

O. I. Morozova, D. D. Uzun

OPERATING SYSTEMS

Part 1

SETUP AND CONFIGURATION

2021

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
National Aerospace University
“Kharkiv Aviation Institute”

O. I. Morozova, D. D. Uzun

OPERATING SYSTEMS

Part 1

SETUP AND CONFIGURATION

Course study guide

Kharkiv “KhAI” 2021

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
National Aerospace University
“Kharkiv Aviation Institute”

O. I. Morozova, D. D. Uzun

OPERATING SYSTEMS

Part 1

SETUP AND CONFIGURATION

Course study guide

Kharkiv “KhAI” 2021

UDC 004.451(075.8)
M80

Викладено базову теорію налаштування та конфігурації операційних систем. Показано особливості операційної системи Windows. Наведено повний та чіткий опис лабораторних робіт з дисципліни.

Для студентів вищих навчальних закладів спеціальностей 123 «Комп'ютерна інженерія», 125 «Кібербезпека».

Reviewers: Doctor of Sciences, Professor K. O. Meteshkin,
Doctor of Sciences, Professor N. V. Sharonova

Morozova, O. I.

M80 Operating systems [Text] : course study guide. In 2 p. P. 1. Setup and configuration / O. I. Morozova, D. D. Uzun. – Kharkiv : National Aerospace University “Kharkiv Aviation Institute”, 2021. – 108 p.

ISBN 978-966-662-809-4

Basic theory for setup and configuration of operating systems are introduced. Features of the Windows operating system are shown. Complete and clear descriptions of laboratory works in the discipline are included reinforce the important points.

For students of higher educational institutions training in specialties 123 “Computer Engineering” and 125 “Cybersecurity”.

Figs 38. Table 1. References: 7 numbers.

UDC 004.451(075.8)

ISBN 978-966-662-809-4

© Morozova O. I., Uzun D. D., 2021
© National Aerospace University
“Kharkiv Aviation Institute”, 2021

INTRODUCTION

Operating systems (OS) is a basic set of programs that manages the hardware of a computer or virtual machine, which provides control of the computing process and organizes interaction with the user.

The history of operating systems covers the period and events from the development of the first operating systems to the present.

Initially, there were computers that work without the OS at all. They ran one program that was downloaded from the punch card.

Then auxiliary software came, such as assemblers, or tapes with auxiliary libraries (for example, for input and output), which became the kernel of the OS.

As the speed increased, the operating time of the programs became an order of magnitude less than the time of transferring the equipment into the hands of another user. There is a need in libraries to control the sequence of loading punch cards, select a magnetic device for reading and writing, logging time and errors. These libraries became invisible software that ran before the user's first task and controlled its download and execution, allocated resources, recorded work results, monitored the proper completion and release of resources, and then immediately moved on to the next task. Such background programs, even before the introduction of the term OS, were called "monitors".

During the transition to the era of personal computers (PC) there was a shift in understanding the concept of OS.

The first OS lacked many features (such as text editors, file managers, etc.), which on the PC have become required components of the OS.

The modern user does not even imagine an OS without a graphical interface. However, some programs, such as databases or spreadsheets, are still considered applications and are provided separately.

And the real descendant of the first OS is what is now called the kernel.

Some technical specialists still use the old concept of the OS, as only the kernel, through the development of embedded systems for various types of devices with data processing capabilities – from watches to industrial robots.

The OS includes: the kernel of the OS, which provides the allocation and management of the computer system resources; basic set of applications, system libraries and service programs.

In general, OS have the same structure: the kernel of the operating system, which is the central part of the operating system, drivers, utilities and interface.

1 BASIC CONCEPTS OF OPERATING SYSTEMS. BIOS AND UEFI INTERFACES

OS classification. According to the number of at the same time performed actions, the OS is divided into:

- single-task – able to execute only one program (for example, DOS);
- multitasking – able to run multiple programs at the same time (for example, UNIX, Windows, Linux);
- multi-user – allows you to serve many tasks of many users at the same time. It is divided into multi-processor and multi-machine (for example, network OS).

According to the method of interaction, the OS system is divided into:

- batch processing systems. They handle the task flow. All control over reading and task execution is taken over by the OS. It is a characteristic of large machines, server and distributed computing facilities;
- dialog or interactive access. The OS executes user directives at the rate of receipt and tries to respond to each user command in the shortest time. Provide support for one (single-terminal) or several (multi-terminal) users at the same time. Command (Linux) or graphical (Windows, MacOS) interfaces can be used to communicate with the user.

According to the methods of building, the OS is divided into:

- micronucleus. The kernel consists of an abbreviated set of procedures that deal only with process scheduling and control of computer system equipment. A striking example of a micronucleus OS is the real-time operating system QNX (Developer Blackberry);
- monolithic. The full set of functions is included in the OS kernel. In the secure mode, both scheduling procedures and resource use planning are performed. You can call monolithic – Windows and Linux. It should be noted that the Windows kernel cannot be modified, its source codes are not available and there is no program to build (compile) this kernel. But in the case of Linux, it is possible to assemble the kernel that we need, including only those software modules and drivers that we consider appropriate to include in the kernel (rather than accessing them from the kernel).

According to the purpose, the OS is divided into:

- OS of large computers (mainframes). The main characteristic of the hardware is the performance of Input/Output (I/O). They are equipped with a significant number of peripherals (disks, terminals, printers, etc.). Such computer systems are used to reliably process the large amounts of data, and the OS must effectively support this processing. An example of an OS of this class is OS / 390 from IBM;
- server OS. It is able to handle a large number of user requests to shared resources. Network support plays an important role for them.

Nowadays, universal operating systems (UNIX or Windows systems) are more often used to implement servers;

- personal OS. Some operating systems in this category were designed for non-professional users (Microsoft's Windows 95/98/Me line), others are simplified versions of universal OS. Particular attention in personal OS is paid to support the graphical user interface (GUI) and multimedia technologies;

- real-time OS. In such a system, each operation must be guaranteed to be performed within a specified time range. Real-time OS can control the flight of a spacecraft, the technological process or the demonstration of videos and playing the music. There are specialized real-time OS such as QNX and WxWorks;

- embedded OS. These include control programs for various microprocessor systems used in military equipment, consumer electronics systems, smart cards and other devices. Such systems have special requirements: placement in a small amount of memory, support for specialized I/O, the ability to flash in a non-volatile storage device. Often embedded OS are developed for a specific device. Universal systems include Embedded Linux and Windows CE (for PDAs, smartphones, and embedded systems).

Operating system structure. Different operating systems have the same structure:

- OS kernel, which is the central part of the OS;
- drivers – programs that translate computer commands into the language of a specific device (printer, scanner, sound or video card) and vice versa.

- utilities – a set of tools designed to service disks, scan your computer, configure certain settings, and other.

- interface – the rules of interaction between the OS and the user, which determine the usability.

The OS kernel is the central part of the OS that implements the interface between application processes and computer hardware. It is loaded into the computer's RAM and interacts directly with the hardware, providing hardware management, support for the simultaneous operation of many users, support for the parallel execution of many processes in the system.

Typically, the OS kernel makes these objects available to application processes through interprocess interaction mechanisms and system calls.

A system call is a way for applications to use OS kernel services. These can be services related to the computer hardware (for example, disk access), process and flow management, and other. System calls provide the interface between the process and the OS.

An interrupt is a signal that notifies the processor of an event that requires immediate attention. In this case, the execution of the current commands sequence is suspended and control is passed to the interrupt

handler, which responds to the event and serves it, and then returns control to the interrupted code.

Depending on the source of the interrupt signal are divided into:

- asynchronous or external (hardware) – events that are created by external sources (for example, peripherals) and can occur at any time (for example, a signal from a timer, network card or disk drive, keystrokes, mouse movement);

- synchronous or internal – exceptional situations in the processor as a result of some conditions violation when executing machine code (for example, division by zero or overflow, access to invalid addresses or invalid operation code);

- software (partial case of internal interruption) – are initiated by executing special instructions in the program code. Software interrupts are typically used to access firmware, drivers, and the OS.

The main task of a computer system is to execute programs. Programs, along with the data to which they have access, must be (at least partially) in Random Access Memory (RAM) during execution. The OS has to solve the problem of memory allocation between user processes and OS components. This activity is called memory management. The part of the OS that is responsible for memory management is called the memory manager.

Computer RAM is collected on semiconductor chips and stores information only while the computer is turned on. When you turn off the computer, its contents are lost. All RAM can be divided into two types:

- 1) dynamic RAM (DRAM);
- 2) static RAM (SRAM).

DRAM is used in most personal computer RAM systems. The main advantage of this memory type is that its cells are tightly integrated, i.e. in a small chip you can put a lot of bits, and therefore, on their basis you can build a memory of greater capacity.

SRAM, in contrast to DRAM, does not require periodic regeneration to store its contents. But this is not its only advantage. SRAM has a higher speed than DRAM, and can run at the same frequency as modern processors.

Virtual memory is a technology that introduces the level of additional transformations between the memory addresses used by the process and the physical memory addresses of the computer.

Such transformations should ensure the protection of memory and the absence of the process binding to the addresses of physical memory.

Due to virtual memory, the physical memory of the process address space can be fragmented, because the main amount of memory occupied by the process, most of the time remains free. Addresses can be moved so that only those sections of the process address space that are actually in use at a particular time match the main memory.

The main problem that arises when using the virtual memory is the efficiency of its implementation. Because address conversions need to be done

every time you access memory, careless implementation of this conversion can have the worst consequences for system-wide performance. If for most memory accesses the system is forced to actually access the disk (which is tens thousands of times slower than the main memory), it will be almost impossible to work with such a system.

Another problem is memory fragmentation, which occurs when free memory cannot be used. There are external and internal memory fragmentation.

External fragmentation is reduced to the fact that due to the allocation and subsequent release of memory in it are formed free blocks of small size – holes. This can lead to a situation in which it is impossible to allocate a continuous memory block of size N , because there is no continuous free block whose size is $S > N$, although in general the amount of free memory space exceeds N .

Internal fragmentation comes down to the fact that on request, memory blocks are allocated larger than they will actually be used, resulting in unused areas inside the allocated blocks that can no longer be used for anything else.

Memory segmentation allows you to represent a logical address space as a set of independent blocks of variable length, called segments. Each segment usually contains data of one purpose, for example, there can be a stack in one, and the program code in another, etc. Each segment has a name and length (for convenience of implementation along with names use numbers).

The logical address consists of the segment number and the offset within the segment; the application works with such addresses. Compilers often create individual segments for different program data (code segment, data segment, stack segment).

Today, segmentation is used to a very limited power, primarily due to fragmentation and the complexity of implementing effective memory free and disk exchange.

The implementation of virtual memory is represented by three classes: page, segment, segment-page distributions. Widespread use of the memory distribution into blocks of fixed length – page memory organization.

With page distribution, virtual memory is divided into parts of the same and fixed size for the system – virtual pages. All RAM is also divided into parts of the same size – physical pages. The page size is chosen divisible by two: 512, 1024, 4096 and so on. The page address is in the context of the process.

The program code and source data can appear on the same page at the same time. This approach does not allow for individual processing, such as protection, sharing, etc. Dividing the address space into "significant" parts eliminates these shortcomings and is called segmental distribution. Examples of segments: program code, source data array, etc. Segment-page allocation is a combination of page and segment memory management mechanisms and aims to realize the benefits of both approaches. Virtual memory is divided into

segments, and each segment into pages. All modern operating systems use this method of organization.

BIOS. The basic input/output system is BIOS. Thus, the BIOS is a set of firmware that implements an application programming interface (API) to work with computer hardware and devices connected to it.

BIOS types:

- BIOS of the IBM PC-compatible computer motherboard;
- BIOS of peripherals;
- NetBIOS.

BIOS purpose:

- equipment checks;
- OS download;
- providing API for working with equipment;
- equipment setup;
- initial computer startup.

BIOS code:

- performs computer hardware testing (power-on self-test (POST));
- reads settings from non-volatile storage device (non-volatile ROM);
- applies settings;
- searches and downloads the bootloader code into RAM;
- passes control to the bootloader.

Initialization and verification of the BIOS is performed by pressing several times the "Delete" key (Figure 1).

Most of the BIOS code consists of firmware designed to initialize the controllers located on the motherboard and the devices connected to the motherboard.

Immediately after turning on the computer, the processor reads the BIOS code with Electrically Erasable Programmable Read-Only Memory (EEPROM) – a permanent storage device that is programmed and cleaned with electricity, one of the types of non-volatile memory. This type of memory can be cleared and filled with information tens thousands of times), writes the BIOS code to memory and transmits control to it.

First of all, the BIOS code starts checking the computer hardware – the POST procedure. POST is self-testing after inclusion. Checking the computer hardware, which is performed when you turn it on. It is executed by the programs which are included in BIOS of a motherboard. If a failure occurs during POST, the BIOS code may provide information to identify the reason of the failure.



Figure 1 – BIOS initialization

If POST is executed without errors, the BIOS code will start searching for the OS boot code. The search is performed on available and allowed in the settings media:

- in the master boot record (MBR) on a solid-state drive (SSD) or hard disk drive (HDD). They differ in the technology of reading / writing information;
- on USB-flash;
- on an optical DVD/CD-ROM;
- on a floppy disk;
- in a network using Preboot eXecution Environment PXE technology (PXE technology) – an environment for booting the computer using the network card without the use of local media.

The BIOS code loads the OS boot code into memory and passes control to it.

You can open the BIOS setup menu by pressing several times a key (Delete, F2, F10, Esc, and F8 keys are often used) during POST.

Some menu options (Figure 2):

- setting the date and time for the system clock;
- setting up peripherals that are not suitable for Plug and Play mode, for example, hard disks released in the early 1990s and running in addressing mode CHS (Cylinder, Head, Sector) – sector addressing system, as a minimum unit of data storage in hard disk drives, floppy disk drives, etc., based on the use of physical addresses of disk geometry, COM-ports () (Communications port, serial port) and LPT-ports (parallel port, printer port, Line Print Terminal);
- start the equipment in "forced" or "light" mode;
- setting the factory settings;
- switching on and off the equipment built into the motherboard;
- disable some tests performed during POST;
- activation of bypass branches for known OS errors. For example, a poorly written driver does not work with hard drives connected via the Serial Advanced Technology Attachment (ATA) interface, the BIOS can emulate the ATA interface – the standard interface for connecting external devices;
- the order of the media from which the computer is loaded: hard disk, USB-drives, DVD/CD-ROM, boot using a network interface card (NIC) with technology PXE, etc.

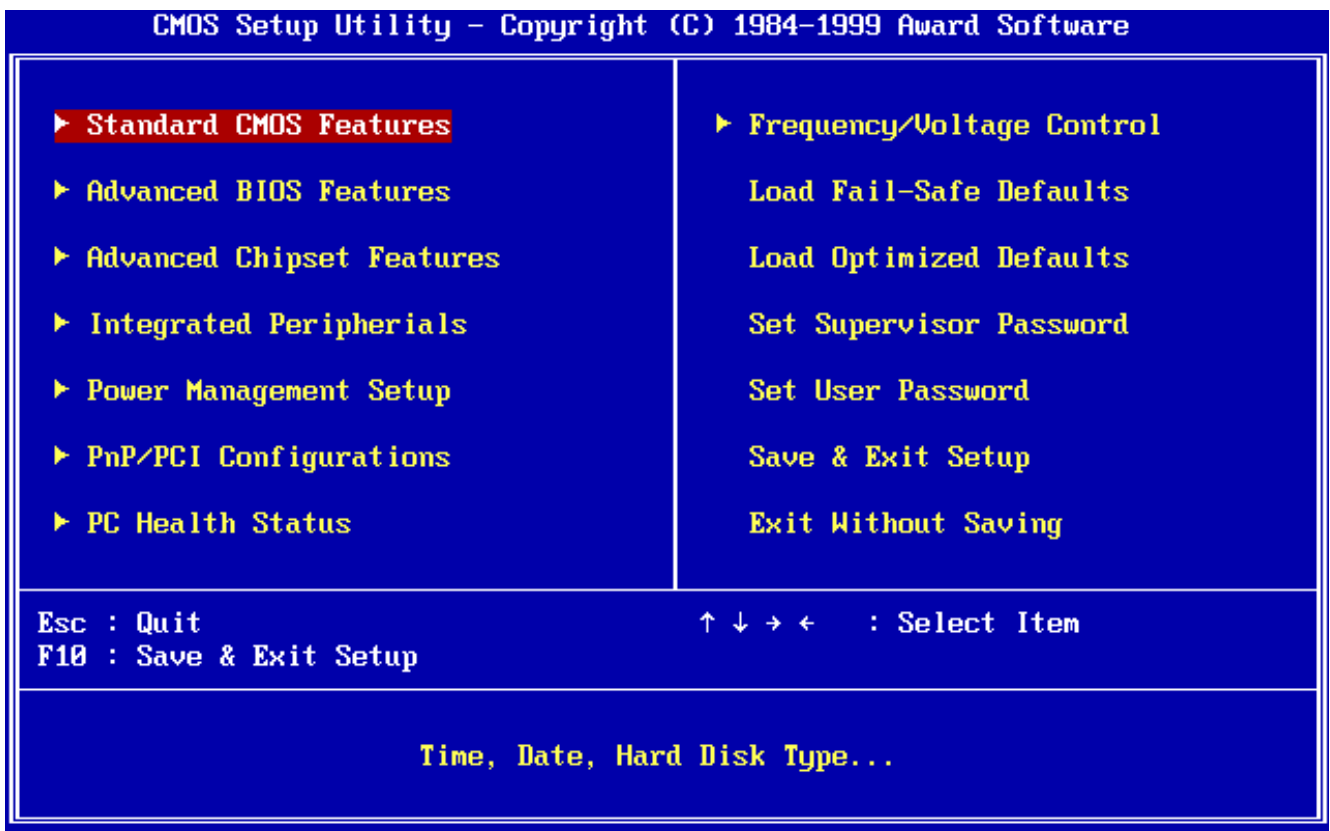


Figure 2 – Hardware configuration window using the menu

Resetting the BIOS can be done by software or hardware. The program is executed using the BIOS menu or by pressing a special key combination during POST. Hardware is performed by installing jumpers on the motherboard. The jumpers on the board are usually marked "CLEAR_CMOS", "CLR_CMOS", "CLRRTC" or by pressing the button located on the motherboard. The button can be placed on the back of the system unit.

With the release of Windows Vista, computer manufacturers began to implement the Software Licensing Description Table (SLIC table) in the BIOS. The SLIC table is the first of three components created for original equipment manufacturer (OEM) activation of the Microsoft Windows family of OS in the off-line mode.

The OEM product code is a special 25-digit license key. Issued only to large component manufacturers. Is the second component of OEM activation in the off-line mode. It is the second component of OEM activation in the off-line mode.

OEM Digital Certificate is an XML file with the extension *.xrm-ms. Published by Microsoft to every prior PC maker. It is the third component of OEM activation in the off-line mode.

UEFI. The first EFI specification was developed by Intel, later the first name was abandoned and the latest version of the standard is called the Unified Extensible Firmware Interface (UEFI). UEFI is currently being developed by the Unified EFI Forum.

Extensible Firmware Interface (EFI) is an interface between OS and firmware that controls low-level hardware functions. Its main purpose: to properly format the equipment when you turn on the system and transfer control of the OS bootloader.

EFI is designed to replace the BIOS interface, which is traditionally used by all IBM PC-compatible personal computers.

The interface defined by the EFI specification includes data tables, which contain information about the platform, boot and runtime services that are available to the bootloader of the OS and the OS itself. Some existing BIOS extensions, such as Advanced Configuration and Power Interface (ACPI) and or System Management BIOS (SMBIOS), a specification that defines the method of accessing data to the contents of the computer's non-volatile memory, are also present in EFI, as they do not require a 16-bit runtime interface. In addition to standard, architecture-dependent device drivers, the EFI specification provides a platform-independent driver environment called the EFI Byte Code (EBC).

The system firmware requires the UEFI specification to have an interpreter for any EBC images that are downloaded or can be downloaded in environment. Some architecture-dependent (non-EBC) types of EFI drivers may have interfaces for using the OS. This allows the OS to use EFI for basic graphics and network support, before downloading the drivers defined in the OS.

The EFI defines "boot services" that include support for text and graphics consoles on various devices, blocks, file services and runtime services such as date, time, and independent memory.

The EFI boot manager is used to select and boot the OS, except for the need for a specialized boot mechanism (the OS bootloader is an EFI application).

The EFI specification does not include a description for file systems, but EFI implementations typically support FAT32 as a file system.

In addition to the standard Master Boot Record (MBR) scheme, EFI has Globally Unique Identifier (GUID) Partition Table (GPT) support, which is free of MBR restrictions. GPT is a standard format for placing partition tables on a physical hard disk. It is part of UEFI. EFI uses GPT where the BIOS uses MBR.

From a GPT partition with EF00 ID and FAT32 file system, the `\efi\boot\boot [architecture name].efi` file, for example, `\efi\boot\bootx64.efi`, is loaded and run by default. That is, for example, to create a bootable flash drive from Windows, simply mark the flash drive in GPT, create an active FAT32 partition on it and copy all the files from the CD with the OS distribution.

The EFI community has created an open shell environment. The user can load the EFI shell instead of loading the OS to perform some operations. The EFI shell is an EFI application; it can be permanently in the platform's permanent storage device (RAM) or on the device for which the drivers are in the RAM. The shell can be used to run other EFI applications, such as configuration, OS installation, diagnostics, configuration utilities, and firmware updates.

The Intel Platform Innovation Framework for EFI is a series of specifications developed by Intel in conjunction with EFI. If the EFI defines the interface between the OS and the firmware, the toolkit defines the structures used to create the firmware at a lower level than the interface between the OS and the firmware.

The toolkit supports at all the steps required to initialize your computer after it is turned on. These internal firmware capabilities are not defined as part of the EFI specification, but are included in the Platform Initialization Specification developed by UEFI.

Released in 2000, Intel systems on the Itanium platform supported EFI 1.02. Released in 2002, Hewlett Packard systems on the Itanium 2 platform supported EFI 1.10; they could download Windows, Linux, FreeBSD and HP-UX. All Itanium or Itanium 2 systems come with EFI-compatible firmware.

In November 2003, Gateway introduced the Gateway 610 Media Center.

In January 2006, Apple Inc. introduced the first Macintosh computers on the Intel platform. These systems use EFI and tools instead of Open Firmware, which was used on previous PowerPC systems.

On April 5, 2006, Apple released the Boot Camp package, which allows create a disk with Windows XP drivers, and includes a non-destructive disk partitioning tool that allows install Windows XP with Mac OS X.

In 2007, Hewlett-Packard released the 8000 Series All-In-One, equipped with firmware compatible with EFI.

Advantages of UEFI over BIOS:

- faster loading due to the absence of the need to search for a bootloader on all disks;
- support for media with a capacity of greater than 2TB;
- easier preparation of bootable media (no need to record different boot sectors);
- having own download manager. In EFI NVRAM (Non-Volatile Random-Access Memory – the common name for independent memory. Independent memory is one in which data is not stored at power shutdown) all records of existing bootloaders are stored normally, and switching between bootable OS is as follows the same as between bootable media;
- more secure boot environment;
- graphical UEFI configuration mode.

Laboratory work № 1

BIOS AND UEFI TOOLS, AS WELL AS VIRTUALIZATION SOFTWARE

The purpose of the work is to explore the means of system initialization and access to BIOS and UEFI hardware. Study of virtualization software.

Problem statement:

1. Get acquainted with the operation principle of BIOS-Setup and UEFI shell.
2. Get acquainted with the Oracle VM VirtualBox virtualization tool.

Execution of work:

1. Describe the basic functionality of BIOS-Setup and UEFI shell.
2. Analyze the differences between BIOS and UEFI, as well as make conclusions about the prerequisites for migrating from BIOS to UEFI.
3. Analyze and describe scenarios for the use of virtual machines.
4. Create a test image of a virtual machine with arbitrary characteristics. Connect two different hard drives to it (Figure 3):
 - a) a disk that expands dynamically as it is filled with information;
 - b) a fixed size disk.
5. Write a report on the work. Contents of the report on laboratory work: topic; goal; setting objectives; the order of execution; research results; conclusion. Copies of screens are provided with the report in accordance with the procedure for conducting research.

