*Ninth LACCEI Latin American and Caribbean Conference (LACCEI'2011), Engineering for a Smart Planet, Innovation, Information Technology and Computational Tools for Sustainable Development, August 3-5, 2011, Medellín, Colombia.*

# Desenvolvimento de Aplicativos Embarcados em Plataforma ST-Linux Aplicados a Receptores do Sistema Brasileiro de Televisão Digital – SBTVD

Marcelo Augusto Costa Fernandes
Departamento de Engenharia de Computação e Automação – DCA
Centro de Tecnologia - CT
Universidade Federal do Rio Grande do Norte - UFRN
Natal, Brasil
mfernandes@dca.ufrn.br

Vicente Idalberto Becerra Sablon[1] e João Marcelo von Zubem[2]
Departamento de Engenharia Elétrica
Unidade Campinas/São José
Centro Universitário Salesiano de São Paulo – UNISAL
Campinas, Brasil
vsablon@sj.unisal.br[1] e joao.vonzuben@gmail.com[2]

## RESUMO

Este artigo faz um estudo sobre o desenvolvimento de aplicativos embarcados para sistema operacional Linux, voltados para equipamentos utilizados no Sistema Brasileiro de Televisão Digital (SBTVD). Testes com desenvolvimento de aplicativos embarcados na plataforma ST-Linux (disponível em vários equipamentos do SBTVD), foram utilizados para validar o estudo.

**Palabras claves:** - Linux; SBTVD; ST-Linux; Sistema Embarcado.

## ABSTRACT

This article is a study on developing applications for embedded Linux operating system, focused on equipment used in the Brazilian System of Digital Television (SBTVD). Tests with application development platform embedded in ST-Linux (available in many devices SBTVD) were used to validate the study.

**Keywords:** Linux; SBTVD; ST-Linux; Embedded System.

## 1. INTRODUCÃO

Um sistema embarcado pode ser definido como um hardware ou software com um propósito especial que é encapsulado pelo dispositivo que o controla. Estes sistemas executam funções dedicadas, ou seja, são responsáveis por uma função especifica ou um conjunto restrito de funções específicas e co-relacionadas. São encontrados na maioria dos equipamentos, na maioria das vezes "escondidos".

O Linux embarcado não é um exatamente um software único o qual se carrega em uma unidade de memória e, quando se liga o computador, ele passa a executar e a controlar o computador. O Linux, tanto em sua forma embarcada quanto em sua forma convencional, é composto de componentes de software que, juntos, realizam as atividades comumente observadas em sistemas operacionais convencionais.

O desafio de embarcar um ambiente Linux completo ainda esta longe do fim. Para comprimir e reescrever alguns gigabytes de códigos fonte, dados de sistema e programas executáveis, dando suporte à maioria dos drives e protocolos de comunicação é realmente uma tarefa difícil.

Na historia da computação, mostra-se que isso não é uma novidade. Na década de setenta, pesquisadores que trabalhavam com mainframes UNIX tiveram este mesmo desafio, começaram a reduzir tanto o tamanho dos programas, bem como a quantidade de memória requerida para a execução. O objetivo era muito parecido com o de hoje, portar um ambiente mainframe para novos aparelhos de menor tamanho e desempenho.

Atualmente existem soluções Linux embarcado em muitos dispositivos, mas estas tendem a serem distribuições dedicadas exclusivamente a um propósito e não contemplam a idéia de um sistema operacional completo embarcado [1] [2] [3] [4] [5] [6] [7].

Os dispositivos projetados para o SBTVD estão sendo desenvolvido, através de soluções embarcadas em sistema operacional Linux. Este tipo de solução é bastante interessante dada sua versatilidade do ponto de vista do fabricante e do desenvolvedor [8] [9] [10].

No Brasil existem diversas alternativas para o desenvolvimento de aplicativos para o SBTVD e uma delas é a construção de aplicativos embarcados com finalidades específicas, porém, soluções deste tipo estão ligadas diretamente ao hardware do fabricante que embarcou o Linux, por outro lado, construir aplicativos diretamente no sistema operacional embarcado, no caso o Linux, é bastante seguro é seu desempenho é melhor que soluções semi-interpretadas [11] [12].

