ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 9, Issue 1 , January 2022

Analysis of Protected File Systems in Operation Systems with Open Source

Ochilov N.N

PhD, State Testing Center under the Cabinet of Ministers of the Republic of Uzbekistan, Uzbekistan, Tashkent

ABSTRACT: The article discusses secure file systems in open source operating systems, as well as various types of file blocks and inod algorithms and mechanisms for their operation in a secure file system, as well as tests on various sizes. The article considers algorithms for creating secure file systems, directories and journals in a file system. An FFS can have three parameters: data, logs, and time-space data, which includes location data and system file metadata, and log space.

KEY WORDS: blocks, directory, header, journal, log files, integrity, attribute, hash.

I. INTRODUCTION

An increasing number of government agencies and commercial firms are beginning to use alternative free software. In particular, many businesses and organizations are starting to use free Linux operating systems instead of the paid Windows operating system. A file system that is part of the operating system is used to create documents and perform various operations [4]. One of the features of Linux operating systems is that they support a file system that runs on large chunks of hard disks, easily measures thousands of files, and works efficiently with files of different sizes. Operating system users mainly work with file systems provided by the system. They rarely create new disk partitions and think less about their settings, and they use only the recommended settings or use pre-formatted ready-made external memory. There are more than a hundred types of file systems, but they consist of metadata and V-trees with the same type of structure. As in any hierarchical system, the V-tree begins with a root entry and is then broken down into final elements - files and their attributes or "leaves". The main purpose of creating such a logical structure is to speed up the search for file system objects in large dynamic arrays - a few terabytes of solid or more powerful RAID arrays [1]. V-trees require much less disk access than other types of V-trees when performing the same operations. This is achieved because the final objects in the V-trees are hierarchically located at the same height and the speed of all operations is proportional to the height of the tree. Like other balanced trees, V-trees have the same path length from root to any leaf.

II. SIGNIFICANCE OF THE SYSTEM

The issues of increasing performance on file systems are now related to the reliability of file systems and data loss. PFS can recommend itself well in this regard, but difficulties in creating PFS, checking, or copying a petabyte file system to run a data bit program are not practical. The file system that is being created should also be able to work with these types of files. This will require an emergency detection tool installed in the file system to verify the metadata immediately.

III. LITERATURE SURVEY

Scientists such as L. Torvalds, R. Herzog, B. Kernigan have conducted research on the development of operating systems belonging to the Linux family. R.Payk, B.Ward, D.Barrett, S.Alapati, A.Robachevskiy, D.N.Kolisnechenko, M.Flenov, S.Nemnyugin, O.Stesik, T.Adelshtayn, B.Lyubanovich, S.L. Research work of foreign and domestic scientists, such as Sklovskaya, in the field of creating a system of data protection in the Linux operating system has been studied. Astra Linux, Zarya operating systems (OS) for the development of information security tools and methods, various methods, models and algorithms of encryption, security tools, theoretical and practical principles of information protection in the scientific development of the CIS countries and the Republic of Uzbekistan, " Alt Linux "and" ROSA "operating systems were studied. Doppix graphical shells used in various government agencies have been studied.
In scientific articles of H.A.Muzaffarov and A.Ikramov the methods of creation of encryption and protection systems with algorithm GOST 28147-89 are studied. In the scientific work of D.N.Kolisnechenko and V.Allen, in all issues of

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 9, Issue 1 , January 2022

Linux Format magazine in 2014, 2015 and 2016, Linux OT kernel algorithms and structure, packet data processing speed, security models, security tools were studied.

IV. METHODOLOGY

The Linux kernel supports the new ext4 file system from version 2.6.28. The ext4 file system is an improved version of ext3, the difference being the presence of a log system on the system. ext4 has the following features:

- can be compatible with ext3. It is possible to switch from ext3 to ext4 with a few commands without having to reformat the disk or reinstall the system. The initial ext3 data structure remains the same and ext4 affects the new data. Thus, the entire file system will have a large volume that works with ext4;

- large file system and large files. Compared to the 16 TB maximum file system and 2 TB maximum file currently supported by ext3, ext4 supports 1EB file systems (1,048,576 TB, 1 EB = 1024 PB, 1PB = 1024TB) and 16 TV files, respectively;

- unlimited number of small catalogs. ext3 currently supports 32,000 subdirectories. ext4 supports an unlimited number of subdirectories;