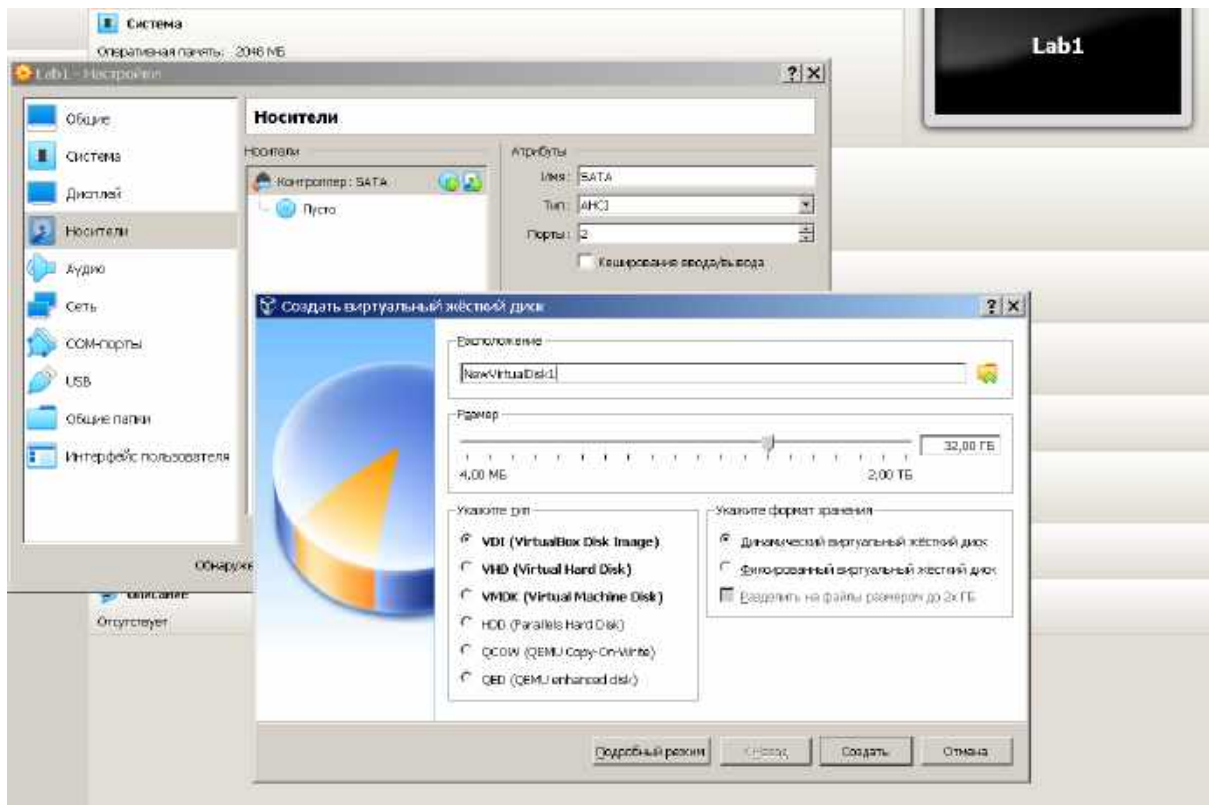


Figure 3 – Connecting the hard drives

Additional information: for work with BIOS-Setup and UEFI shell it is possible to use either real tools of the computer, or the created image in the Oracle VM VirtualBox environment (Figure 4).

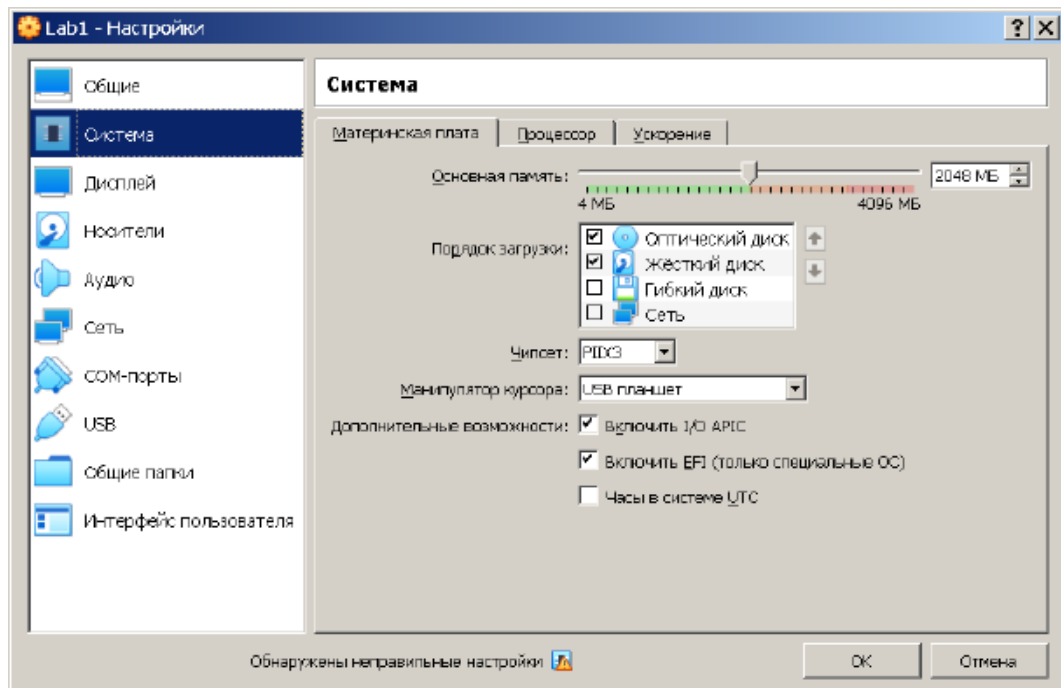


Figure 4 – Turn on the EFI interface

2 PHYSICAL AND LOGICAL STRUCTURE OF A HARD DRIVE

The concept of "media format" includes the structure of information on it and ways to address the elements of this structure.

Physical disk format. The information is placed in sectors of fixed length located on concentric tracks of the disk surface. The number of sectors on the track is set programmatically. Each sector consists of a data field and a field of service information that limits and identifies it.

The physical address of the sector on the disk is represented by a triad

$$[t-h-s],$$

where t is the number of the cylinder (track on the surface of the disk), h is the number of the working surface of the disk (magnetic head), and s is the number of the sector on the track.

The number of the cylinder t is in the range $0..T-1$, where T is the number of cylinders.

The number of the working surface of the disk h belongs to the range $0..H-1$, where H is the number of magnetic heads in the drive.

The sector number on track s is in the range $1..S$, where S is the number of sectors on the track.

For example, the triad $[1-0-2]$ addresses sector 2 on track 0 (usually the upper working surface) of cylinder 1.

Logical disk format. The hard drive can be divided into several partitions used by different OS. There are two types of sections: primary and advanced.

The maximum number of such sections is four. In the primary partition of DOS and Windows only one logical disk can be formed, and in the advanced – any of them.

Each logical section is "controlled" by its logical drive.

At the logical level, it is assumed that the sectors of the logical disk have a continuous numbering from 0 to $N-1$, where $N = T \times H \times S$ – the number of sectors on the disk. The correspondence between the physical address of the sector and its logical number n is determined by the following formula:

$$n = (t \times H \times S) + (h \times S) + s - 1.$$

Each logical drive on the hard drive has its own (relative) logical numbering. The physical addressing of the hard disk is end-to-end.

The formula for determining the physical address of the sector by its logical number:

$$t = n / (H \times S);$$

$$h = (n - (t \times H \times S)) / S;$$

$$s = n - ((t \times H \times S) + (h \times S)) + 1.$$

The above formulas for floppy disks with a single logical drive can be used without any corrections.

There is a little different for hard disks: the resulting t , h and s must be increased by the values corresponding to those used for other purposes of the previous area of disk space.

The File Allocation Table (FAT) is a classic file system architecture that is still widely used for flash drives because of its simplicity. Used in floppy disks, memory cards and some other media. Previously used on hard drives.

There are four versions of FAT: FAT12, FAT16, FAT32 and FAT64. They differ in the bit size of the records in the disk structure, i.e. the number of bits allocated to store the cluster number.

FAT12 is used mainly for floppy disks, FAT16 is for small disks. Based on FAT, a new exFAT file system (extended FAT) was developed, which is used mainly for flash drives. The logical disk space of any logical disk is divided into two areas: the system area and the data area.

The system area of the logical drive is created and initialized during formatting, and then updated when manipulating the file structure.

The logical disk data area contains files and directories that are subordinate to the root. Unlike the system area, it is accessible through a user interface.

The system area consists of the following components located in the logical address space in a row:

- 1) Boot Record (BR);
- 2) Reserved Sector (RSec);
- 3) File Allocation Table (FAT);
- 4) Root Directory (RDir).

The BR is located in the sector with the physical address [0-0-1] (for the floppy disk) and contains the Disk Parameter Block (DPB), as well as the System Bootstrap (SB). The sector that contains BR is called the start.

The boot sector of the logical drive with BR is RSec. Several RSec can be located behind the BR. There is usually the only RSec on the logical drive. BR system occupies 3 sectors in the FAT32 file system.

FAT is a very important information structure. It is a map (image) of the data area, which describes the state of each cluster and linked in a chain of clusters belonging to one file.

A cluster is the minimum unit of disk memory allocated to a file. A cluster is one or more contiguous sectors in a logical disk address space

Clusters as a set of sectors instead of using single sectors has the following meaning:

- possible file fragmentation is reduced;
- the size of FAT, and hence the volume of the system area of the logical drive decreases;

- accelerates access to the file, as several times the length of the chains of fragments of disk space allocated to it.

BR and FAT store the media descriptor for quickly determine the format of the logical drive.

Features of the format of the logical drive on the hard drive are as follows:

- increased cluster size;
- increased sizes of FAT and RDir;
- FAT elements can be 16-bit and 32-bit (FAT16, FAT32);
- the maximum number of elements of the root directory is usually 512;
- media descriptor is code FBH or F8H. The FBH code means that the data is correct and the entire data field can be read. Code F8H indicates that the data is erased.

The hard disk has a single-sector MBR, which contains Non-System Bootstrap (NSB), as well as Partition Table (PT) and has a physical address (0-0-1). Thus, in the starting sector of the physical hard disk is not BR, and MBR. PT describes the location and characteristics of the partitions available on the hard drive. NSB is used to copy SB from BR logical drive in the active partition to RAM and transfer control to it, which is carried out when loading the OS. The MBR is followed by sections.

The primary partition includes only the system logical drive without any additional information structures.

An advanced DOS partition can be thought of as a hard disk sketch: it contains Secondary MBR (SMBR), which instead of PT includes Logical Disk Table (LDT) similar to it. LDT describes the location and characteristics of the partition that contains a single logical drive, and can specify the next SMBR.

A logical partition formatted for the FAT file system contains the following partitions:

- boot sector contains the program of OS initial boot;
- main copy of FAT;
- FAT backup;
- the root directory occupies a fixed area of 32 sectors (16 kB), which allows you to store 512 records about files and directories (each record – 32 B);
- the data area is used to place all files and all directories except the root directory.

The features of the FAT system. The FAT index can have the following values:

- the cluster is free;
- the cluster is used (number of the next cluster);
- last file cluster;
- defective cluster;
- backup.

Bit elements:

- FAT12 – 12 bits – maximum 4096 clusters;
- FAT16 – 16 bits – maximum 65536 clusters;
- FAT32 – 32 bits – maximum $2^{32} = 4294967296$ clusters.

Each entry in the directory – 32 B. File name – 11 B in the format 8.3. The file name is up to 255 double-byte Unicode characters. The long name is placed in portions of 13 characters in the records immediately following the main directory entry (each such entry contains another 6 B of service information).

Disk system area component formats:

- 1) the structure of the main boot record and partition table;
- 2) the structure of the extended section;
- 3) the structure of the boot record;
- 4) catalog structure;
- 5) the structure of the file placement table.

The structure of the main boot record and partition table. MBR is divided into two logical parts – NSB and PT. The NSB will provide regular DOS or any other OS from the logical drive in the active partition by analyzing the contents of the PT. The PT is at the end of the MBR and contains four lines according to the maximum number of partitions on the hard drive. Each line of PT is called a descriptor of the section.

The structure of the extended section. An extended section contains one or more logical disks, for each of which an SMBR is created. All SMBRs are linked in a chain (list). The link to the first SMBR of this list is in the PT MBR. The structure of SMBR is similar to MBR, but NSB is absent, and LDT is used instead of PT. The latter has the same format as PT, but contains two descriptors instead of four. The first descriptor specifies a section similar to the primary (system code 01H or 04H). The second descriptor LDT specifies the next SMBR (system code 05H) or is empty (system code 00H). The addresses here are set absolutely.

The structure of the boot record. BR is the first on the logical drive (it has a physical address [0-0-1 on the floppy]). It consists of two parts – Drive Parameter Block (DPB) and System Bootstrap (SB). DPB is used to identify the physical and logical formats of a logical drive, and SB plays a significant role in the DOS boot process. The first two bytes of BR are occupied by the unconditional JMP to SB command. The third byte contains the code 90H (NOP – no operation). Next is an eight-byte system ID that includes information about the developer and DOS version. This is followed by DPB and then SB.

Catalog structure. Root and non-root directories have the same structure. The directory consists of a sequence of 32-byte elements. Each element of the directory describes the included file or directory, contains a label back to whether it is free. The catalog element consists of eight fields. The

name field contains the name of the file (directory), supplemented if necessary, by spaces. The code in the first byte of this field determines the object or state of the given element of the directory described by this element. For its full identification the field of attributes is involved also.

The contents of the first byte of the name field are interpreted as follows:

1) 00H – the directory element has never been used. This means that all the following items are empty, so you can stop searching the directory with the confidence that the entire directory has been reviewed;

2) 05H – the first character of the file name (directory) has the code E5H, i.e. is the character "^". This recoding is used because the E5H code is used for another purpose;

3) 2EH – the directory element describes this directory (link to yourself). The number 2EH is the code of the symbol "point". If the second byte of the name field contains the code 2EH, then the directory element describes the parent directory (..) of this directory. The remaining bytes of the username and extension fields contain spaces;

4) E5H – the element was used by a file (directory), but has already been released. This code is written when deleting an existing file (directory), but no further changes are made to the directory element.

Any other content of the first byte of the name field is interpreted as the code of the first character in the file name. The extension field stores the file name extension. In the attribute field, each bit specifies a specific attribute of the file or describes the object represented by this directory element. The time field contains the time of creation (last modification) of the file or the time of the directory creation. Similarly, the date field contains the date of creation (last modification) of the file or directory.

The 1AH offset field contains the number of the first cluster belonging to the file (directory). The list of other clusters is stored in FAT. The size field contains the length of the file (directory), which is measured in bytes.

The structure of the file placement table. FAT is an image of the logical disk data area and contains comprehensive information about using as well as the status of each cluster. In particular, it stores information about the location of each file and non-root directory. FAT consists of a sequence of elements, each of which, except for the first two, describes the corresponding cluster.

The first two elements (numbered 0 and 1) store a copy of the media descriptor (the first, i.e. the least significant, byte), supplemented to the required number of bytes in binary units (FFH). A positional correspondence is established between the other FAT elements and the data area clusters.

Each m-th FAT element contains either the number of the next cluster belonging to the file (non-root directory) or a special code. The following values are used as special:

(0) 000H – the cluster is free;

(F) FF0H ... (F) FF6H – the cluster reserved for DOS use;
(F) FF7H – the cluster is defective;
(F) FF8H ... (F) FFFH – the cluster is the last in the file (non-root directory).

Technically, accessing a file on disk is done by DOS as follows:

- 1) the corresponding element of the catalog is searched;
- 2) the FAT content determines the number of the required cluster;
- 3) the cluster number is converted into a logical sector number;
- 4) interrupt 25H is issued for direct reading of the sector from the disk, after which the HMD driver is turned on;
- 5) the driver converts the logical number of the sector into its physical address and provides control of the drive by reading the required sector.

RAID (redundant array of independent disks) is a data virtualization technology that combines multiple disks into a logical element to reliably store information and increase drive performance.

The abbreviation "RAID" was originally deciphered as "surplus array of low-cost disks" ("redundant (spare) array of low-cost disks", as they were much cheaper than Single Large Expensive Drive (SLED)). This is how RAID was presented by its creators David A. Patterson, Garth A. Gibson and Randy H. Katz in 1987.

A disk array is a set of disk devices that work together to increase the speed and/or reliability of an I/O system. This set of devices is controlled by a special RAID controller (array controller), which provides the functions of placing data in the array; and for the rest of the system allows you to represent the whole array as one logical I/O device.

By performing read and write operations on multiple disks in parallel, the array provides an increased exchange rate compared to one large disk. Arrays can also provide secure data storage so that data is not lost if one of the disks fails. Depending on the RAID level, there is either duplication or even distribution of data on the disks.

The University of California at Berkeley has identified the following levels of RAID, which have been adopted as the standard. Each of the four basic RAID levels uses a unique method of writing data to disk, and therefore all levels provide different advantages.

RAID levels 1,3 and 5 provide data duplication or storage of parity bits and therefore, allow you to recover information in case of failure of one of the disks. View RAID:

- 1) RAID 1 – mirror disk array;
- 2) RAID 2 – reserved for arrays that use Hamming code;
- 3) RAID 3 and 4 – disk arrays with alternation and dedicated parity disk;
- 4) RAID 5 – disk array with alternation and no dedicated parity disk.

Modern RAID controllers provide additional levels of RAID specification:

1) RAID 0 – disk array of high performance with alternation, without fault tolerance. Strictly speaking, it is not a RAID array, because there is no redundancy in it;

2) RAID 6 – disk array with alternation, using two checksums, which are calculated in two independent ways;

3) RAID 10 – RAID 0 array built from RAID 1 arrays;

4) RAID 01 – RAID 1 array built from RAID 0 arrays (has low fault tolerance).

A hardware RAID controller can support multiple RAID arrays of different levels at the same time.

In this case, the controller built into the motherboard in the BIOS settings has only two states (on or off), so the new hard drive connected to the idle connector of the controller when RAID mode is activated, can be ignored by the system until it is associated as one RAID array of Just a bunch of disks (JBOD) type, consisting of one disk. JBOD is a disk array in which a single logical space is distributed across the hard disks sequentially.

RAID 0 technology is also known as data striping. With this technology, information is broken into pieces (fixed amounts of data, usually called blocks). These pieces are written to disks and read from them in parallel.

In terms of implementation, this means two main advantages: increased serial I/O bandwidth by loading multiple hard drives at the same time; reduced latency of random access; multiple queries to different small segments of information can be executed at the same time.

Disadvantage: RAID level 0 is designed solely to increase implementation, and does not provide reliable data storage. Therefore, any disk failures require recovery of information from backup media.

RAID 1 technology is also known as disk mirroring. In this case, copies of each piece of information are stored on an individual disk; or each (used) disk has a "twin" that stores an exact copy of that disk. If one of the main disks fails, it is replaced by its "twin". Random reading performance can be improved if the "twin" whose head is located closer to the required block are used to read the information.

Recording time can be slightly longer than for one disc, depending on the recording strategy: recording on two discs can be done either in parallel (for speed) or strictly sequentially (for reliability).

RAID 1 is well suited for cases that require high reliability, low latency when reading, and if you do not need to minimize the cost. RAID 1 provides redundant storage of information, but in any case, should support data backup, as this is the only way to recover accidentally deleted files or directories.

Advantages: provides an acceptable write speed (the same as without duplication) and a gain in reading speed when parallelizing queries; has high

reliability – works as long as at least one disk in the array functions. The probability of two disks failure at once is equal to the product of the probabilities of each disk failure, i.e. much lower than the probability of a single disk failure. In practice, if one of the disks fails, urgent measures should be taken to restore the redundancy. For this purpose, with any level of RAID (except zero) it is recommended to use disks of a reserve.

The disadvantage of RAID 1 is that for the price of two hard drives, the user actually gets the volume of only one.

RAID 1E (Enhanced – extended) is an array of three (or more) disks, combining RAID 0 and RAID 1. The system works on the principle of alternating blocks throughout the array, then shifting the "alternation" to one disk. The solution is not very popular, but it exists and is supported.

The array is similar to RAID 0 (alternation), where each block is written 2 times – on 2 disks. That is, in a system of 3 discs, the 1st block will be written to the 1st and 2nd disc, the 2nd block to the 3rd and 1st, the 3rd block to the 2nd and 3rd, and so on.

Calculating the location of a block on a physical disk and the number of this disk is a non-trivial task.

RAID 2 arrays are based on the use of Hamming code. Disks are divided into two groups: for data and for error correction codes.

The data is distributed on disks, which are designed to store information, as well as in RAID 0, i.e. they are divided into small blocks by the number of disks. The remaining disks store error correction codes, according to which in the event of failure of any hard disk, information can be restored.

The advantage of the RAID 2 array is the increase in the speed of disk operations compared to the implementation of a single disk.

The disadvantage of a RAID 2 array is that the minimum number of disks at which it makes sense to use it – 7. This requires a structure of almost twice the number of disks (for $n = 3$ data will be stored on 4 disks), so this type of array is not widespread. If the disks are about 30-60, the overuse is 11-19%.

In a RAID 3 array of n disks, the data is divided into pieces smaller than a sector (divided into bytes or blocks) and distributed over $n-1$ disks. Another disk is used to store parity blocks.

RAID 2 used an $n-1$ disk for this purpose, but most of the information on the control disks was used to correct errors "on the fly", while most users are satisfied with a simple recovery of information in the event of a disk failure, for which there is enough information on one dedicated hard drive. Difference between RAID 3 and RAID 2: inability to correct errors "on the fly".

Advantages:

- high speed of reading and writing data;
- the minimum number of disks to create an array is three.

Disadvantages:

- an array of this type is good only for single-task work with large files, as the access time to a single sector, divided into disks, is equal to the maximum of the access intervals to the sectors of each disk. For small blocks, the access time is much longer than the read time;
- a large load on the control disk, and, as a result, its reliability is greatly reduced compared to data storage disks.

RAID 4 is similar to RAID 3, but differs from it in that the data is divided into blocks, not bytes. Thus, it was possible to partially "overcome" the problem of low data rates of small volumes. Recording is slow because the parity for the block is generated during recording and is written to a single disc.

Of the widely used storage systems, RAID 4 is used on NetApp storage devices (NetApp FAS), where its shortcomings are successfully eliminated by running the discs in a special group write mode, which is determined by the internal Write Anywhere File Layout (WAFL) file system ("File structure with recording everywhere" – internal high-performance file system).

The main disadvantage of RAID levels 2 to 4 is the inability to perform parallel write operations, as a separate control disk is used to store parity information. RAID 5 does not have this disadvantage.

Data blocks and checksums are cyclically written to all disks of the array, there is no asymmetry of the disk configuration.

Checksums mean the result of an XOR (excludes or) operation. XOR has a feature that makes it possible to replace any operand with a result, and, using the XOR algorithm, get the resulting missing operand.

For example: $a \text{ xor } b = c$ (where a, b, c are three disks of the RAID array), if a refuses, we can get it by putting c in its place and drawing xor between c and b: $c \text{ xor } b = a$.

This can be applied regardless of the number of operands: $a \text{ xor } b \text{ xor } c \text{ xor } d = e$. If c refuses, then e takes its place and, holding xor, we obtain c: $a \text{ xor } b \text{ xor } e \text{ xor } d = c$. This method essentially provides fault tolerance in version 5. To store the result xor requires only 1 disk, the size of which is equal to the size of any other disk in RAID. The minimum number of disks used is three.