Assim, este artigo tem como objetivo apresentar um estudo sobre o desenvolvimento de aplicativos em Linux embarcado mostrando suas características e possibilidades. Para validar o estudo foi utilizada a plataforma ST-Linux da ST Microelectronics [7] [13] [14]. Este artigo esta dentro de um projeto maior chamado de STB-SCAN no qual apresenta soluções de monitoramento remoto dos pontos de TVD para auxilio em tempo real para as operadoras de TV.

## 2. LINUX EMBARCADO

O ambiente Linux embarcado, como mostra a Figura 1, pode ser dividido em duas camadas. O *kernel* com seus componentes (drives e módulos) fica em uma camada mais baixa e controla a alocação de memória, dispositivos periféricos, interfaces de conexões, como também envia tarefas a Unidade Central de Processamento (CPU). Assim, todos os problemas existentes quando se manipula blocos de memória é tratado por esta camada [12] [15] [16].

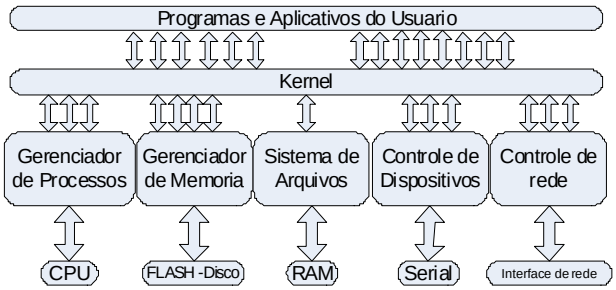


**Figura 1. Arquitetura Simplificada do Sistema Operacional Linux.**

Já o espaço do usuário tem uma segurança mais baixa e contem programas de um nível mais alto que acessam recursos específicos do *kernel*, como por exemplo, um *pipe*. Os aplicativos também executam processos completos, sem se preocupar com a divisão deste processo em pequenas tarefas para a CPU.

Uma diferença entre o ambiente Linux de alta plataforma e um embarcado, é o processamento de multitarefa. Por mais que seja dito que o ambiente embarcado é multitarefa, assim como o de alta plataforma, o processo para a execução de multitarefas não é o mesmo [12] [15] [16].

Um ambiente de alta plataforma é capaz de processar diferentes tarefas, para diferentes usuários, enviadas por diferentes programas. Já o embarcado, executa apenas tarefas de um usuário e cria uma fila de prioridades para simular o processamento multitarefas. Assim, quando programas concorrentes enviam processos a CPU embarcada, cada processo é dividido em pequenas tarefas e estas recebem uma prioridade para execução. Intercalando tarefas oriundas de processos distintos é que se consegue simular uma CPU multitarefa. Particularmente, quando se executa uma tarefa em um ambiente embarcado, toda a memória requerida já é alocada antes da sua execução por meio das unidades de gerenciamento de memória MMUs (*Memory management units*). Isto faz com que outros programas não consigam escrever em um espaço de memória que já foi alocado e assim, garante a integridade da execução da tarefa [12] [15] [16].

## 3. SET-TOP-BOX

O Set-Top-Box (STB) é um hardware dedicado que faz a recepção do sinal de televisão digital e para que isto seja desenvolvido de maneira a satisfazer toda especificação, outras interfaces e memórias foram incorporadas. O diagrama de blocos mostrado na Figura 2 apresenta um set-top-box genérico com seus principais componentes.

