Australian Journal of Basic and Applied Sciences, 9(2) February 2015, Pages: 368-374

ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Design of a Boot Loader for Operating System

[1]Alycia Sebastian and [2]Dr. K. Siva Sankar

[1]*Birla Institute of Technology, Muscat Branch, Computer Science & Engineering* Department, Alycia Sebastian, P O. Box. 197, Muscat, Sultanate of Oman.
[2]*Noorul Islam University, Department of Information Technology, Dr. K. Siva Sankar, Thuckalay, Tamil Nadu 629180, India*

ARTICLE INFO

Article history:
Received 12 December 2014
Received in revised form 26 December 2014
Accepted 28 January 2015
Available online 1 February 2015

Keywords:
Boot loader, BIOS, Live USB, dynamic loader, OS

ABSTRACT

The growth in semiconductor technology has introduced storage devices with faster access time which leads to the design of different boot mechanisms to optimize the boot time of the operating system. The Boot Loader is the most crucial program for the initialization of the operating system. Whether it is PC, embedded system or smart phone its concept is same i.e., loading of OS. The boot process may either run from local hard disk or from external memory storage like USB memory, optical disk or network interface card with user interaction. This paper intends to explore the boot loading process and analyze the different ways of boot loading the system and discuss the need to dynamically boot load the OS from external storage to optimize the booting time of the OS. This paper shows the design of a dynamic Boot Loader to automatically identify the Live USB to remove the dependency on BIOS and lower the boot time of OS.

To Cite This Article: Alycia Sebastian and Dr. K. Siva Sankar., Design of a Boot Loader for Operating System. ***Aust. J. Basic & Appl. Sci.,*** *9(2): 368-374, 2015*

INTRODUCTION

In this technology world, a system performance is measured in terms of speed and efficiency. These features depend upon the type of machine and the type of operating system installed in that machine. The system type is based on the cost and the standards of manufacturers who build the machine. But the performance of operating system mostly depends upon the features provided by the loaders, as they are considered as the init of the operating systems.

The Boot Loader is the first software that starts running after BIOS. Boot loaders are responsible for initializing hardware and loading the kernel image into RAM. Boot loaders are highly processor and board specific (Raghavan *et al*., 2006) since its initialization is carried out by the boot loader. With increase in the advancement in computers, to include all its features, multi stage boot loaders are used. Processors like ARM and x86 fetches code from specific address after power on or reset. The boot program is stored in that address location in ROM or flash so that it runs every time the system is reset. Over the years, the advancement in ICs has made the possibility of using nonvolatile memory like NOR and NAND flash memory to store boot loaders due to its high density and low operating cost thereby reducing the load time of OS. The use of flash memory is encouraged rather than ROM because the boot program can be updated as per required features when in the system.

Operating system could not run without support of boot loader, so boot loaders for different architectures has been designed and used. A system is bootable from hard disk, floppy disk, optical disk or flash memory. The beginning of the boot-up process varies depending on the hardware platform being used. For a partitioned disk, the boot code is in the VBR of the active partition and for storage device like floppy disk that is not partitioned, the boot code is in the first sector of the disk. Boot loader transfers code from internal or external memory into RAM. However once the kernel is found and loaded by the boot loader, the default boot-up process is identical across all architectures.

The boot loaders are tailored as per the requirement; it may be specific features, easy portability or faster boot. The boot loader should be multi boot complaint i.e., it should be capable of loading any operating system available in the system. The multi boot loader provides user with interface to give options for user to select which kernel to be initialized.

This paper discusses about the initialization of the system known as the boot process and different

Corresponding Author: Alycia Sebastian, Birla Institute of Technology, Muscat Branch, Computer Science & Engineering Department, Alycia Sebastian, P O. Box. 197, Muscat, Sultanate of Oman.
E-mail: alycia.sebastian@gmail.com; GSM: 0096896595230

techniques adapted in the booting of the OS. Different boot mechanisms such as flash memory, hard disk, Ethernet or external USB disk are used. When boot loaded from external devices, BIOS setting is accessed to change the boot priority to the particular device. This change in boot settings every time you want to run from USB or optical disk results in extra boot time. This paper shows the design of dynamic Boot Loader to identify the Live USB and to boot load the OS.

