

Интерактивный терминал: Perl, XML и Firefox

Доклад больше касается веб-разработки, но Perl сыграл важную роль. Об этом я и расскажу.

Интерактивные терминалы

- платежные терминалы
 - оплата по счетам
 - банкоматы
- информационные терминалы
 - справочные терминалы по товарам и услугам

Характеристики терминалов

- обычный компьютер: процессор 2.6 Гц, память 256 Мб (wikipedia)
- ОС Windows XP
- Delphi / C++ Builder



Особенности терминалов

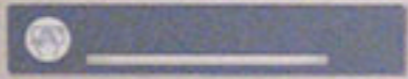
- не работает
 - ошибки Windows
 - “синий экран смерти”
 - перезагрузка системы

– ошибки Windows в виде всплывающий окон (если не сенсорный экран, то убрать невозможно)

– сбой питания => перезагрузка => никто не застрахован :)

WINCOM
WINCOM

The screen displays the Korona Bank logo at the top left, with the text "ХАЙНОВ" and "Банк родко" below it. In the center, there is a crown icon and the word "Золотая" above "Корона". Below this, a blue box contains a warning message in Russian: "The software has been modified, changed and updated new configurations. Before using please the program, it is not allowed to use the software. Please report to support when you have any issue." Below the warning, the text "ПЛАТЕЖНАЯ СИСТЕМА" and "www.korona.net" are visible. At the bottom of the screen, there is a row of logos for "Золотая Корона", "Билайн", "МТС", and "МЕГАФОН". To the right of these logos, the time "16:55:09" and the date "13/08/2008" are displayed. The screen is framed by a grey border with navigation buttons on the left and right sides.



A numeric keypad with buttons for digits 1 through 9, 0, and a star key. To the right of the keypad are three function buttons: "CANCEL" (red), "CORRECTION" (yellow), and "ENTER" (green).

BT

BT Internet kiosk

Internet Web email Text Telephone

WWW @ [Envelope Icon] [Phone Icon]

How to use the services

www.bt.com/multimedia

BT Internet kiosk terminal with a handset on the left, a screen displaying text, a numeric keypad, and a full QWERTY keyboard on a tray below.

Справочный терминал по товарам

- для небольших гаджетов
 - мобильные телефоны
 - цифровые фото
 - мп3-плееры

Справочный терминал по товарам (в медиа-центрах)

Основная задача

- пришел в магазин
- выбрал модель(и)
- получил полноценную (сравнительную) информацию
- купил :)

Требования к терминалу

- небольшой
- легкий
- дешевый
- без ошибок :)

Данные условия продиктовали требования не только для подбора железа, но и к разрабатываемому софту.
Главное – lightweight & powerful.

Характеристики железа

- материнская плата и процессор via
(все в одном)
- память 256 Мб
- USB флеш для системы

Самое интересное: варианты

- ОС
- Хранилище данных
- Клиентская часть
- Серверная часть ?

Нужно определиться, что будем использовать в качестве:

- ОС
- Х.д. - некоторая БД для хранения информации по товарам
- Клиентская часть и возможно серверная (в случае с Delphi все в одном)

Операционная система

- Linux Slackware
 - вся ОС содержится на USB флешке
 - размер ОС ~ 130 Мб
 - UTF-8

К решению данной проблемы мы подошли как веб-разработчики :)

Хранилище данных: варианты

- текстовая информация
 - СУБД: MySQL, PostgreSQL
 - BerkeleyDB, SQLite
 - ?
- медиа-контент

Основное условие – минимум всего.

MySQL – лишний демон, отказались, такие удобства здесь не нужны.