- extensions. Ext3 uses indirect block customization, which is very inefficient when working with large files. For example, for a 100 MV file, ext3 creates a mapping table consisting of 25,600 data blocks (each data block is 4 KV). ext4 introduces the concept of extensions that are popular in modern file systems. Each extension is an adjacent set of data blocks. The above file data is stored in the next 25,600 data blocks, which significantly increases efficiency [2];

- multi-block distribution. when writing data to an ext3 file system, the ext3 data separator can only allocate 4 KV blocks at a time, and to write a 100 MV file, it must refer to the data block splitter 25,600 times, while the ext4 multiblock allocator (mballoc) supports multiple block separation;

- distribution delay. ext3's block allocation strategy is to distribute as quickly as possible, while ext4 and other modern file system strategies should delay partitioning as much as possible and start writing data blocks to disk before the file is cached, a method that can optimize the entire file system;

- fast FSK. Previously, the first fsck step was very slow because it had to check all inodes. ext4 now adds a list of unused inodes to each group's inode table;

- inspection log. The log is the most commonly used part, it can easily cause disk drive corruption, and recovering data from a broken log can lead to more data corruption. ext4 log checking can easily determine if log data is corrupted and integrates ext3's two-step log recording mechanism into one step, which in turn increases security [3];

- properties related to inodes. ext4 supports larger inodes The standard inode size is 128 bytes compared to ext3. It has a 256 byte inode size on ext4 to accommodate extended attributes (e.g., nanosecond time or inode version). ext4 also supports fast extended attributes and inode band;

- The disk is equipped with an internal cache to reset the order of data recording operations in batch mode and optimize production.

Creating PFS systems is a very complex process, and good results can be achieved in the process of processing large and medium-sized files by using thought-out disk space-sharing algorithms and managing user queries with effective parallelism methods [3].

The PFS system can allocate a special partition on the disk in real time, where simplified access mechanisms and distribution of inodes are performed to stabilize the execution time of operations performed on blocks. The Linux operating system does not have such capabilities, so real-time algorithms and mechanisms are offered.

The contents of the directory in the PFS file system are stored directly in the inode or addressed through the file tree. All operations in the catalog structures are logical, i.e. the file system refers to the file map to find the physical state of some elements and makes the appropriate changes.

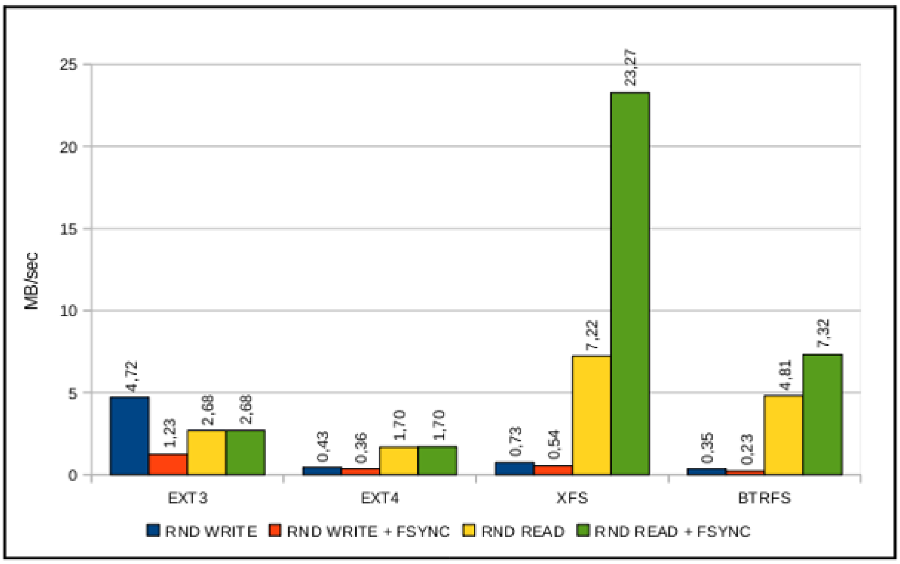
If the catalog does not fit into a single block, it expands the tree structure for storage in PFSs, which contain data from the catalog elements. It consists of filenames and blocks containing index information in their inodes and data blocks, and the names and hash values of the corresponding elements within the directory. Empty blocks are attached to these structures. The logical space inside the directory file is considered to be 8-bit words. These structures can only be called tree structured systems [2]. In fact, in the PFS system, directories are indexed using hashing. All of this causes PFSs to functionally lag behind extended file systems such as reiserfs.