1. Bootup process:

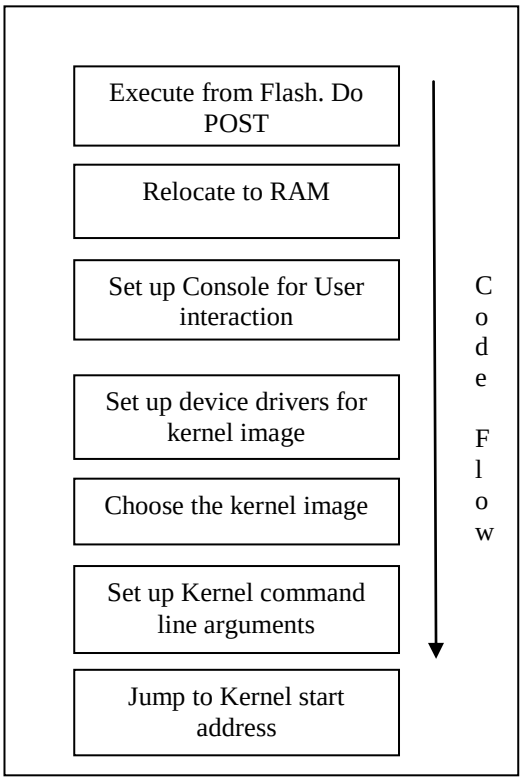

Fig. 1: Boot Loader boot up sequence (Raghavan *et al*., 2006).

2.1 BIOS:

After power on, Basic Input Output System (BIOS) at address 0xFFFFFFF0 performs self-test known as POST (Power on Self-Test) to identify and initialize CPU, memory, hard disk and device drivers. BIOS is a piece of code stored on a nonvolatile ROM on the motherboard. Now BIOS are stored in EEPROM, so that it can be rewritten without removing the chip (Winzent, 2012). The booting time is critical in many embedded system, so BIOS should run fast. Therefore depending on the type of booting, BIOS perform testing and initialization of the hardware. If the booting was warm boot i.e., using alt + ctrl + Del key combinations or restart button, then full POST process is not run to save time and avoid rewriting the codes which are already available. After POST, the BIOS runs the boot sequence to find the boot sector and load the MBR from the boot sector to memory to continue the booting process.

2.2 MBR:

The boot sector of hard disk contains Master Boot Record (MBR) which is of 512 bytes of which 446 bytes contains the primary boot loader program, 64 bytes is partition table which contains the partitions available in the disk and remaining two bytes marked as 0x55AA for error validation check for MBR.

The MBR is loaded at address 0000:7C000 after validation. MBR size limits the primary partition on the disk to four. The MBR code finds the active partition from the partition table and loads the code from its first sector to memory.

2.3 Boot Loader:

With only 512 bytes in boot sector, the boot loader program is divided into two stages. BIOS loads the primary boot loader in memory and its only function is to load the second stage boot loader from the storage medium. The second stage boot loader contains additional features like error checking, multi boot i.e., providing user interface to select OS if more than one OS kernel resides in the system and diagnostic services. MBR starts the primary boot loader in real mode and this then loads the kernel image, decompresses in protected mode and setup the environment for the OS to run and handover control to the OS.

|2.4 Kernel:

The kernel is a compressed binary image bzImage and not an executable file (Dokeun Lee, Youjip Won, 2012). The kernel image is a self-extracting file. When the kernel is loaded, it immediately initializes and configures the computer's memory and configures the hardware attached to the system, including all processors, I/O subsystems, and storage devices. The boot time is mainly determined by the time taken for loading the kernel image from external memory and its decompression. Significant time is taken for loading and decompression of kernel image.

The binary file head.S is to activate the virtual address by MMU and the misc.o code extracts the compressed kernel image which is the piggy.o. The loading and decompression of kernel image takes up the significant amount of booting time. Many techniques to reduce the boot time of the system have been designed by eliminating the decompression part by using directly uncompressed kernel data in flash memory.

2. Literature Review:

There are various implementations of the boot process. The Boot Loaders can be a basic boot loader, complex or a multi-boot. Because of limited capabilities in MBR and to add more features, the boot loader is stored in internal or external flash memory or optical disk. The main concern is to limit the boot time of an operating system. Much ongoing research is focused on taking advantage of the growth in nonvolatile memory and designating it as the primary boot device. This experiment shows positive results on the basis of boot time.

3.1 Booting From Local Memory:

An efficient boot loader is the most crucial component in any system to improve the performance of the system by effectively reducing the start-up time. The local memory can be ROM or a flash memory. In embedded system for immediate boot up, the boot code is stored in internal flash memory.

