DEVELOPING MOBILE BASED SYSTEM WITH QTOPIA SDK IN VIRTUAL OPERATING SYSTEM & LINUX KERNEL – MOBILE TECHNOLOGY

Dr. Prashant M. Dolia, Milan S. Bhatt

ABSTRACT

The introduction of Linux operating system into the embedded sector has been one of the most exciting changes in the last few years. Based on the open-source model, it offers new possibilities to embedded engineers traditionally used to commercial operating systems. Qt for Embedded Linux (formerly known as Qtopia Core) is the leading application framework for the developing a (For Region Language - Gujarati) mobile-application and mobile based kernel system as well as single-purpose devices powered by embedded Linux. It provides a robust and proven development environment inherited from the Qt cross-platform application framework and key components developed specifically for embedded Linux. Qt for Embedded Linux enables manufacturers to efficiently create devices with applications that are tailored to market needs specifically Gujarati Language.
Keywords: Qt, Linux, Qtopia, Cross-platform, Gujarati Language, mobile application, mobile based kernel.

1. INTRODUCTION

The Information Technology, without doubt, has enormous power to improve how people live and work. Thousands of tools – hardware, software, and embedded – are developed to make life of mankind an efficient and convenient for the Gujarati Language. A revolution is taking place today in the way of people, how to access, learn, and interact Gujarati Language with information in the mobile technology in India.

Researchers have also concentrated on the developing new mobile-based kernel for mobile technology with Qtopia Core and QT - Designer. Researchers have also concentrated on the developing Gujarati Desktop and Gujarati Application interface with context to Indian Culture System. It provides unrivaled flexibility for customizing an advanced user experience like used Linux based application in region languages. Therefore, further research work is required to develop more **Flexible Gujarati System** in areas like **WYSIWYG** concept in the system, database design & administration, gaming, multimedia systems, browsing web pages, small application like notepad, calculator, To Do application, alarm, calendar, etc. in region language-Gujarati.

This research work is intended to develop system for the Mobile Technology and Virtual Operating System with Qtopia SDK and Qt Designer both especially to facilitate Gujarati People.

Ultimate objective of technology is to facilitate human being for easy use of Mobile with Gujarati Language Desktop and its application. Nowadays, there is worldwide campaign going on to use the Microsoft Operating System working in the different mobile, so that, we works on the developed Gujarati Desktop and application in Linux based mobile and Virtual Operating System.

2. RESEARCH METHODOLOGY

In approach, a problem is identified, and solved by using available technologies and/or extension of technologies. This approach is **problem centric**.

In another approach, technology is extended / invented, which is useful in solving, completely or partially, many problems related to use to the Linux based technology in India. This approach is **technology centric**.

This research work will be carried out by technology centric approach, so that many applications may be developed based on the outcome of the work, i.e. developing new mobile kernel, software tools, basically used in the Linux based Mobile technology specifically for Gujarati Language.

3. LINUX KERNEL WITH QTOPIA EMBEDDED SYSTEM

This section explains the steps taken during compilation of the Linux kernel and the output produced at each stage. The build process depends on the architecture so we would like to emphasize that we only consider building a Linux kernel for the Linux based Mobile. In this part understand the kernel for devices and the collaboration of the other architecture, which is used for the mobile basically on the Linux, based Operating System.

Below the image is for the understanding the Linux Kernel and the Qtopia architecture for used in the Linux Based Mobile, which is used for the India Culture Mobile with Gujarati Language.

4. DETAILS OF THE RESEARCH WORK

This research work uses Linux Kernel because it's totally free and easy to use. It creates a new mobile kernel for the mobile devices in Gujarati Language and its interface for the regional people. This approach works with the Kernel structure and changes it to make it suitable for the Linux Based Mobile and Virtual Operating System in the PCs with the Gujarati Language for the regional people.

- Kernel Source: We modified the existing kernel into new proposed kernel for mobile kernel.
- Qtopia Core: For the Embedded System in the Mobile for the developing a new kernel and device independent application for the mobile devices.
- Qt – Designer: For the developing New GUI Application
- An alternate form of querying a Virtual Operating system and its structure.

- Also be using the Linux Kernel Structure and Kernel Programming.