The PFS system can allocate a special partition on the disk in real time, where simplified access mechanisms and distribution of inodes are performed to stabilize the execution time of operations performed on blocks. The Linux operating system does not have such capabilities, so real-time algorithms and mechanisms are offered.

The issues of increasing performance on file systems are now related to the reliability of file systems and data loss. PFS can recommend itself well in this regard, but difficulties in creating PFS, checking, or copying a petabyte file system to run a data bit program are not practical. The file system that is being created should also be able to work with these types of files. This will require an emergency detection tool installed in the file system to verify the metadata immediately.

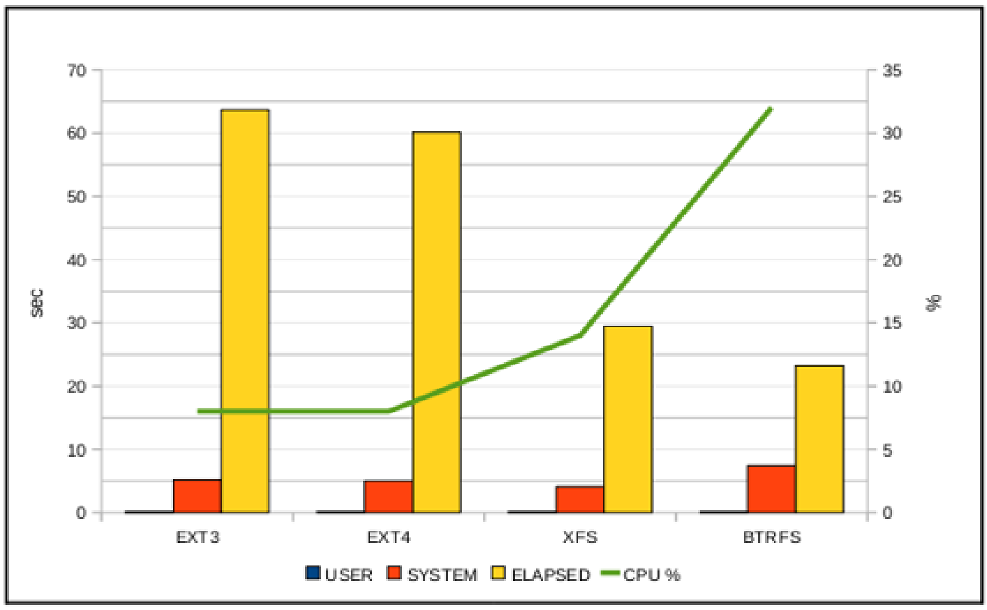
V. EXPERIMENTAL RESULTS

The random read and write capability of file systems is 100 requests per second - 16 KV for each block. We find that the maximum read speed is 1600KV / s. XFS and BTRFS exceed this value (perhaps the data may not be random or the buffer will affect the result) (Figure 1).
ext3 random write indicator is best suited for database processing and high volume virtual machine systems. As a test, you can also see that in the Linux operating system, the cat command works on file systems (Figure 2). During the process of opening the files of the kernel of the Linux operating system, the creation speed of 32,000 files was evaluated on the efficiency of processing in file systems (Figure 3). In addition, there is an analytical comparison table on information security in file systems (Table 1).
Today, the XFS file system is used in the storage and processing of large amounts of data in many well-known scientific institutions.
Apparently, it is advisable to use the XFS file system to develop and create PFS (protected file system). The XFS file system can support large volumes of files and ensure good streaming I / O (input / output) performance. It is also used by other Linux file systems.
The presence of a subsystem to protect files stored in the operating system significantly increases the overall level of protection of the operating system.