Optimizing the boot time has become a critical point in the initialization of the system. Non Volatile memory is gaining popularity due to its capacity to retain information after power off and power efficiency. The two important technologies PCM (Phase Change Memory) and memristors have made it possible to build cheaper NVRAMs with faster access time (Katelin Bailey *et al*., 2013).

With development in semiconductor technology, NVRAM are finding significant advantage over legacy DRAM. In paper (Wonsik Lee, Youjip Won , 2012) the authors has proposed to use a nonvolatile memory NVRAM in place of DRAM as main memory and storing one kernel image for booting. This eliminates the time needed for loading and decompression of kernel image. Here global objects are initialized for every booting to refrain from using previous values because of nonvolatile characteristics.

NVRAM is byte addressable like RAM. In terms of power consumption and performance NVRAM is attaining demand than DRAM but it is still in evolution stage and not completely adapted because of its limitations with the existing architecture. NVRAM is yet to be used for DRAM because of its performance limitations.

This paper (Minnich, R.G, 2004) discusses a new design of their own bootstrap using trusted OS after reload. This innovation reduces the risk of locating, verifying and loading a new OS image because here OS does the booting process. This tricky boot is implemented by opening and validating a new image and moving it to the kernel space, now prepare for hardware reboot and load the new kernel to the existing running kernel, and entirely move to the new kernel. This rebuilds the concept of tailoring the boot strap according to the requirements of the application. Even though boot mechanisms have all such advantages users still have to rely on the boot times. Some systems took more boot time to load the required functions than traditional booting process.

Resistive Memory technologies like RRAM-Resistive Random Access Memory are most likely to replace DRAM as the main memory in the coming years. It's a type of non-volatile memory and faster when compared with the traditional flash memory. In many companies like Hp and Crossbar Intensive research is going on to overcome the limitations of RRAM compatibility with present CMOS technology (10) and to develop a single chip that can store up to 20 terabyte of data and improves data transfer by 20 times (11). It is expected soon to build a high performance, high capacity and reliable memory to replace the DRAM.

Security has become an important concern in modern systems. The need to build a secure architecture has significantly increased over the years. The security for the system starts from the boot process, so a secure booting is needed. A secure Bootstrap process known as AEGIS guarantees a secure booting by validating the integrity of boot code (William Arbaugh *et al*., 1997). At each step, the integrity of each component is verified by doing hash function and comparing with the digital signature stored with each component before passing control over to the higher layer. The secure architecture is designed on the assumption that ROM, motherboard and BIOS are not compromised.

BIOS can be protected from unauthorized users by protecting with password. But BIOS password can be erased by physical removal of CMOS battery unless it is stored elsewhere. In a system where it contains sensitive data, BIOS password can be encrypted and stored in Trusted Platform Module (TPM) NVRAM (Kuan-Jen Lin,Chin-Yi Wang, 2012 Junkai Gu, Weiyong Ji, 2009). The TPM is required

to establish the identity of the MBR, then boot loader and so on forming a secure chain booting. Another most common attack on system is boot sector virus or MBR virus which replaces boot code with its own code thereby compromising the security of the system. To protect against virus attack boot programs can be stored in separate ROM chip (Umakant Mishra, 2012) and protected using encryption techniques.

3.2 Booting From External Memory:

A modest growth in the flash memory developed from EEPROM is the most commonly used storage medium. NAND Flash memory are commonly used in main memory, USB and SD card and NOR flash memory is finding itself an alternative for ROM.

Many techniques have been devised using nonvolatile memory either as external or internal memory to achieve optimal boot time by storing boot loader and running kernel image from the flash memory. A new method known as Hybrid-Execute-In-Place Architecture have been implemented (Tony Benavides *et al*., 2007) to reduce the Boot Time to some extend by storing and executing the code and data in a NOR flash memory designated as an external memory and only the required code is brought to RAM for execution. Such a design provided more efficiency in boot loaders. Even though NOR flash memory produces faster random reads but it is slower in block writes (Ken Curewitz, 2011). The rise of VLSI technology made the development of embedded loaders into a single chip.

Due to the rapid development in USB memory and its suitability to store large amount of portable data has resulted in rise in using USB as the boot device to create a portable environment. The user can use live USB in any system to create his own personalized secure environment. The author Anil Kumar Karna has proposed that an External USB hard disk can be used as an installation resource for various operating systems and also as a bootable media. It is the best substitute for laptop to carry various operating systems along with other personal data. The external USB hard disk is loaded through BIOS for both installing and booting an operating system (Anil Kumar Karna, 2010).