RAID 5 has become widespread, primarily due to its cost-effectiveness. The volume of a RAID 5 disk array is calculated by the formula $(n-1) * \text{Hard-Drive size}$ (HDD size), where n is the number of disks in the array, and HDD size is the size of the disk (the smallest if the disks have a different size). And as the number of disks in the array increases, the savings continue to increase.

RAID 5 provides high read speeds – the gain is achieved through independent data streams from multiple disks in the array, which can be processed in parallel.

RAID 5 performance is significantly lower on Random Write operations, in which performance drops by 10-25% of RAID 0 performance, as it requires more disk operations.

When at least one disk fails, the reliability of the volume immediately decreases to the level of RAID 0 with the corresponding number of disks $n-1$, i.e. $n-1$ times lower than the reliability of one disk – this state is called critical (degrade). To return the array to normal operation requires a long recovery process associated with a significant loss of productivity and increased risk.

During recovery (rebuild or reconstruction), the controller performs a long intensive read, which can make come the failure of one or more disks of the array.

In addition, during the reading, previously undetected reading failures may be detected in data arrays that are not accessed during normal operation of the array, archival and inactive data, which prevent recovery.

If before the full recovery of the array there is a failure, or there is a non-recoverable read error on at least one other disk, the array is destroyed and the data on it can not be restored by conventional methods.

RAID 6 is similar to RAID 5, but has a higher degree of reliability – three data disks and two parity control disks. Based on Reed – Solomon codes and provides performance after the simultaneous failure of any two disks.

Typically, the use of RAID 6 provides about 10-15% drop in disk group performance, relative to RAID 5, which is provided by the large amount of work for the controller (more complex algorithm for calculating checksums), as well as the need to read and overwrite more disk blocks when writing each block.

In addition to the basic RAID 0 – RAID 6 levels described in the Common RAID Disk Drive Format (DEF standard), there are combined levels with names of the type "RAID $\alpha+\beta$ " or "RAID $\alpha\beta$ ", which usually means RAID β compiled of several RAID α (sometimes manufacturers interpret this in their own way).

The combination of RAID 0 and RAID 1 forms RAID 0+1. This level provides redundancy through mirroring.

RAID 10 (or 1+0) combines RAID 0 and RAID 1, i.e. mirroring a group of drives combined in RAID 0 for maximum performance. This level provides redundancy due to the mirror image. That is, it is RAID 0, composed of several (or at least two) RAID 1 (mirrored pairs).

RAID 10 – to reflect an array in which data is written sequentially on several disks, as in RAID 0. This architecture is an array of type RAID 0, the segments of which instead of individual disks are arrays of RAID 1. Accordingly, an array of this level must contain at least 4 disks (and always an even number).

The claim that RAID 10 is the most reliable option for data storage is fully justified by the fact that the entire RAID 10 array will fail only after the failure of

all drives in the same RAID 1 array. If one drive fails, the chance of failure in the order of the second in the same array is equal to $1/3 * 100 = 33\%$.

Combined levels inherit both the advantages and disadvantages of their "parents": the appearance of alternation in the level of RAID 5+0 does not give it any reliability, but has a positive effect on performance.

The RAID 1+5 level is probably very reliable, but not the fastest and, moreover, extremely uneconomical: the usable capacity of the volume is less than half of the total disk capacity.

RAID 7 is a registered trademark of Storage Computer Corporation, there is no separate level of RAID. The structure of the array is as follows: n-1 disks store data, one disk is used to store parity blocks. Writing to disks is cached using RAM, the array itself requires a required uninterruptible power supply. In fact, it is RAID 4 with UPS (uninterruptible power supply). RAID 7 does not exceed RAID 6 – rather it is a marketing move.

RAID 7.3 is an interleaving unit with triple parity distribution, developed and patented by RAIDIX. Analogue of RAID 6, but has a higher degree of reliability – three checksums are calculated according to different algorithms, the capacity of three disks is allocated for checksums. Thus, RAID 7.3 arrays can withstand the complete failure of three disks in one group.

There is a modification of RAID 6 from NetApp – RAID DP. The difference from the traditional array is the allocation of the checksums of two separate disks.

Due to the interaction of RAID DP and the WAFL file system (all write operations are sequential and performed in a free space), the performance drop disappears both in comparison with RAID 5 and in comparison, with RAID 6.

Hybrid RAID is some of the usual RAID levels, but in combination with additional software and SSDs that are used as a cache for reading. As a result, system performance is increased, as SSDs have significantly better speed characteristics compared to HDDs. There are several implementations, such as some budget-class Adaptec controllers.

In Hybrid RAID, read operations are performed from a faster solid-state drive, and write operations to perform redundancy are performed on both solid-state drives and hard drives. Hybrid RAID is ideal for low-level data applications such as an Internet gateway, file server, or virtual machine.

Matrix RAID is a technology implemented by Intel in the southern bridges of its chipsets, starting with ICH6R. This technology is not a new level of RAID (and its analogue exists in high-level hardware RAID controllers), it allows, using a small number of disks, to organize on different partitions of these disks several arrays of RAID 1, RAID 0 and RAID 5. This allows provide increased reliability for some data and high performance for others.

To implement RAID, you can use not only hardware, but also fully software components (drivers). For example, systems on the Linux kernel have special kernel modules, and RAID devices can be managed using the mdadm (multiple devices) utility. Software RAID has its advantages and disadvantages. On the one hand, it costs nothing. On the other hand, software RAID uses Central Processing Unit (CPU) resources, and at peak times on the disk system, the processor can spend much of its power on servicing RAID devices.

The idea of RAID arrays is to combine disks, each of which is considered as a set of sectors, and as a result the driver of the file system "sees" as if a single disk and works with it, ignoring its internal structure.

However, it can significantly improve the performance and reliability of a disk system if the file system driver "knows" that it is not working with a single disk, but with a set of disks. If the file system driver has placed each file on one disk, and the directory structure is properly organized, then the destruction of any of the disks will lose only the files that were on this disk.

In some RAID controllers, the JBOD mode is a mode in which the controller works as a normal IDE or SATA controller, without using the mechanisms of connecting disks into an array, i.e. in this case, each disk will be visible as a separate device in the OS. This fact indicates that the term "JBOD" as a mode of operation of disks has not yet been finally formed.

Some IT-professionals interpret it literally as a "bundle" or "pile" of disks, each of which operates independently of each other, and the concept of "connecting" (i.e. "data coverage" of multiple disks) is no longer referred to as JBOD, but to RAID technology, because there is an organization of disks in the simplest array. JBOD array characteristics:

1) the capacity of the array is equal to the sum of the capacities of the component disks;

2) the probability of failure corresponds to the probability of failure of any disk in the array;

3) the speed of reading and writing depends on the data area; it is not higher than the fastest disk in the array and not lower than the slowest;

4) the load on the processor during operation is minimal (comparable to the load when working with a single disk).

Features of the JBOD array:

– failure of one disk allows you to recover files on other disks (if none of their fragments belong to a damaged disk);

– in some cases, it is possible to ensure high speed of several applications (provided that the programs work with data areas on different disks);

– the array may consist of disks of different capacity and speed;

– the array is easily expanded with additional disks as needed.

Laboratory work № 2

PHYSICAL AND LOGICAL STRUCTURE OF A HARD DRIVE

The purpose of the work is to study of the physical and logical structure of the PC hard disk. Introduction to software utilities for changing the logical structure of the hard disk.

Problem statement:

1. Create an image of a virtual machine and connect a virtual floppy disk from the provided image.
2. In the created image of the virtual machine to partition the hard drive into at least three user-accessible partitions.
3. Insert the system from a floppy disk into one of the partitions so that this partition becomes the boot partition of the hard disk.

Execution of work:

1. Use Oracle VM VirtualBox to create a hard disk image. Use the provided iso file as a floppy disk image.
2. Use the FDISK.EXE utility to partition the hard disk.
3. Write a report on the work. Contents of the report on laboratory work: topic; goal; setting objectives; the order of execution; research results; conclusion. Copies of screens are provided with the report in accordance with the order of research.

Additional information: types of partitions are divided into primary and advanced partitions. The primary partition must have been present on the physical disk first. Accordingly, these OS could only be installed on the primary partition. This partition always contains only one file system. An extended partition is a primary partition that does not contain its own file system, but contains other logical partitions. The number of logical partitions is limited only by the size of the disk.

An example of partitioning a hard drive is shown in Figure 5.

How to Use the Fdisk Tool and the Format Tool to Partition a Hard Drive:

1. At the a: prompt type fdisk then hit enter.
2. Click yes you want to use large disk support.
3. To Delete the existing partition, Type 3 and press Enter.
4. Select your partition type you want to delete (Primary/Extended/Logical/Non-Dos)
5. Select shown partition by typing the displayed number of partition and press Enter for confirmation for deleting partition.
6. To re-create a new partition, Type 1 and press Enter.
7. Select what type of partition you want to create and type related value and press Enter for creating new partition.

8. When you create Primary partition in Volume Label type your Surname restart your PC.
9. At the a: prompt type fdisk/mbr (that fdisk your master boot record).
10. When done, restart PC.
11. At the a: prompt type format c:/s/u and type "y" when prompted.
12. Call fdisk and select Display Partition Information (Figure 6).

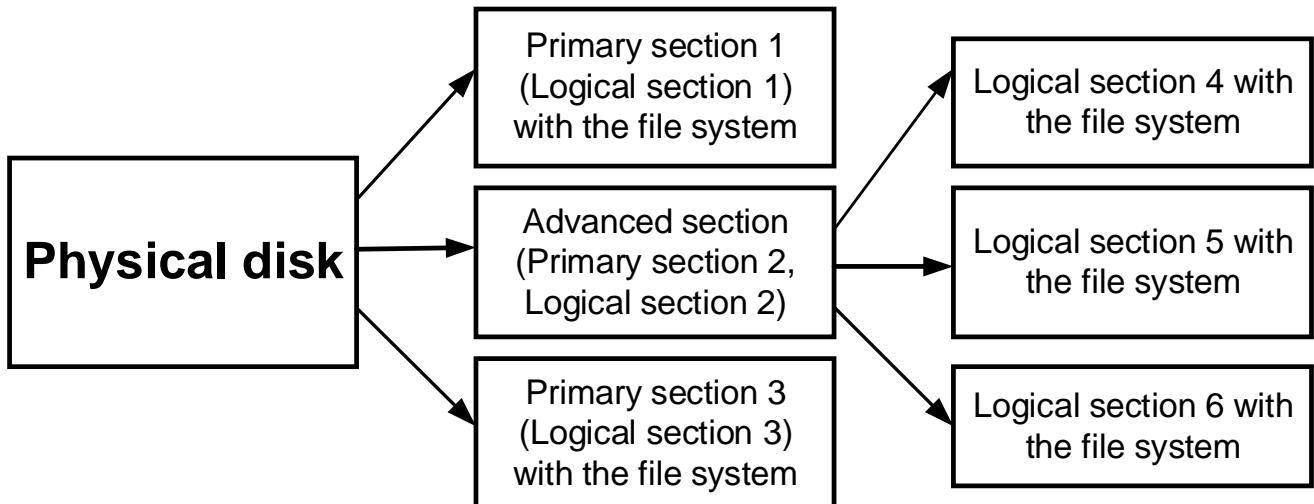


Figure 5 – The scheme of partitioning the hard disk

```

          Display Partition Information

Current fixed disk drive: 1

Partition  Status   Type   Volume Label  Mbytes   System   Usage
C:  1          PRI  DOS     SYSTEM         2047   FAT16    25%
   2          EXT  DOS         2047         25%

Total disk space is 8033 Mbytes (1 Mbyte = 1048576 bytes)

The Extended DOS Partition contains Logical DOS Drives.
Do you want to display the logical drive information (Y/N).....?[Y]

Press Esc to return to FDISK Options
  
```

Figure 6 – Display Partition Information

3 FILE SYSTEM

File system (FS) is the procedure that determines the method of organization, storage and naming of data on media in computers, as well as in other electronic equipment: digital cameras, mobile phones, etc.

FS determines the format of the content and the method of physical storage of information that is usually grouped as files. A specific file system determines the size of file names and directories, the maximum possible size of a file and partition, a set of file attributes. Some FS provides service capabilities, such as access sharing or file encryption.

FS connects the media on the one hand and the API to access files – on the other. When an application accesses a file, it has no idea how the information is located in a particular file, as well as on what physical type of media (DVD/CD, hard disk, tape, flash drive, or other), it is recorded.

All the program knows the file name, its size and attributes. It receives this data from the file system driver. It is the file system that determines where and how a file will be written to physical media (such as a hard drive).

A more complex structure is used in NTFS (New Technology File System) and HFS (Hierarchical File System).

In these file systems, each file is a set of attributes. Attributes are not only traditional read-only, system, but also file name, size and even content. Thus, for NTFS and HFS, what is stored in the file, is just one of its attributes.

Following this logic, a single file can contain several content options. Thus, in one file you can store several versions of the document, as well as additional data (file icon associated with the program file).

The main functions of any FS are aimed at solving the following tasks:

- file naming;
- software interface for working with files for applications;
- mapping the logical model of the file system to the physical organization of the data warehouse;
- organization of FS resistance to power failures, hardware and software errors;
- the content of the file parameters necessary for its proper interaction with other objects of the system (kernel, applications, etc.).

A file is a set of data that can be accessed by name. Files are the most common tools of organizing access to information stored in non-volatile memory.

The FS is an OS subsystem that is designed to provide the user with a user-friendly interface for working with data stored in non-volatile memory (disk) and to allow file sharing by multiple users and processes. The FS provides applications with file abstraction.

The FS includes:

- set of all files;

- sets of data structures used for file management: directories, descriptors, disk space allocation tables;
- a set of system software that implements file management, including creating, deleting, reading, writing, naming, searching and other operations.

File types:

1) regular files:

- text – strings of characters (some characters may have a special meaning – the end of the line, the end of the file);
- binary – a sequence of bytes (bits);
- executable files;

2) directory files:

- on the one hand it is a group of files;
- on the other hand, it is a special file that contains information about the files that are part of it;

3) special files:

- device files;
- block-oriented devices – files with direct access;
- byte-oriented (character) devices – files with serial access;
- allow you to replace specialized I/O operations with standard read and write operations to a file.

The directory contains a list of files and matches the files and their characteristics (attributes):

- name;
- type (binary/symbolic, catalog, connection, special);
- file size;
- security attributes (owner, read only, hidden, system, temporary, access attributes);
- time attributes (creation time, last modification time).

Directories can contain all information (MS-DOS) or only table references (UNIX). Directories can form a hierarchical tree structure (MS-DOS) or a network (UNIX). File path is a list of directories through which you can access the file, which can be absolute and relative.

Partition is a part of the physical disk space, which is designed to accommodate the structure of a single file system and from a logical point of view, which is considered as a whole. A partition is a logical device that, in terms of the OS, functions as a separate disk, which may correspond to the entire physical disk or most often corresponds to part of the physical disk.

Each partition can have its own file system and may be used by different OS. There are two approaches to implementing the relationship between partitions and file system directories:

- partitions with file systems installed on them are combined (mounted) into a single directory tree. This is typical of UNIX-like systems and hides the partition structure from the user;

– each section is a separate logical volume that has its own notation (C:, D: etc.), its own root directory, its own directory tree.

There are two types of links:

1. Hard links. Hard linking is actually giving the file an extra name (in most cases, it's in a different directory). All hard links reference the same file with its attributes (a descriptor table must be supported). All hard ties are equal. Attempting to delete a file deletes only its individual name and reduces the number of names by one, the file itself is deleted only after deleting the last of the names.

2. Symbolic links. A symbolic link is a special file that contains the name of the file it points to. Deleting the link does not affect the file in any way. Deleting or moving the file will invalidate the connection (restoring the file will restore the connection).

Basic file operations:

1. Open the file. Needed to get started with the file. It usually involves loading a file descriptor into memory, which determines its attributes and location on a disk.

2. Close the file. The structure created when the file is opened and is deleted from memory. All changes that have not been saved are written to disk.

3. Create a file. A zero-length file is created on the disk. At the same time, it is automatically opened.

4. Delete the file. Structures that describe the file (index handle, directory entry) are deleted or marked as invalid. The disk space occupied by the file is marked as free (but usually not cleared). This operation is generally not allowed for open files.

5. Reading from a file. Forwarding a certain number of bytes from a file, starting at the current position, to a pre-allocated user mode buffer.

6. Write to a file. Carried out from the current position. If data is already recorded in this item, it is overwritten. Data is sent from a pre-allocated buffer. The operation may change the file size.

7. Move the pointer. You can set the current position pointer anywhere inside the file, or directly at the end of the file.

8. Obtaining and assigning file attributes. Operations allow you to read all or some attributes of a file and set their new values.

Basic operations on directories:

1. Creating a new directory. The newly created directory is usually empty (in some implementations, the elements are added immediately).

2. Delete the directory. At the system call level, this operation is only allowed for an empty directory.

3. Opening and closing the directory. The directory, like a regular file, must be open before use and closed after use. Some directory access operations are only allowed for open directories.

4. Reading directory elements. Reads one directory item and moves the current position to one item.

5. Go to the beginning of the directory. Moves the current position to the beginning of the directory.

Logical file organization is the structure that a programmer deal with. An element of the logical structure of a file is called a logical record. A logical entry is the smallest piece of data that a programmer can handle when exchanging with an external device. Physical exchange with the device is usually carried out in large portions (blocks). Logical records can be:

- fixed length;
- variable length;
- using index tables.

Logical records can have a special field called a key to identify them. In modern file systems, files usually have the simplest structure in the form of a sequence of single-byte records. In them the length of the logical record is fixed and equal to 1 byte, the key is missing.

The physical organization of files describes the rules for placing a file on an external storage device. The file consists of physical units – blocks. A block is the smallest unit that is exchanged with an external device. In some file systems, the term "cluster" is used instead of "block".

Schemes of physical organization:

- continuous placement of files;
- placing files in linked lists: simple linked lists or linked lists with a file placement table;
- indexed file placement.

General FS model:

1. Symbolic level. The symbolic name of a file determines its unique name.

2. Basic level. The unique name of the file determines its characteristics (attributes). Access rights are checked. When you open a file, its attributes are moved to RAM.

3. Logical level. The coordinates of the logical record in the file are determined.

4. Physical level. The block number is determined.

There are two main approaches to determining access rights:

- selective access, when for each file and each user the owner can determine the permissible operations;
- required approach, when the system gives the user certain rights in relation to each resource that (in this case the file) depending on which group the user belongs to.

Classification of file systems:

1. For random access media (for example, hard disk): FAT32 (File Allocation Table), HPFS (High Performance File System), ext2 (Second Extended File System for Linux), etc. Because disk access is several times slower than RAM access, many file systems use asynchronous write changes

to disk to increase performance. To do this, use either a log, such as ext3 (Third Extended File System for Linux), ReiserFS for Linux, JFS (Journaled File System), NTFS (New Technology File System), XFS, or soft updates, and others. Logging is widespread in Linux and used in NTFS. Soft updates – in BSD (Berkeley Software Distribution) systems.

2. For media with sequential access (for example, magnetic tapes): QIC (Queensland Investment Corporation), etc.

3. For optical media – CDs and DVDs: HFS (Hierarchical File System), UDF (Universal Disk Format), etc.

4. Virtual file systems: AEFS (Advanced Encrypted File System), etc.

5. Network file systems: NFS (Network File System), CIFS (Common Internet File System), SSHFS (Secure SHell FileSystem), GmailFS (using Gmail as a filesystem), etc.

6. For flash memory: YAFFS, ExtremeFFS, exFAT.

7. Slightly fall out of the general classification of specialized file systems: ZFS (actually the file system is only part of ZFS), VMware VMFS (so-called cluster file system, which is designed to store other file systems), and others.

FS for optical media. ISO 9660 is a standard issued by the International Organization for Standardization that describes the file system for CD-ROMs. Also known as CDFS (Compact Disc File System). The purpose of the standard is to ensure media compatibility under different operating systems, such as Unix, Mac OS, Windows. The standard extension, called Joliet, adds support for long filenames and non-ASCII characters in names.

DVDs can also use ISO 9660, but the UDF file system is more suitable for them, as it supports large media and is better suited for modern operating systems. Blu-ray Disc (BD) use only the UDF file system.

Universal Disk Format (UDF) is an operating system-independent file system format specification for storing files on optical media.

Versatility and support in different operating systems allows you to use UDF as a FS not only for optical drives, but also for other removable media, such as flash drives and portable hard drives.

UDF allows you to write files to CD-R or CD-RW on a single file without significant loss of disk space. UDF also takes into account the possibility of selective erasure of some files on rewritable CD-RW media, freeing up disk space.

File system metadata, such as the root directory, can be anywhere on the disk, the "root" of the metadata must be in two of the following three places: sector 256, sector (N-257) and (N-1), where N is track size.

UDF is also better suited for DVDs, as it has better support for large volumes – there is no limit of 2 and 4 GB on file size.

Fragmented files are allowed. The version 2.60 (March 1, 2005) attaches a pseudo-overwrite method to sequentially recorded discs (supported since Windows Vista, Linux 2.6, Mac OS X 10.5, NetBSD, OpenBSD 4.7).

Fourth extended FS ext4 or ext4fs. Logged FS, which is used in OS with a Linux kernel. Based on ext3 FS, previously used by default in many GNU/Linux distributions.

The first experimental implementation of ext4 was written by Andrew Morton and released on October 10, 2006 as a patch for Linux kernels versions 2.6.19-rc1-mm1 and 2.6.19-rc1-git8.

The main changes compared to ext3:

- increase the maximum volume of one disk partition to 1 exbybyte (260 bytes) with a block size of 4 kibytes;
- increase the size of one file to 16 tebybytes (244 bytes);
- introduction of a mechanism of spatial (extent) file recording, which reduces fragmentation and increases productivity. The essence of the mechanism is that new information is added to the end of the disk area, selected in advance next to the area occupied by the contents of the file.

The file system exFAT, sometimes called FAT64 – proprietary file system, designed primarily for flash drives. The main advantages of exFAT over previous versions of FAT are: reduction of the number of overwrites of the same sector, which is important for flash drives in which memory cells are irreversibly worn out after a certain number of write operations (this is greatly mitigated by wear leveling), built into modern USB drives and SD cards. This was the main reason for the development of exFAT. Theoretical limit on file size is 264 bytes. The maximum cluster size has been increased to 225 bytes (32 megabytes). Improving the allocation of free space by introducing a bitmap of free space, which can reduce disk fragmentation. Support for the list of access rights has been introduced. Transaction support (optional, must be supported by the device).

FS Btrfs (B-tree FS, «Better FS» or «Butter FS») is FS for Linux, based on the structures of B-trees and works on the principle of "copy-on-write". Published by Oracle Corporation in 2007 under the GNU General Public License (GPL). One of the initial goals of developing this file system was to provide decent competition to the popular ZFS.

It was originally planned to release Btrfs v1.0 (and fix the disk storage format) in late 2008, but the format was not fixed until June 12, 2010.

List of current Btrfs features:

- efficient packaging of small files and indexed directories;
- dynamic allocation of node (there is no limit to the maximum number of files in the file system);
- snapshots;
- various internal roots of file systems – subvolumes;
- object level of reflection and stratification of data;
- data and metadata hashes (improved integrity guarantee, current hash algorithm – CRC-32C, hardware acceleration of which is implemented in the SSE 4.2 instruction set, alternative in plans);

- transparent compression (currently there is zlib (default) and lzo, activated during mounting with the option -o compress = <type>);
- logging of read-records of all data and metadata;
- strong integration with Device Mapper (DM) – a subsystem (module) of the Linux kernel that allows you to create Virtual Block Devices (VBU) to support multiple devices with multiple built-in algorithms for working with RAID – data virtualization technology, which combines several disks into a logical element for redundancy and performance);
- effective incremental backup and mirroring of the FS;
- migration from the ext3 (ext4) FS and back (until the update);
- mode of optimized operation under Solid-state Drive (SSD) – activated during mounting with the -o ssd option);
- defragmentation in working mode;
- deduplication of data after recording by special utilities.