BDB – в принципе ок, но визуально трудно проверить, нужен клиент ~ mysql

Хранилище данных: варианты

- текстовая информация
 - СУБД: MySQL, PostgreSQL
 - BerkeleyDB, SQLite
 - XML
- медиа-контент

XML – удобный формат данных, “стандарт” обмена данными, легко генерить, работать, индуктивен

Пример XML каталога

```
<?xml version="1.0" encoding="utf-8"?>
<catalog>
  <category name="mobile" title="...">
    <product id="1">
      <title>Sony-Ericsson P990i</title>
      ...
    </product>
    ...
  </category>
  ...
</catalog>
```

В XML хранится вся информация о товарах

Каждая категория товаров содержит примерно несколько тысяч товаров, размер ~ 2-3 Мб, каталог ~10 Мб

Контроль XML

- XML + XSLT
 - легко написать преобразования для визуального контроля
 - картинка есть, но не читается
 - проверка корректности данных

Легко написать XSLT-преобразование для визуального контроля всего каталога или конкретной категории.

Например, картинка есть, но не читается

Генерация XML

- источник данных <http://market.yandex.ru>
- необходимо
 - собрать данные
 - нормализовать данные
 - сгенерировать XML

Нужна полноценная БД товаров.

Отличным источником данных является маркет яндекс.

Сбор данных

- Perl + POE
- POE::Component::Client::HTTP
 - наполнение БД категорий и товаров
 - получение конечных страниц товаров и изображений

На 20 слайде наконец появилось упоминание про Perl :)

POE – событийная машина, кот позволяет легко реализовать неблокирующую сетевую многозадачность и межпроцессорное взаимодействие.

БД – MySQL

1. паук проходит по всем категориям, производителям и разбивкам по страницам и наполняем базу товаров: url конечной страницы и url изображений

2. “тупой” паук – просто получает и сохраняет данные

Спасибо Яндекссу за
отказоустойчивость :)

Иногда при получении изображений — 500 ошибка

Пауки работают параллельно. Второй зависит от первого.
За несколько минут собирается необходимая база товаров.
Если 500, то при повторном проходе – все собирается.

Нормализация данных

- скрипты-нормализаторы на Perl
 - все строки — нужной длины !
 - ресайз изображений
- при необходимости — сокращение данных
 - факт наличие на сайте производителя

XML есть, Perl помог

генерация XML

Клиентская часть

- стенд с гаджетами
- при поднятии гаджета — выводится информация на монитор
- сравнить — поднять N-штук одновременно
- админ для управления

Выбрать категорию



- Выбрать категорию
- MP3 плееры
- Мобильные телефоны
- Цифровые фотоаппараты

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24

Очистить Отменить изменения Сохранить Отмена

Очистить Отменить изменения Сохранить Отмена

Мобильные телефоны

1  Nokia 5070 удалить	2  Nokia 6021 удалить	3  Nokia 6131 удалить	4  Nokia 6220 Classic удалить	5  Nokia 6500 Slide удалить	6  Nokia 8800 Sirocco Gold удалить	7  Nokia E61 удалить	8  Nokia E70 удалить
9  Nokia N73 удалить	10  Nokia N78 удалить	11  Nokia N91 8 Gb удалить	12  Nokia N93i удалить	13  Nokia N96 удалить	14  Samsung SCH-B100 удалить	15  Samsung SPH-I700 удалить	16  Samsung SPH-V840 удалить
17  Samsung SGH-Z550 удалить	18  Samsung SGH-i400 удалить	19  Sony-Ericsson G700 удалить	20  Sony-Ericsson K618i удалить	21  Sony-Ericsson W880i удалить	22  Sony-Ericsson Z770i удалить	23  Sony-Ericsson XPERIA X1 удалить	24  Sony-Ericsson W760i удалить

- | | | | |
|---------------|---|---------------------|---|
| Motorola | ↑ | Sony-Ericsson W710i | ↑ |
| Nokia | | Sony-Ericsson W760i | |
| Panasonic | | Sony-Ericsson W800i | |
| Rover | | Sony-Ericsson W810i | |
| Samsung | | Sony-Ericsson W830i | |
| Sony-Ericsson | ↓ | Sony-Ericsson W850i | ↓ |

Очистить Отменить изменения Сохранить Отмена

Очистить Отменить изменения Сохранить Отмена

SONY-ERICSSON W760i

На стенде размещается N-датчиков (24), которые посылают сигналы по COM-порту. Старо, но работает. Данная разработка секретна :)

Клиентское приложение: варианты

- аналог Delphi — Kylix
- графические тулкиты: Tk, Gtk
- ?