Protecting the system from unauthorized users is another concern that needs to be addressed. Security of a system should starts from the boot up process. Securing the boot up will provide a safe and reliable environment for the users. A new portable security system designed by incorporating syslinux boot loader and customized puppy loader into plug and play USB 2.0 and integrating with fingerprint authentication to provide security (GuodongLi , HuChen, 2010). The USB is hardware encrypted and authorization information is stored in USB rather than computer memory. It is a combination of both hardware and software.

UEFI (Unified Extensible Firmware Interface) was developed by Intel to replace BIOS to provide a standard booting procedure. A secure architecture is designed using USB as the TPM and booting media (Abhishek Singh Kushwaha, 2013). In this method the ESP (EFI system partition) is implemented inside the USB to store the boot loader and make it as the primary boot mechanism.

USB is fast evolving as the popular plug and play storage device with its ability to store up to one terabyte data (9) and its nonvolatile characteristics. Since USB doesn't have any moving parts, it's more reliable and durable. But the flash memory in USB faces limited read and write cycles of up to 10,000 to 100,000 for multi and single cell respectively (Zhou Di-bo, Pan De-lu, 2009, Anand Lal Shimpi, 2008). With technology improvement, some companies are providing warranty for USB for 5 years. Along with encryption, USB provides a safe, secure and portable environment.

The boot loaders stored in flash or ROM is slower, so a new design using Scratch-Pad Memory (SPM) is implemented to improve booting speed by loading from Scratch-Pad Memory (SPM), which is small, isolated and located on a single chip. The second stage boot loader is stored in SPM in this design and loaded using a SPMOS loader. This made the booting speed to some extent. They implemented this technique in a Novel loading process. In this technique boot loader initializes the hardware and the complicated functions are performed by the SPM (Nan Zhang *et al*., 2008). But this chip based loaders faced serious disadvantages. If we need to modify this loader it needs a flash or a new chip. So researchers focused on a programmable loader.

3.4 Booting Through Network:

Network booting is becoming popular in large organization where the booting of the system is carried out through network rather than from the local drive. This type of booting is advantageous in terms of maintenance overhead and cost saving and also the user can be given the new version with minimal effort. From various experiments conducted on network booting with hard disk booting, hard disk booting is recommended (John M. Ostrowick, 2004) for better performance. Since all the users in network booting have to rely on a single boot server which causes network congestion leading to more boot time than hard disk booting.

For network booting, DHCP (Dynamic Host Configuration Protocol) and TFTP (Trivial File Transfer Protocol) servers are required (Sumanth Vidyadhara, Arun Patil, 2006). DHCP responds to request by providing TFTP server location and the filename to be downloaded to start the booting process.

The authors have proposed to use Ethernet boot loader to boot the ARM processor from external flash. In this the boot loader in MBR is altered to

include the external device address to automatically transfer the control to continue with boot loading. The system must have an Ethernet chip to be controlled over LAN. The boot loader program is stored in the external memory (Sayali A. Kulkarni *et al*., 2006).

The Network booting is achieved (FEI Luo *et al*., 2013) by designing an intelligent booting system with distributed modules which automatically performs the virtual partition on the client system and performs loading and passing of the boot parameters for boot process.

4. Dynamic Boot Loader:

Boot loaders are simple code that loads the OS. Every loader has its own specifications. It may be small, large or may be multi-boot. But even with its characteristics they have to rely on BIOS as their root because of its capability to load every loader of all operating systems. No loaders are BIOS independent. Moreover the operating systems in-cooperating such loader takes a huge disk space when installed. And if to be a portable one the disk space consumed will be more.

In many boot loading techniques discussed above, the secondary boot program is from external device. Normally the first priority is for floppy disk, then hard disk and so on. The boot order can be changed using CMOS settings. When you have to boot load from USB, every time you start the system, BIOS boot order need to be changed to USB by the user as the boot option to continue booting. The end user has to have knowledge about BIOS settings to change the boot priority to USB.