Zettabyte FS (ZFS) is a FS originally created in Sun Microsystems for the Solaris operating system.

This FS supports large amounts of data, combines the concepts of FS and manager of logical drives (volumes) and physical media, innovative data structure on disks, lightweight file systems, as well as easy management of storage volumes.

ZFS is an open source project and is licensed under the CDDL (Common Development and Distribution License).

The main advantage of ZFS is its complete control over physical and logical media. Knowing exactly how the data is located on disks, ZFS is able to provide high speed access to them, control their integrity, as well as minimize data fragmentation. This allows you to dynamically allocate or free up disk space on one or more media for a logical FS.

In addition, there is a variable block size, which best affects performance, concurrency of read-write operations, as well as a 64-bit mechanism for using checksums, which minimizes the probability of invisible data destruction.

FS HFS Plus or HFS+ or Mac OS Extended is FS developed by Apple Inc. HFS, the main FS on Macintosh computers, was previously used for replacement. An iPod player can also work with this file system. HFS+ can be seen as an enhanced version of HFS to enhance the features of Mac OS. During development, this system was called Sequoia.

Like its predecessor, HFS+ uses a tree structure called the B*tree to store most of the metadata.

HFS+ is an improved version of HFS with support for large files (32-bit addressing instead of the old 16-bit) and uses UTF-16 encoding for file and folder names.

FS NTFS was developed as the primary file system for Windows NT in the early 90's. Supports large files and large disks (up to 264 bytes). Resistance to failures. Provides high speed of operations. It has a flexible

structure that allows you to add new record types and file attributes. Supports long character names (up to 255 Unicode characters). Provides access control to directories and files.

The structure of the NTFS volume. The basis of the structure is the main file table (Master File Table, MFT). The MFT contains at least 1 record for each volume file (including itself), the record size is the same for the entire volume and can be 1, 2 or 4 kB (typically 2 kB).

Like any other system, NTFS divides all useful space into clusters – data blocks that are used once. NTFS supports almost any cluster size – from 512 bytes to 64 KB, a certain standard is a cluster size of 4 KB. The NTFS disk is conventionally divided into two parts. The first 12% of the disk is allocated to the so-called MFT zone – the space in which the MFT metafile grows (see below). It is not possible to write any data to this area. The MFT area is always kept empty – this is done so that the most important, service file (MFT) is not fragmented as it grows. The remaining 88% of disk space is normal file storage space.

All files are therefore identified by a file number, which is determined by its position in the MFT. The entire volume is a sequence of clusters (not just a data area). The sequence number of a cluster in a volume is called the Logical Cluster Number (LCN). Cluster numbers are stored in 64-bit pointers.

The basic unit of disk space allocation in NTFS is not a single cluster, but a continuous area of clusters called a segment or extent.

The extent is given by the logical number of its first cluster and the number of clusters in the extent.

The boot sector contains:

1. Standard block of BIOS parameters.
2. The number of clusters in the volume.
3. The initial logical number of the cluster of the main copy of the MFT and the mirror copy of the MFT.

A copy of the boot sector is in the middle of the volume. The first extent of MFT contains 16 standard entries that are created during formatting about NTFS system files. The null MFT record contains a description of the MFT itself, in particular, the addresses of all its extents.

Immediately after formatting, the MFT has one extent at which the system files are written.

When the first non-system file is created, the second MFT extent is created.

The file consists of the attributes set. Each of the attributes is an independent stream of bytes that can be created, deleted, read, and written. Some attributes are standard for all files:

1. Name (File Name).
2. Version (File Version).
3. Standard information about the file, including creation time, recovery time, etc. (Standard Information).

Other attributes depend on the purpose of the file. The directory has an Index Root attribute – a data structure that contains a list of files in that directory. The data attribute contains all the data in the file.

In NTFS, a file can contain more than one data attribute, which is distinguished by name (multiple named data streams are allowed inside the file). By default, a file contains a single, unnamed data stream.

Small attributes that are stored directly in the MFT record are called resident. The Data attribute can be resident.

Large attributes are stored on disk separately, they are called non-resident.

Small files consist of at least the following attributes:

1. Standard information (SI).
2. File name (FN).
3. Data.
4. Security descriptor (SD).
5. Can be completely placed in the MFT record (all attributes are resident).

In large files, the data attribute is non-resident, and pointers to all extents are stored in its resident part.

Very large files have many attributes or are highly fragmented, the data attribute does not contain all pointers to extents. In this case, the file occupies several MFT records. The main record is called the base file record, and the rest – overflow records

The base record of extremely huge files may not have enough space for overflow records.

The Attribute List can be made non-resident. They can use double indirect addressing (non-resident attribute refers to other non-resident attributes).

Laboratory work № 3

COMMAND ROW. WORKING WITH FILES

The purpose of the work is to consolidate knowledge and acquire skills to work with the command line.

Problem statement:

1. The student must know the commands of the FS; must know the theoretical part, which is necessary for work.
2. Required workstation with the Windows OS installed.

Execution of work:

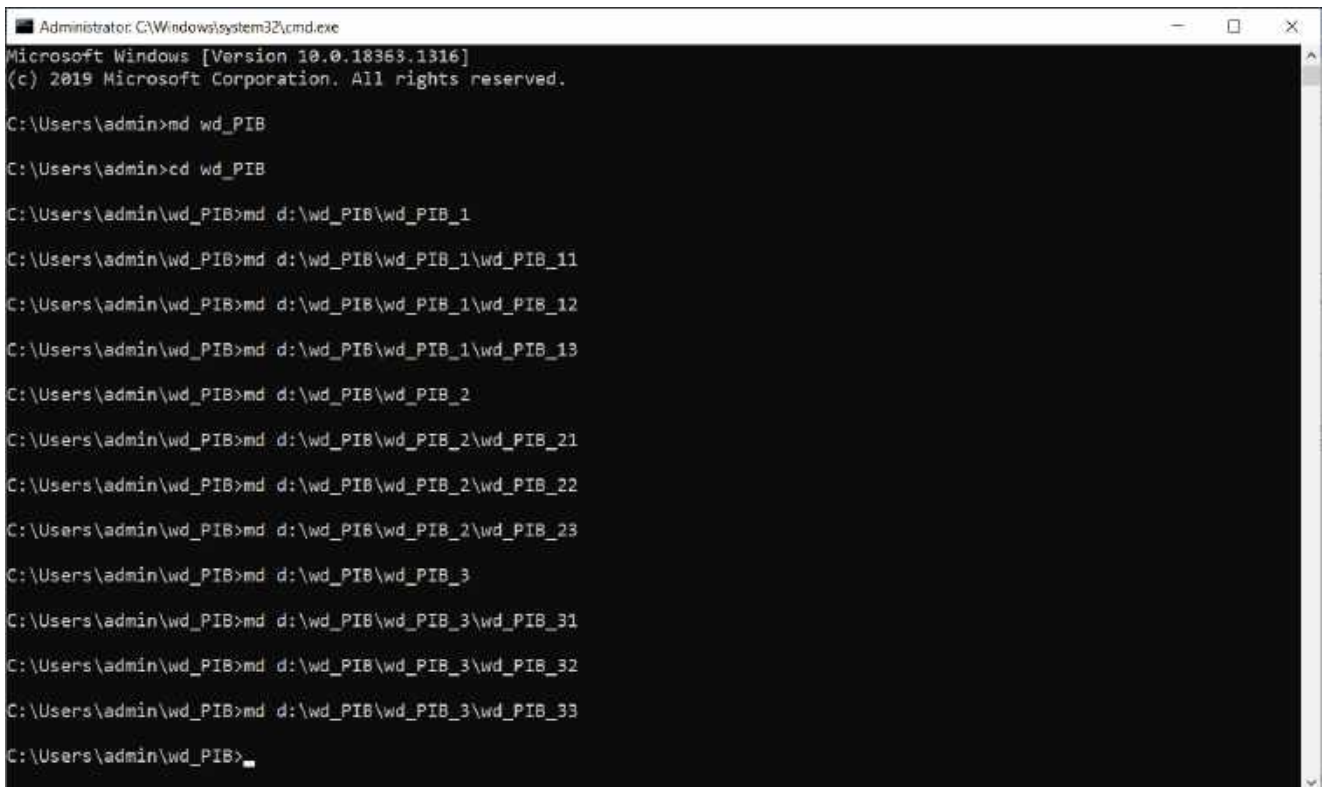
1. On a working disk, for example D:\, create a folder with the name wd_PIB by means of the md command where PIB are the first letters of a surname, a name, a patronymic of the student.

Then go to the created folder in which we create the following folders (Figure 7):

```

md wd_PIB
cd wd_PIB
md D:\wd_PIB\wd_PIB_1
md D:\wd_PIB\wd_PIB_1\wd_PIB_11
md D:\wd_PIB\wd_PIB_1\wd_PIB_12
md D:\wd_PIB\wd_PIB_1\wd_PIB_13
md D:\wd_PIB\wd_PIB_2
md D:\wd_PIB\wd_PIB_2\wd_PIB_21
md D:\wd_PIB\wd_PIB_2\wd_PIB_22
md D:\wd_PIB\wd_PIB_2\wd_PIB_23
md D:\wd_PIB\wd_PIB_3
md D:\wd_PIB\wd_PIB_3\wd_PIB_31
md D:\wd_PIB\wd_PIB_3\wd_PIB_32
md D:\wd_PIB\wd_PIB_3\wd_PIB_33

```



```

Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1316]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\admin>md wd_PIB
C:\Users\admin>cd wd_PIB
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_1
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_1\wd_PIB_11
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_1\wd_PIB_12
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_1\wd_PIB_13
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_2
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_2\wd_PIB_21
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_2\wd_PIB_22
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_2\wd_PIB_23
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_3
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_3\wd_PIB_31
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_3\wd_PIB_32
C:\Users\admin\wd_PIB>md d:\wd_PIB\wd_PIB_3\wd_PIB_33
C:\Users\admin\wd_PIB>_

```

Figure 7 – Display of created folder

2. Change the current folder with the CD command and create a folder wd_PIB_331 and wd_PIB_332.

It is necessary to go directly to disk **cd /d D:\wd_PIB:**

```

D:\wd_PIB>cd wd_PIB_3
D:\wd_PIB\wd_PIB_3>md wd_PIB_331
D:\wd_PIB\wd_PIB_3>md wd_PIB_332
D:\wd_PIB\wd_PIB_3>cd ..

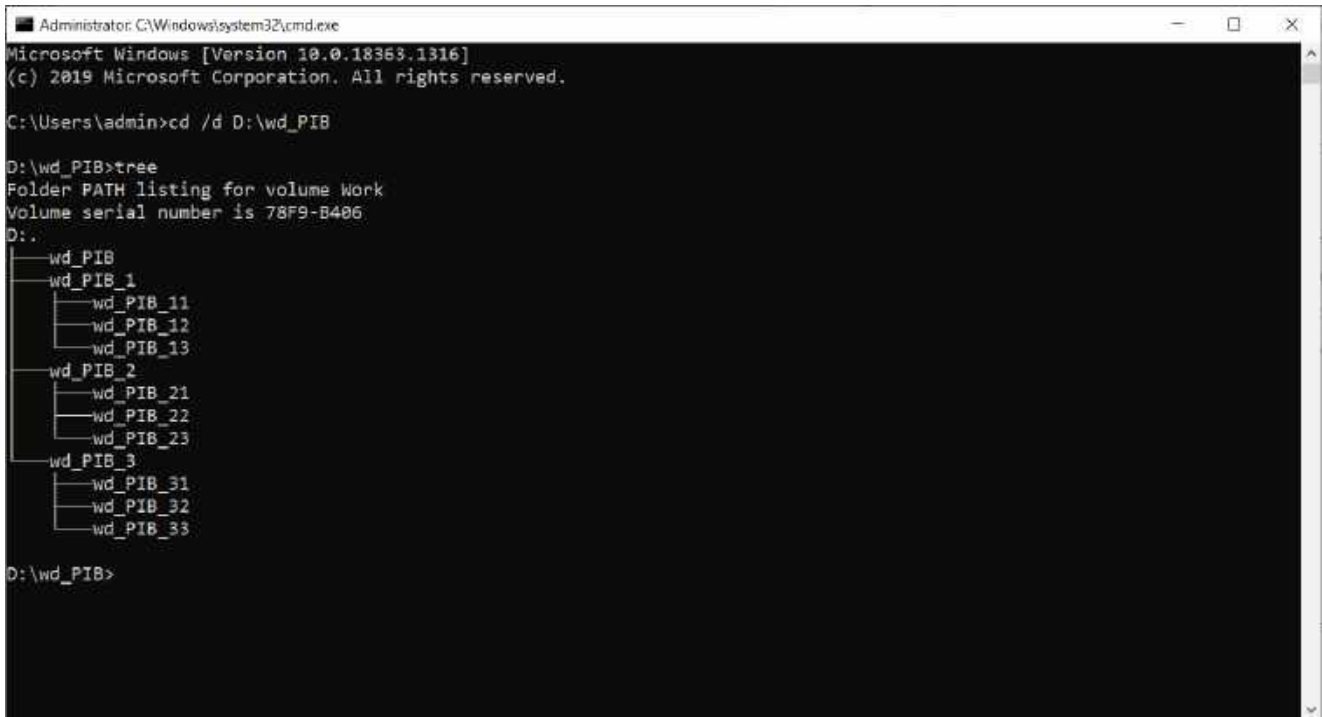
```



```
D:\wd_PIB>cd ..  
cd D:\wd_PIB
```

Use the TREE command to display the folders that were created (Figure 8):

```
D:\wd_PIB>tree
```



```
Administrator: C:\Windows\system32\cmd.exe  
Microsoft Windows [Version 10.0.18363.1316]  
(c) 2019 Microsoft Corporation. All rights reserved.  
  
C:\Users\admin>cd /d D:\wd_PIB  
  
D:\wd_PIB>tree  
Folder PATH listing for volume Work  
Volume serial number is 78F9-B406  
D:.  
├── wd_PIB  
│   ├── wd_PIB_1  
│   │   ├── wd_PIB_11  
│   │   ├── wd_PIB_12  
│   │   └── wd_PIB_13  
│   ├── wd_PIB_2  
│   │   ├── wd_PIB_21  
│   │   ├── wd_PIB_22  
│   │   └── wd_PIB_23  
│   └── wd_PIB_3  
│       ├── wd_PIB_31  
│       ├── wd_PIB_32  
│       └── wd_PIB_33  
  
D:\wd_PIB>
```

Figure 8 – Display command tree

3. Use the RD command to delete directories from the folder wd_PIB_331 and wd_PIB_332:

```
D:\wd_PIB>cd wd_PIB_3  
D:\wd_PIB\wd_PIB_3>rd wd_PIB_331  
D:\wd_PIB\wd_PIB_3>rd wd_PIB_332  
D:\wd_PIB\wd_PIB_3>tree
```

4. To create files, use the echo command. The command is implemented as:

```
echo Text to file> filename.extension
```

where “Text to file” is the text that will be placed in the file.

If the file already exists, its contents will be replaced with "Text to file":

```

D:\wd_PIB>echo FILE1 > f1.txt
D:\wd_PIB>echo FILE2 > f2.txt
D:\wd_PIB>echo FILE3 > f3.txt
D:\wd_PIB>echo FILE4 > f4.pas
D:\wd_PIB>echo FILE5 > f5.pas
D:\wd_PIB>echo FILE6 > f6.pas
D:\wd_PIB>echo FILE7 > f7.doc
D:\wd_PIB>echo FILE8 > f8.doc
D:\wd_PIB>echo FILE9 > f9.doc
D:\wd_PIB>echo FILE10 > f10.bat
D:\wd_PIB>echo FILE11 > f11.bat
D:\wd_PIB>echo FILE12 > f12.bat
D:\wd_PIB>echo FILE13 > f13.exe
D:\wd_PIB>echo FILE14 > f14.exe
D:\wd_PIB>echo FILE15 > f15.exe
D:\wd_PIB>echo FILE16 > f16.gif
D:\wd_PIB>echo FILE17 > f17.gif
D:\wd_PIB>echo FILE18 > f18.gif
D:\wd_PIB>echo FILE19 > f19.com
D:\wd_PIB>echo FILE20 > f20.com
D:\wd_PIB>tree

```

5. You can attach streams to files by executing a command

```
echo thread content > f1.txt:1
```

The command is used to view the contents of the stream:

```

more < f1.txt:1
D:\wd_PIB>echo flow of file 1 > f1.txt:1
D:\wd_PIB>echo flow of file 2 > f2.txt:2
D:\wd_PIB>echo flow of file 3 > f3.txt:3
D:\wd_PIB>more < f1.txt:1
D:\wd_PIB>more < f2.txt:2
D:\wd_PIB>more < f3.txt:3

```

6. Copying files is done using the command COPY:

```

D:\wd_PIB>copy *.txt D:\wd_PIB\wd_PIB_2\wd_PIB_21
f1.txt
f2.txt
f3.txt
Files copied: 3.
D:\wd_PIB>copy *.gif D:\wd_PIB\wd_PIB_2\wd_PIB_22

```

```
f16.gif
f17.gif
f18.gif
Files copied: 3.
D:\wd_PIB>copy *.pas D:\wd_PIB \wd_PIB_2\ud_PIB_23
f4.pas
f5.pas
f6.pas
Files copied: 3.
```

7. Perform the following steps with the command FOR:

```
D:\wd_PIB> FOR %c IN (*.doc *.pas) DO copy %c
D:\wd_PIB\wd_PIB_3\ud_PIB_31\*.*
D:\wd_PIB>copy f7.doc D:\wd_PIB\wd_PIB_3\wd_PIB_31\*.*
Files copied: 1.
D:\wd_PIB>copy f8.doc D:\wd_PIB\wd_PIB_3\wd_PIB_31\*.*
Files copied: 1.
D:\wd_PIB>copy f9.doc D:\wd_PIB\wd_PIB_3\wd_PIB_31\*.*
Files copied: 1.
D:\wd_PIB>copy f4.pas D:\wd_PIB\wd_PIB_3\wd_PIB_31\*.*
Files copied: 1.
D:\wd_PIB>copy f5.pas D:\wd_PIB\wd_PIB_3\wd_PIB_31\*.*
Files copied: 1.
D:\wd_PIB>copy f6.pas D:\wd_PIB\wd_PIB_3\wd_PIB_31\*.*
Files copied: 1.
```

8. Follow these steps to move the files using the command MOVE:

```
D:\wd_PIB>move *.doc d:\ud_PIB\ud_PIB_1\ud_PIB_12
D:\wd_PIB\f7.doc
D:\wd_PIB\f8.doc
D:\wd_PIB\f9.doc
D:\wd_PIB>move *.bat d:\ud_PIB\ud_PIB_1\ud_PIB_13
D:\wd_PIB\f10.bat
D:\wd_PIB\f11.bat
D:\wd_PIB\f12.bat
```

9. Write a report on the work. Contents of the report on laboratory work: topic; goal; setting objectives; the order of execution; research results; conclusion. Copies of screens are provided with the report in accordance with the procedure for conducting research.

Additional information: shell is a separate software product that provides a direct connection between the user and the OS. The text command

line user interface provides an environment in which applications and utilities with a text interface run. In the shell, programs are executed and the result is displayed on the screen in a form similar to the Command.com MS-DOS interpreter. The Windows shell uses the cmd.exe command interpreter, which loads applications and directs the data flow between applications to translate the entered command into an understandable form. The MD command creates a directory:

MD [drive:] path or MKDIR [drive:] path

The CD command displays the name or changes the current directory:

```
CD [/ D] [disk:] [path]
CD [/ ..]
CHDIR [/ D] [disk:] [path]
CHDIR [/ ..],
```

where .. means going to the parent directory.

CD drive command: Displays the name of the current directory of the specified drive.

The CD-free command displays the name of the current disk and directory.

The /D switch is used to change the current disk and directory at the same time.

The name of the current directory in the call string is converted to the same case as for the existing names on the disk.

For example, the C:\ TEMP command will make the current C:\Temp directory, if it exists.

The command does not treat spaces as delimiters, which allows you to go to a subdirectory whose name contains spaces without enclosing the entire directory name in quotation marks.

The TREE command reproduces a graphical representation of the folder or path structure: TREE [disk:] [/F] [/A],

/F – output file names in each folder.

/A – use ASCII characters instead of national alphabet characters.

The RD command deletes the directory: RD [/S] [/Q] [disk:] path,

/S – delete the directory tree, not only the specified, but also all files and subdirectories contained in it.

/Q – disable confirmation request when deleting folder trees with the /S switch.

The MORE command sequentially outputs data in one screen size.

The COPY command copies one or more files to another location.

The FOR command executes the specified command for each set file.

The MOVE command moves files and renames files and folders.

4 MICROSOFT MANAGEMENT CONSOLE WINDOWS

All system management tasks must be performed in a single environment that uses the same techniques and a single user interface. The environment should allow you to perform multiple tasks at the same time. The set of administration tasks must be configured and expanded or reduced at the request of the administrator.

Access to the tools must be performed both on the local machine and remotely, from a variety of clients.

There should be an easy way to transfer tools from one administrator to another at minimal cost.

Microsoft Management Console (MMC) is a network server component that provides a single environment for all administrative programs.

MMC provides a common environment for managing various system and network resources in the Windows environment. The MMC console is actually a shell where modules called snap-ins work and contain real resource management tools.

For example, Windows resources are managed with the Computer management icon, components are managed with the Component services icon, work logs are viewed with the Event Viewer icon, Internet information services are managed with the Internet information services icon, and performance is monitored with the Performance icon. They are all located in the Administrative Tools folder.

In fact, the MMC console does not provide any management functions. Instead, the MMC environment provides full snap-in integration. This allows administrators and other users to create their own management tools from snap-ins created by different administrators.

Administrators can save the tools they have created for later use and share them with other administrators and users. This model allows administrators to delegate administrative tasks by creating different tools of different levels of complexity and transferring those funds to users who will perform those tasks. For example, you can create a special console in which a user is only allowed to add users, configure mailboxes, and create shared folders in an exchange organization.

The management console itself does not provide any control functions, but is only a snap-ins environment.

MMC windows load control programs called snapshots, each of which performs a specific administrative role. Administrators define the minimum control units for the management console.

The snap-ins are organized in the form of a tree, and all together form a complete set of administrative tools. At the same time, the administrator can download several snap-ins he needs and manage the system from one console, which saves him from downloading different programs and constantly switching between them.

Administrators can not only independently perform control functions, but also due to the developed software interfaces to act as elements of integration of various applications. The console thus becomes an environment where programs work.

According to the method of use, all equipment is divided into two types:

- standalone or snap-in ensures that its functions are performed even in the absence of other equipment, such as Computer Management;
- extension snap-in that can only work when connected to the parent snap-in.

The task of the snap-expansion is to expand the functionality of the nodes included in the parent equipment. Examples include the Administrative Templates and Extended View extensions, which are part of the Group Policy Object Manager.

The Microsoft Management Console 3.0 interface allows you to open multiple documents (Multiple Document Interface, MDI) at the same time.

The console window contains the main menu and toolbar, a set of which is specific to each connected snap-in. The main menu provides file and window management functions, as well as access to the help system. The working console window contains a preview panel (left panel), a results panel (center panel), and an actions panel (right panel).

The overview panel displays the MMC tool namespace as an object tree. This tree contains all visible nodes, each of which is a managed object, task, or viewer.

You can manage the system using ready-made snap-ins collected in the program group Administration (Administrative Tools). However, in some cases you need additional snap-ins or you may need to create a console with its own set of features.

1. Type `mmc` in the Start menu or in the command prompt window – an empty console window will open.

2. On the File menu, click Add/Remove Snap-in. A window will open containing a list of all snap-ins available in the system. The Available snap-ins list contains available snap-ins, and the Selected snap-ins list shows the snap-ins that are already connected.

3. Select the Event Viewer snap-in and click the Add button.

If you are creating an MMC for another user with limited rights, it may be helpful to restrict the console from being changed. This requires the following operations:

1. On the Console (File) menu, click Options.

2. In the window, in the Console mode list, select the value for user mode – full access. In this mode, the user will not be able to add new snap-ins to the tool, but will be able to change the location of windows.

