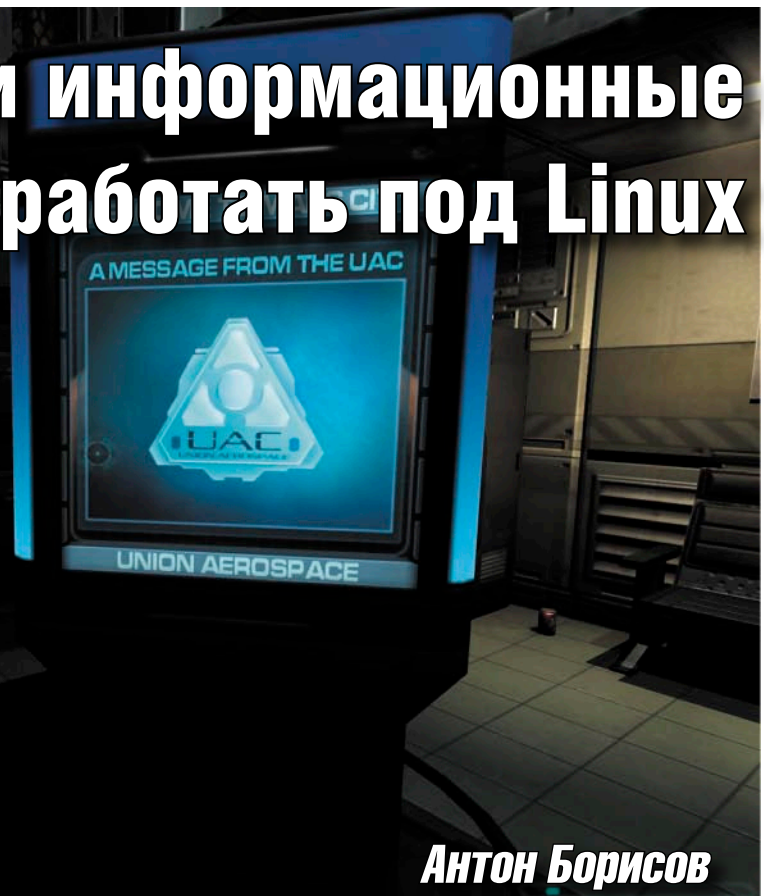


Могут ли информационные киоски работать под Linux



Антон Борисов

Как вы можете заметить, информационные киоски окружают нас повсюду: начиная от железнодорожных вокзалов и заканчивая кинотеатрами. Зачастую мы даже не подозреваем, на какой именно операционной системе они работают. Честно говоря, догадаться можно, было бы желание и наметанный глаз. И узнать знакомые очертания виджетов GDI и Internet Explorer от одной небольшой компании на северо-востоке США не составит большого труда. Правда ли, что информационные киоски несовместимы с Linux? Предлагаю проверить.

Первый вопрос, который задаст неискушенный читатель: «А что такое информационный киоск?» И будет совершенно прав в своем порыве докопаться до сути дела. Информационный киоск, если верить wikipedia, это – «автоматизированный программно-аппаратный комплекс, предназначенный для предоставления справочной информации. Их собирают на базе персонального компьютера, оснащенного сенсорным монитором и установленного в эргономичный вандалостойкий корпус. В отличие от обычного справочного киоска электронный информационный киоск работает автономно».

Достаточно сухо и не совсем информативно. На самом деле киоск – это обычная ПЭВМ, которую вставили в металлическую тумбу, добавили сен-

сорный экран, источник бесперебойного питания, по необходимости – мини-атюрный принтер и купюроприемник. Также в состав комплектации может входить GSM-модем, но это в случаях, когда невозможно дотянуть ethernet-кабель или же требуется организовать независимый интернет-канал.

Со всей указанной периферией мы сталкиваемся и в обычной жизни, дома и на работе, кроме, пожалуй, купюроприемников и сенсорных экранов. Если первые активно используются в платежных системах и банкоматах, и на которых мы не будем пока акцентировать внимание, то на сенсорные экраны следует обратить особое внимание. Почему? Да потому, что они будут выполнять роль как клавиатуры, так и мышки, т.е. это единственный источник ввода информации для киоска.

Концепция и первая реализация сенсорных экранов была разработана еще в начале 70-х годов прошлого века. К сегодняшнему дню существует несколько видов сенсорных экранов: матричные, резистивные, ёмкостные, проекционно-ёмкостные, сенсорные экраны на поверхностно-акустических волнах, инфракрасные и оптические сенсорные экраны. Для киосков применяются только некоторые из перечисленных: ёмкостные и проекционно-ёмкостные, а также экраны на основе поверхностно-акустических волн. Давайте рассмотрим, в чем их принципиальное различие.

Ёмкостный сенсорный экран

В ёмкостных экранах применяется принцип: предмет с большей ёмкос-

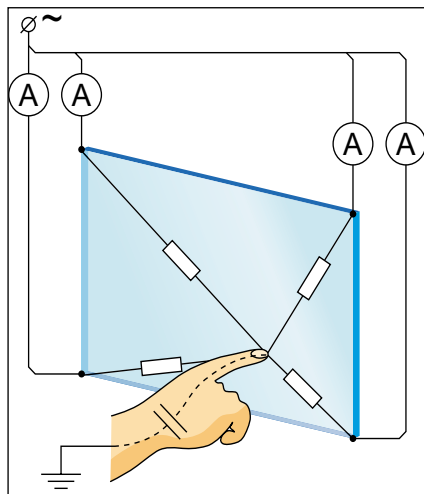


Рисунок 1. Схема работы ёмкостного сенсорного экрана

тью лучше проводит переменный ток. Из курса физики вы помните, что шансов поразиться электрическим током с сухими ладонями меньше, нежели их предварительно намочить (автор надеется, что читатели не будут проверять данное утверждение на практике). Хотя, строго говоря, любой элемент, обладающий электрической ёмкостью, проводит переменный ток. Просто сила тока будет изменяться в зависимости от ёмкости и частоты тока. Экран же представляет собой стеклянную панель, покрытую проводящим материалом. Электроды, расположенные по углам экрана, подают на проводящий слой небольшое переменное напряжение (одинаковое для всех углов). При касании экрана пальцем или другим проводящим предметом появляется утечка тока. При этом чем ближе палец к электроду, тем меньше сопротивление цепи, а значит, сила тока больше. Ток во всех четырёх углах регистрируется датчиками и передаётся в контроллер, вычисляющий координаты точки касания.