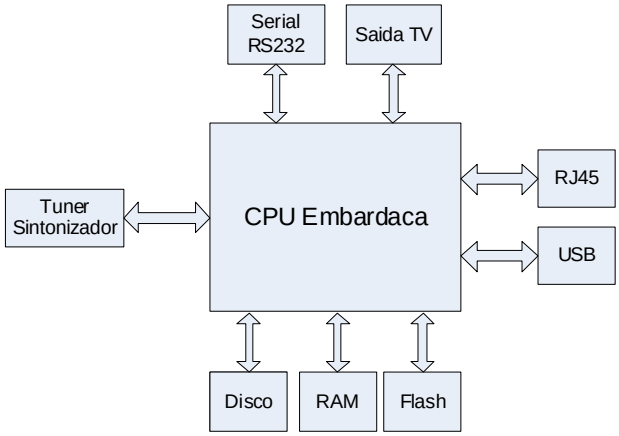


**Figura 2. Arquitetura de um set-top-box genérico com seus principais componentes**.

O centro de todo sistema é a CPU embarcada que é responsável por controlar todos periféricos, decodificar as *streams* de vídeo codificados no padrão MPEG4 (*Moving Picture Experts Group*) [8] e de áudio codificadas no padrão ACC (*Advanced Audio Coding*) [8]. Esta *stream* de dados é sintonizada pelo TUNER que transmite os dados para o decodificador da CPU. Ele processa os dados e encaminha a informação de áudio e vídeo para a saída do receptor de televisão [8] [9] [10] [11]. Existem outros periféricos incorporados a esta solução como as memórias: RAM e FLASH, interfaces de comunicação: RJ45, RS232 e USB.

A memória FLASH pode ser dividida em três partes, sendo que cada uma delas contem apenas uma imagem comprimida dos dados e não podem ser usados antes de expandi-los: A primeira contendo um programa que gerencia a inicialização do sistema embarcado. Este é um procedimento muito particular, pois alem de checar a integridade de cada periférico, ainda inicializa alguns deles. No segundo seguimento da memória FLASH pode-se armazenar o *kernel*. Esta possivelmente é a maior partição de memória, pois todo sistema é baseado neste *kernel*.

E por fim, a terceira parte que pode ser definida como um espaço para o sistema de arquivos do usuário, possibilitando que o usuário tenha um pequeno espaço de leitura e escrita de seus dados.

Durante o processo de inicialização, estas imagens contidas na FLASH são expandidas na memória RAM e assim, o sistema ganha em desempenho, já que o acesso a memória RAM é mais rápido que o acesso a FLASH e também deixa o produto com um preço menor, pois a memória RAM é consideravelmente mais barata que a memória FLASH [10] [11].

Pode-se também destacar as diferentes interfaces e seus respectivos propósitos, como por exemplo, a interface de rede RJ45, que permite a conexão via cabo, do STB a uma rede de computadores. Esta foi uma solução encontrada para desenvolver o canal de retorno e estabelecer a comunicação oriunda do telespectador a emissora. Outra interface importante é a RS232, com ela podemos conectar via terminar no ambiente Linux, permitindo ajustes e manipulação de programas já embarcados.

## 4. ST-LINUX

O ST-Linux é um sistema operacional Linux de código aberto (*open source*), embarcado e baseado em pacotes RPM (*RedHat Package Manager*). O ST-Linux foi desenvolvido pela ST Microelectronics [7], mas conta com a colaboração de muitos desenvolvedores ao redor do mundo. Desenvolvido nas linguagens de programação C e C++ tem seu *kernel* na versão 2.6 para a arquitetura ST40 e conta com uma gama muito grande de *drivers* escritos e disponibilizados pela STAPI.

A ST Microelectronics também criou um sistema completo de desenvolvimento e para isso foi projetada uma arquitetura utilizando NFS (*Network File System*), que permite o acesso remoto do sistema de arquivos embarcado. Em uma maquina de alta plataforma com sistema operacional Linux baseado em pacotes RPM (Fedora e RedHat) é instalado um ST Linux e um servidor NFS. O set-top-box é configurado para que na inicialização procure o *kernel* e o sistema de arquivos na maquina alta plataforma. Assim, o desenvolvedor não precisa fazer o *deploy* (instalar a aplicação para disponibilizar ela para seus usuários), nem embarcar a aplicação a cada teste. Pode desenvolver o código na própria maquina alta plataforma e rodar o teste pelo set-top-box. A Figura 3 ilustra este tipo de configuração, exemplificando que esta comunicação entre o STB e o servidor NFS se da por meio de interfaces de rede usuais.