3. You can also prevent another user from changing the appearance of the console by clearing the Allow the user to customize views check box.

The MMC tool, which combines the maximum number of control functions, has become a computer management administrator.

It allows you to manage system services, server resources (shared network resources connected to the server by users and files opened on it), system devices, view events logged, monitor events.

This application is essentially an administrator tool, with its help you can view all installed services in the system, find out their status, as well as determine the account on whose behalf they are launched, and the order of launch. To get information about services, you need to select the branch Services and applications\Services (System Tools\Services). If you select one of the services, a dialog box appears in which you can manage the service.

Here you can specify the name of the service, specify the file to run, select the type of startup, specify the account on whose behalf the service will be started, specify actions to restore the service after a failure, etc.

Disk management includes three branches:

- resources;
- sessions;
- open files.

With these branches you can get information about the resources provided by the computer for network sharing; create new resources; view active computer sessions and, if necessary, terminate them; receive information about all files opened by connected users.

The event log viewer allows you to view three logs: applications (Application Log), security (Security Log) and system (System Log).

The application log contains data related to the operation of applications and programs. Entries in this log are created by the applications themselves. The decision on what events to log is made by the developer.

The system log contains event logs registered by Windows 2000 system components. For example, the system log records failures when loading a driver or other system components at system startup. The types of events that are logged in the system are predefined.

The security log records security events, such as legal and illegal logon attempts, and resource-related events (such as creating, opening, or deleting files). Decisions about events, information about which is entered in the safety log, be made by administrator. For example, after a login audit is allowed, information about all login attempts is logged in the security log.

Three types of messages are entered in the logs:

- error;
- warning;
- message.

The system monitor consists of two parts: System Monitor Graph, which is used to display the chart on the screen, and the System Monitor Log Service, designed to log the monitoring data.

Windows receives performance data from local computer components. All running system components generate performance data. This data is described as a performance object, commonly referred to as a data-generating component. For example, a Processor object is a set of processor performance data on a local system.

Active Directory allows you to store information about all network objects in a hierarchical structure and gives administrators a quick search and access to it. Active Directory provides:

- one-time registration in the network;
- a single point of administration of all objects in the network;
- execution of queries from any attribute of any object;
- hierarchy and extensibility of directories.

The operating system directory service stores information about system objects and allows you to manipulate them.

Active Directory allows you to centrally administer all resources, any arbitrary objects and services: files, peripherals, databases, web-connections, accounts, and more.

The directory service is a model of domains, and the domain is a set of computers with a common database of user accounts and a single security policy.

Each domain has its own unique name. It can reflect either its functional purpose, or geographical location, or something that is clear only to one author of the name.

MMC is very similar to Microsoft Windows Explorer. MMC uses a multi-document interface, i.e. you can download and display several console windows in the MMC parent window at the same time.

The MMC interface allows you to place snap-ins inside the overall management console. This allows you to display all snap-ins in the same way, although they may perform their tasks differently. Actually, the console does not contain any control functions; it just acts as a shell for snap-ins. The equipment is always in the console, they are never performed by themselves.

One of the main advantages of the MMC console is the support for tool customization. You can create specialized MMC consoles for custom management tasks and then delegate those consoles to other administrators. These tools can be specialized for certain management requirements of different administrators' groups.

Laboratory work № 4

OPERATING SYSTEM ADMINISTRATION UTILITIES

The purpose of the work is to study the OS administration utility Windows – Microsoft Management Console.

Problem statement:

1. Get acquainted with the possibilities of using MMC snap-ins.

2. Learn the basics of administering MMC snap-ins.

Execution of work:

1. Analyze the MMC snap-ins and group them by purpose.
2. Identify the allocated resources of devices, as well as possible conflicts in the allocation of these resources.
3. Create a security template and read the list of policies that are set by this template.
4. Create your configuration for MMC (Figure 9), to which add:
 - Active Directory equipment;
 - Dynamic Host Configuration Protocol (DHCP) equipment;
 - print server management;
 - computer management;
 - service management;
 - set of scripts and utilities;
 - set of network resources;
 - set of remote desktops.



Figure 9 – The screenshot of the empty console

5. Rename the console and create a new taskbar (Figure 10).

6. Write a report on the work. Contents of the report on laboratory work: topic; goal; setting objectives; the order of execution; research results; conclusion. Copies of screens are provided with the report in accordance with the order of research.

Additional information: The main menu line of the MMC console always contains three menus: Console, Window and Help. The Window and Help menus are arranged in the standard way. The Window menu allows you to manage console windows if more than one window is open in the MMC. The Help menu provides access to the general help of MMC Help, as well as help for snap-ins that are currently downloaded. The Console menu contains most of the actions. From this menu you can open and save consoles and even create new consoles. You can also add snap-ins and remove them from open consoles, as well as set general MMC settings. You can set the following parameters:

- Console Title specifies the name of the console that appears in the MMC header line.
- Console Mode.

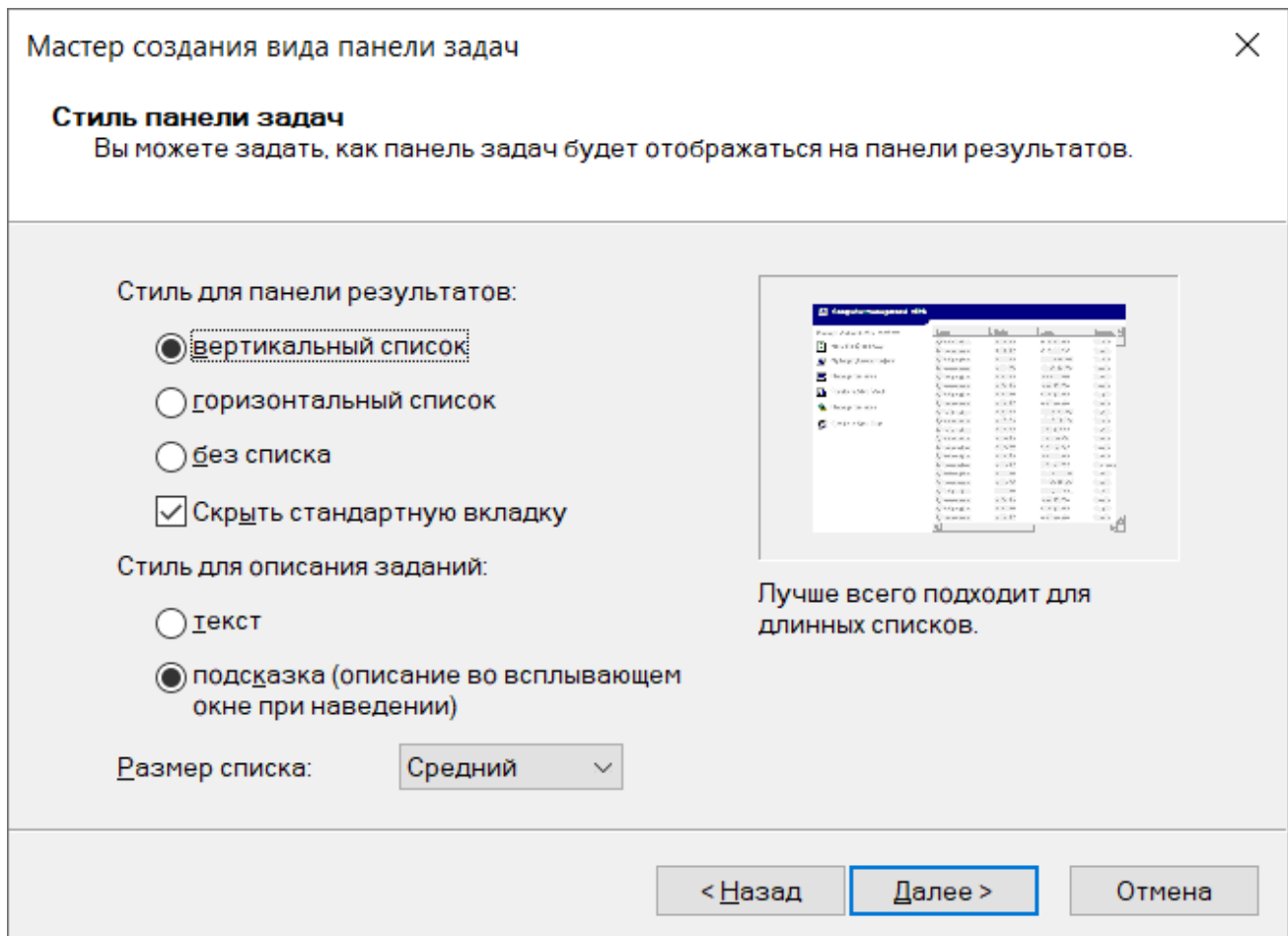


Figure 10 – Form of the taskbar layout creation

Author mode gives the user full access to all MMC functions. User mode is used in three versions:

- 1) Full Access mode gives the user access to all MMC commands, but does not allow you to add or remove snap-ins or change console properties;
 - 2) Limited Access Multiple Window mode gives the user access only to those parts of the console tree that were present on the screen at the time of saving the console, as well as open new windows;
 - 3) Limited Access Single Window mode works in the same way as Limited Access Multiple Window mode, but users cannot open new windows.
- Other settings determine whether users can access icons in taskbars, save changes made to the console, and customize presentation forms.

The MMC toolbar is a standard Windows toolbar that provides access to the most commonly used MMC commands. By default, these are the commands to open and save the console, create a new console, and open a new window.

The snap-in action bar actually combines a menu bar and a toolbar; you can split them if you want by dragging. Although the menus themselves (Action, View and Favorites) remain the same regardless of the console you open, the commands of these menus may vary slightly according to the open console.

The Action menu usually contains commands that apply to any container or object that you select in the appropriate console. These commands duplicate the commands that appear in the context menu when you right-click on an object.

The Action menu allows you to perform actions such as creating new views for viewing and objects in containers, opening object property pages, and accessing tasks.

The View menu allows you to control the form of information presented in the Details window of your console. You can, for example, change the view in the form of icons to a view to the list form or adjust the view of the columns.

The Favorites menu allows you to add items to the list of this menu and organize the list by categories. The Favorites list can contain shortcuts to tools, console items, or tasks. The Favorites tab in the scope window allows you to view the items in the Favorites menu list. The part of the action bar corresponding to the toolbar provides quick access to some of the most commonly used actions related to selected objects. Many of the toolbar icons change depending on the object highlighted in the scope window or in the details window of your console.

The scope window contains a hierarchical container structure called the console tree. Some containers are displayed as unique icons that graphically represent the type of items they contain. Other containers are displayed as folders, indicating that they simply contain other objects. To open a container and display its objects, you need to click on the "plus" sign next to this container. To close the container again, click the "minus" sign.

The detail window changes depending on the contents of the container selected in the scope window. In other words, this window displays the results of the currently selected scope.

The details window can use different ways of displaying information, called representations. With the View menu, you can get standard views – large or small icons, a list or a detailed list. The View menu also allows you to customize the appearance of the columns that are displayed in the scope windows and details. The details window you can change the order of the column in alphabetical order or by time.

In addition to standard views, you can also create views in the form of a taskbar shown in the Details panel. The view in the form of a taskbar corresponds to a dynamic HTML page (DHTML), which contains commands available for the selected item in the scope window. Each command displays a task consisting of an image, a label, a description, and the mechanism by which the equipment should execute that command. Users can execute these commands by clicking on the task icon.

Purpose of taskbars:

- 1) include icons for all tasks that a particular user may need to perform;
- 2) group tasks by type of activity or users by creating multiple views in the form of a taskbar inside the console;
- 3) create simplified to-do lists. For example, you can include tasks in the taskbar and then hide the console tree;
- 4) simplify component tasks. For example, if a user frequently performs a task that consists of multiple snap-ins and other tools, you can organize icons in one area of the screen for those tasks to launch relevant property pages, command lines, dialog boxes, or scripts.

The root node is the upper nodal point of the snap-in; its name is usually based on the name of the relevant product or task. The MMC console supports standalone snap-ins.

Standalone snap-ins, such as Exchange System, provide management features without the need for support from other snap-ins. There is only one root node for each stand-alone snap-in. Parental snap-in in the console tree is required for snap-expansion. Equipment-extensions increase the functionality of other equipment.

Exchange Server is a great example of an object-oriented hierarchical cataloging environment. All elements that make up the Exchange structure are objects that interact with each other in any way. Objects that can be seen in the windows of the scope and details can be divided into two types.

Containers can contain both container objects and objects of another type. Container objects can also appear in the details window. They are used to logically group all the objects that make up the control environment. The administrator uses container objects to organize the tree and then move through that tree.

A list object is simply an object that cannot contain other objects. The most common email objects that the administrator works with on a daily basis are servers and connectors.

All objects in the MMC console are managed by using the properties page. The properties page is a dialog box that can be opened by selecting an object and then selecting Properties from the Action menu. This dialog box consists of one or more tabs that contain controls for setting a group of desired properties.

Each MMC tool consists of instances set of smaller tools called MMC equipment. A snap-in is a minimal console expansion unit that represents a unit of control behavior. Equipment to perform its task can trigger other supporting controls and dynamically connected libraries (DLLs).

The equipment expands the MMC console by setting and activating control behavior. They can perform this behavior in a number of ways. For example, a snap-in can add items to the container tree or extend the action of a particular tool by adding context menu items, toolbars, tabs on property pages, or Help to existing snap-ins. There are two main types of equipment:

- standalone equipment provides control functions, even if only this equipment is inside the console. This equipment is not based on other snaps present in the console. An example of such a standalone snap-in is the Exchange System;

- snap-ins can perform a variety of functions, but only in conjunction with the parent snap-in. Some snap-ins expand the console namespace, while others simply expand the context menus.

Many equipment items support both modes of operation, introducing some stand-alone features as well as extending the functionality of other snap-ins.

Equipment is usually delivered in groups, called packages. For example, the Microsoft Windows XP OS includes one or more snap-in packages. In addition, some companies may supply products that consist entirely of snap-in packages. Grouping of equipment on packages is convenient for loading on communication channels and installation. It also allows multiple snap-ins to share core DLLs so that these DLLs do not need to be placed in each snap-in.

The MMC console has the functionality to create specialized management tools. Administrators can create, save, and then delegate a custom console that contains several snap-ins considering for special tasks. Administrators can combine these special features into a tool (also called a document) that works in one of the instances of MMC.

After composing this tool, the administrator can save it in a MSC file and download this file later for instant playback of this tool. You can also email an MSC file to another administrator, who can then download the file and use the tool.

You can characterize the MMC console as a control panel. By adding all the necessary tools, you can save a lot of time. Everything you need is always in front of you, in one place.

The modular system allows to unite in one place various equipment, it can be both the monitor of resources of the server and the most usual browser of web pages. The management console is on its way:

C:\Windows\System32\mmc.exe.

MMC consoles have two startup modes:

- author's – in which there is complete freedom of action and it be able to remove and add snap-ins;

- intended for the user – prohibits changes in the structure and is suitable for the end user.

You can change the startup mode in the console itself or, for example, for authoring mode, start with option.

5 STUDY OF THE INTERNAL DEVICE OF THE WINDOWS OPERATING SYSTEM

In most multi-user operating systems, programs are separated from the actual OS: its kernel code is executed in a privileged mode CPU (called kernel mode), which provides access to system data and hardware.

Application code runs in non-privileged CPU mode (called user-defined) with an incomplete set of interfaces, limited access to system data, and no direct access to hardware.

When a custom mode program calls a system service, the CPU intercepts the call and switches the calling stream to kernel mode. After the end of the system service, the OS switches the flow context back.

Windows is a monolithic OS – in the sense that much of its code and drivers use the same protected kernel mode memory space. This means that any OS component or device driver is potentially capable of corrupting data used by other OS components.

Although some declare it as such, Windows is not a microkernel-based OS in the classical sense of the term. In such systems, the main components of the OS (memory managers, processes, I/O) are executed as separate processes in their own address spaces.

An example of a modern system with an architecture based on a microkernel – Mach OS, developed at Carnegie Mellon University. It implements a tiny kernel that includes thread scheduling, messaging, virtual memory, and device driver services. Everything else, including a variety of APIs, file systems, and network support, runs in user mode.

There are four types of user processes:

- fixed system support processes – for example, the process of processing the login and the session manager, which are not Windows services (i.e., started by the service management manager);

- service processes – carriers of Windows services such as Task Scheduler and Spooler. Many Windows server applications, such as Microsoft SQL Server and Microsoft Exchange Server, also include components that run as services;

- user applications – there are six types:

- 1) 32-bit Windows;
- 2) 64-bit Windows;
- 3) 16-bit Windows 3.1;
- 4) 16-bit MS-DOS;
- 5) 32-bit POSIX;
- 6) 32-bit OS/2;

- environment subsystems – implemented as part of the OS environment support provided to users and programmers.

Initially, Windows NT came with three subsystems: Windows, POSIX, and OS/2. The latter was removed in Windows 2000. As for Windows XP, it only comes with the Windows subsystem.

Windows includes the following kernel mode components:

1. Windows executive system, which contains basic OS services that provide memory management, processes and flows, protection, I/O and interaction between processes.

2. Kernel Windows, which contains low-level OS features that support, for example, scheduling, interrupt and shutdown scheduling, and synchronization when using multiple CPUs. It also provides a set of procedures and basic objects used by the executive system to implement higher-level structures.

3. Device drivers, which include hardware device drivers that broadcast user I/O requests to device-specific requests, as well as network drivers and file system drivers.

4. Hardware abstraction layer (HAL) that isolates the kernel, drivers, and Windows executable system from the specifics of the hardware on that hardware platform (for example, from differences between motherboards).

5. A windowing and graphics system that implements GUI functions, better known as the Windows functions of the user and Graphics Device Interface (GDI) modules. These features provide support for windows, user interface controls, and graphics.

Windows portability between systems with different hardware architectures and platforms is achieved in two main ways.

Windows has a multilevel structure. CPU or platform-specific low-level parts of the system are made in separate modules. Due to this, the high-level part of the system does not depend on the specifics of architectures and hardware platforms. The key components that provide OS migration are the kernel (contained in the Ntoskrnl.exe file) and the hardware abstraction level (HAL) (contained in the Hal.dll file). Functions specific to a particular architecture (flow context switching, trap scheduling, etc.) are implemented in the kernel.

The vast majority of Windows components are written in C and only some of them are written in C++. Assembler language was used only when creating system parts that interacted directly with system hardware (for example, when writing an interrupt trap handler) or required exceptional performance (for example, when switching context). Assembler code is not only in the kernel and HAL, but also in some other parts of the OS: procedures that implement interlocking, the mechanism for calling local procedures (LPC), part of the Windows subsystem running in kernel mode, and even in some user libraries mode (for example, in the code to start processes in Ntdll.dll – system library).

Multitasking is an OS mechanism that allows you to use a single CPU to run multiple threads. However, true simultaneous execution of, for example,

two threads is possible only if the computer has two CPUs. With multitasking, the system only creates the appearance of simultaneous execution of multiple threads, while a multi-CPU system actually executes several threads at once – one on each CPU.

Windows is an OS that supports symmetric multiprocessing (SMP).

This model does not have a main CPU. The OS, like custom threads, can run on any CPU. In addition, all CPUs use the same memory. In asymmetric multiprocessing (ASMP), the system, on the contrary, selects one of the CPUs to execute the OS kernel code, and the other CPUs execute only the user-defined code.

Windows XP and Windows Server 2003 support two types of multi-CPU systems: logical processors (hyperthreading) and NUMA (Non-Uniform Memory Architecture).

Logic processors are technologies created by Intel, one physical CPU there can be some logical. Each logical CPU has its own state, but the execution engine and the onboard cache are common. This allows one of the logical CPUs to continue running while the other logical CPU is busy (for example, interrupt processing that prevents threads from running on that logical CPU). Scheduling algorithms in Windows XP have been optimized for computers with such CPUs.

In NUMA systems, CPUs are grouped into blocks called nodes. Each node has its own CPU and memory, and it connects to other nodes with a special bus. Windows in the NUMA system still works as an SMP system, in which all CPUs have access to all the memory – just access the memory local to the node is faster than the memory in other nodes. The system seeks to increase performance by allocating time to streams on the CPU, which are in the same node as the memory used. It also tries to allocate memory within the node, but if necessary, allocates memory from other nodes.

Windows was originally designed to support up to 32 CPUs, the multiprocessor model does not have any internal features that would limit the number of CPUs used to 32. Just this number is easy to imagine a bit mask using a machine 32-bit data type. Indeed, 64-bit versions of Windows support up to 64 CPUs, because the word size on 64-bit CPUs is 64 bits.

The actual number of CPUs supported depends on the specific release of Windows. This number is stored in the registry parameter

HKLM\SYSTEM\CurrentControlSet\Control\Session\Manager\Licensed-Processors

Scalability is one of the key goals of multi-CPU systems. For correct execution in SMP-systems, the OS must strictly meet certain requirements. Solving the problems of competition for resources and other issues in multi-CPU systems is more difficult than in single-CPU, and this must be taken into

account when designing a system. Some features of Windows proved crucial to its success as a multi-CPU OS:

- ability to execute OS code on any available CPU and on several CPUs at the same time;
- several threads of one process can be executed in parallel on different CPUs;
- fine synchronization within the kernel (spin-lock, spin-lock with queues, etc.), device drivers and server processes allow you to run more components on multiple CPUs at the same time;
- mechanisms such as I/O completion ports that facilitate the efficient implementation of multithreaded server processes that are well scalable in multi-CPU systems. The scalability of the Windows kernels later improved. For example, in Windows Server 2003, there are scheduling queues that are individual to each CPU, which allows you to schedule threads in parallel on multiple machines.

The differences between client and server versions differ in the following parameters:

- number of supported CPUs;
- the amount of supported physical memory;
- possible number of simultaneous network connections (for example, in the client version a maximum of 10 simultaneous connections with the service of access to shared files and printers is allowed);
- availability of non-Professional services in Server editions (for example, directory services, clustering support and multi-user terminal service).

A special debug version of Windows is called a checked build. It is only available to Professional (or higher) Microsoft Developer Network (MSDN) subscribers. The release check is to recompile the Windows source code for which the "DBG" flag (a debugger and profiler with open source that forces you to include debug and trace the compilation stage) has been set to TRUE.

The validation release is intended primarily for device driver developers, as this version performs more stringent error checking when calling kernel mode functions by device drivers or other system code.

Windows has three subsystems: OS/2, POSIX, and Windows. Starting with Windows XP, the basic POSIX subsystem does not come with Windows, but a much better version can be obtained for free as part of the Services for UNIX product.

The POSIX subsystem, which is abbreviated to "portable operating system interface based on UNIX", is a set of international standards for UNIX-type operating system interfaces.

The Windows subsystem differs from the other two in that Windows cannot work without it (this subsystem handles everything related to the keyboard, mouse and screen, and is needed even on servers in the absence of

interactive users). The Windows subsystem always works, the start information of the subsystem is stored in the registry key:

HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\SubSystems

Environment subsystems provide applications with a subset of the basic services of the Windows executive system. Each subsystem provides access to different subsets of built-in Windows services.

Each executable image (EXE) belongs to one – and only one – subsystem. When the image is started, the code responsible for creating the process receives the subsystem type specified in the image header and notifies the corresponding subsystem of the new process. The type is specified by the /SUBSYSTEM specification in the link command in Microsoft Visual C++, it can be viewed using the EXE type utility, which is part of Windows resources.

When the application calls one of the functions of the DLL subsystem may be one of three:

- the function is fully implemented in the user mode inside the DLL subsystem. In other words, no messages are sent to the environment subsystem process, and Windows executable services are not called.

- this feature requires one or more Windows Executive calls. For example, Windows functions ReadFile and WriteFile access internal undocumented I/O services – according to NtReadFile and NtWriteFile.

- the function requires the performance of any operations in the environment subsystem process (such processes running in user mode are responsible for servicing client applications that run under their control).