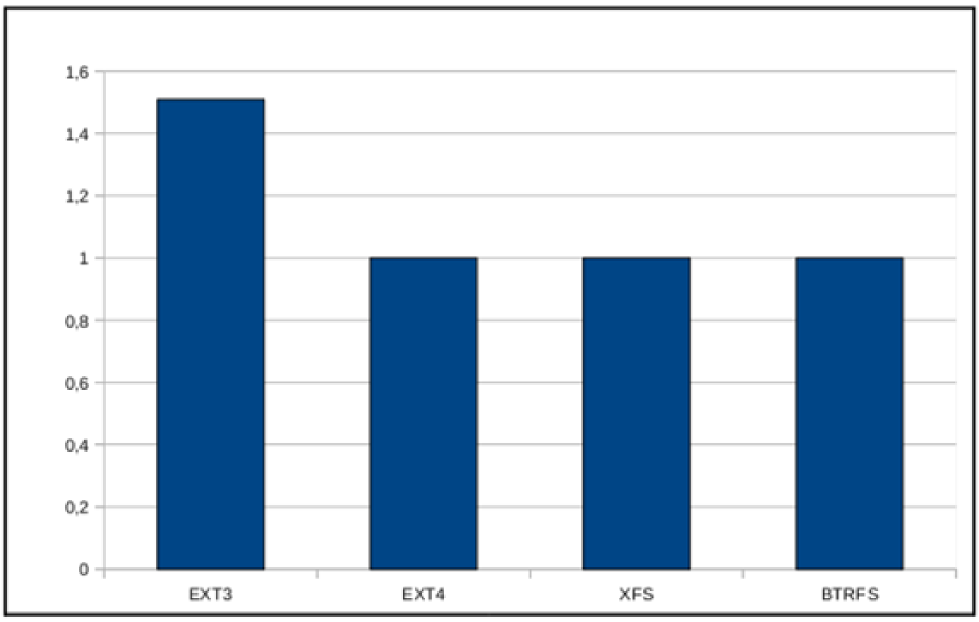


1. Figure. Random read and write analysis of file systems

No matter what perfect access control tools are used in operating systems, these tools only work on subjects that are available in the OT model. The intruder can simply remove the hard drive from the PC and analyze the stored data. Protected file systems (PFS) are used in practice to protect against this type of threat.



2. Figure. Analysis of the performance of the cat command on file systems



3. Figure. Estimate the creation speed of 32,000 files

We also evaluate the efficiency of opening files in Linux 2.6.36 kernel version (Figure 4).
The high efficiency of EFS is due to the use of different types of encryption algorithms. It is known that symmetric encryption has high speed properties. EFS uses 3DES or AES symmetric algorithms to encrypt the main volume of the file system. The use of symmetric encryption inevitably leads to the need to solve basic data management problems in the operating system, and they are:

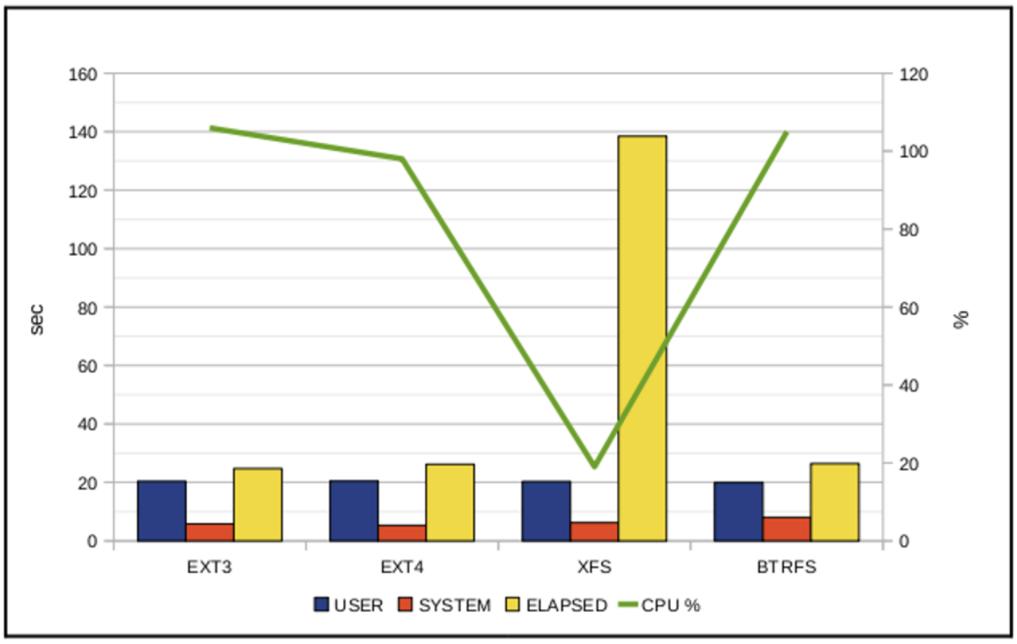
- storage of key data for each file (it is not advisable to use a single key to encrypt all files);

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 9, Issue 1 , January 2022

- Inability to share a single file.