В более ранних моделях ёмкостных экранов применялся постоянный ток – это упрощает конструкцию, но при плохом контакте пользователя с землёй приводит к сбоям.

Ёмкостные сенсорные экраны надёжны (порядка 200 млн нажатий), не пропус-

кают жидкости и отлично терпят непроводящие загрязнения. Прозрачность на уровне 90%. Впрочем, проводящее покрытие всё ещё уязвимо. Поэтому ёмкостные экраны широко применяются в автоматах, установленных в охраняемом помещении. Не реагируют на руку в перчатке (см. **рис. 1**).

Проекционно-ёмкостный сенсорный экран

На внутренней стороне экрана нанесена сетка электродов. Электрод вместе с телом человека образует конденсатор; электроника измеряет ёмкость этого конденсатора (подаёт импульс тока и измеряет напряжение).

Прозрачность таких экранов до 90%, температурный диапазон чрезвычайно широк. Очень долговечны (узкое место – сложная электроника, обрабатывающая нажатия). На таких экранах может применяться стекло толщиной вплоть до 18 мм, что позволяет позиционировать их как весьма вандалоустойчивые. Хотя против такого элегантного и простого инструмента, как лом, все равно не найдется симметричный ответ. На непроводящие загрязнения не реагируют, проводящие легко подавляются программными методами. Поэтому проекционно-ёмкостные сенсорные экраны применяются в автоматах, устанавливаемых на улице. Реагирует на руку в перчатке. Невысокая точность дополняется параллаксом от толстого вандалоустойчивого стекла.

Отличают нажатие рукой от нажатия проводящим пером. В некоторых моделях поддерживается обработка событий от нажатий несколькими пальцами (см. **рис. 2**).

Сенсорный экран на основе ПАВ

Экран представляет собой стеклянную панель с пьезоэлектрическими преобразователями (ПЭП), находящимися по углам. По краям панели находятся отражающие и принимающие датчики. Принцип действия такого экрана заключается в следующем. Специальный контроллер формирует высокочастотный сигнал (ультразвук) и посылает его на ПЭП. ПЭП преобразует этот сигнал в ПАВ, а отражающие датчики его соответственно отражают. Эти отраженные волны принимаются

соответствующими датчиками и посылаются на ПЭП. ПЭП в свою очередь принимают отраженные волны и преобразовывают их в электрический сигнал, который затем анализируется с помощью контроллера. При прикосновении к экрану происходит изменение в распространяющихся волнах (характер прохождения ультразвука), что и фиксируется принимающими датчиками.

Реагирует на касание предметом, способным поглотить волну (палец, рука в перчатке, пористая резина). Реагирует на силу нажатия. Высокая надёжность. Не реагирует на предмет, не поглощающий ультразвук (перо, карточка). Любой посторонний предмет (например, прилепленная жвачка) полностью блокирует работу экрана. Не удаётся надёжно загерметизировать края экрана (см. **рис. 3**).

Предельно высокая прозрачность (не нужны никакие электроды; мало того, ультразвук можно возбуждать прямо на экране). Экраны применяют только в охраняемом помещении.

По большому счету сенсорный экран – это чувствительная накладка на TFT-панель. Производитель информационного киоска берет тумбу, закрепляет на ней обычный TFT-экран или TFT-телевизор и накладывает сверху сенсорный экран. Все вместе органично выглядит как единое целое.

По типу подключения различают экраны как с USB-подключением, так и с подключением по порту RS232. Принципиальных различий нет – требования по скорости настолько низкие, что можно было бы оставить только RS232. Если бы не одно «но». Современные ПЭВМ все чаще делают без COM-портов, но с увеличенным количеством USB-портов. Поэтому для таких случаев производители сенсорных экранов предлагают экраны с USB-подключением. И похоже, это становится уже тенденцией.

Разрешающая способность сенсорных экранов, как правило, не больше 4096 точек по горизонтали и по вертикали. Соответственно, графический режим вывода на TFT-панель не может превышать данную цифру. Впрочем, TFT-матрицы видео-экрана в киосках не превышают 17, 19 дюймов. И графические режимы там, как правило, стандартны – не более 1280x1024 то-

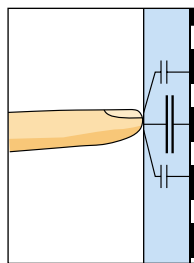


Рисунок 2. Схема работы проекционно-ёмкостного сенсорного экрана

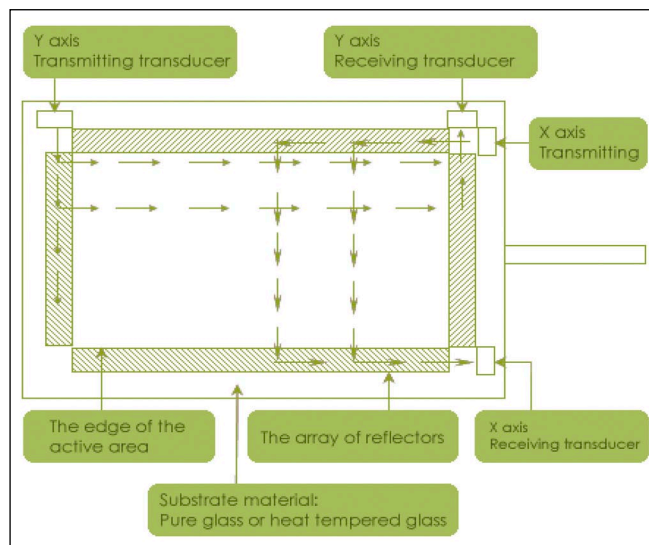


Рисунок 3. Схема работы сенсорного экрана на основе ПАВ

чек. Сравните с 4096x4096 – разрешающей способностью сенсорного экрана.

Программное обеспечение информационных киосков

Очень часто, а здесь надо читать как «практически всегда», в качестве операционной системы используется Windows. Либо в виде настольной серии (Windows 2000/Windows XP), либо для встраиваемых систем (Windows XP Embedded), либо вообще Windows CE. И поверх операционной системы работает Win32-приложение.

Применима ли здесь другая расстановка сил? Ответ – да. Для того чтобы киоск заработал, нам нужны:

1. Операционная система.
2. Получение событий на нажатие от сенсорного экрана.
3. Взаимодействие с периферией.
4. Приложение, выполняющее обработку информации.

