Develop Linux Device Driver to Optimize Power Consumption While Streaming Media

Gondaliya Bhavdeep D[1] Shankar Patel[2]
[1,2] VLSI & Embedded Systems Design Department
[1,2] Gujarat Technological University, Ahmedabad, India.

Abstract— The goal of the thesis is to design streaming media from flash by writing driver in user space and loaded in kernel space. The main objective is to make low power consumption mechanism useful for media streaming. In Present mechanism , a media is streaming from hard-disk in that the movement of mechanical part is continue so that's why more power consumption occurs, so instead of present mechanism the concurrent mechanism will implement in that the media is streaming from flash so that's why the mechanical part's movement is reduce and power Saving occurs which ultimately results in power optimization. The project involves writing a character driver in blocking and non-blocking mode using Ioctl system call so that instead of using address space of secondary storage devices like hard disk, the new mechanism will use the address space of ram, which results in fast data caching so without any delay the media will be play which finally saves lot of time as well as power. The process overall includes creating multithreads, making open system call in blocking or non-blocking mode which utilizes user daemon services to call, wait(blocking or non-blocking mode) and acknowledge the process.

Keywords: streaming media , device driver , demon, Ioctl.

I. INTRODUCTION

Today electronic industry there are seeking how to power optimization occur so there are a very interesting topic which name is streaming media[1] from flash in that the data is read from flash and hard disk will going to sleeping mode so that's why mechanical part of movement is reduce so that's why power optimization is occur.

In this thesis I use a apache server for browsing any file. Apache server is a http server. Apache is s a web server application notable for playing a key role in the initial growth of the World Wide Web. Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache quickly overtook NCSA HTTPd as the dominant HTTP server, and has remained the most popular HTTP server in use since April 1996. In 2009, it became the first web server software to serve more than 100 million websites. and it is available for a wide variety of operating systems, including Unix,FreeBSD, Linux, Solaris, NovellNetWare, OSX, Mic rosoft Windows, OS/2, TPF, OpenVMS and eComStation. Released under the Apache License, Apache is open-source software.

In this thesis, I work with source in side which is an application for windows os but when this application is run in Linux then it run through wine application. These wine application uses for windows application run in Linux. And also write a Linux character device driver[2] for opening, reading, writing and releasing the file. For blocking and non-blocking operation of character driver[3] which is also done by linux kernel[4] and it gives a four type of API for blocking and non-blocking operation. Character devices can be compared to normal files in that we can read/write arbitrary bytes at a time .They work with a stream of bytes.

For controlling hardware operation IOCtl is use IOCtl is a system call for device-specific input/output operations and other operations which cannot be expressed by regular system calls. So for input-output control operation IOCtl is use.

Ultimately above descript topic are main topic which are use for made and understand my project effectively.

II. OBJECTIVES

The objective of the proposed work is to study

(1) To develop a character device driver for streaming media from flash by implementing blocking and non-blocking read/write operation .
(2) Modifying ioctl for communication operation between hardware and device.
(3) analyzing daemon services
(4) making multi threading mechanisme for muti-user.

analyze the character driver in blocking an non blocking mode and daemon services features and its support that its offer to different hardware which has Ethernet cable or wifi port .

III. PROJECT BLOCK DIAGRAM

In the present implementation mechanism is show in the fig (a) in that the data is read from hard disk by this following steps:

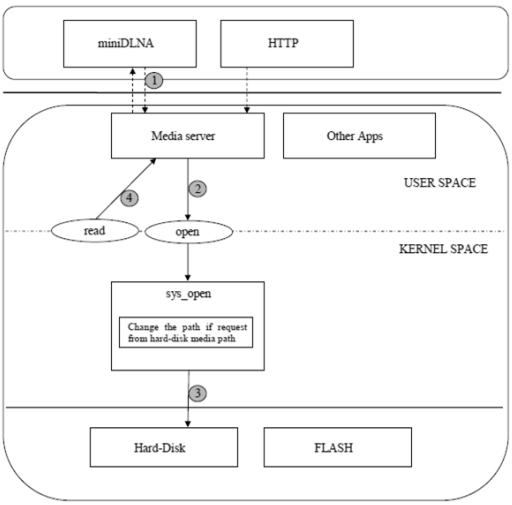

Fig. 1: Present Implementation Mechanism for media streaming

Client application request media content for streaming.