Kylix – сразу нет

Tk – есть опыт разработки, нужны дополнительные навыки :)

Решили пойти дальше...

Клиентское приложение: варианты

- аналог Delphi — Kylix
- графические тулкиты: Tk, Gtk
- **Firefox**

Firefox – браузер, отличное клиентское приложение

Firefox

- верстка интерфейсов: HTML+CSS+JS, XML+XSLT
- поддержка мультимедиа: флеш, аудио, видео, svg (не очень :)
- большое количество плагинов и дополнений (плагин rkiosk)
- протокол взаимодействия — HTTP

Очень просто и быстро, любой веб-разработчик способен создавать интерфейсы. Изменили дефолтные настройки FF, связанные с кэшированием – все через JS. Для Firefox нужен сервер, с которым он будет взаимодействовать по HTTP.

Серверная часть будет!

Perl? :)

Об этом позже

Порядок работы: варианты

- AJAX опрашивает сервер, нет ли нового сигнала один раз в N-миллисекунд
- push-технология: сервер сам сообщает клиенту о новом сигнале
- трюк: открывать новый таб в FF при появлении сигнала

Как клиент должен работать?

Самое главное – отреагировать на поднятый гаджет, т.е. на сигнал

Трюк – сильно заметен, мелькает белый экран, табов не видно, но их число растёт.

Push – самое то, но я узнал об этом позже.

Порядок работы: варианты

- **AJAX** опрашивает сервер, нет ли нового сигнала один раз в N-миллисекунд
- push-технология: сервер сам сообщает клиенту о новом сигнале
- трюк: открывать новый таб в FF при появлении сигнала

AJAX-опрос – стабильный вариант, работает 24на7 :)

Серверная часть: варианты

- Apache (mod_cgi/mod_perl/mod_fcgi)
- nginx (FastCGI)
- ?

Помним – минимум всего и функционально.

Веб-сервер будет совсем не нагружен, явно не highload.

Apache – тяжело, лишнее звено

nginx – отдача статики, динамика аналогична с Apache (плюс только видели, не пробовали)

Серверная часть: варианты

- Apache (mod_cgi/mod_perl/mod_fcgi)
- nginx (FastCGI)
- Perl + **POE**

POE я тогда увлекался :)

Веб-сервер на Perl

- POE::Component::Server::SimpleHTTP
 - отдача статики (img, css, js, xslt)
 - генерация нужных xml, соответствующих запросу
 - уведомление клиента о появлении сигнала от датчиков

SimpleHTTP – быстро и легко можно создать неблокирующий веб-сервер на POE.
– статики совсем не много
– xml – получение информации всего каталога, конкретной категории, производителя или группы товара

Сигналы

- Сигнал — последовательность 0 и 1, длина N (000...0, 100...0, 111...1)
- `cat /dev/ttyS0`
- `POE::Wheel::Run`

Как хорошо, что у нас Unix, чтение сигнала – очень просто.
Wheel::Run – не блокирующее взаимодействие с дочерними процессорами, в данном случае только чтение

Порядок работы: без вариантов

- при старте — читает XML каталог
- конфигурация PoCo::Server::SimpleHTTP
- POE::Session — события по обработке запросов от клиента
- POE::Session + POE::Wheel::Run — получение сигнала от датчиков

Порядок работы POE-машины:

– при старте – читает XML каталог, парсит и больше не обращается (если принудительно не попросить, н-р -s HUP или restart) – это достаточно, т.к. каталог меняться не будет, почти не будет (обновление хранилища)

```
POE::Component::Server::SimpleHTTP->new(  
  'ALIAS'      => 'HTTP_SERVER',  
  'ADDRESS'    => '0.0.0.0', # for all eth  
  'PORT'       => 80,  
  'HOSTNAME'   => 'hostname',  
  'HEADERS'    => {'Server' => '...'},  
  'HANDLERS'   => [  
    {  
      'DIR'      => '^/$',  
      'SESSION'  => 'HTTP_SERVER_SESSION',  
      'EVENT'    => 'got_index',  
    },  
    ...  
  ],  
);
```

В XML хранится вся информация о товарах

Каждая категория товаров содержит примерно несколько тысяч товаров, размер ~ 2-3 Мб, каталог ~10 Мб

Обработка запросов

- статика — отдача с диска, без кеша
- индекс — XML только нужных товаров
- сигнал — XML представление сигнала
- админ — пустой XML
- установка порядка — сохранение XML

- статика, не кешируем, т.к. это лишнее
- индекс, только те товары, которые были выбраны в админе (сразу все 24), т.к. другие показаны быть не могут
- сигнал: время генерации, id и на каком месте какой товар
- админ - пустой, вся логика XSLT + JS
- сохранение в XML placing.xml кросс-связи: место и товар

Получение сигнала

```
POE::Wheel::Run->new (  
    'Program'          => ['cat', '/dev/ttyS0'],  
    #'Program'        => './ttyS0.pl',  
    'StdoutEvent'     => 'stdout',  
    'StdoutFilter'    => POE::Filter::Stream->new,  
  
    'ErrorEvent'      => 'error',  
    'CloseEvent'     => 'close',  
  
);
```

При получении сигнала, вызывается событие stdout.

Преимущество POE – если нет /dev/ttyS0, то можно тестировать через свой скрипт – эмулятор.

Эмулятор сигнала

```
local $| = 1;
local $\ = "\n";

for (;;) {
    print pack "b*",
        join ' ', map { int rand 2 } 1..24;

    sleep ($ARGV[0] || 1);
}
```

Код на Си получился на несколько строк больше :)

Получение сигнала

- распаковать сигнал: `unpack "b*"`
- нюансы
 - короткий сигнал
 - длинный сигнал
- изменить `$SIGNAL`
- в идеале — `POE::Filter::Signal`

Может прийти короткий или длинный сигнал, тогда нужно либо накопить, либо обрезать, оставшееся в буфер.

Наполняем глобальную переменную `$SIGNAL`, которая при запросе на сигнал преобразуется в XML.

В идеале должен быть фильтр :)

Преимущества

- Легко, просто, быстро и дешево :)
- Решение — **OpenSource**
- Решение — кросс-платформенное
(за исключением /dev/ttyS0)
- Может работать без флешки
(изменения в админе не учитываются)
- Интересный подход: Perl + XML + Firefox

нет проблем с лицензией

Не забыть:

- отключить логи (на флешки объем ограничен и небольшой)
- победить проблему с прокруткой :)

Недостатки

- **Открытый код** (решение: обфускация, криптование раздела)
- Могут возникнуть проблемы с отрисовкой изображений в Firefox при слабых мощностях (решение: видео-драйвер?)
- При больших объемах данных и слабых мощностях — Firefox медленно парсит XML/JSON (решение: увеличить таймаут)

Только что у нас это было преимуществом, теперь недостаток.
Кто захочет, тот расшифрует :)

Хоть из кеша, хоть с сервера – медленно отрисовываются картинки
Здесь не возникло такой проблемы

Решение увеличить таймаут или уменьшить порции данных

Для каждой задачи —
свое решение

Мне понравилось :) Работает. Стабильно.

Спасибо за внимание!

Анатолий Шарифулин

South Perl