В качестве пункта 1 была выбрана Linux, а точнее Thinstation – по большому счету это конструктор дистрибутива. С его помощью можно на основной системе приготовить образы ISO (загрузка через CD-ROM), SysLinux (загрузка, например, через USB-брелок), PXE (сетевая загрузка). На вариант сетевой загрузки мы и будем ориентироваться. В качестве основной системы, где будет подготавливаться сетевой образ для киоска, выступает openSUSE 10.3.

Для пунктов 2 и 3 необходимо найти драйверы устройств. На первый взгляд кажется, что задача невыполнимая. Однако это не так. Большая часть производителей уже давно на своих сайтах разместила данное программное обеспечение. Надо только внимательно искать. Если у вас используется сенсорный экран от законодателя мод – компании Elo TouchSystems, то здесь еще проще. Драйвер для этого производителя можно найти в любом современном дистрибутиве.

Ну а что выбрать на роль приложения? Писать что-то специфическое не особо хотелось, поэтому я решил, что в качестве приложения достойно выступит следующая «команда»: opera, nginx, fastcgi и тестовое приложение. Opera будет отображать веб-страницы с неким содержимым, nginx – перенаправлять запросы к fastcgi, ну а fastcgi – запускать тестовое

приложение, которое в свою очередь будет выводить на принтер некое содержимое, выбранное на веб-странице. По мере усложнения можно также добавить и выборку из SQL-базы. Звучит сложно, но реализация проста (см. рис. 4).

Посмотрим, каково аппаратное обеспечение типичного информационного киоска. В моем случае киоск называется SFOUR Tadpole.

```
kiosk:~# lspci
```

```
00:00.0 Host bridge: Intel Corporation 82945G/GZ/P/PL
Memory Controller Hub (rev 02)
00:02.0 VGA compatible controller: Intel Corporation 82945G/GZ
Integrated Graphics Controller (rev 02)
00:1b.0 Audio device: Intel Corporation 82801G (ICH7 Family)
High Definition Audio Controller (rev 01)
00:1c.0 PCI bridge: Intel Corporation 82801G (ICH7 Family)
PCI Express Port 1 (rev 01)
00:1c.1 PCI bridge: Intel Corporation 82801G (ICH7 Family)
PCI Express Port 2 (rev 01)
00:1d.0 USB Controller: Intel Corporation 82801G (ICH7 Family)
USB UHCI Controller #1 (rev 01)
00:1d.1 USB Controller: Intel Corporation 82801G (ICH7 Family)
USB UHCI Controller #2 (rev 01)
00:1d.2 USB Controller: Intel Corporation 82801G (ICH7 Family)
USB UHCI Controller #3 (rev 01)
00:1d.3 USB Controller: Intel Corporation 82801G (ICH7 Family)
USB UHCI Controller #4 (rev 01)
00:1d.7 USB Controller: Intel Corporation 82801G (ICH7 Family)
USB2 EHCI Controller (rev 01)
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev e1)
00:1f.0 ISA bridge: Intel Corporation 82801GB/GR (ICH7 Family)
LPC Interface Bridge (rev 01)
00:1f.1 IDE interface: Intel Corporation 82801G (ICH7 Family)
IDE Controller (rev 01)
00:1f.2 IDE interface: Intel Corporation 82801GB/GR/GH
(ICH7 Family) SATA IDE Controller (rev 01)
00:1f.3 SMBus: Intel Corporation 82801G (ICH7 Family)
SMBus Controller (rev 01)
02:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd.
RTL8101E PCI Express Fast Ethernet controller (rev 01)
```

```
kiosk:~# cat /proc/cpuinfo
```

```
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 15
model          : 4
model name     : Intel(R) Celeron(R) CPU 2.80GHz
stepping       : 9
cpu MHz        : 2794.238
cache size     : 256 KB
fdiv_bug       : no
hlt_bug        : no
f00f_bug       : no
coma_bug       : no
fpu            : yes
fpu_exception  : yes
cpuid level    : 5
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep
                mtrr pge mca cmov pat pse36 clflush dts acpi mmx
                fxsr sse sse2 ss ht tm pbe lm constant_tsc pni
                monitor ds_cpl tm2 cid cx16 xtpr lahf_lm
bogomips       : 5593.32
clflush size   : 64
```

```
kiosk:~# free
```

	total	used	free	shared	buffers
Mem:	906896	120396	786500	0	38584
Swap:	0	0	0		
Total:	906896	120396	786500		

Для отображения и предоставления информации вычислительной мощности более чем достаточно, впрочем, как и ОЗУ (в киоск вставлена планка на 1 Гб, часть используется для нужд интегрированного видео).

Подключаем сенсорный экран к машине с openSUSE и в dmesg наблюдаем появление информации:

```
usb 6-2: new full speed USB device using uhci_hcd and address 2
usb 6-2: new device found, idVendor=04e7, idProduct=0007
usb 6-2: new device strings: Mfr=1, Product=2, SerialNumber=3
usb 6-2: Product: Elo TouchSystems IntelliTouch 2500U
usb 6-2: Manufacturer: Elo TouchSystems, Inc.
usb 6-2: SerialNumber: 07A27613
usb 6-2: configuration #1 chosen from 1 choice
input: Elo TouchSystems, Inc. Elo TouchSystems IntelliTouch 2500U
as /class/input/input5
input: USB HID v1.00 Pointer [Elo TouchSystems, Inc. Elo
TouchSystems IntelliTouch 2500U] on usb-0000:00:1d.0-2
```

Убедились, что проданный нам киоск действительно содержит сенсорный экран, что указан в проспекте на изделие.

Загружаем дистрибутивы Opera 9.52 [6] (как Static QT package) и Thinstation-src2.3beta1 [7] (раздел Sources) вместе с Thinstation-2.3beta2 (раздел Main Distribution).