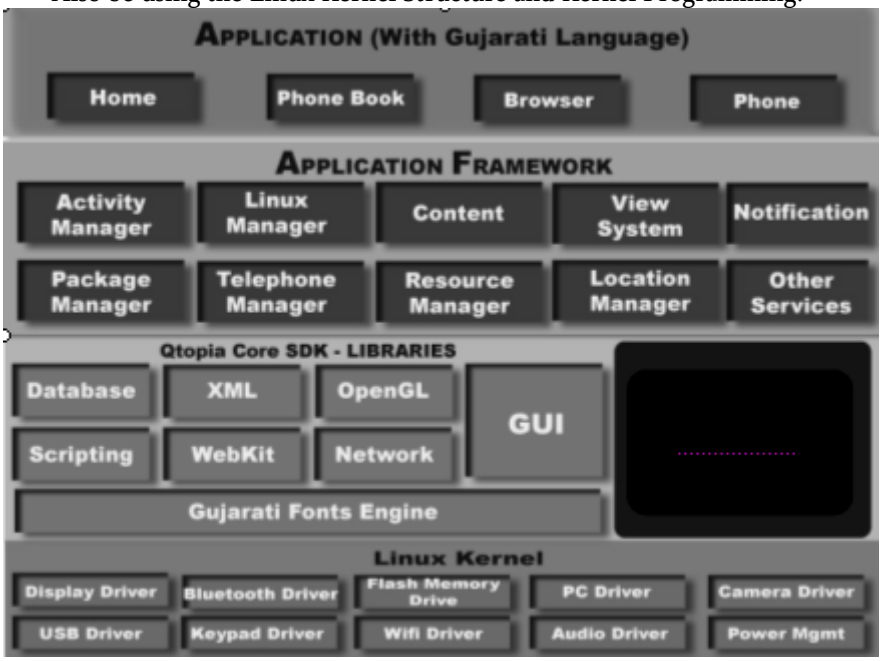


Fig. 1.0: Mobile Kernel Architecture with proposed new system

5. CONFIGURING A NEW KERNEL

Our next step is to run the configuration utility. On the 2.6.x kernels there are four main frontend programs: config, menuconfig, and xconfig. **config** is the least user-friendly option as it merely presents a series of questions that must be answered sequentially. Alas, if an error is made you must begin the process from the top.

oldconfig will read the defaults from an existing .config and rewrite necessary links and files. Use this option if you've made changes to source files or need to script the rebuild process.

menuconfig is an *ncurses* based frontend. Your system must have the ncurses-devel libraries installed in order to use this utility. As the help text at the top of the screen indicates, use the arrow keys to navigate the menu. Press **Enter** to select sub-menus. Press the highlighted letter on each option to jump directly to that option.

make xconfig

Once you have decided which configuration option to use, start the process with **make** followed by either **config**, **menuconfig**, or **xconfig**. For example:

$ make menuconfig

6. BOOTING OVERVIEW

The boot process details are architecture–specific, so we shall focus our attention on the mobile devices Due to old design and backward compatibility, the PC firmware boots the operating system in an old–fashioned manner.

6.1 Booting: bootsector and setup

The bootsector used to boot Linux kernel in the Mobile devices could be either

- Linux bootsector (arch/i386/boot/bootsect.S),
- LILO (or other bootloader's) bootsector, or
- no bootsector (loadlin etc)

We consider here the Linux bootsector. The first few lines initialise the convenience macros to be used for segment values:

```
29 SETUPSECS = 4 /* default nr of setup–sectors */
30 BOOTSEG = 0x07C0 /* original address of boot–sector */
31 INITSEG = DEF_INITSEG /* we move boot here – out of the way */
32 SETUPSEG = DEF_SETUPSEG /* setup starts here */
33 SYSSEG = DEF_SYSSEG /* system loaded at 0x10000 (65536) */
34 SYSSIZE = DEF_SYSSIZE /* system size: # of 16–byte clicks */
```

6.2 Module gets into the Kernel

You can see what modules are already loaded into the kernel by running **lsmod**, which gets its information by reading the file /proc/modules. How do these modules find their way into the kernel? When the kernel needs a feature that is not resident in the kernel, the kernel module daemon kmod[1] execs modprobe to load the module in. modprobe is passed a string in one of two forms:

A module name like softdog or ppp.
A more generic identifier like char–major–10–30.
If modprobe is handed a generic identifier, it first looks for that string in the file /etc/modprobe.conf.[2] If it finds an alias line like:
alias char–major–10–30 softdog
it knows that the generic identifier refers to the module softdog.ko.

Next, modprobe looks through the file /lib/modules/version/modules.dep, to see if other modules must be loaded before the requested module may be loaded. This file is created by **depmod –a** and contains module dependencies. For

example, msdos.ko requires the fat.ko module to be already loaded into the kernel. The requested module has a dependency on another module if the other module defines symbols (variables or functions) that the requested module uses. Lastly, modprobe uses insmod to first load any prerequisite modules into the kernel, and then the requested module. Modprobe directs insmod to /lib/modules/version/[3], the standard directory for modules. Insmod is intended to be fairly dumb about the location of modules, whereas modprobe is aware of the default location of modules, knows how to figure out the dependencies and load the modules in the right order. So for example, if you wanted to load the msdos module, you'd have to either run:

insmod /lib/modules/2.6.11/kernel/fs/fat/fat.ko
insmod /lib/modules/2.6.11/kernel/fs/msdos/msdos.ko

Now we do the how the developing a new programmed into the mobile based kernel and its source code is as under.