(1) Media server requests kernel to open the file.
(2) Kernel open the media file from hard disk
(3) Media server read the file from hard-disk and stream to client.

During that whole processes the data is reading continuously from hard disk so that' why the mechanical part of movement is increase so that's why the process is take a more power and that's why the alternative mechanism will implement so in that t the data is read from flash and hard disk will going to sleeping mode so that's why mechanical part of movement is reduce so that's why power optimization is occur.

An the alternative mechanism is show in fig b in that the data is read from flash rather than the hard disk by following steps

(1) Client application request media content for streaming .
(2) Media server requests kernel to open the file.
(3) Kernel checks if the file is requested from hard-disk media path. If yes go to step # 3.1, otherwise go tostep#3.6.
    (a) Lookup the media file and change the path of media file if required.
    (b) Generate a path change event and blocks for response.
    (c) User space daemon listen the event and start file copy from hard-disk to FLASH
    (d) Daemon gives path change response.
    (e) sys_open resumes.
    (f) Sys_open() opens the file from FLASH
    (g) sy_open resumes.
    (h) Sys_open() opens the file from FLASH.
(4) Media server read the file from FLASH and stream to client.

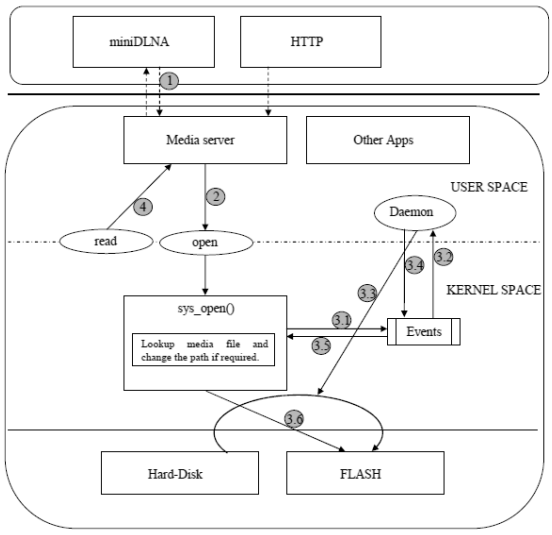

Fig. 2: Alternative Implementation Approach for media streaming

IV. TOOLS USED FOR DEVELOPMENT

A. *Hardware*

(1) Am335x series Board.
(2) PC with Linux supported OS (my case ubuntu 12.0.4).
(3) Sd card 4gb with reader.

B. *Software*

(1) VI, gedit and GCC.
(2) Source Insight.
(3) Minicom.

V. RESULT AND DISCUSSION

Project have been divided into two parts

- Front End
- Back end

Front End part taken out of having the basic of Linux environment, driver information, how to write a driver, how to load a module into kernel, data transfer mechanism, read the different kernel source code, study about character driver in blocking and non blocking mode, file system mechanism, daemon services, multithreading mechanism and modify open sys and ioctl

Back End part is totally base on hardware platform which enhanced to optimize more power by streaming media from flash and get the final output. Driver testing on the hardware kit also include this part. This part also contains the voltage outputs with the special APIs which are being carried out.