Thinstation-2.3beta2 – это конструктор, где из множества компонентов собирается загрузочный образ, а Thinstation-src2.3beta1 – среда, где мы будем компилировать те компоненты, что отсутствуют в конструкторе, а именно nginx и fastcgi. Распаковываем конструктор Thinstation на рабочую машину, например в директорию /media/hyperspace/ts/:

```
# tar xjvf Thinstation-2.3beta2.tar.gz -C /media/hyperspace/ts/
# cd /media/hyperspace/ts/Thinstation-2.3
```

Отредактируем файл build.conf, в котором содержится описание тех компонентов, которые используются для функционирования киоска. Моя конфигурация выглядит так:

```
module pcm
module serial
module acpi
module pcspkr
```

```
module agpgart
module ati-agp
module intel-agp
module r8169

module snd-hda-intel
module snd-intel8x0

module usb-hid

module elo

package rdate
package xorg6vnc
package xorg6-i810
package keymaps-en_us
```

```
package sshd
package scp
```

```
package opera.kiosk
```

```
param rootpasswd RootPass
param xorgvncpasswd VNCPass
param bootlogo true
param bootresolution 1024x768
param defaultconfig thinstation.conf.buildtime
param basename thinstation
param basepath . files
param baseurl http://thinstation.sourceforge.net
param localpkgs false locally
param fulllocales false
param icaencryption false
param bootverbosity 1
```

В данном случае при загрузке будет выставлен режим 1024x768 и в процессе загрузки будет красоваться картинка (параметр bootlogo подразумевает включение режима framebuffer). Для доступа к киоску по протоколу SSH необходимо будет авторизоваться в качестве пользователя root с паролем RootPass. Для доступа по VNC-протоколу следует использовать пароль VNCPass.

Второй файл, в который надо внести изменения, – thinstation.conf.buildtime. В нем определяется конфигура-