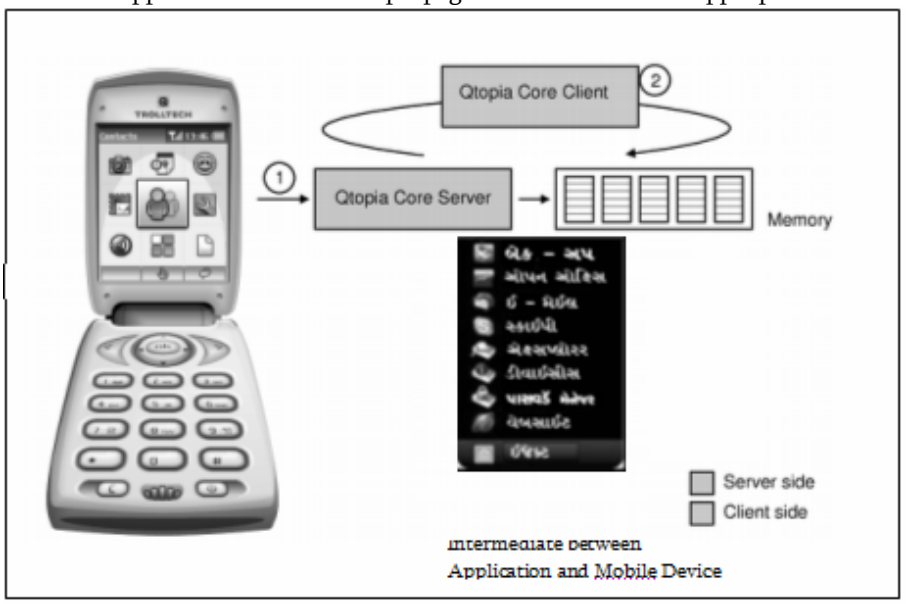
```
/*
 * first.c – The simplest kernel module.
 */

 #include <linux/module.h>       /* Needed by all modules */

 #include <linux/kernel.h>       /* Needed for KERN_INFO */

 int init_module(void)
 {
 printk(KERN_INFO "Research work on Mobile Kernel.\n");

 /*
 // A non 0 return means init_module failed; module can't be
 loaded.
 */
 return 0;
 }

 void cleanup_module(void)
 {
 printk(KERN_INFO "Goodbye Gujarati Desktop.\n");
 }
```

7. QTOPIA CORE ARCHITECTURE

A Qtopia Core application requires a server application to be running, or to be the server application itself. Any Qtopia Core application can act as the server. When more than one application is running, the subsequent applications connect to the existing server application as clients.

The server and client processes have different responsibilities: The server process manages pointer handling, character input, and screen output. In addition, the server controls the appearance of the screen cursor and the screen saver. The client process performs all application specific operations.

The server application is represented by an instance of the QWSServer class, while the client applications are represented by instances of the QWSClient class. On each side, there are several classes performing the various operations.
All system generated events, including keyboard and mouse events, are passed to the server application which then propagates the event to the appropriate client.



Green Phone: Courtesy from Trolltech Company and proposed

Figure 2.0: Qtopia Core Architecture with researcher applications

7.1 Client/Server Communication

The running applications continuously alter the appearance of the screen by adding and removing widgets. The server maintains information about each top-level window in a corresponding QWSWindow object.

Whenever the server receives an event, it queries its stack of top-level windows to find the window containing the event's position. Each window can identify the client application that created it, and returns its ID to the server upon request. Finally, the server forwards the event, encapsulated by an instance of the QWSEvent class, to the appropriate client.

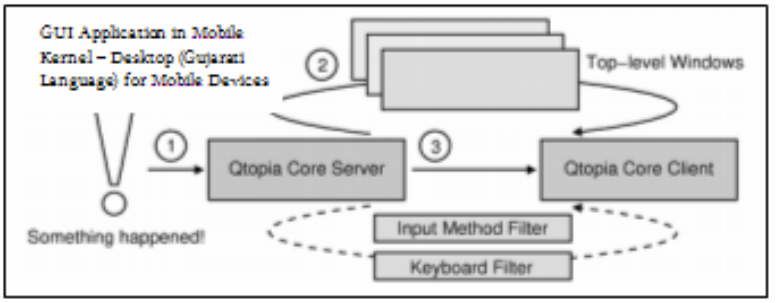


Fig. 2.0.1: Communication between Client and Serve in Gujarati Language

7.2 Character Input Layer

- QWSKeyboardHandler
- QKbdDriverPlugin
- QKbdDriverFactory

The keyboard driver (represented by an instance of the QWSKeyboardHandler class) is loaded by the server application when it starts running, using Qt's plugin system.