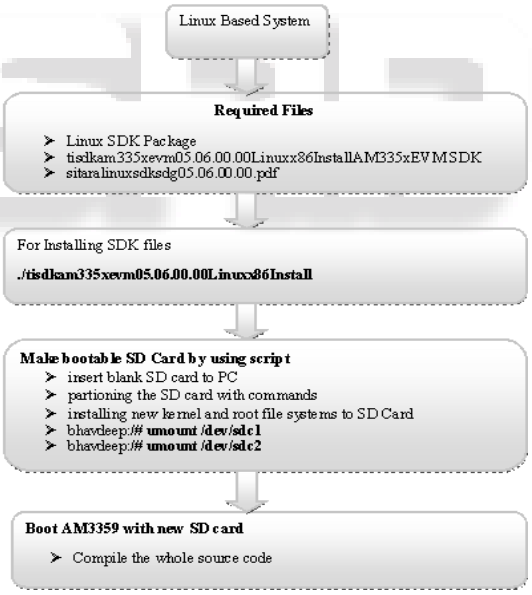

Fig. 3: Testing flow for the project

A. *Driver compilation and verification which generate uImage file.*

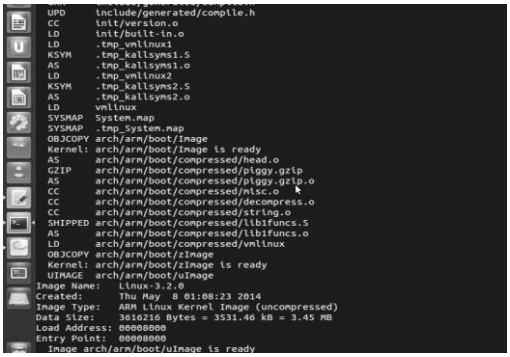

Fig. 4: Driver compilation and verification which generate uImage file

B. Final generated u-boot file after successful

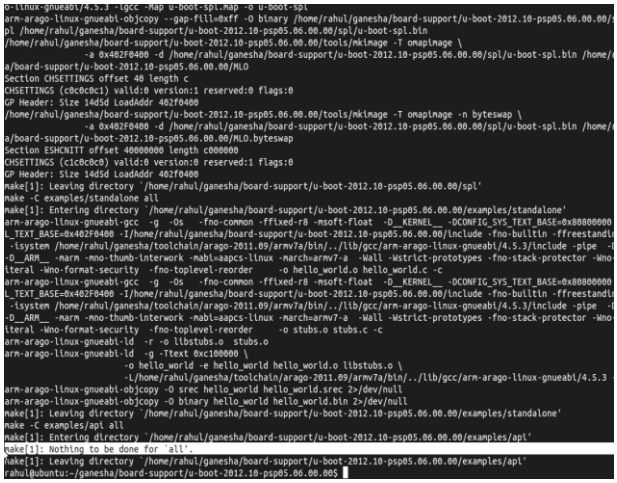

Fig. 5: Final generated u-boot file after successful

C. U-boot and x-loader final source

Fig. 6: U-boot and x-loader final source.

D. Hardware setup and cross compilation

1) Here I have used Am335x EV for the cross compilation.For the cross compilation I have follow the following steps:

Set path for the new hardware

Run minicom by usiong command sudo minicom.

Fig. 7: Run minicom by usiong command sudo minicom

2) After setting the path, Switch on Power button and the booting process is on.The name given if figure "Angstrom" indicates that booting process is over.

Fig. 8: the booting process

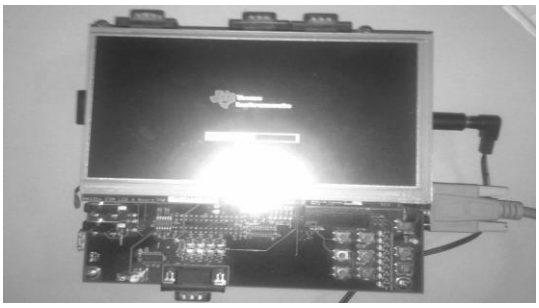

Fig. 9: booting process is over

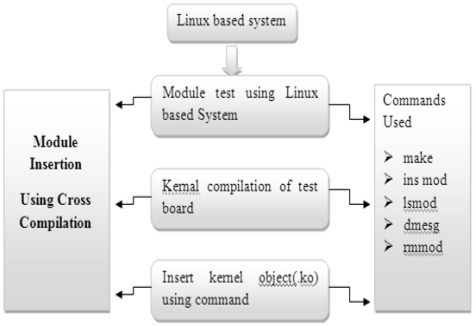

Fig. 10: module compilation process

E. How to load module into kernel

Step. 1 : Write any module in high language in C called module.c

Step. 2 : Compile it by make file.It creates .ko(kernel object)file.