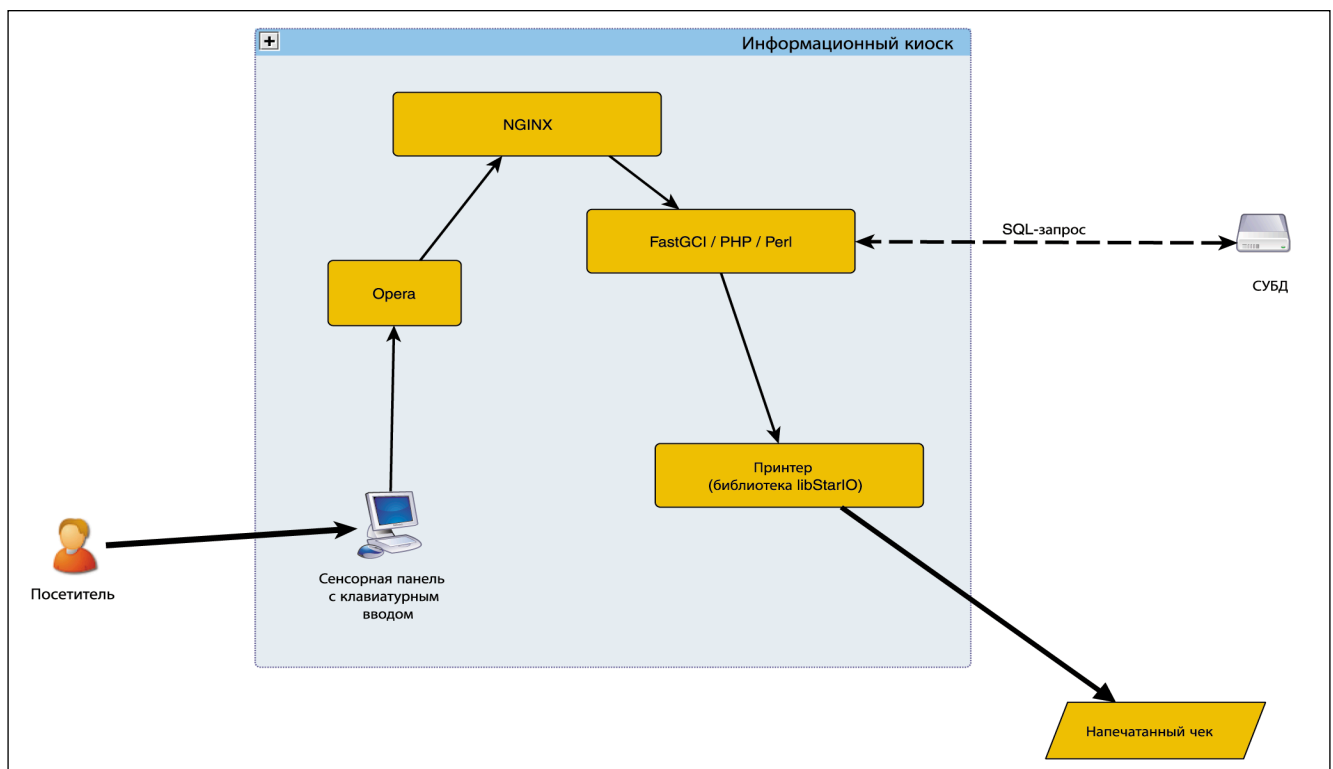


Рисунок 4. Схема взаимодействия элементов киоска

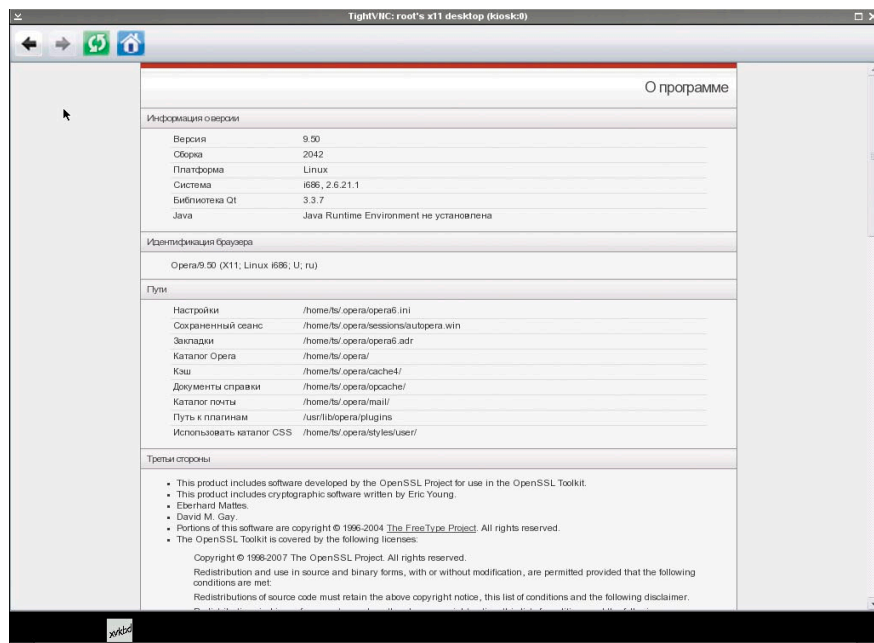


Рисунок 5. Экран киоска передается по VNC-протоколу

ция компонентов X-сервера, например, размещение оконного менеджера на X-экране, размер самого экрана, чувствительность мышки, а также некоторые другие параметры. Его содержимое такое:

```
AUDIO_LEVEL=67
KEYBOARD_MAP=ru RU.UTF-8
TIME_ZONE="UTC+10:00"
USB_ENABLED=On
DAILY_REBOOT=On
CUSTOM_CONFIG=Off
RECONNECT_PROMPT=Off

SCREEN=0
WORKSPACE=1
AUTOSTART=Off
ICONMODE=AUTO
ICONHOTPLUG=YES

SESSION_0_ICON=On
SESSION_0_TYPE=opera
SESSION_0_AUTOSTART=On

SCREEN_RESOLUTION="1024x768 | 800x600 | 640x480 | *"
SCREEN_COLOR_DEPTH="16 | 8 | *"
SCREEN_HORIZSYNC="56-86 | *"
SCREEN_VERTREFRESH="30-83 | 60 | 56 | 70 | 72 | 75"
SCREEN_BLANK_TIME=0
SCREEN_STANDBY_TIME=0
SCREEN_SUSPEND_TIME=0
SCREEN_OFF_TIME=0
MOUSE_RESOLUTION=100

X_DRIVER_OPTION1="swcursor On"
X_SERVERFLAGS_OPTION1="DontVTSwitch On"
X_SERVERFLAGS_OPTION2="DontZap On"
X_MONITOR_OPTION1="DPMs On"
X_MONITOR_MODELINE="'1680x1050" 146.2 1680 1784 1960 1
2240 1050 1053 1059 1089'

NET_HOSTNAME=kiosk

NET_USE_DHCP=On
NET_MASK=255.255.255.0

NET_DNS1=1.2.3.4
```

Обратить внимание следует на параметры:

```
SESSION_0_ICON=On
SESSION_0_TYPE=opera
SESSION_0_AUTOSTART=On
```

Они означают – запустить X-Server и в качестве главного приложения использовать Opera.

Если собираетесь использовать DNS, то параметр NET-DNS1 должен содержать правильный адрес:

```
NET_DNS1=1.2.3.4
```

Видеорежим, который будет установлен на киоске, – 1024x768. Если X-Server не сможет его установить, то будут опробованы режимы 800x600 и 640x468.

Браузер Opera я распаковал в поддиректорию packages/opera.kiosk/.

Со структурой зависимостей пакета можно ознакомиться в примере – директория packages/template/.

Теперь подключим взаимодействие X-Server с сенсорным экраном, используем для этого файл packages/

xorg6vnc/lib/X11/vnc_keyboard.tpl:

```
Section "InputDevice"
    # vncKeyboard: keyboard actions from vnc
    Identifier "vncKeyboard"
    Driver "rfbkeyb"
EndSection

Section "InputDevice"
    # vncMouse: mouse actions from vnc
    Identifier "vncMouse"
    Driver "rfbmouse"
EndSection

Section "InputDevice"
    Identifier "Touchscreen0"
    Driver "evtouch"
    Option "Device" "/dev/input/event3"
    Option "TouchMode" "0"
    Option "AlwaysCore"
    Option "screenno" "0"

    Option "MinX" "0"
    Option "MaxX" "4000"
    Option "MinY" "0"
    Option "MaxY" "4000"

    Option "SwapX" "1"
EndSection
```

Зачем нам требуется подключать VNC-поддержку? Во-первых, для контроля работы сенсорного экрана. Если киоск работает продолжительное время не выключаясь, то может возникнуть ситуация, что экран перестает обрабатывать сигналы. Проще говоря, «завис». Тогда и зайти по протоколу VNC не удастся. Хотя система в целом продолжает функционировать. Связано ли это с определенным модельным рядом, свойством драйвера или экрана – трудно сказать. Чтобы компенсировать проблему, и предусмотрен вариант с удаленным заходом по SSH. Необходимо только перезапустить X-Server. Во-вторых, можно удостовериться, что пользователь, подошедший к киоску, занимается правильным делом, а не старается выйти, например, в Интернет или запустить пасьянс – бывают разные случаи.

Устройство, которое посылает сигналы от сенсорного экрана, обрабатывается драйвером `evtouch` и располагается по адресу `/dev/input/event3`. Архив драйвера я взял с ресурса [8] и поместил нужный мне драйвер в конструкторе Thinstation как файл: `packages/xorg6/lib/X11/modules/input/evtouch_drv.so`.

Этого достаточно, можно приступить к сборке загрузочного образа:

```
# cd /media/hyperspace/ts/Thinstation-2.3
# ./build
```

Сборка занимает несколько минут, и результат можно найти в директории `/media/hyperspace/ts/Thinstation-2.3/boot-images/pxe`.

Остается настроить DHCP- и TFTP-сервер, положить полученные файлы в директорию, например `/tftpboot`, и запустить информационный киоск на загрузку с PXE-источника.

При правильном составлении конфигурации компонентов и самого браузера Opera вы должны лицезреть на сенсорном экране похожую картинку (см. **рис. 5**).

В моем случае были произведены оптимизация и настройка компонентов Opera, так что не особенно удивляйтесь, если у вас будут видны и меньшие пиктограммы, и строка адреса. У вас так и должно быть.

Если внимательно присмотреться, то в правом нижнем углу можно видеть виртуальную клавиатуру (приложение `xvkbd`, которое я использую отдельно для ввода символов).

Если у вас сенсорный экран обрабатывает нажатия, то клавиатура на экране тоже может пригодиться. В противном случае надо разбираться, почему X-Server не принимает сигналы от драйвера сенсорного экрана.

Кстати, не следует рассчитывать, что с помощью сенсорного экрана удастся эмулировать двухкнопочную мышь и получить сигнал от правой кнопки. В большинстве случаев это может быть невозможно, т.к. драйвер сенсорного экрана, скорее всего, такую ситуацию обработать будет не в состоянии.