To solve these problems, EFS uses asymmetric encryption algorithms to encrypt file encryption keys. The use of asymmetric encryption provides:

- secure storage of file encryption key;
- Integration with corporate public key infrastructure (PKI).



4. Figure. Evaluate file opening in Linux 2.6.36 kernel version

One of the most popular protected file systems is the Microsoft Encrypted File System (EFS). We will learn how to build a similar PFS on a Linux operating system. The EFS file system is very simple and the resulting algorithm is transparent [5].

Table 1. Information security in file systems

File system	Access rights	Data encryption
FAT16	no	no
FAT32	no	no
exFAT	ACL	no
NTFS	ACL	yes
Ext3	Unix, ACL	no
Ext4	POSIX	yes
XFS	POSIX, ACL	yes

Figure 5 shows a block diagram of the file encryption algorithm in the EFS file system. The process of encrypting files consists of the following steps:

- create a file encryption key (individual for each file);

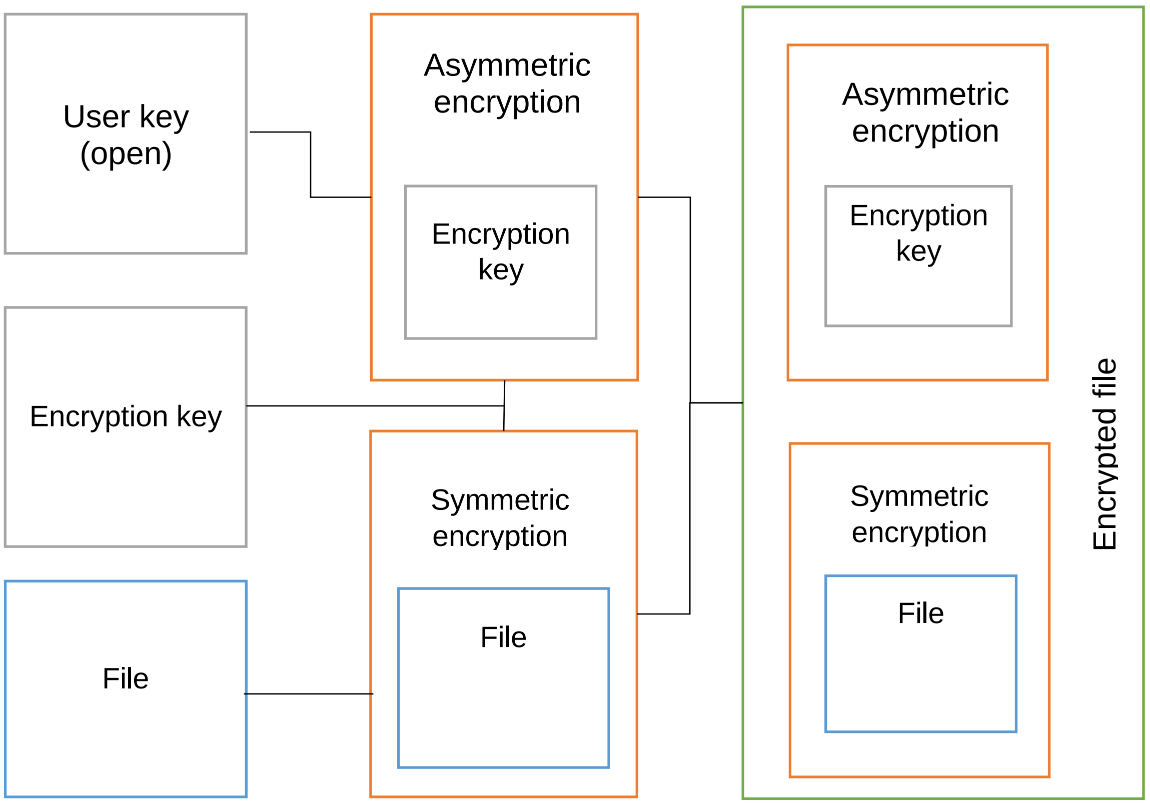
ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 9, Issue 1 , January 2022

- Encrypt the file body with a file encryption key (symmetric encryption algorithm);
- Encrypt the file encryption key with the user's private public key;
- Attach the encrypted file encryption key to the encrypted body of the file.