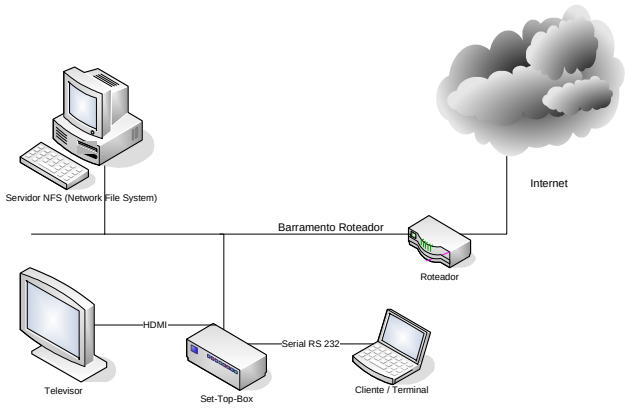


**Figura 3. Plataforma de desenvolvimento para o ST-Linux**.

Também foi customizada a IDE (*Integrated Development Environment*) eclipse [7] [13] [14] com *plugins* que permitem desenvolver aplicações e alterar o *kernel* de maneira muito mais agradável, pois permite analisar o consumo de memória, fazer um "*Trace*" para verificar o ponto de decisão da CPU, bem como debugar os

programas em tempo real. A esta nova interface de desenvolvimento atualmente ela esta na versão R 4.0.1 e deram o nome de ST *Workbench*.

Outra forma de comunicação importante que o set-top-box oferece é a interface serial RS232. Através desta porta de comunicação é possível conectar um cliente/terminal ao ambiente de desenvolvimento. Também é usada pelo ST *Workbench* como meio de comunicação para debug.

A principal linguagem de programação para o ST-Linux é o C e C++. São aceitos dois tipos de compilação, Nativa e Cruzada. Para ambos ainda existem dois cenários de execução: um mais robusto contendo todas as bibliotecas GNU C (glibc) e outro menor, contendo, apenas as bibliotecas primordiais (uclibc). O segundo ambiente é otimizado em relação ao primeiro, porem limitado por carregar apenas bibliotecas essenciais. A opção pelo uso de cada um dos ambientes fica a critério do projetista/desenvolvedor, que deve ponderar entre um conjunto maior de bibliotecas para o desenvolvimento, ou uma execução mais rápida pelo usuário.

## 5. RESULTADOS

Com o ambiente de desenvolvimento montado e o ST-Linux instalado, foram recompilados os aplicativos padrões, *drivers* da STAPI e o *kernel* 2.6.

Para isso, foi primeiro necessário instalar os pacotes RPM do ST-Linux na versão 2.3, arquitetura SH4. Estes pacotes podem ser encontrados no site do ST-Linux, sendo que em sua maioria possuem licença *General Public License* (GPL).

Foi também instalado o software para comunicação do *kernel* e integração deste a toda plataforma. Este software é chamado *Multicom* e necessita de determinados *patches* para possibilitar esta integração. Fazendo uso do compilador GCC 4.1 e de scripts *make* previamente estabelecidos, foi compilado primeiro o kernel na versão 2.6, depois os *drivers* e por ultimo os aplicativos.

Também foram copiados alguns programas já compilados para locais específicos do ambiente, como por exemplo, o programa de teste da STAPI (main_cocoref_gold_7109_LINUX_29BITS.out). Com este programa é possível testar o funcionamento completo da solução. Desde a interface física de entrada o TUNER, até a saída da para televisão (saída HDMI ou três elementos), tendo todo o controle gerenciado pelo ST-Linux.

Assim é notado que o ST-Linux atua como gerenciador de controle de todo hardware, fazendo a comunicação entre os periféricos que estão ao redor da CPU, através dos drivers/módulos de acesso de cada periférico.

Com a plataforma de desenvolvimento, apresentada na Figura 3, montada é possível desenvolver qualquer tipo de aplicativo em C ou C++ utilizando o GCC associado à STAPI como apresentado em [7] [13] [14].