The Windows subsystem consists of the following basic elements.

1. Process subsystem environment (Csrss.exe), which provides:

- support for console (text) windows;
- support for creating and deleting processes and threads;
- partial support for 16-bit DOS virtual machine (VDM) processes;
- many other features, such as GetTempFile, DefineDosDevice, ExitWindowsEx, and several natural language support features.

2. Kernel mode driver (Win32k.sys), including:

- Window Manager, which controls the drawing and display of windows on the screen, accepts input from the keyboard, mouse and other devices, as well as transmits user-defined messages with applications;

- GDI, which is a library of functions for graphics output devices. GDI includes features for manipulating graphics and playing lines, text, and shapes.

DLL-modules of subsystems (Kernel32.dll, Advapi32.dll, User32.dll and Gdi32.dll), which translate calls of documented Windows API functions into calls of corresponding (and mostly undocumented) kernel mode services with Ntoskrnl.exe and Win32k.sys.

Graphics device drivers, which are hardware-specific graphics display, printer, and miniport video card drivers.

To create on-screen user interface controls, such as windows and buttons, applications can call standard user functions. Window Manager forwards these calls to the GDI, which in turn transmits these calls to the graphics device drivers, where they are formatted for display.

GDI provides a set of standard two-dimensional graphics features that allow applications that have no idea about graphics devices to access them. GDI functions act as an intermediary between applications and display and printer drivers. GDI interprets application requests for graphics output and sends appropriate driver requests.

POSIX standards have encouraged manufacturers to maintain the compatibility of their UNIX-like interfaces, thus allowing programmers to easily migrate their applications between systems.

The OS/2 environment subsystem, like the POSIX subsystem, has very limited functionality and supports only 16-bit OS/2 applications version 1.2 with symbolic or graphical I/O.

In addition, Windows prohibits applications from directly accessing hardware and therefore does not support OS/2 programs, uses advanced video I/O, or includes privileged I/O segments that attempt to execute IN/OUT instructions (to access some hardware devices).

Ntdll.dll is a special library of system support, required mainly when using DLL subsystems. It contains two types of functions:

- 1) system service dispatch stubs to Windows runtime services;
- 2) internal support functions used by DLLs, subsystems, and other OS components.

The first group of features provides an interface to Windows runtime services that can be called from user mode. There are more than 200 such functions.

The executive system is at the top level of Ntoskrnl.exe (the kernel is at the lower level). It includes functions of the following type:

1. Exported functions available for calling from user mode. These functions are called system services and are exported through in Ntdll, most services are available through the Windows API or other subsystem APIs. However, some of them are not available due to documented functions.

Device driver functions called through the DeviceControl function. The latter is a universal interface from user mode to kernel mode to perform functions in non-read or write device drivers.

Exported features are available for call only from kernel mode and are documented in the Windows Driver Development Kit (DDK) or Installable File System (IFS) Kit (www.microsoft.com/whdc/ddk/ifskit.default.mspx).

Exported functions that are only available for calling from kernel mode, but are not described in the Windows DDK or IFS Kit (for example, functions

used by a video driver that are in the boot stage and whose names begin with Inbv).

Functions defined as global but not exported characters. Include internal support functions called in `ntoskrnl.exe` (NT OS kernel – "NT operating system kernel"); their names begin with `lop` (I/O Manager support function) or `Mi` (memory management support function).

Internal functions in any module that are not defined as global characters.

The executive system consists of the following main components:

1. Configuration Manager, which is responsible for implementing and managing the registry.

2. Process and thread manager, which creates and final processes and threads. Low-level support for processes and threads is implemented in the Windows kernel, and the executive system complements these low-level objects with its semantics and functions.

3. Security status monitor, which implements security policies on the local computer. It protects OS resources by auditing and controlling access to objects during execution.

4. I/O Manager, which implements hardware-independent input-output and is responsible for sending I/O commands to the required device drivers for further processing.

5. Plug and Play (PnP) Manager, which determines which drivers are needed to support a particular device, and downloads them. The requirements of each device in hardware resources are determined in the process of enumeration. Depending on the requirements of each device, the PnP manager allocates resources such as I/O ports, IRQ (Interrupt ReQuest), Direct Memory Access channels (DMA), and memory areas. He is also responsible for sending appropriate messages about changes in system hardware (when adding or removing devices).

6. Power Manager, which coordinates power-related events and generates power management system messages sent to drivers. When the system is not busy, the manager can be set to stop the CPU to reduce power consumption. Changing the power consumption of individual devices relies on their drivers, but is coordinated by the power manager.

7. Windows Management Instrumentation routines that allow drivers to publish information about their performance and configuration, as well as receive commands from the Windows Management Instrumentation (WMI) service for user mode.

8. Cache Manager, which increases the performance of I/O file by storing in the main memory of disk data that has recently been accessed.

9. Memory Manager, which implements virtual memory – a memory management scheme that allows you to allocate each process a large closed

address space, the amount of which may exceed the available physical memory.

10. Logical preselection tool that accelerates the start of the system and processes by optimizing the loading of data that is accessed when starting the system or processes.

The executive system consists of four main groups of support functions used by the above components. About a third of them are described in the DDK (Driver Development Kit), because drivers also use them.

Object Manager creates, manages, and deletes executable system objects and abstract data types that are used to represent OS resources such as processes, threads, and various synchronizing objects.

Low Pin Count (LPC) mechanism transmits messages between client and server processes on one computer. LPC is a flexible, optimized version of Remote Procedure Call (RPC).

Large set of standard library functions for string processing, arithmetic operations, data type conversion and security structure processing.

Subroutines to support the executive system, for example, to allocate system memory (pools pumps and do not pump pages), memory access with interlocking, as well as two special types of synchronizing objects – resource and high-speed mutex).

In Windows, the kernel consists of a set of functions that are in the Ntoskrnl.exe file. This set provides the basic mechanisms (flow scheduling and synchronization services) used by the components of the executable system, as well as support for low-level architecture-dependent hardware (for example, interrupt and exception scheduling), which has differences in the architecture of each processor.

Like various auxiliary functions, a number of kernel functions are documented in the Windows Driver Kit (WDK) and their descriptions can be found when searching for functions beginning with the K-prefix, because they are required to implement device drivers.

In addition, the kernel consists of functions set in Ntoskrnl.exe that provide the fundamental mechanisms used by executive system components and low-level hardware-dependent support tools (interrupt and exception scheduling) that are different in each CPU architecture. The kernel code is written mainly in C, and the assembler was used only to solve specific problems that are difficult to implement in programming language C.

Like executive system support functions, some kernel functions are described in the DDK because they are required to implement device drivers.

Also, the kernel consists of low-level, well-defined and well-predictable primitives and OS mechanisms that allow higher-level executive system components to perform their functions. The core is separated from the rest of the executive system; it implements systemic mechanisms and does not participate in decision-making related to systemic policy.

Outside the kernel, the executive system represents flows and other collective resources in the form of objects. Management of these objects requires certain costs, as you need descriptors to manipulate objects, means of protection and quotas of resources reserved when they are created.

The kernel itself is separated from the rest of the executable system by implementing the mechanisms of the OS. It leaves almost all decisions, except for the planning and scheduling of flows implemented by the kernel, to the executive system.

Outside the kernel, the executable system represents streams and other shared resources as objects. These facilities require some costs.

In a kernel that implements a set of less complex objects, called "kernel objects", such costs are eliminated, which helps the kernel to manage the main processing and support the creation of executable objects.

You can avoid such costs in the kernel because it implements a set of simpler objects called kernel objects. These objects allow the kernel to control CPU data processing and support executable system objects.

One of the kernel objects groups, called control objects, defines the semantics of controlling various OS functions. This group includes deferred procedure call (DPC) objects, and several objects used by I/O Manager.

Another group of objects – dispatcher objects implements synchronization tools that allow you to change the scheduling of flows.

The group of such objects includes:

- kernel thread;
- mutex;
- event;
- semaphore;
- timer, expected timer (waitable timer) and some others.

Using kernel functions, the executive system creates kernel objects, manipulates them, and constructs more complex objects that are provided in user mode, dispatcher objects implement synchronization tools that allow you to change the flow schedule.

Another major task of the kernel is to abstract or isolate executing systems and device drivers from the differences in hardware architectures supported by Windows. Even for those hardware-dependent features, the kernel design tries to maximize the amount of total code. The kernel supports a set of portable and semantically identical interfaces for different architectures. The main part of the code that implements these portable interfaces is also identical for different architectures.

Some of these interfaces for different architectures are implemented differently or partially implemented using architecture-specific code.

Other examples of architecture-specific kernel code include interfaces to support translation buffer and CPU cache. Because of the ways to implement cache, this support requires the use of different code for different architectures.

Another example is context switching. Although the same algorithm is used at a high level to select a thread and switch the context (the context of the previous thread is saved, the context of the new thread is loaded, and a new thread is started), there are architectural differences in implementations on different processors. Because the context is described by the state of the processor (its registers, etc.), the stored and loaded context varies depending on the architecture.

Another important task of the kernel is to abstract or isolate the executive system and device drivers from discrepancies between hardware architectures supported by Windows (i.e., differences in interrupt handling, exception scheduling, and synchronization between multiple CPUs).

The kernel architecture aims to maximize code generalization – even in the case of hardware-dependent functions. The kernel supports a set of semantically identical and portable interfaces. Much of the code that implements portable interfaces is also identical for different architectures.

But one part of these interfaces is implemented differently on different architectures, and the other includes code specific to a particular architecture. Architecturally independent interfaces can be called on any machine, and the semantics of the interface will be the same – regardless of the specific code for a particular architecture.

The kernel also contains a small portion of code with x86-specific interfaces needed to support older MS-DOS programs. These interfaces are not portable in the sense that they cannot be called on a machine with a different architecture where they are simply absent.

One of the most important elements of Windows design is its portability between different hardware platforms.

Hardware abstraction layer (HAL) is a key part of this portability. HAL is a bootable kernel mode module (Hal.dll) that provides a low-level interface to the hardware platform on which Windows is running.

It hides hardware-dependent details, such as I/O interfaces, interrupt controllers, and processor interoperability, any function that has both architectural and machine dependencies.

Therefore, instead of directly accessing the hardware, internal Windows components, as well as user-written device drivers, support platform portability by calling for platform-dependent information when needed. For this reason, HAL routines are documented in the WDK. Contact the WDK for more information on HAL and its use by device drivers.

Although several HAL modules are included in the operating system (Table 1), Windows has the ability to determine which HAL module to use during boot.

One of the most important features of the Windows architecture is portability between different hardware platforms. The key component that provides such portability is the level of HAL.

HAL hides from the OS the specifics of a particular hardware platform, including its I/O interfaces, interrupt controllers and mechanisms of interaction between CPUs, i.e. all the functions depending on the architecture and the specific machine.

Table 1 – The level of hardware abstractions

Name of HAL-file	Supported systems
Halacpi.dll	Personal computers with Advanced Configuration and Power Interface (ACPI). Intended only for a single-processor machine without the support of an advanced programmable interrupt controller – APIC (the presence of any of these controllers will force the system to use instead the HAL-module)
Halmacpi.dll	Personal computers with an Advanced Programmable Interrupt Controller (APIC) have ACPI. The presence of APIC implies support for symmetric multiprocessing – Symmetric Multiprocessing (SMP)

Device drivers are downloadable kernel mode modules (usually .sys files), they form the interface between the I/O Manager and the corresponding hardware. These drivers run in kernel mode in one of three contexts:

- in the context of the user stream that initiated the I/O procedure;
- in the context of the kernel mode system flow;
- as a result of an interrupt (and therefore not in the context of any process or flow that was current at the time of the interrupt).

There are several types of device drivers.

Hardware device drivers that control (via HAL) equipment write the output data to them and receive data input from a physical device or from a network. There are many types of such drivers – drivers for buses, interfaces, mass storage devices, etc.

FS drivers are Windows drivers that take I/O file requests and translate them into device-specific I/O requests. FS filter drivers that provide disk mirroring and encryption, I/O interception, and some additional information processing before taking it to the next level.

Network redirectors and servers, which are file system drivers that transmit file system requests to I/O from other computers on the network and receive similar requests from them.

Protocol drivers that implement network protocols such as Transmission Control Protocol/Internet Protocol (TCP/IP), Network Basic Input/Output System (NetBIOS), and internetwork packet exchange/sequenced packet exchange (IPX/SPX).

Kernel streaming filter drivers that operate on a chain to process streaming data, such as when recording and playing audio and video information.

Because installing a device driver is the only way to add kernel mode to a third-party code system, some programmers write drivers simply to access internal functions or OS data structures that are not accessible from user mode.

Device drivers in Windows do not work with the hardware directly, but call to interface with the hardware function in HAL.

Drivers are usually written in C (sometimes in C++) and therefore, when used correctly, HAL routines can be transferred as source code between the CPU architecture supported by Windows, and as an executable file can be transferred within the architectural family.

In terms of the Windows driver model (WDM), there are three types of drivers:

1. A bus driver that serves a bus controller, adapter, bridge, or any other device that has child devices. Tire drivers are required for the system to work and are generally supplied by Microsoft. Each type of bus (PCI, PCMCIA and USB) has its own driver in the system.

2. Function driver is the main driver of the device that provides its functional interface. Required, except when the device is used without drivers (i.e. I/O is performed by the bus driver or bus filter drivers, as in the case of SCSI Pass Through Direct (SCSI PassThru)).

3. The filter driver supports additional functionality of the device (or existing driver) or modifies I/O requests and responses from other drivers (this is often used to correct devices that provide incorrect information about their hardware requirements). Such drivers are optional, and there may be several. They can operate both at a higher level than a functional driver or bus driver, and at a lower level.

In a WDM environment, all aspects of a device are controlled by more than one driver: the bus driver sends PnP reports to devices connected to its bus, and the functional driver controls the device itself.

PnP technology literally translates as "plug and play" is a technology designed to quickly identify and configure devices in your computer and other technical devices.

In most cases, the filter drivers at the lower level change the behavior of the device.

Top-level filter drivers usually give the device some additional properties. For example, a device driver such as a keyboard at the top level may impose additional security checks.

The Windows Driver Foundation (WDF) toolkit simplifies the development of Windows drivers by providing two environments: the KernelMode Driver Framework (KMDF) and the User-Mode Driver Framework (UMDF).

The KMDF environment provides a simple interface to WDM and hides all its complexities from the driver creator without changing the original bus-

function-filter model. KMDF drivers are responsible for events that they can register and make calls to the KMDF library to perform work that is not specific to the equipment they manage, such as general power management or synchronization.

The UMDF environment allows you to create drivers of certain classes (mainly based on USB or other buses that use the protocol with long delays). Essentially, UMDF runs each user mode driver as a user mode service and uses advanced local procedure call (ALPC) to communicate with the kernel mode driver, which provides actual access to the hardware. If the UMDF driver crashes, the process terminates and usually reboots, so the system remains stable – the device simply becomes unavailable until the service that hosts the driver reboots.

Each Windows system runs the following system processes:

1. Process idle (includes one thread per CPU to account for idle CPU time).
2. The System process (contains most of the kernel mode system threads).
3. Session Manager (Smss.exe).
4. Windows Subsystem (Csrss.exe).
5. Login process (Winlogon.exe).
6. Service Management Manager (Services.exe) and its created child service processes (for example, universal process for hosting services, SvcHost.exe).
7. Server process of local authentication (Lsass.exe).

To understand the relationship between these processes, it is useful to review the "tree" of processes, which reflects the relationship between parent and child processes. Seeing who creates a process makes it easier to understand where each process comes from.

Laboratory work № 5

WINDOWS SCRIPT HOST PROGRAMMING

The purpose of the work is to consolidate knowledge and acquire Windows Script Host programming competencies.

Problem statement:

1. The student must know the programming commands in Windows Script Host and must know the theoretical part, which is necessary for work.
2. Required workstation with the Windows family OS installed.

Execution of work:

1. Let's create the first educational script in the usual notebook. Launch Notepad. Listing for the Hello.js file (Figure 11):

```
WScript.Echo ("Hello, Name!");
```

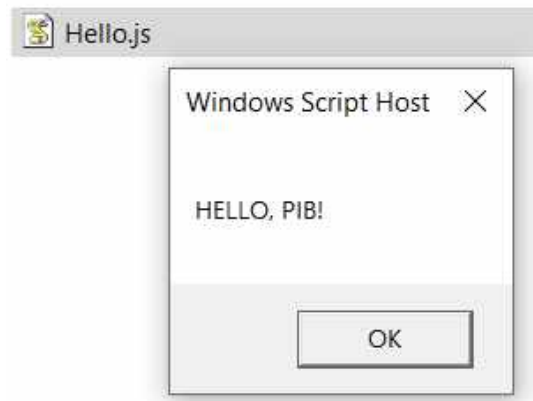


Figure 11 – Result for the Hello.js file

Save the file with a .js extension (instead of .txt). Let it be Hello.js. Open Explorer and click on the created file. WSHost will start the JScript engine and execute the script.

As a result of the actions the message will be "Hello, full name!"

2. Drag-and-Drop in WSH

As mentioned above, WSH has rich capabilities for working with files, the registry and so on. In addition, WSH supports Drag'n'Drop. With the WScript.Arguments object in your script, you get the names of the files you need to work with. Save the dragndrop.vbs file (listed below) to disk, open Windows Explorer, and now simply drag any files to the script that will show their names (paths to them). Listing for the dragndrop.vbs file (Figure 12):

```
' Example Drag'n'Drop
Set objArgs = WScript.Arguments
For I = 0 to objArgs.Count - 1
    WScript.Echo objArgs(I)
Next
```

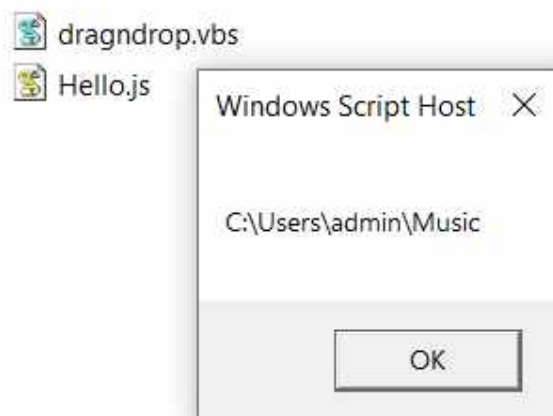


Figure 12 – Result for the dragndrop.vbs file

3. Create WSH shortcuts

Consider an example of creating a shortcut to the calculator and place it in the startup. Listing for the createshortcut.vbs file (Figure 13):

```
'This example shows how to create a shortcut to a program
'and copy it to the specified folder

L_Welcome_MsgBox_Message_Text    = _
"This script will create a shortcut to the calculator and
place it in the Startup"
L_Welcome_MsgBox_Title_Text      = "For Programmer Magazine"

Call Welcome()

'Create a shortcut

Dim WSHShell
Set WSHShell = WScript.CreateObject("WScript.Shell")

Dim MyShortcut, MyDesktop, StartupPath

'Find out the path to the special Startup folder
StartupPath = WSHShell.SpecialFolders("Startup")

'Create a shortcut for Startup
Set MyShortcut = WSHShell.CreateShortcut(StartupPath & _
"\Calculator shortcut.lnk")

'Set the properties for the shortcut
MyShortcut.TargetPath =
WSHShell.ExpandEnvironmentStrings("%windir%\calc.exe")
MyShortcut.WorkingDirectory =
WSHShell.ExpandEnvironmentStrings("%windir%")
MyShortcut.WindowStyle = 4
MyShortcut.IconLocation = _
WSHShell.ExpandEnvironmentStrings("%windir%\calc.exe, 0")
MyShortcut.Save

WScript.Echo " Calculator shortcut created and placed in
Startup "

Sub Welcome()
    Dim intDoIt

    intDoIt = MsgBox(L_Welcome_MsgBox_Message_Text, _
                    vbOKCancel + vbInformation, _
                    L_Welcome_MsgBox_Title_Text )
    If intDoIt = vbCancel Then
        WScript.Quit
    End If
End Sub
```

Verify creating a shortcut with the msconfig command from the command line.

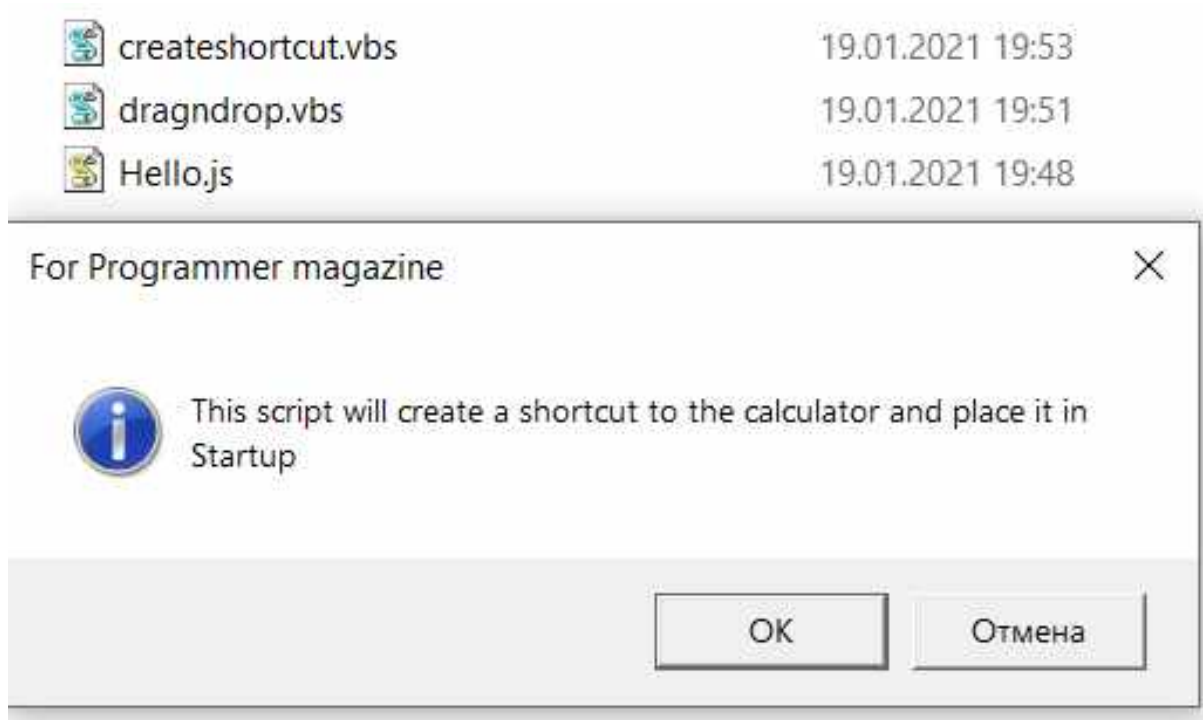


Figure 13 – Result for the createshortcut.vbs file

4. Special folders

In the previous task, we used one such Startup folder. But the fact is that in the English version of Windows a similar folder is called Startup, and if you specify the exact path like C:\WINDOWS\Main menu\Programs\Startup, it will work in the other version of Windows and nowhere else. To avoid these problems, Microsoft has created a list of special folders. You can use the SpecialFolders property of the WshShell object to access any special folder. Listing for the specfol.js file (Figure 14):

```
//Desktop of the current uservar  
WshShell = WScript.CreateObject("WScript.Shell");  
strDesktop = WshShell.SpecialFolders("Desktop");  
WScript.Echo(strDesktop);  
//Desktop common to all users  
strDesktop = WshShell.SpecialFolders("AllUsersDesktop");  
WScript.Echo(strDesktop);  
//Startup  
strDesktop = WshShell.SpecialFolders("Startup");  
WScript.Echo(strDesktop);  
//Main menu  
strDesktop = WshShell.SpecialFolders("StartMenu");  
WScript.Echo(strDesktop);
```

 createshortcut.vbs	19.01.2021 19:53
 dragndrop.vbs	19.01.2021 19:51
 Hello.js	19.01.2021 19:48
 specfol.js	19.01.2021 19:55

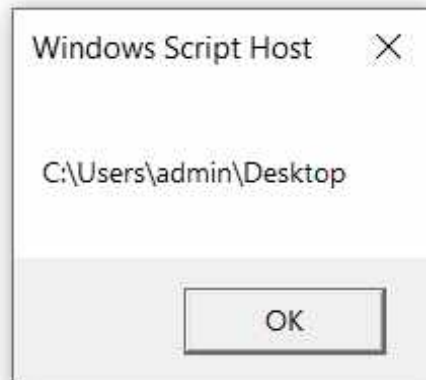


Figure 14 – Result for the specfol.js file

5. Work of WSH with the register

Working with the registry provides even more opportunities to realize your tasks. To do this, use the object `Wscript.Shell` with its methods `RegRead`, `RegWrite`, `RegDelete`.