The above steps are performed in EFS and are transparent to the user. The performance of modern processors makes the process of encrypting files invisible to the user (Intel processors now have their own cryptographic processor). The following to decrypt the files



5. Figure. EFS file encryption algorithm

It is necessary to complete the steps:
- extract the encrypted file encryption key from the file body;
- decrypt the file encryption key using the user's private key;
- decrypt the file using the received encryption key.

EFS is a simple and effective secure file system. The operating system uses the concept of "all objects file". Therefore, to ensure the security of the operating system, it is necessary to start with the security of the file system first. Modern operating systems use a virtual file system.

The structure of a virtual file system is based on an analysis of existing file systems. Because this system is virtual, it does not have the following functions, such as file systems:
- registration (to organize the reversal of actions);
- creation of loading environment (loading sector);
- implementation of various types of data backup tools.

All of the above functions are performed at low levels (virtual). Protecting stored data is just as important as protecting file system objects (files and directories). Therefore, the following requirements are placed on the virtual file system during the creation phase:

- Ensuring the security of stored data through cryptographic modification;
- the ability to implement different methods of distinguishing access to file system resources (access tabs for objects in the file system, access matrices, etc.).

Table 2. File title

№	Name	Function
1	char name [16]	The symbolic name of the file
2	U16 flag	File description (file type, storage location, etc.)
3	U8 metka	Reserved
4	U8 ovner	File owner
5	U8 group	File owner group
6	U32 size	File size (number of building blocks)
7	void * fo	"Open" file icon (link to open operation title)
8	void * p	Link to the file body

A file system consists of two types of file objects: actually files and directories. In the context of a virtual file system, a file is a link. This link contains a file header that contains information about the file (Table 2).
The directory structure of a file system is a directory tree that starts from the root directory. The root directory is a special dynamic directory. The structure of the root directory is also formed during the startup of the operating system. The file header is 17 bytes in size. Table 3 shows the catalog structure of the developed PFS system. Depending on the actual location of the data (internal memory, external storage device), the values of the internal directory addresses are written to the body of the main directories.

Table 3. The structure of the file system

№	Name	Function
1	/	Root directory
2	/etc	Basic application settings directory
3	/mnt	Installation directory
4	/mnt/CSA	Catalog of secure storage areas for cryptographic containers
5	/mnt/0	External memory installation location
6	/bin	Application catalog
7	/proc	RAM

The security mechanisms used to prevent unauthorized access to files depend on the data stored and the physical storage location.

V. CONCLUSION AND FUTURE WORK

In the protected file system, a file system with a B + file tree structure and a tree structure program were proposed. Storage within 32-bit values of indicators is one of the main mechanisms for using allocation groups. On average, each separation group is 0.5-4 Gb in size and has its own data structures to manage the placement of inodes and blocks within its boundaries.
In the protected file system, file blocks and inodes algorithms and their operation block diagram were tested and evaluated in different sizes. The results show that the proposed algorithm for reading and writing files from fixed PFSs from the volume of data increases the efficiency of work on systems running on ext3 / 4 by about 0.5 times.
It is advisable to use the XFS file system to develop and create PFS (protected file system). The XFS file system can support large volumes of files and ensure good streaming I / O (input / output) performance.

REFERENCES

[1] Silicon Graphics Inc., XFS Filesystem Structure, 2nd edition;
[2] Dave Chinner and Jeremy Higdon, “Exploring High Bandwidth Filesystems on Large Systems,” Proceedings of the Ottawa Linux Symposium 2006;

[3] Silicon Graphics Inc., XFS Overview and Internals;
[4] Dave Chinner and Barry Naujok, Fixing XFS Filesystems Faster;
[5] Dr. Stephen Tweedie, “EXT3, Journaling Filesystem,” transcript of a presentation at the Ottawa Linux Symposium 2000;

AUTHOR’S BIOGRAPHY

	Ochilov Nizomiddin Nazhmiddin ugli, was born September 30, 1991 th year in Tashkent city, Republic of Uzbekistan. Has more than 30 published scientific works in the form of articles, journals, theses and tutorials. Currently works as head of the department, State Testing Center under the Cabinet of Ministers of the Republic of Uzbekistan.