## 6. CONCLUSÕES

Este artigo apresentou um estudo sobre o desenvolvimento de aplicativos embarcados em plataforma Linux. Foram apresentadas e discutidas as características e funcionalidades dos sistemas embarcados baseados no Linux bem como sua arquitetura. Para validar o estudo foram apresentados os procedimentos para o desenvolvimento de aplicativos na plataforma ST-Linux, plataforma esta desenvolvida pela ST Microelectronics.

## REFERÊNCIAS


[1] S. Moon, J. Kim, K. Bae, J. Lee and D. Seo. Embedded Linux Implementation on a Commercial Digital TV System. In IEEE Transactions on Consumer Electronics, Vol. 49, No. 4, pp. 1402 – 1407, Nov. 2003.

[2] K. Baker, R. Pulles and P. Sasno. Interactive TV: Combining MHP with MPEG4. In 2006 International Conference on Consumer Electronics, ICCE 2006, pp 445–446, Feb. 2006.

[3] The Lynuxworks website. [Online]. Available: http://www.lynuxworks.com, 01/04/2010

[4] The Montavista website. [Online]. Available: http://www.mvista.com, 01/04/2010

[5] The Timesys website. [Online]. Available: http://www.timesys.com, 01/04/2010

[6] The Handhelds website. [Online]. Available: http://www.handhelds.org, 01/04/2010

[7] The ST Linux website. [Online]. Available: http://www.stlinux.com, 01/04/2010

[8] V. Zlokolica, R. Uzelac, G. Miljkovic, T. Maruna, M. Temerinac and N. Teslic. Video processing real-time algorithm design verification on embedded system for HDTV. In 2009 First IEEE Eastern European Conference on the Engineering of Computer Based Systems. pp 150–151, Sep. 2009.

[9] B. Hu, G. Wu, L. Pan, H. Ni and M. Zhu. An Implementation of Interactive Set-Top-Box and Its Applications. In 2006 8th International Conference on Signal Processing, Beijing. Nov. 2006.

[10] L. Kumar, R. Kushwaha and R. Prakash. Design & Development of Small Linux Operating System for Browser Based Digital Set Top Box. 2009 First International Conference on Computational Intelligence, Communication Systems and Networks. pp. 227 – 281, India, Jul. 2009.

[11] Q. Li, M. Guo, Digital Recordable Integrated Television Based on Embedded Linux Operating System. CSIE '09 Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering. IEEE Computer Society Washington. Vol. 7, pp. 81 – 84, 2009.

[12] I. McLoughlin and A. Aendenroomer. Linux as a Teaching Aid for Embedded Systems. 2007 International Conference on Parallel and Distributed Systems. pp. 1 – 8, Taiwan, Dec. 2007.

[13] STMicroelectronics. ST40 STLinux: Distribution and development environment for ST40 platforms. Doc. 8153893 Rev. A, Sep. 2008.

[14] STMicroelectronics. RN0053 Release note: Linux support package for SPEAr (LSP) v 2.3. Doc ID 17552 Rev 1, May. 2010.

[15] M. Jin, X. Zhou, P. Duan, Z. Tian and J. Zhou. The Design and Implementation of Embedded Configuration Software Based on Embedded-Linux. In Proceeding CSSE '08 Proceedings of the 2008 International Conference on Computer Science and Software Engineering. IEEE Computer Society Washington. Vol. 4, pp. 98 – 101, 2008.

[16] Y. Tian, G. Ren and Q. Wu. Implementation of Real-time Network Extension on Embedded Linux. In ICCSN '09 Proceedings of the 2009 International Conference on Communication Software and Network. IEEE Computer Society Washington. pp. 163 – 167, 2009.

### *Autorización y Renuncia*

*Los autores autorizan a LACCEI para publicar el escrito en las memorias de la conferencia. LACCEI o los editores no son responsables ni por el contenido ni por las implicaciones de lo que esta expresado en el escrito.*