The main booting process time depends on the operating system being used. For example, puppy Linux average boot time is 26 seconds and Ubuntu has the fastest boot time of 10 seconds (Jack Wallen, 2011). With external booting, time is spent in changing the boot sequence which depends on the knowledge of the user in computer settings which will considerably affects the boot time of the OS. Not all end users have this knowledge of changing BIOS settings. So, it should be possible to boot automatically from the USB without any setting change. There is a need to develop a dynamic loader which can automatically detect the boot media so as to remove the dependency on BIOS which will sufficiently reduce the boot time of the system.

4.1 System Design:

The dynamic boot loader is designed to provide maximum flexibility which when boot loaded ignores the BIOS boot priority settings and displays all installed operating systems both in USB drive and hard disk and Live-USB is boot priority first.

Dynamic Boot Loader is implemented using VC++, WMI (Windows Management Instrumentation) code creator and Shell Programming.

Windows Management Instrumentation (WMI) is a core Windows management technology that is used to get information about the internal state of computer systems, much like the disk drive information, network configurations, operating system detail etc. This system objects are modeled using classes such as Win32_LogicalDisk, Win32_DiskDrive, Win32_NetworkAdapter, Win32_NetworkAdapterConfiguration and Win32_OperatingSystem (12).

Table 1: Description of WMI Classes.

S.NO	CLASS	DESCRIPTION
1	Win32_LogicalDisk	represents the logical disks installed on a computer
2	Win32_DiskDrive	represents a physical disk drive as seen by a computer running the Win32 operating system
3	Win32_NetworkAdapter	represents a network adapter of a computer running a Windows operating system
4	Win32_NetworkAdapter Configuration	represents the attributes and behaviors of a network adapters
5	Win32_OperatingSystem	Returns an instance for the currently active operating system and list of operating system installed.

These classes help to identify the disk drives in the system. This information helps to boot OS from Live USB. The equation to identify the Live-USB drive is

Live-USB = LogicalDisk(Type 5) + LogicalDisk(Type 2) -1

where LogicalDisk(Type 5), LogicalDisk(Type 2) € N

$0 <$ LogicalDisk(Type 5) <3

$0 <$ LogicalDisk(Type 2) $<= 127$

$1 <$ LiveUSB < 133

LogicalDisk(Type 5) – Number of hard disk present in the system

LogicalDisk(Type 2) – Number of USB disk present in the system

The dynamic boot loader includes the above WMI classes to gain information about the USB device dynamically when used in a system.

A three stage boot loader is designed. Stage 1 resides in the MBR and stage 1.5 is installed between MBR and first partition. Stage 1.5 interprets the different types of file systems. Everything is considered as file. So the physical location of the Kernel image is not required. Only the device number and the partition number are required to

identify the boot sector. For example (hd0, 0) means the first hard disk and first partition.

Stage 2 loads the configuration file which displays the different kernel images installed in the system. The menu screen is displayed which allows the user to select the OS as per their need. For the dynamic boot loader, the configuration file is written in such a way that the Linux in USB will be the default OS to be loaded in to memory if no interaction.

The below code shows the configuration file to boot load the Linux OS from USB.

```
default=0
Timeout=3
title Linux Operating System
root (hd1,0)
kernel  vmlinuz append root=/dev/ram0
initrd  /initrd.gz
# To load Windows
title Windows
rootnoverify (hd0,0)
chainloader +1
```

This configuration files allows the Linux to load from Live USB after a timeout of 3 seconds. This setting gives the user the ability to load Linux OS from USB as well as windows from hard disk. The user only has to select which OS from the menu instead of changing the BIOS settings for external booting.

A 128MB Flash memory is selected for storing light weight Linux OS. Taking into consideration the limitation in read/write cycles of flash memory, the OS is especially designed to have no writes to the Flash drive during a session, enormously extending its life span.

4.2 Performance Analysis:

In normal boot process, BIOS transfers control to the device as per the order in the boot menu and then the boot loader starts its process of loading OS. In the existing boot loader, the boot time from external memory depends on the user interaction time i.e., time taken to change the BIOS settings to USB. The dynamic boot loader is designed with a menu interface, listing all the available kernel entries for easier selection thereby optimizing the booting time. If any new OS is installed and changes are made in the configuration file, the boot loader automatically updates and displays the new OS in the menu because the boot loader is dynamically configurable. As explained before, disk drive and partitions are considered as files.

5. Conclusion:

*Ope*rating system relies on the support of the boot loader to run. With the technology improvement, boot loaders are now commonly stored in external Flash memory. With the available boot loaders, the booting process from external device is dependent on BIOS. It is not anticipated that all users who use a portable Live USB will have the knowledge of BIOS settings. The dynamic boot loader is designed independent of BIOS, so it automatically identifies the boot mechanism and starts the booting process. The dynamic boot loader will effectively improve the boot time of the system.