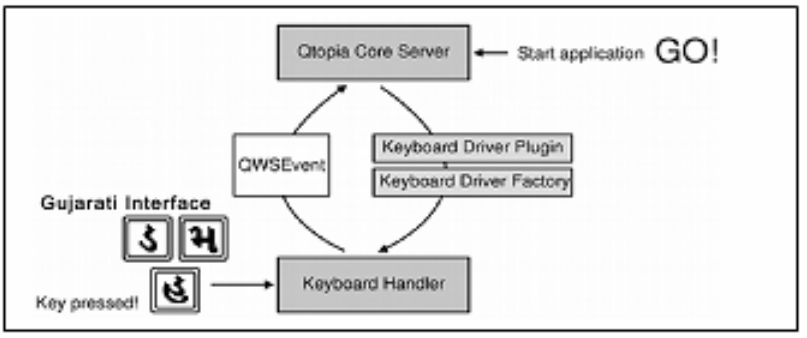


Fig. 2.0.2 Character Input Layer

A keyboard driver receives keyboard events from the device and encapsulates each event with an instance of the QWSEvent class which it then passes to the server.

Qtopia Core provides ready-made drivers for several keyboard protocols; see the character input documentation for details. Custom keyboard drivers can be implemented by subclassing the QWSKeyboardHandler class and creating a keyboard driver plugin. Qtopia Core's implementation of the QKbdDriverFactory class will automatically detect the plugin, loading the driver into the server application at runtime.

enum Qt:: WA_PaintOnScreen

This enum type is used to specify various widget attributes. Attributes are set and cleared with QWidget::setAttribute(), and queried with QWidget::testAttribute(), although some have special convenience functions which are mentioned below.

Constant	Value	Description
Qt::WA_PaintOnScreen	8	Indicates that the widget wants to draw directly onto the screen. Widgets with this attribute set do not participate in composition management, i.e. they cannot be semi-transparent or shine through semi-transparent overlapping widgets. The flag is set or cleared by the widget's author. This flag is required for rendering outside of Qt's paint system; e.g. if you need to use native mobile device painting primitives.

7.3 Accelerated Graphics

It is possible to add an accelerated graphics driver to take advantage of available hardware resources.

8. CONCLUSION

In India, mobile user may use Symbian operating system in the Mobile as well as Windows CE technology, but Linux based Operating System Mobile is very rarely used by the mobile users. So, the present paper has discussed the anatomy and features of a Linux Based Mobile Kernel for the user of Gujarati Language in application and desktop, who easily understand the mobile application in the regional language like Gujarati. It is an embedded system for the mobile particularly Linux mobile in India and can be easily used by the non-technical people and mobile users. These applications also work on “what you see is what you get concept”; to demonstrate the understating of how to work on the Linux based Mobile. Also the proposed system is beneficial for the Mobile based kernel

and Virtual Operating system for Linux Operating System in the Personal Computer for Gujarati Language.

REFERENCES:

1. [October, 2000] Understanding the Linux Kernel, Deniel P. Bowet & Marco Cesati, By O'Reilly
2. Online Reference Documentation, http://www.*doc.trolltech.com*
3. http://trolltech.com/solutions: For the Online solution by trolltech site.
4. [July, 1996-1997] The Linux Kernel, David A. Rusling
5. An Overview of Linux Kernel Structure, Jennifer Hou, Department of Computer Science, University of IIIinois at Urbana-Champaign
6. C++ GUI Programming with QT 3, Jasmin Blanchette, Mark Summerfield, Prentice Hall in association with Trolltech Press
7. [April, 2003 – 2004] The Linux Kernel Module Programming Guide, Peter Jay & Ori Pomerantz Salzman http://www.faqs.org/docs/kernel
8. Linux Kernel Module Programming: http://tldp.org/LDP/lkmpg/2.6/html/lkmpg.html#AEN319
9. http://www.ibm.com/developerworks/linux/library/l-linuxboot
10. Monolithic Kernel vs. Microkernel, Benjamin Roch, TU Wien
11. A Virtual Operating System, Dennis E. Hall, Deborah K. Scherrer & Joseph S. Sventek – Lawrence Berkeley Laboratory