Проверим, как зарегистрировались обработчики `input`-событий в тонком клиенте:

```
kiosk:~# cat /proc/bus/input/handlers
```

```
N: Number=0 Name=kbd
N: Number=1 Name=mousedev Minor=32
N: Number=2 Name=evdev Minor=64
```

```
kiosk:~# cat /proc/bus/input/devices
```

```
I: Bus=0019 Vendor=0000 Product=0002 Version=0000
N: Name="Power Button (FF)"
P: Phys=button_power/button/input0
S: Sysfs=/class/input/input0
H: Handlers=kbd event0
B: EV=3
B: KEY=100000 0 0 0

I: Bus=0019 Vendor=0000 Product=0001 Version=0000
N: Name="Power Button (CM)"
P: Phys=PNP0C0C/button/input0
S: Sysfs=/class/input/input1
H: Handlers=kbd event1
B: EV=3
B: KEY=100000 0 0 0
```

```
I: Bus=0010 Vendor=001f Product=0001 Version=0100
N: Name="PC Speaker"
P: Phys=isa0061/input0
S: Sysfs=/class/input/input2
H: Handlers=kbd event2
B: EV=40001
B: SND=6
```

```
I: Bus=0003 Vendor=04e7 Product=0007 Version=0100
N: Name="Elo TouchSystems, Inc. Elo TouchSystems IntelliTouch 2500U"
P: Phys=usb-0000:00:1d.0-1/input0
S: Sysfs=/class/input/input3
H: Handlers=event3
B: EV=b
B: KEY=10000 0 0 0 0 0 0 0
B: ABS=100 3
```

Если вы не видите названия сенсорного экрана, то это означает, что модуль, отвечающий за обмен данными с сенсорным экраном, не загружен.

```
kiosk:~# lsmod | grep elo
```

```
elo                6336  0
```

У меня он загружен. В случае когда ничего похожего не наблюдается, необходимо форсированно его загружать, например, через скрипт запуска X-Server либо в одной из служб, например `ssh`.

Второй интересный момент. Ваш сенсорный экран может общаться не по USB-шине, а через порт RS232. Так, например, работают некоторые экраны производства Generic Touch. Тогда в файл `packages/xorg6vnc/lib/X11/vnc_keyboard.tpl` надо внести следующие строки:

```
Section "InputDevice"
    Identifier "Touchscreen0"
    Driver "gentouch"
    Option "Device" "/dev/ttyS0"
    Option "TouchMode" "0"
    Option "AlwaysCore"
    Option "screenno" "0"
    Option "MinX" "150"
    Option "MaxX" "3300"
    Option "MinY" "970"
    Option "MaxY" "3470"
    Option "UntouchDelay" "3"
    Option "ReportDelay" "1"
EndSection
```

В этом случае драйвер называется `gentouch`, и события поступают с устройства `/dev/ttyS0`, т.е. с того порта, куда подключили экран.

Если и в этом случае выявились проблемы, то в качестве источника выявления ошибок обратитесь к файлу `/var/log/Xorg.0.log`. Здесь вы найдете не только причину ошибки, но и массу интересной информации, вплоть до производителя TFT-матрицы экрана информационного киоска.

Следует дальше. Загрузка работает, сенсорный экран откликается. Остается только подготовить HTML-страницу с нужными параметрами и попробовать их распечатать на принтере киоска. Звучит сложно? Отнюдь. Распаковываем исходные коды конструктора Thinstation:

```
# tar xjvf thinstation_src-2.3beta1.tar.bz2 -J
-C /media/hyperspace/ts/
# cd /media/hyperspace/ts/thinstation_src-2.3
# ./RUNME
```

Появится приглашение от Thinstation. Мы в самом сердце среды компиляции.

You have entered the Thinstation Build Environment

sh-3.1#

Заберем веб-сервер nginx [9] и распакуем в /media/hyperspace/ts/thinstation_src-2.3/source. Почему именно он, а не Apache? Из-за своей легковесности. Да, сам он не умеет обрабатывать CGI, но его легко научить.

```
sh-3.1# cd /source/nginx-0.6.32/
sh-3.1# ./configure --prefix=/usr/local/nginx
sh-3.1# make && make install
```

Открываем новое окно терминала и в нем будем производить дополнительные манипуляции.

Добавляем в описание конструктора Thinstation (файл build.conf) новый пакет под названием nginx и копируем в /media/hyperspace/ts/Thinstation-2.3/packages/nginx/ всю директорию из /media/hyperspace/ts/thinstation_src-2.3/usr/local/nginx.

Затем создадим файл fastcgi_params в директории nginx/conf/:

```
fastcgi_param QUERY_STRING      $query_string;
fastcgi_param REQUEST_METHOD    $request_method;
fastcgi_param CONTENT_TYPE      $content_type;
fastcgi_param CONTENT_LENGTH    $content_length;

fastcgi_param SCRIPT_NAME       $fastcgi_script_name;
fastcgi_param REQUEST_URI       $request_uri;
fastcgi_param DOCUMENT_URI      $document_uri;
fastcgi_param DOCUMENT_ROOT     $document_root;
fastcgi_param SERVER_PROTOCOL   $server_protocol;

fastcgi_param GATEWAY_INTERFACE CGI/1.1;
fastcgi_param SERVER_SOFTWARE   nginx/$nginx_version;

fastcgi_param REMOTE_ADDR       $remote_addr;
fastcgi_param REMOTE_PORT       $remote_port;
fastcgi_param SERVER_ADDR       $server_addr;
fastcgi_param SERVER_PORT       $server_port;
fastcgi_param SERVER_NAME       $server_name;

# PHP only, required if PHP was built
# with --enable-force-cgi-redirect
fastcgi_param REDIRECT_STATUS 200;
```

Конфигурационный файл nginx должен выглядеть примерно так:

```
1) user nobody;
2) worker_processes 1;

3) events {
4)     worker_connections 1024;
5) }

6) http {
7)     include mime.types;
8)     default_type application/octet-stream;

9)     #access_log logs/access.log main;

10)    #sendfile        on;
11)    #tcp_nopush      on;

12)    keepalive_timeout 65;

13)    #gzip on;

14)    server {
15)        listen      80;
16)        server_name localhost;
17)        location / {
18)            root      html;
19)            index      index.html index.htm;
20)        }
```

```
21)         error_page 500 502 503 504 /50x.html;
22)         location = /50x.html {
23)             root      html;
24)         }

24)         location ~ \.cgi$ {
25)             root      html;
26)             fastcgi_pass 127.0.0.1:9000;
27)             fastcgi_index index.cgi;
28)             fastcgi_param SCRIPT_FILENAME _
                /scripts$fastcgi_script_name;
29)             include     fastcgi_params;
30)         }
31)     }
32) }
```

Небольшой комментарий. Сервер работает от имени пользователя nobody (строка 1). Чтобы он действительно функционировал, а не висел в памяти, надо закомментировать (строка 10) использование функции sendfile():

```
#sendfile        on;
```

Параметр sendfile в nginx используется для передачи файлов больших размеров (графические файлы), при этом используется функция ядра sendfile(). Это актуально для условий хостинга, когда экономятся кванты времени на обработку тысячи запросов в секунду, за счет передачи управления непосредственно ядру. В данном случае в ядре Thinstation функция sendfile() не включена, поэтому nginx не будет «отдавать» вообще никакие файлы, даже HTML.

