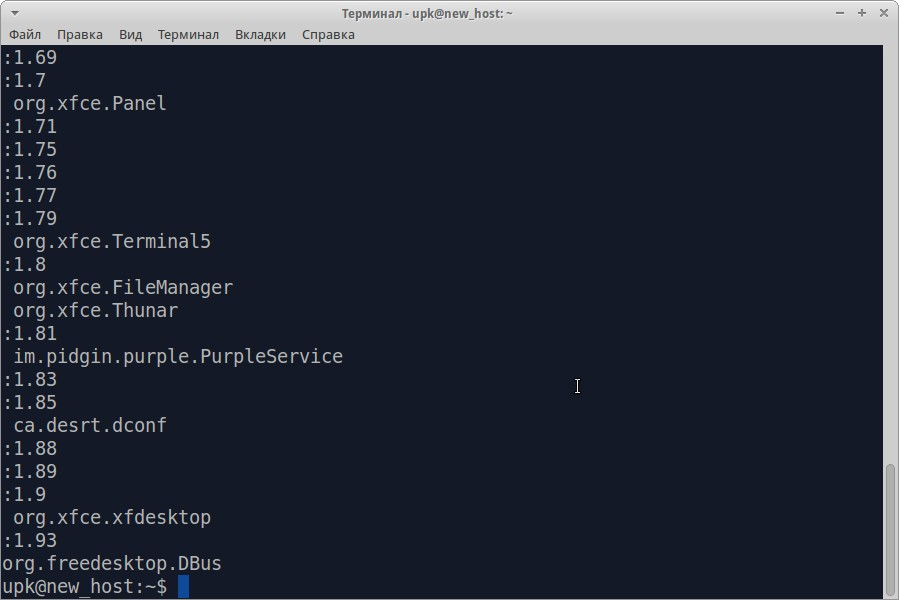


Рисунок 1.8 — Результат запуска qdbus без параметров