Here is the task of creating the entry in the registry that allows you to change the title in Internet Explorer. Listing for the `ie_title.vbs` file (Figure 15):

```
'To change the title to Internet Explorer

Dim WshShell, bKey
Set WshShell = WScript.CreateObject("WScript.Shell")

WshShell.RegWrite "HKCU\Software\Microsoft\Internet
Explorer\Main\Window Title", " I read the Programmer magazine",
"REG_SZ"

WScript.Echo "Launch your browser and look at the header!"

'to delete the created line,
'remove the comment from the next line
'WshShell.RegDelete "HKCU\Software\Microsoft\Internet
Explorer\Main\Window Title"
```

6. Launch other applications

The `WshShell` object allows you to run other programs. In the following task, it necessary to run two copies of Notepad – a new copy and the contents of the script.

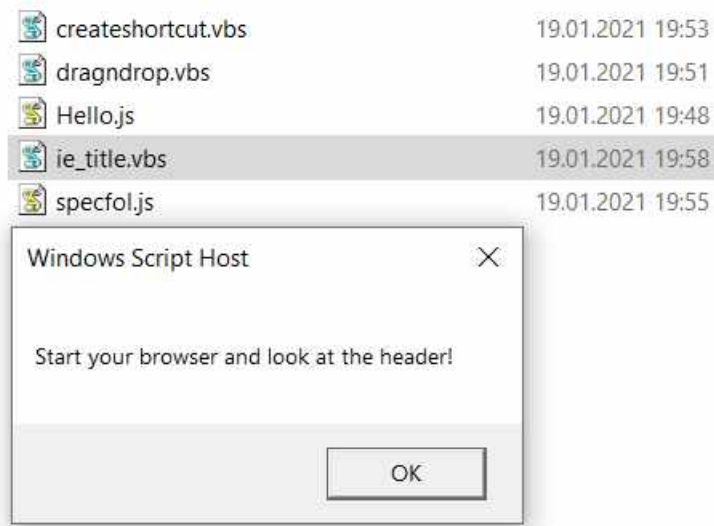


Figure 15 – Result for the ie_title.vbs file

Listing for the book.vbs file (Figure 16):

```
Set WshShell = WScript.CreateObject("WScript.Shell")
'Launching a new instance of Notepad
WshShell.Run "%windir%\notepad"
'Opening our script in Notepad
WshShell.Run "%windir%\notepad " & WScript.ScriptFullName
```

book.vbs	19.01.2021 20:03	Файл сценария V...	1 КБ
createshortcut.vbs	19.01.2021 19:53	Файл сценария V...	2 КБ
dragndrop.vbs	19.01.2021 19:51	Файл сценария V...	1 КБ
Hello.js	19.01.2021 19:48	файл JavaScript	1 КБ
ie_title.vbs	19.01.2021 19:58	Файл сценария V...	1 КБ
specfol.js	19.01.2021 19:55	файл JavaScript	1 КБ

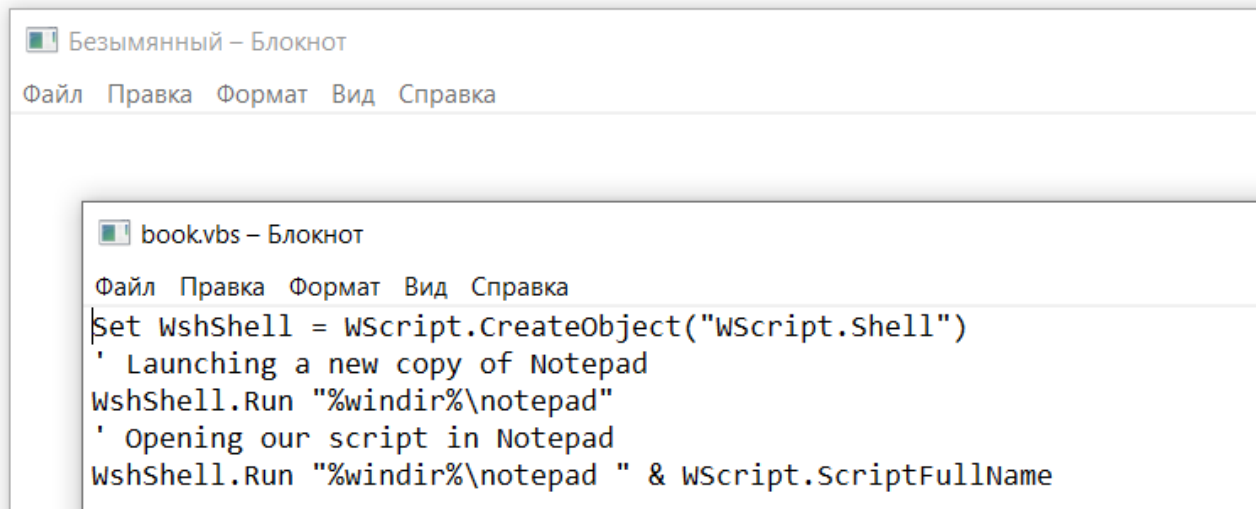


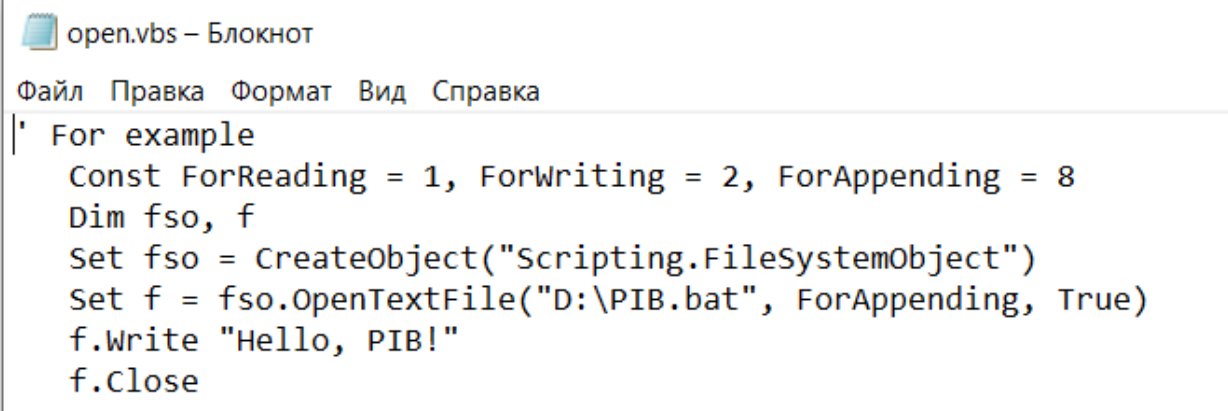
Figure 16 – Result for the book.vbs file

7. WSH work with files

It can also easily copy, move and delete files and folders. It is not difficult to work with viewing the contents of the file and editing it. Having mastered these capabilities, it get the opportunity to work with files autoexec.bat and config.sys, check the values of parameters in win.ini, keep logs of users. To do this, initialize the file system object (FSO), and then use one of its methods to open the file. Listing for the open.vbs file (Figure 17):

```
'For example only!  
Const ForReading = 1, ForWriting = 2, ForAppending = 8  
Dim fso, f  
Set fso = CreateObject("Scripting.FileSystemObject")  
Set f = fso.OpenTextFile("D:\auto.bat", ForAppending, True)  
f.Write "Hello, PIB!"  
f.Close
```

 book.vbs	19.01.2021 20:03	Файл сценария V...
 createshortcut.vbs	19.01.2021 19:53	Файл сценария V...
 dragndrop.vbs	19.01.2021 19:51	Файл сценария V...
 Hello.js	19.01.2021 19:48	файл JavaScript
 ie_title.vbs	19.01.2021 19:58	Файл сценария V...
 open.vbs	19.01.2021 20:07	Файл сценария V...
 specfol.js	19.01.2021 19:55	файл JavaScript



```
open.vbs – Блокнот  
Файл Правка Формат Вид Справка  
' For example  
Const ForReading = 1, ForWriting = 2, ForAppending = 8  
Dim fso, f  
Set fso = CreateObject("Scripting.FileSystemObject")  
Set f = fso.OpenTextFile("D:\PIB.bat", ForAppending, True)  
f.Write "Hello, PIB!"  
f.Close
```

Figure 17 – Result for the open.vbs file

Therefore, OpenTextFile opens the file, if it does not exist, a new file is created (Figure 18). Of the parameters passed to the method, only the first is required – the file name. Other parameters may not be specified. The second parameter sets the file opening mode. Possible values are:

- 1) 1 – the file is opened for reading only;
- 2) 2 – the file is opened for writing. Moreover, all the contents of the file, which was before, will be destroyed;

3) 8 – the file is opened to add data. Everything you write down will be added to the end of the file. Quite convenient for keeping different logs.

The third variable can be true or false. It shows whether to create a new file if it did not exist before. If the value is true, a new file will be created.

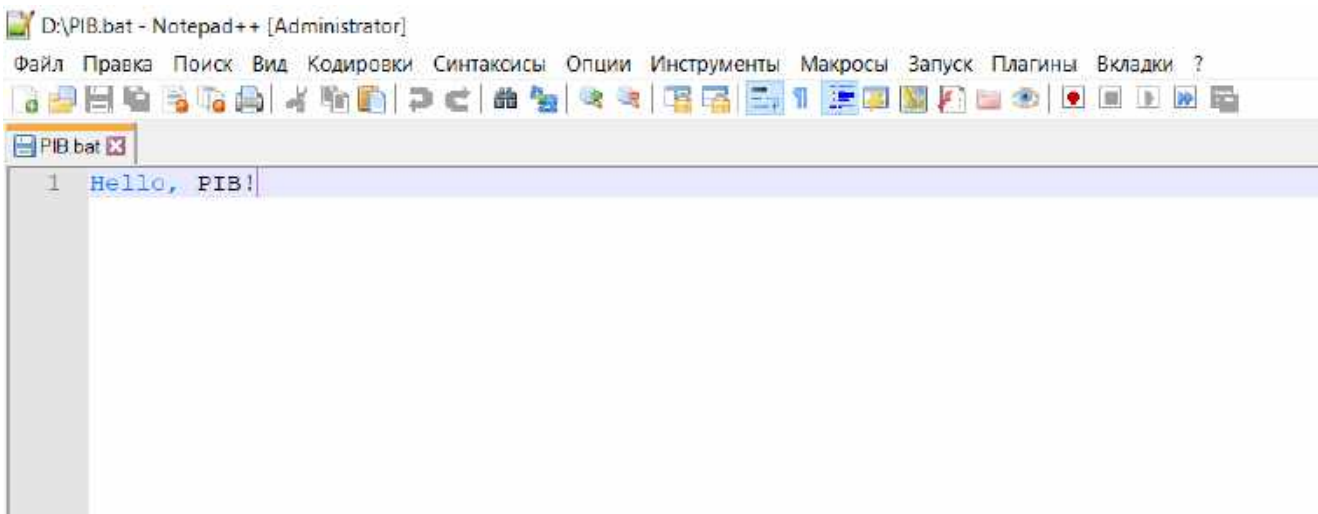


Figure 18 – Created file with result for the open.vbs file

8. Write a report on the work. Contents of the report on laboratory work: topic; goal; setting objectives; the order of execution; research results; conclusion. Copies of screens are provided with the report in accordance with the order of research.

Additional information: Windows Script Host (WSH) is a Microsoft Windows component designed to run scripts in the JScript and VBScript scripting languages, as well as in other optional languages. That is, it is a special component of the OS that allows you to run scripts written in JS (Java Script), VBS (Visual Basic Script) and other languages. If it does not work properly, various failures can occur during Windows startup and operation. WSH files are written in scripting languages, which include JScript, VBScript, Active Perl, Python and some others. They allow you to perform a certain sequence of actions on the device. There are several advantages to bat files. The most important advantage is the increased range of actions:

1. Create shortcuts for applications.
2. Turn off the device.
3. Making adjustments to the OS registry.
4. Working with the network.
5. Switch users and get information about them.
6. Making changes to environment variables.
7. Screen output of different types of information messages.

Only some of the possible actions are listed above. The complete list is diverse and allows to write scripts for almost any situation.

Windows Script Host was a great improvement for the OS, which replaced MS-DOS. What is especially important, learning to use the proposed opportunities is quite simple and will not be difficult. WSH is especially useful for administrators. This is due to the peculiarities of making changes to the register. Does not provide for the display of any messages about the need to confirm actions. This factor allows you to perform the desired work without distracting the user.

Windows Script Host documents can be divided into two main categories, depending on the extension:

- 1) js – JScript is used for writing;
- 2) vbs – are created using VBScript.

There are some differences in syntax, but they are little. If you are familiar with one of the described languages, then there will be no problems in understanding the other. Special libraries must be used to read the files. On Windows OS, they are built-in and will not require additional downloads.

Some versions of the library package do not exist and should be downloaded from the official site. To clarify this point is quite simple – it needs to enter in the search wscript.exe. It must be used a text editor to write a WSH document. You can use specialized programs, which simplifies the task for the user.

Windows Script Server can be used for a variety of purposes, including logon scripts, management, and general automation. Microsoft describes it as a management tool. WSH provides an environment for running scripts – this calls the appropriate script handler and provides a set of services and objects for the script to run.

These scripts can be run in any GUI mode (WScript.exe) or in command line mode (CScript.exe) offering user flexibility for interactive or non-interactive scripts.

The Windows Management Toolkit also scripts this way. WSH implements an object model that provides a set of Component Object Model (COM) interfaces.

Unless otherwise noted, any WSH scripting engine can be used with different Windows server software packages to provide Common Gateway Interface (CGI) scripts – the standard interface used to communicate an external program with a web-server.

Current versions of the default WSH motors and all or most third-party motors have sockets capability as well; like the CGI script or otherwise, PerlScript is the choice of many programmers for this purpose and VBScript and various Rexx engines are also rated as quite powerful in communication and word processing.

6 WINDOWS OPERATING SYSTEM ARCHITECTURE. PROCESSES AND FLOWS IN OPERATING SYSTEMS

A program is a static set of commands, and a process is a container for a set of resources used to execute a program.

Components of the process:

- sequence of executable instructions of the processor;
- a set of memory addresses (address space) in which processor commands and data for them are located.

At the highest level of abstraction, the process in Windows includes the following:

- closed virtual address space – a range of virtual memory addresses that can be used by the process;
- executable program – executable code and data projected on the virtual address space of the process;
- list of open descriptors (handles) of various system resources – semaphores, communication ports, files and other objects available to all streams in this process;
- security context, called an access token, and identifies the user, security groups, and privileges associated with the process;
- unique process identifier (in intrasystem terminology also called client identifier: process and flow identifiers are generated from a single namespace and never overlap);
- at least one thread.

To view (and modify) processes and related information, there is a set of utilities that include:

- in Windows itself;
- in Windows Support Tools;
- Windows Debugging Tools;
- Windows resources;
- Platform Software development kit (SDK);
- number of utilities can be found at <https://docs.microsoft.com/en-us/sysinternals/>.

In multithreaded systems, a process is a collection of one or more threads and a secure address space in which those threads run.

A thread is some entity within a process that receives CPU time to execute. Therefore, it is a set of sequentially executed processor instructions. Without a flow, the process program cannot run.

The stream includes the following most important elements:

- the contents of a set of CPU registers that reflect the state of the CPU;
- two stacks, one of which is used by the thread when running in kernel mode, and the other – in user mode;

- closed area of memory, called local memory flow (thread-local storage, TLS) and used by subsystems, libraries of executing systems;
- Run-time libraries and DLLs;
- unique flow identifier;
- sometimes threads have their own security context, which is usually used by multi-server applications, replacing the security context of the served clients.

Models of processes and flows:

1. Single-task systems – one address space (one process), in which one thread can run.
2. Some modern embedded systems – one address space (one process), in which several threads can run.
3. Single-threaded process model (traditional UNIX systems) – many processes, but in each of them only one thread.
4. Multithreading, or thread model (most modern OS) – many processes, each of which can have many threads.

Advantages of multithreading:

- implementation of different types of parallelism (concurrency) – multiprocessor computing, I/O, user interaction, distributed applications;
- scalability (especially with increasing number of processors);
- less resources are needed than to support processes;
- efficient data exchange via shared memory.

Disadvantages of multithreading:

- complexity of development and adjustment of multithreaded applications;
- reducing the reliability of applications (possible "races", memory leaks, data loss);
- the ability to reduce the performance of applications.

In a multitasking computer system, processes can take on different states. These states may not actually differ from the OS kernel, but they are a convenient abstraction for understanding processes.

The various states of the process are shown in the state diagram, in which the arrows show the transitions between states. As you can see, some processes are stored in main memory and others in secondary (virtual) memory.

In all types of computer systems, processes have the following states:

1. Created.
2. Ready.
3. Launched.
4. Blocked.
5. Completed.

The most important task of the OS when managing processes and threads is to organize the switching of the context – the transfer of control from

one thread to another while maintaining the state of the processor. The general principles of context switching are followed in most systems, but their implementation is determined by a specific architecture. It usually needs to do the following:

- save the state of the stream processor in some area of memory (storage area of the state of the stream processor);
- determine which flow should be performed next;
- download the processor status of this thread from its storage area;
- continue executing the code of the new thread.

Context switching is usually done with hardware support.

Special registers and memory plots can be used to store information about the current task, as well as special processor instructions for working with these registers and memory plots.

To save the state of the processor of each task (the contents of the associated registers of the processor) use a special area of memory – a Task State Segment (TSS).

The address of this area can be obtained from the Task Register (TR). This is the system address register. To switch tasks, it is enough to load new data into the TR. As a result, the CPU register values of the current task will be automatically saved in its state segment, after which the processor registers of the new task will be loaded into the processor registers and its instructions will be executed.

In the process of execution, a thread can be interrupted not only to switch the context to another thread, but also in connection with a software or hardware interrupt (context switching is also associated with interrupts, in fact, with an interrupt from the timer).

With each interrupt comes additional information (for example, its number). Based on this information, the system determines where the address of the interrupt handler procedure will be placed (the list of such addresses is stored in a special area of memory and is called the interrupt vector).

Here is an example of a sequence of actions during interrupt processing:

- maintaining the state of the flow processor;
- setting the interrupt handler stack;
- start of the interrupt handler (OS code); to do this, a new value of the command counter is loaded from the interrupt vector;
- restore the state of the flow processor after the execution of the handler and continue the execution of the flow.

Process creation and completion tools allow you to dynamically change the set of running programs in the OS. Thread creation and completion tools are the basis for creating multithreaded programs.

There are four main events that lead to the creation of processes:

1. System initialization. (winload.exe, winresume.exe, ntoskrnl.exe, smss.exe, hal.dll, wininit.exe, services.exe; init, rc, ttytab, getty).

2. Execution by the started process of a system call of creation of process.

3. User request to create a process.

4. Initialization of a batch task.

Transferring control to the interrupt handler, as well as context switching, can occur at almost any time. The main difference is that the address to which control is transferred is set based on the interrupt number and stored in the interrupt vector, and that the handler code does not continue from where the execution was interrupted, but starts running again and again.

On UNIX systems, the system call `fork()` is used to create the process, on Windows – `CreateProcess()`.

After creating a new process, both processes have their own address space.

The initial state of the child process is an exact copy of the parent process. Necessary actions to create a process:

- create information structures that describe the process (control unit: descriptor, context);

- allocate RAM;

- load the code segment of the process into RAM;

- put the process descriptor in the queue of finished processes.

The process ends in one of the following four ways:

1. Normal completion (voluntary).

2. Completion due to error (voluntary):

- no file;

- restriction of access.

3. Completion due to a fatal error (forced):

- incorrectly written program;

- division by zero;

- reference to a non-existent or forbidden area of memory.

4. Destruction by another process (forced).

Make a system call `exit()` on UNIX or `ExitProcess()` on Windows.

The process address space consists of virtual memory addresses set that this process can use.

Addresses can be associated with RAM, and can with resources stored in memory.

The process address space is not available to other processes.

The process has system resources such as files, network connections, I/O devices, synchronization objects.

The process does not have direct access to the processor.

The process contains startup information for the threads that are created in it.

The process must contain at least one thread.

Scheduling – the distribution of CPU time between processes or threads.

Planning tasks:

- determining the time to change the flow being performed;
- selection of the next thread to perform;
- switching contexts.

The first two tasks are solved mostly by software, the third – mainly by hardware using the interrupt mechanism.

Thus, scheduling is the provision of alternate process access to a single processor.

The scheduler is the responsible part of the OS.

Scheduling algorithm – the algorithm used for scheduling.

Situations when planning is necessary:

- process creation;
- completion of the process;
- process blocking on I/O operations, semaphores, etc.;
- when interrupting I/O.

Types of planning:

1. Long-term planning. Long-term scheduling tools determine which programs need to be loaded into memory to run. This layout is also called static because it does not depend on the current state of the system.

2. Medium-term planning. Medium-term scheduling tools control the transition of flows from a suspended state to a state of readiness and back. The control blocks of ready-to-run threads are organized in memory into a structure called a ready-made thread queue.

3. Short-term scheduling, or CPU scheduling. Responsible for the following actions: when stopped a thread? which ready-to-run stream needs to be transferred to the processor at this time?

A short-term scheduler is an OS subsystem that, if necessary, interrupts an active thread and selects from the queue of ready-made threads the one to be executed. The highest demands are placed on his productivity, because he gets control very often.

There is also a dispatcher, which directly transmits control to the selected stream (switches the context).

The format of the queue of finished flows depends on the implementation of short-term planning. This queue can be organized according to the FIFO (First In, First Out) principle, be a queue with priorities, a tree or an unordered linked list.

Planning levels:

1. Task planning is used as long-term process planning. It is responsible for generating new processes in the system, determining its degree of multiprogramming, i.e. the number of processes that are at the same time in it.

2. CPU usage planning is used as short-term process planning. It is carried out, for example, when the process is running, to the I/O devices or simply after a certain time interval.

Planning criteria:

1. Fairness – to guarantee each task or process a certain amount of time using the processor in a computer system, trying to prevent a situation where the process of one user is constantly occupying the processor, while the process of another user has not actually started.

2. Efficiency – try to occupy the processor for 100% of working time, not allowing it to stand waiting for processes ready to run. In real computer systems, CPU usage ranges from 40 to 90%.

3. Reducing the total execution time (turnaround time) – to ensure the minimum time between the start of the process or setting the task in the queue for download and its completion.

4. Reducing the waiting time – to reduce the time spent in the state of readiness and tasks in the queue for download.

5. Reduction response time – minimize the time required by the process in interactive systems to respond to user requests.

There are two main planning strategies:

1. Non-displacement multitasking algorithm (non-priority) – threads can run indefinitely and cannot be interrupted by the OS. The threads themselves transfer control to the system when they shut down or go into standby mode (can be used in real-time systems, batch processing systems).

2. Displacement multitasking algorithm (priority) – threads can be interrupted by the OS scheduler to transfer control to other threads. The process works only for the allotted period of time, after which it is stopped on a timer to transfer control to the scheduler (clearly must be implemented in interactive systems).

Requirements for planning algorithms:

1. Were predictable. The same task should be performed in about the same time. The application of the planning algorithm should not lead, for example, to the extraction of the square root of four per hundredth of a second at one start and a few days – at the second start.