Затем следует включить обработчик CGI-скриптов, строки 24-30. На порту 9000 локального хоста, а именно информационного киоска, должен крутиться CGI-wrapper, который и будет передавать управление FastCGI, который в свою очередь и занимается выполнением CGI-скриптов.

В качестве данного подмастерья идет wrapper из комплекта lighttpd [10] под названием spawn-fcgi. Распаковываем lighttpd и компилируем его по аналогии с nginx. Устанавливать его не надо, а требуется только переписать файл /source/lighttpd-1.4.19/src/spawn-fcgi в пакет nginx.

Остается собрать наше тестовое приложение печати в формате FastCGI и запустить все это вместе. Забираем FastCGI [11], распаковываем в директорию /source/fcgi-2.4.0/, компилируем и устанавливаем и затем приступаем к взаимодействию с принтером.

Используемый в сегодняшнем примере информационный киоск комплектуется принтером TUP900 производства Star Micronics. Он подключается через порт RS232, поэтому для изучения его управления я воспользовался примером, что шел в комплекте с его драйверами [12]. Правда, слегка модифицировав, чтобы оно было больше похоже на CGI-приложение. Драйверы ставить не нужно, т.к. печатью управлять будет наше приложение. Для более сложных вариантов, требующих растровой печати, поставляются драйверы для CUPS-подсистемы. Но в Thinstation не был обнаружен простой способ создать работающий CUPS-демон. И по большому счету я не увидел преимущества растровой печати, кроме как явного замедления печати.

```
(1)#include <stdio.h>
(2)#include <string.h>
(3)#include <sys/time.h>

(4)#include <stario/stario.h>
(5)#include <stdlib.h>
```

```

(6)#include <fcgi_stdio.h>

(7)void main(void)
(8){
(9)    long res = STARIO_ERROR_SUCCESS;
(10)    int i = 0;

(11)    int count = 0;
(12)    char BigText[1024];
(13)    char BigString[1024];
(14)    int len = 0;

(15)    char PortID[]      = "/dev/ttyS0";
(16)    char PortParams[]  = "9600,none,8,1,hdwr";

(17)    char Month[32];
(18)    char Year[16];

(19)    char* header[] = {
(20)        "+-----+\\n",
(21)        "| CP1251: Информационный заголовок |\\n",
(22)        "| CP1251: Выбирается год и месяц |\\n",
(23)        "+-----+\\n"
(24)    };

(25)    /*----- Начало CGI-цикла -----*/

(26)    while(FCGI_Accept() >= 0)
(27)    {
(28)        len = atoi(getenv("CONTENT_LENGTH"));
(29)        gets(BigText);

(30)        /*----- Передать браузеру html-header -----*/
(31)        printf("Content-type: text/html\\r\\n"
(32)            "\\r\\n"
(33)            "<title>Printer response page \\n"
(34)            "</title>"
(35)            "<h1>Printer response page</h1>"
(36)            "Request number %d running on \\n"
(37)            "host <i>%s</i><br>",
(38)            ++count, getenv("SERVER_NAME"));

(39)        res = writePort(PortID, "\\x1b", 1,
(40)            sizeof("\\x1b") - 1);
(41)        res = writePort(PortID, "\\x1d", 1,
(42)            sizeof("\\x1d") - 1);
(43)        res = writePort(PortID, "\\x74", 1,
(44)            sizeof("\\x74") - 1);
(45)        res = writePort(PortID, "\\x22", 1,
(46)            sizeof("\\x22") - 1);

(47)        /*----- Напечатать заголовок header -----*/
(48)        for( i = 0; i < 4; i++ )
(49)            res = writePort(PortID, header[i], 1,
(50)                strlen(header[i]) );

(51)        /*----- Выбрать подстроку 'month' -----*/
(52)        strcpy(BigString, 1,
(53)            strstr(BigText, "month"));
(54)        for( i = 0; i < strlen(BigString); i++ )
(55)            if(BigString[i] == '&') break;

(56)        BigString[i] = '\\0';
(57)        strncpy(Month, BigString + 1,
(58)            (strlen("month") + 1), 1,
(59)            i - strlen("month"));

(60)        /*----- Выбрать подстроку 'year' -----*/
(61)        strcpy(BigString, 1,
(62)            strstr(BigText, "year"));
(63)        for( i = 0; i < strlen(BigString); i++ )
(64)            if(BigString[i] == '&') break;

(65)        BigString[i] = '\\0';
(66)        strncpy(Year, BigString + 1,
(67)            (strlen("year") + 1), 1,
(68)            i - strlen("year"));

```

```

(58)        /* Напечатать выбранные год и месяц */
(59)        sprintf(BigString, "Выбранный год: \\n"
(60)            "%s, месяц: %s\\n", Year, Month);
(61)        res = writePort(PortID, BigString, 1,
(62)            strlen(BigString));

(63)        /*----- Обрезать бумагу -----*/

(64)        res = writePort(PortID, "\\x1b""d3", 1,
(65)            sizeof("\\x1b""d3") - 1);

(66)        res = closePort(PortID);

(67)        printf("\\nInput buffer length is: \\n"
(68)            "%i symbols", len);
(69)        printf("\\nBigText: %s", BigText);
(70)        printf("\\nYear: %s", Year);
(71)        printf("\\nMonth: %s", Month);

(72)        /*----- Конец CGI-цикла -----*/
(73)    }
(74) }

```

В строке 4 подключаем заголовочный файл от библиотеки принтера, а в строке 6 – от библиотеки FastCGI. Изначально делаем буфер BigText и BigString достаточно большими, чтобы вместить запрос POST-запрос с HTML-страницы (строки 12-13). Принтер TUP900 подключен к порту COM1 (строка 15) и работает с параметрами, указанными в строке 16. Максимальная скорость, на которую его можно перевести с помощью переключателей, составляет 38400 бод. Впрочем, и при используемых по умолчанию параметрах скорострельность высокая – печать чека осуществляется менее чем за секунду.