Step. 3 : For insert into kernel type command sudo insmod.

Step. 4 : To show this module in kernel type demsg command.

F. Cross compilation process of any module using kit.

(1) Write Driver in high level language like C having the name module.c

(2) To enter into kernel type command sudo insmode

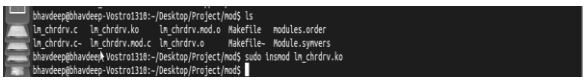

Fig. 11: Command line

(3) For viewing this the init function arguments type command dmesg

(4) then after create a interface and after that create a folder and change the path of media and run from ram.

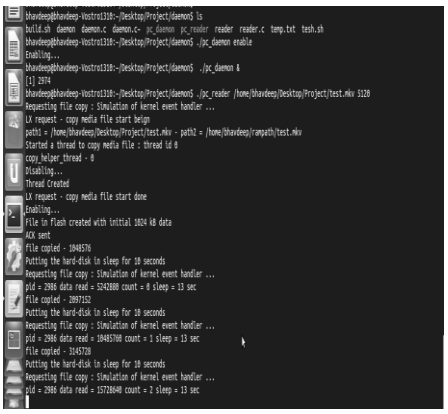

Fig. 12: the path of Code

(5) At the end the result is show that when copy is done from secondary memory(hard disk) to flash memory(ram) then after media(audio,video.txt) is running from flash memory and hard disk is going to the sleep mod.

VI. CONCLUSIONS

By analyzing the problems mentioned in my thesis about more power consumption in conventional methods of streaming media directly from Secondary storage memory, I will utilize flash memory of System to directly stream the media rather than from hard disk, by writing a character device driver in blocking and non-blocking operation which ultimately saves the power and thus optimization of power occurs. Thus the advantage will be the more battery life of the device by approximately 20 percentages more than the normal battery life of the device.

VII. FUTURE WORK

In embedded market there are coming new mechanism in that you can implementing this mechanism in your portable device and enhance power optimization while streaming media.

REFERENCES

Papers

[1] Yang Chuandong, Yu Zhenwei, Wang Xinggang, Zhang Junqing. A Survey of the End to End Transporting Technology of Streaming Media over Internet(J). Computer Engineering And Applications. 2005, 41(8)

[2] Baohua Song Linux Device driver development explanation Ve rsion 2 Beijing:Posts and Telecommunications Press 2010,pp. 248-260.

[3] JonathanJonathan Corbet ,Alessandro Rubini ,Greg Kroah-Hartman.Linux device drivers .Sebastopol O Reilly Media 2005

[4] SD Group, SD Memory Card Specifications Part 1 PHYSICAL LAYER SPECIFICATION Version 1.01, https://www.sdcard .org/ , 2001.

[5] A. Ganapathi, V. Ganapathi, and D. Patterson, "Windows XP Kernel Crash Analysis," in Proc. of 20th Large Installation System Administration Conference, 2006, pp. 149–159.

Book

[1] Peter jay salzman, Michael burian, Ori pomerantz, "The linux kernel module programming guide", opensource TLPD

[2] Linux Device Drivers,3rd Edition ,Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman,Oreilly media, February2005.

Website

[1] https://help.ubuntu.com/community/BinaryDriverHowto/ATI

[2] http://news.netcraft.com/archives/2011/10/06/october-2011-web-server-survey.html

[3] http://lwn.net/Kernel/LDD3/