REFERENCES:

Abhishek Singh Kushwaha, 2013. “A Trusted Bootstrapping Scheme Using USB Key Based on UEFI”, International Journal of Computer and Communication Engineering, 2(5).

Anand Lal Shimpi, 2008. “ Intel X-25 M SSD: Intel Delivers One of the World’s Fastest Drivers”.

Anil Kumar Karna, 2010. “Multipurpose USB Hard Disk: Your Mini Laptop”, Second International Conference on Information Technology and Computer Science, 357-360.

Bruce Liao, 2013. “Customizing Boot Media for Linux Direct Boot, Intel Corporation”.

Dokeun Lee, 2012. Youjip Won,”Bootless Boot: Reducing Device Boot Latency with Byte Addressable NVRAM”.

Luo, F.E.I., 2013. “A Novel Intelligent Network Booting System”, Advanced Materials Research, 658: 497-501.

GuodongLi, HuChen, 2010. “ A New High-Level Security Portable System Based on USB Key with Fingerprint”, International Conference On Computer Design And Applications (ICCDA 2010), 159-162.

https://www.ibm.com/developerworks/library/l-linuxboot.

http://en.wikipedia.org/wiki/USB_flash_drive#Booting_operating_systems

http://www.crossbar-inc.com/technology/resistive-ram-overview.html

http://venturebeat.com/2013/08/05/crossbar-says-it-will-explode-the-60b-flash-memory-market-with-resistive-ram-which-stores-a-terabyte-on-a-chip/

http://msdn.microsoft.com/en-us/library/aa392727(v=vs.85).aspx#_hmm_drivers

John, M., Ostrowick, 2004.”Network Booting versus hard disks: Costs and Implications”, Conference for IT in Tertiary Education.

Junkai Gu, Weiyong Ji, 2009. “A secure bootstrap based on trusted computing, International Conference on New Trends in Information and Service Science”, 502-504. ((Jack Wallen , “The Five fastest- booting Linux distributions”, 2011.

Katelin Bailey, 2013. “Operating System Implications of Fast, Cheap, Non-Volatile Memory”, Proceedings of the 13th USENIX conference on hot topics in operating systems.

Ken Curewitz, 2011. Booting from NOR/NAND/Serial Flash/e·MMC™, Micron Technology.

Kuan-Jen Lin, Chin-Yi Wang, 2012. "Using TPM to Improve Boot Security at BIOS Layer" **,** IEEE International Conference on Consumer Electronics (ICCE), 376-377.

Minnich, R.G., 2004. "Give your bootstrap the boot: using the operating system to boot the operating system, International Conference on Cluster Computing", 439-448.

Nan Zhang, 2008. "SPM-based Boot loader", The 2008 International Conference on Embedded Software and Systems Symposia (ICESS2008), 164-168.

Raghavan, Amol Lad, Sriram Neelakandan, 2006. "Embedded Linux System Design and Development".

Sayali, A., Kulkarni, 2006. "Ethernet Boot loader for ARM Processor", International Journal of Scientific & Engineering Research, 4-11.

Sumanth Vidyadhara, Arun Patil, 2006. "Best Practices in Boot Loader Design, Emdedded Systems Conference-San Jose.

Tony Benavides et.al, 2007. "The Implementation of a Hybrid-Execute-In-Place Architecture to Reduce the Embedded System Memory Footprint and Minimize Boot Time", IEEE International Conference on Information Reuse and Integration, 473-479.

Umakant Mishra, 2012,"Detecting Boot Sector Viruses- Applying TRIZ to Improve Anti-Virus Programs".

WinZentTechnologies, 2012. http://www.winzenttech.com/files/Download/WinZent/WhitePaper/BIOS/ for/Embedded.pdf.

Wonsik Lee, Youjip Won, 2012. "Booting Linux Faster" , Proceedings of 3rd IEEE International Conference on Network Infrastructure and Digital Content, 665-668.

William Arbaugh, 1997. "A Secure and Reliable Bootstrap Architecture", Proceedings of the 1997 IEEE Symposium on Security and Privacy, 65.

Zhou Di-bo, Pan De-lu, 2009. "A method for developing plug-and-play Web GIS , International Conference on Environmental Science and Information Application Technology", IEEE, 377-380.