Цикл программы в CGI-стиле ограничен строками 26-69. В нем мы получаем содержимое POST-запроса (строка 29), выводим ответ браузеру в виде HTML-страницы (строки 31-36), открываем COM-порт с указанными параметрами (строка 37), переключаем в принтере кодовую страницу на CP1251 (строки 39-42). Затем печатаем заголовок чека (строки 44-45). Находим в POST-запросе значения параметров year и month (строки 47-57). После этого печатаем значения этих параметров (строки 59-60), подаем команду принтеру обрезать бумагу (строка 62) и закрыть порт (строка 63). И довыводим в HTML-страницу полученные результаты (строки 64-67).

Я специально не использую динамическое выделение буфера при обработке POST-запроса, так как, во-первых, пример должен быть понятен, и во-вторых, считаю, что для более серьезной работы вы все-таки выберете Perl или PHP, ведь вы будете выбирать данные из базы, не правда ли?

Собирается он следующим образом:

```

sh-3.1# gcc -Wall -o t4 t4.c -s -I. -I./lib -lstaro 1
-I /source/fcgi-2.4.0/include/ 1
-L /usr/local/fcgi/lib/ -lfcgi

```

Результат сборки кладете в пакет nginx.

Если вы правильно указали все участвующие в процессе компиляции библиотеки, а это LibStarIO (от принтера) и FCGI (для FastCGI), то перед вами рабочее приложение в формате FastCGI, которое будет способно распечатать параметры с HTML-страницы. Его также надо вместе с spawn-fcgi и скриптом старта nginx.sh переписать в пакет nginx.

```

kiosk:~# cat /usr/local/nginx/bin/nginx.sh

#!/bin/sh

```



```
chmod a+rw /dev/ttyS0
/usr/local/nginx/bin/spawn-fcgi -a 127.0.0.1 -
-p 9000 -u nobody -g nobody -
-f /usr/local/nginx/bin/t4
/usr/local/nginx/sbin/nginx
```

Сценарий запуска, файл packages/nginx/etc/init.d/nginx.init, создан по аналогии с примером пакета template в Thinstation.

```
#!/bin/sh
. $TS_GLOBAL
case "$1" in
init)
if ! pkg_initialized $PACKAGE; then
pkg_set init flag $PACKAGE
sh /usr/local/nginx/bin/nginx.sh
fi
;;
*)
exit 1
;;
esac
exit 0
```

В качестве HTML-страницы будем использовать такое творение мысли:

```
<html>
<meta http-equiv="Content-Type" content="text/html;
charset=cp1251">
<body>
<FORM ACTION="12.cgi" METHOD="post">
<select name="month">
<option value="January">January</option>
<option value="February">February</option>
<option value="March">March</option>
<option value="April">April</option>
<option value="May">May</option>
<option value="June">June</option>
<option value="July">July</option>
<option value="August">August</option>
<option value="September">September</option>
<option value="October">October</option>
<option value="November">November</option>
<option value="December">December</option>
</select>
<select name="year">
<option value="2008">2008</option>
<option value="2007">2007</option>
<option value="2006">2006</option>
<option value="2005">2005</option>
<option value="2004">2004</option>
<option value="2003">2003</option>
<option value="2002">2002</option>
<option value="2001">2001</option>
<option value="2000">2000</option>
<option value="1999">1999</option>
<option value="1998">1998</option>
<option value="1997">1997</option>
</select>
<input type="submit" name="submit" value="Try Me!">
</form>
</body>
</html>
```

То, что вызывается 12.cgi, не должно вводить вас в заблуждение, т.к. nginx среагирует на расширение .cgi и передаст управление spawn-cgi, а уж он-то и запустит приложение t4. Вместо 12.cgi можете придумать любое название, лишь бы оно было с правильным расширением.

Следует выбрать месяц и год и нажать на кнопку

«Try Me!». После этого забрать напечатанный чек с выбранным годом и месяцем:

```
+-----+
| CP1251: Информационный заголовок |
| CP1251: Выбирается год и месяц |
+-----+
Выбранный год: 2008, месяц: February
```

На экране информационного киоска будет высвечено сообщение (см. **рис. 6**), клиенту выдан чек, а в браузере напечатана информация.

Вполне понятно, что для более сложного сценария, когда требуется производить выборку из базы данных и необходимо пользователю предоставлять более интересное оформление разметки HTML-страницы, на одном таком простом CGI далеко не уедешь. Тем более что распарсивать сложные параметры, например, с русскими буквами и спецсимволами, на C напоминает очередное изобретение велосипеда. В этом случае на помощь могут прийти Perl и PHP, которые nginx может также запускать в качестве CGI-приложения. Никто не мешает выбрать в качестве веб-сервера еще более легкие решения, например GoAhead. Благо там тоже есть возможность запускать CGI-скрипты, а размер исполняемого файла составляет 80 Кб вместо 300 Кб для nginx.

Заключение

Мы не рассмотрели механизм работы с купюроприемниками, которыми могут комплектоваться киоски. Несмотря на то, что на первый взгляд кажется, что работа с ними таит какую-то специфику, на самом деле это далеко не так. «Общение» с этими устройствами не сложнее, чем «общение» с принтерами. Тот же коммуникационный порт RS232 (иногда бывает LPT и «токовая петля»), тот же возвращаемый статус (в данном случае это «купюра распознана», «купюра принята» или «купюра не принята»). Впрочем, это совсем другая история.

Отдельно хотелось бы поблагодарить сотрудников компании Star Micronics EMEA, Лоуренса Оуена (Lawrence Owen) и Джени Миллер (Jennie Miller) за помощь в подготовке статьи.

1. http://en.wikipedia.org/wiki/Internet_kiosk.
2. http://en.wikipedia.org/wiki/Automated_teller_machine.
3. http://ru.wikipedia.org/wiki/Информационный_киоск.
4. http://ru.wikipedia.org/wiki/Сенсорный_экран.
5. <http://www.comexim.com.ua/i18410.html>.
6. <http://www.opera.com/download/linux>.
7. <http://thinstation.sourceforge.net/wiki/index.php/ThDownload>.
8. <http://www.conan.de/touchscreen/evtouch.html>.
9. <http://sysoev.ru>.
10. <http://www.lighttpd.net/download/lighttpd-1.4.19.tar.gz>.
11. <http://www.fastcgi.com/dist/fcgi-2.4.0.tar.gz>.
12. http://www.star-micronics.co.jp/eng/dl/dl02_06_02spsd.htm.

Printer response page

```
Request number 1 running on host localhost
Input buffer length is: 41 symbols
BigText: month=February&year=2008&submit=Try+Me%21
Year: 2008
Month: February
```

Рисунок 6. Информационное сообщение в киоске