2. Were associated with minimal overhead. If for every 100 milliseconds allocated to the process to use the processor, you will have 200 milliseconds to determine which process the processor will have at its disposal, and to switch the context, then such an algorithm is probably not worth applying.

3. Evenly load the resources of the computer system, preferring those processes that will take up little-used resources.

4. Were scalable, i.e. did not immediately lose performance with increasing load. For example, doubling the number of processes in a system should not increase the total execution time of processes by an order of magnitude.

All scheduling parameters can be divided into two major groups: static parameters and dynamic parameters.

Static parameters do not change during the operation of the computer system, dynamic parameters on the contrary, subject to constant change.

The static parameters of a computer system include the limit values of its resources (the amount of RAM, the maximum amount of memory on the disk for swapping, the number of connected I/O devices, etc.). Dynamic system parameters describe the amount of free resources at the moment.

Variable registers, stacks, and local memory areas are called thread contexts. Because this information is different on each hardware platform on which Windows can run, the corresponding data structure is platform-specific. The Windows `GetThreadContext` function provides access to this hardware-dependent information (called the `CONTEXT` block).

Although threads have their own execution context, each thread within one process shares (as well as other process-related resources). This means that all threads in the process can write and read the contents of the memory of any of the threads of this process.

In addition to a closed address space and one or more threads, each process has security identification and a list of open descriptors of objects such as files and partitions of shared memory, or synchronizing objects such as mutexes, events and semaphores.

Each process has a security context that is stored in the object – the access token. The access token contains the identification of the security and defines the authority of this process. By default, the stream does not have its own access token, but it can get it, and this will allow it to replace the security context of another process (including running on a remote Windows system).

Virtual address descriptors (VADs) are data structures used by the memory manager to account for virtual addresses involved in a process.

Windows provides an extension for the process model – tasks. They are designed mainly so that groups of processes can be operated and managed as a whole.

The task object allows you to set certain attributes and impose restrictions on the process or processes associated with the task. This object also stores information about all the processes that were mapped to the task, but have so far been completed.

In some respects, the task object compensates for the lack of a hierarchical process tree in Windows, and in some respects, it even exceeds the UNIX process tree in its capabilities.

Windows implements a virtual memory system based on a linear address space. It creates for each process the illusion that it has its own large and closed address space. Virtual memory gives a logical representation, not necessarily corresponding to the structure of physical memory.

At runtime, the memory manager, using hardware support, translates, or designs, virtual addresses to physical ones that actually store data. By managing the design and protection of memory pages, the OS ensures that no process will interfere with another and will not be able to damage the data of the OS itself.

Because most computers have much less physical memory than the total amount of virtual memory involved in running processes, Memory Manager moves or puts pages, part of the contents of memory to disk.

Pumping data to disk releases physical memory for other processes or the OS itself. When a thread accesses a virtual memory page reset on disk, Virtual Memory Manager loads this information from disk back into memory.

No additional code is required to take advantage of paging in applications, as the memory manager relies on the hardware support of this mechanism.

The size of the virtual address space depends on the specific hardware platform. On 32-bit x86 systems, the theoretical maximum for the total virtual address space is 4 GB.

By default, Windows allocates the lower half of this space (in the address range from 0x00000000 to 0x7FFFFFFF) to processes, and uses the other half (in the address range from 0x80000000 to 0xFFFFFFFF) for its own purposes.

Windows 2000 Advanced Server, Windows 2000 Datacenter Server, Windows XP (SP2 and above), and Windows Server 2003 support the /3GB and /USERVA boot options specified in the Boot.ini file.

In this regard, 32-bit Windows has a mechanism of Address Windowing Extension (AWE), which allows a 32-bit application to allocate up to 64 GB of physical memory, and then project views, or windows, on its 2 GB virtual address space.

The application of AWE complicates the management of projections of virtual memory on the physical, but removes the problem of direct access to the amount of physical memory that exceeds the limits of the 32-bit address space of the process.

To prevent applications from accessing critical OS data and eliminate the risk of modifying them, Windows uses two CPU access modes, even if it supports more than two modes:

- intended for the user (user mode);
- kernel mode.

Application code runs in user mode, while OS code (such as system services and device drivers) runs in kernel mode. In kernel mode, all system memory is accessed and any machine CPU commands are allowed.

Although each Windows process has its own (closed) memory, OS code and kernel mode device drivers share a single virtual address space.

Each page in the virtual memory is indicated by a tag that determines in which mode the CPU should work to read and/or write this page.

Pages in the system space are only available in kernel mode, and all pages in the custom address space are in custom mode.

Read-only pages (for example, containing only executable code) are not writable in any mode.

Be careful when downloading the device driver from a third-party supplier: by switching to kernel mode, it will have full access to all OS data. This

vulnerability is one of the reasons why Windows has introduced a driver verification mechanism. The Driver Verifier mechanism helps device driver developers find errors in them (for example, memory leaks or buffer overflows).

Applications can switch from custom mode to kernel mode by accessing the system service. To switch from user mode to kernel mode is performed by a special CPU command.

Terminal Services in Windows provide support for multiple interactive user sessions on a single system. With Terminal Services, a remote user can set up a session on another machine, log on to it, and run applications on the server. The server provides the client with GUI, and the client returns the server for user input.

The first login session on the physical console of the computer is considered a console session, or session zero. Additional sessions can be created using the Remote Desktop Connection program (Rdpfc.exe), in Windows XP – through the mechanism of fast user switching.

Windows 2000 Server and Windows Server 2003 support two simultaneous remote sessions. Windows 2000 Advanced Server, Datacenter Server, and all editions of Windows Server 2003 and later are capable of supporting more than two sessions at once, provided that the system is properly licensed and configured as a terminal server.

For applications that need to know if they are running in a terminal server session, there is a set of Windows API functions that allow you to programmatically recognize this situation and control various aspects of terminal services.

In Windows, an object is a single instance of a run-time instance of a statically specific object type. An object type consists of a system-wide data type, functions that operate on instances of that data type, and a set of attributes.

An object attribute is a data field in an object that partially determines the state of that object. For example, a process object has attributes that include a process ID, a base priority, and a pointer to an access token object. Object methods (means for manipulating objects) usually read or modify any attributes.

The facilities are very convenient to support four important OS functions:

- assigning clear names of system resources;
- division of resources and data between processes;
- protection of resources from unauthorized access;
- link accounting (this allows the system to know when an object is no longer in use and automatically destroys it).

Not all data structures in Windows are objects. Objects contain only those data that need to be divided, protected, named or made available by the programs of the user mode (through system services).

Windows was originally developed as a secure system that meets the requirements of various government and industry security standards, such as

the Common Criteria for Information Technology Security Evaluation (CCITSE) specifications.

The basic security features in Windows are:

- selective protection of any shared system objects (files, directories, processes, threads, etc.);
- security audit (to account for users and the transactions initiated by them);
- password authentication when logging in and preventing one user from accessing uninitialized resources released by another user.

Windows supports two types of access control:

1. Discretionary access control is a mechanism that most users associate with protection. This is a method in which owners of objects (such as files or printers) allow access to them to other users. Upon login, the user receives a set of security credentials, or security context. When it attempts to access an object, its security context is checked against the access control list (ACL) for that object to determine if it has permission to perform the requested operation.

2. Privileged access control is necessary in cases where electoral access control is insufficient. This method ensures that the user can access protected objects even if their owner is not available. For example, if an employee quits a company, the administrator needs to access files that could only be accessed by the former employee. In such cases, Windows allows the administrator to become the owner of these files and, if necessary, manage access rights to them.

Protection permeates the entire Windows API. The Windows subsystem implements object-based protection in the same way as the OS itself. The first time an application tries to access a shared object, the Windows subsystem checks to see if the application has the appropriate rights. If the scan completes successfully, the Windows subsystem allows the application access.

The Windows subsystem implements protection for common objects, some of which are based on native Windows objects. Windows objects include desktop objects, menus, windows, files, processes, threads, and a number of synchronizing objects.

Windows differs from most other OS in that it uses Unicode as an internal format for storing and processing text strings.

Unicode is a standard encoding that supports many world-famous character sets and in which each character is represented by 16-bit (double-byte) code.

Because many applications deal with 8-bit (single-byte) ANSI characters, Windows functions that take string parameters exist in two versions: for Unicode and for ANSI.

When Windows uses an outdated service or code fragment written in ANSI strings, this OS will be forced to convert ANSI characters to Unicode. However, Windows never converts data inside files – only the applications themselves decide which encoding to store textual information in the files.

Starting with Windows 2000, all language editions contain the same Windows features. The only Windows binary code base for all countries is able to support many languages by simply adding the necessary language support components.

System failure, in all cases caused by the KeBugCheckEx function described in the Windows Driver Kit (WDK). This function accepts a stop code, or bugcheck code, and four parameters that are interpreted depending on this code.

KeBugCheckEx masks all interrupts on all system processors, then switches the video adapter to low-resolution VGA graphics mode (which is supported by all Windows-compatible graphics cards) and displays the stop code on a blue screen, accompanied by recommendations for further user action.

The first line of the "Technical information" section in the blue screen example shows the stop code and four additional parameters that are passed to the KeBugCheckEx function.

The line at the top of the screen is the text equivalent of the numeric stop code ID. In this example, stop code 0x00000D1 corresponds to error DRIVER_IRQL_NOT_LESS_OR_EQUAL.

If the parameter contains the address of a part of the OS or device driver code, Windows displays the base address of the module in which the error occurred, the date stamp, and the file name of the device driver.

There are more than 300 unique stop codes, but most of them are used extremely rare or never at all.

Most crashes in Windows are accompanied by several typical stop codes.

In addition, the stop code determines the content of the four additional parameters (not all stop codes provide extended information transmitted through these parameters).

However, analysis of the stop code and parameter values can help diagnose a malfunctioning component.

Safe Mode is a boot configuration that consists of a minimal set of device and service drivers. By using only those drivers and services that you can't download without, Windows avoids other drivers that can cause a system crash.

There are usually three options:

- 1) Safe Mode;
- 2) Safe Mode with Networking;
- 3) Safe Mode with Command Prompt.

4) Directory Services Restore. Used to boot the system with the Active Directory Domain Controller service disabled and without opening its database. In this mode, all drivers and services are downloaded, except Active Directory. It is designed to troubleshoot when you cannot log on because of Active Directory corruption.

When booting in safe mode, the loader (Winload) passes the kernel (Ntoskrnl.exe) command line parameter along with other parameters specified in the BCD for the current boot.

If the download is in safe mode, Winload sets the binary-coded decimal (BCD) parameter to safeboot, the value of which corresponds to the type of mode you selected. For standard safe mode, this is minimal, and for network-enabled mode, it is network. If you want to boot in safe mode with command line support, Winload adds a safebootalternateshell parameter to the minimal parameter. When booting in directory service recovery mode, the dsrepair parameter is used.

When the system boots in safe mode, Winload passes to the kernel, along with the parameters required by this mode, the string specified by the bootlog parameter. During the initialization process, the kernel checks for the bootlog parameter, regardless of whether safe mode is set.

For example, if the IopSafebootDriverLoad function prevents the I/O manager from downloading any driver, the manager calls the IopBootLog function to fix this fact.

Similarly, after the IopLoadDriver function successfully loads the driver included in the safe mode configuration, it calls the IopBootLog function to enter the appropriate log entry.

If the failing driver is in the Safe group, you will not be able to boot in safe mode.

The Windows Recovery Environment (WinRE) allows you to deal with these problems. It provides a range of tools and automated recovery technologies to solve the most common boot problems.

Startup Repair. An automated tool that recognizes the most common Windows startup problems and automatically tries to solve them.

System Restore. Allows you to return to the restore point in cases where Windows cannot boot even in safe mode.

System Image Recover. In previous versions of Windows, this tool was similar to Complete PC Restore and Automated System Recovery (ASR).

Windows Memory Diagnostic Tool. Analyzes computer memory for hardware problems.

Command Prompt. In cases where manual intervention is required to solve problems, you can use the command line as a Windows shell, which allows you to run almost any program if you have the necessary dependencies.

Laboratory work № 6

PROCESS AND FLOW MANAGEMENT IN WINDOWS USING THE POWERSHELL

The purpose of the work is to consolidate knowledge and acquire process and flow management skills in Windows using the PowerShell.

Problem statement:

1. The student must know the programming commands in Windows using the PowerShell; must know the theoretical part necessary for work.
2. Required workstation with the Windows family operating system installed.

Execution of work:

1. Using the Windows + R hotkeys, call the execution window and execute the PowerShell command. After pressing the Enter key, the command editing window will open. Enter the first command (Figure 19):

```
PS C:\Users\Administrator> start-process PowerShell
```

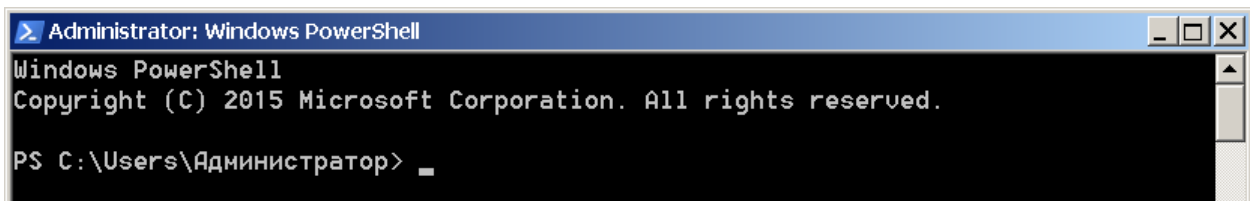


Figure 19 – Command demo window «Start-process PowerShell»

2. Enter in the terminal window, the tasklist command line will list the entire workflow on this computer. All data will be displayed in tabular form, but you can change the data type to another format. Use the /fo option and everything will be displayed as a list (or Comma-Separated Values (CSV)), and if you use the /v option, this list will include more detailed information about each process (Figure 20).

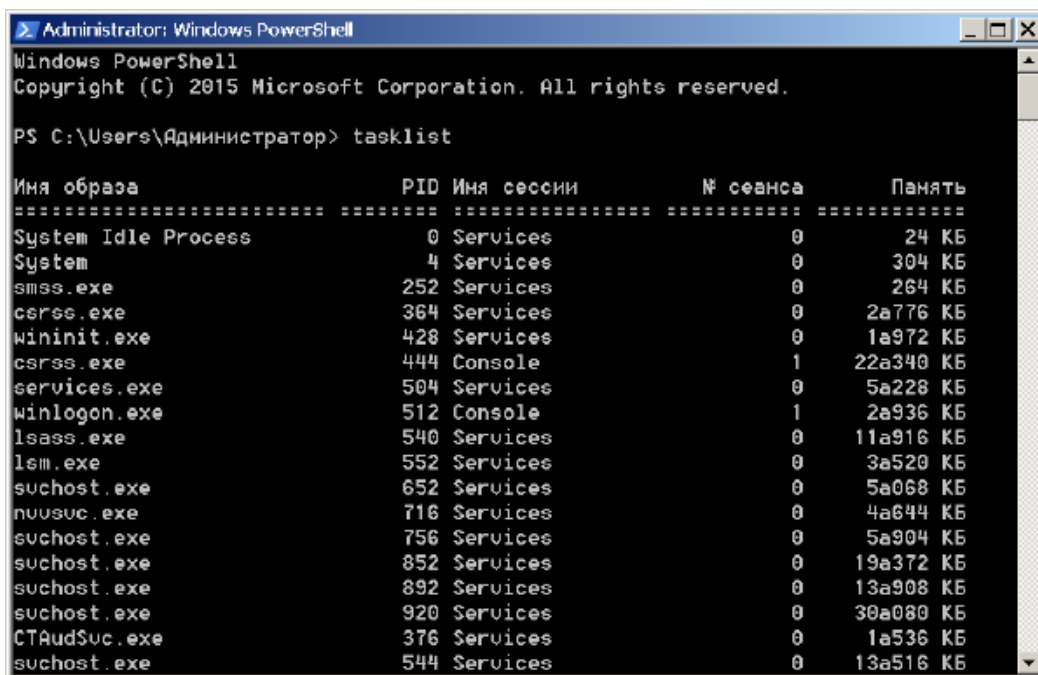


Figure 20 – Tasklist command demonstration window

3. Enter the `tasklist /v /fo list` (without quotes) and approximately the following picture should appear in the window (Figure 21).

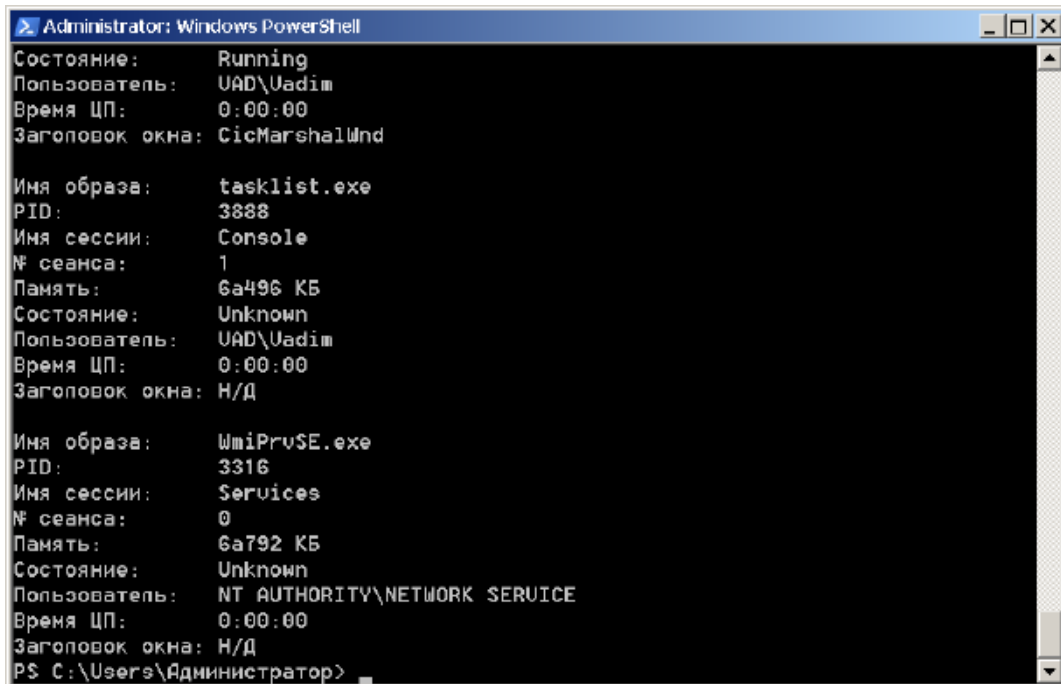


Figure 21 – Demo window of the "tasklist /v /fo list" command

4. Consider the command "taskkill". To terminate the process, you can use its name or PID, and you can terminate the process using different filters (Figure 22).

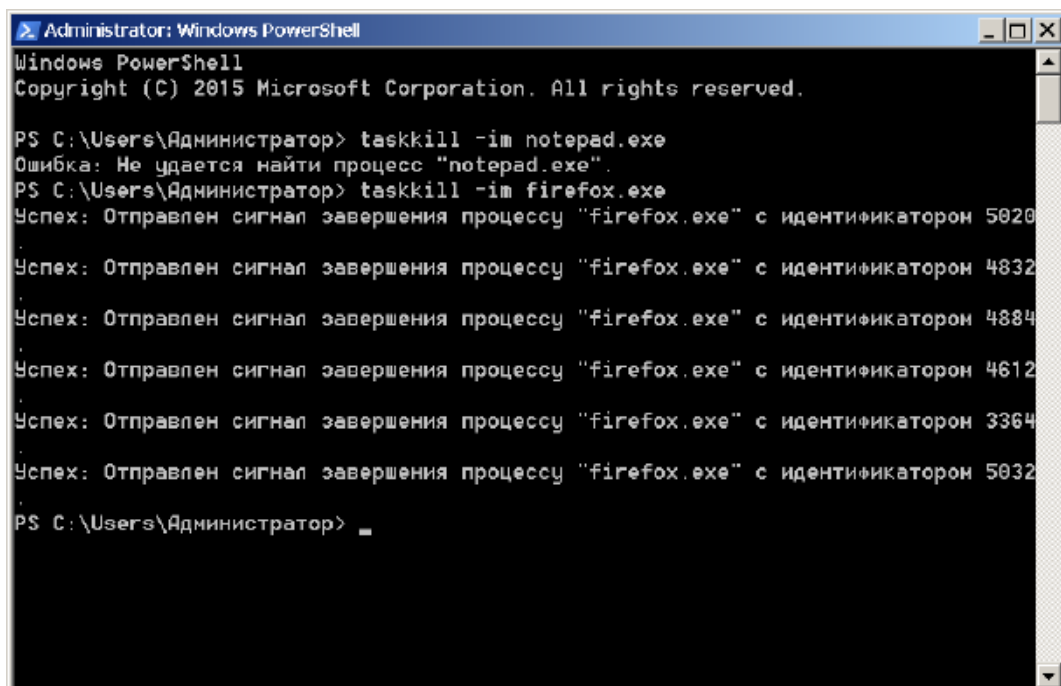


Figure 22 – Taskkill command demonstration window

5. Use the /f switch to force the process, and if you add the /t switch, all others that were started because of it will be terminated. To learn more about the tasklist and taskkill commands, add the /? switch to them. It displays complete help for these utilities. Let's not forget about such a powerful tool as PowerShell. We can use it directly in the same command prompt window. To view the entire list of processes, run the Get-Process command (Figure 23).

Handles	NPM(K)	PM(K)	WS(K)	UM(M)	CPU(s)	Id	ProcessName
79	8	1412	1924	31	0,02	2740	alg
58	7	1416	4720	55	0,25	944	conhost
610	12	2156	2768	47	0,42	364	csrss
510	16	25208	22076	132	23,42	444	csrss
85	9	1272	1536	47	0,03	376	CTAudSvc
114	10	3264	8112	59	0,03	3004	DiscSoftBusService
88	8	1984	4940	69	0,13	3832	dwm
761	68	39904	57840	332	9,67	3852	explorer
325	39	65032	105816	1465	1,34	1016	firefox
889	79	200060	261700	1781	9,06	2904	firefox
327	33	52656	71936	1418	0,55	4104	firefox
309	30	44076	52912	1401	0,27	4304	firefox
101	9	1496	528	46	0,00	3040	GoogleCrashHandler
93	8	1520	528	43	0,00	1596	GoogleCrashHandler64
126	20	25600	9232	69	3,92	1692	hasplms
68	7	1560	2460	49	0,06	3148	HP1006MC
0	0	0	24	0		0	Idle
65	10	1076	980	29	0,00	1832	KMSSSS
866	30	8228	11936	50	2,25	540	lsass
214	10	2780	3496	33	0,13	552	lsm
87	11	3072	7692	79	0,08	4728	MathType

Figure 23 – Get-Process command demonstration window

6. To finish viewing the list of processes, type the command (Figure 24):

```
PS C:\Users\Administrator> get-process firefox | stop process
```

Handles	NPM(K)	PM(K)	WS(K)	UM(M)	CPU(s)	Id	ProcessName
1319	85	22256	30072	163	2,41	920	svchost
729	40	13792	12732	122	2,45	1092	svchost
44	4	936	1052	13	0,02	1252	svchost
311	34	9492	7304	73	1,09	1304	svchost
433	36	9416	98900	221	2,72	1664	svchost
101	13	1932	2800	33	0,02	2028	svchost
139	10	106760	101212	179	55,08	2256	svchost
356	26	10588	14512	107	0,77	2944	svchost
886	0	108	304	3	92,94	4	System
170	16	3128	4608	68	0,17	3628	taskhost
407	45	20372	31152	197	1,91	3708	TeamViewer
395	31	7944	10412	92	1,27	2296	TeamViewer_Service

Figure 24 – Demonstration window completes the list of processes

7. Next, learn a list of all the properties of the PowerShell process. This can be done with the following command (Figure 25): `Get-Process -Name powershell | Get-Member -MemberType property`

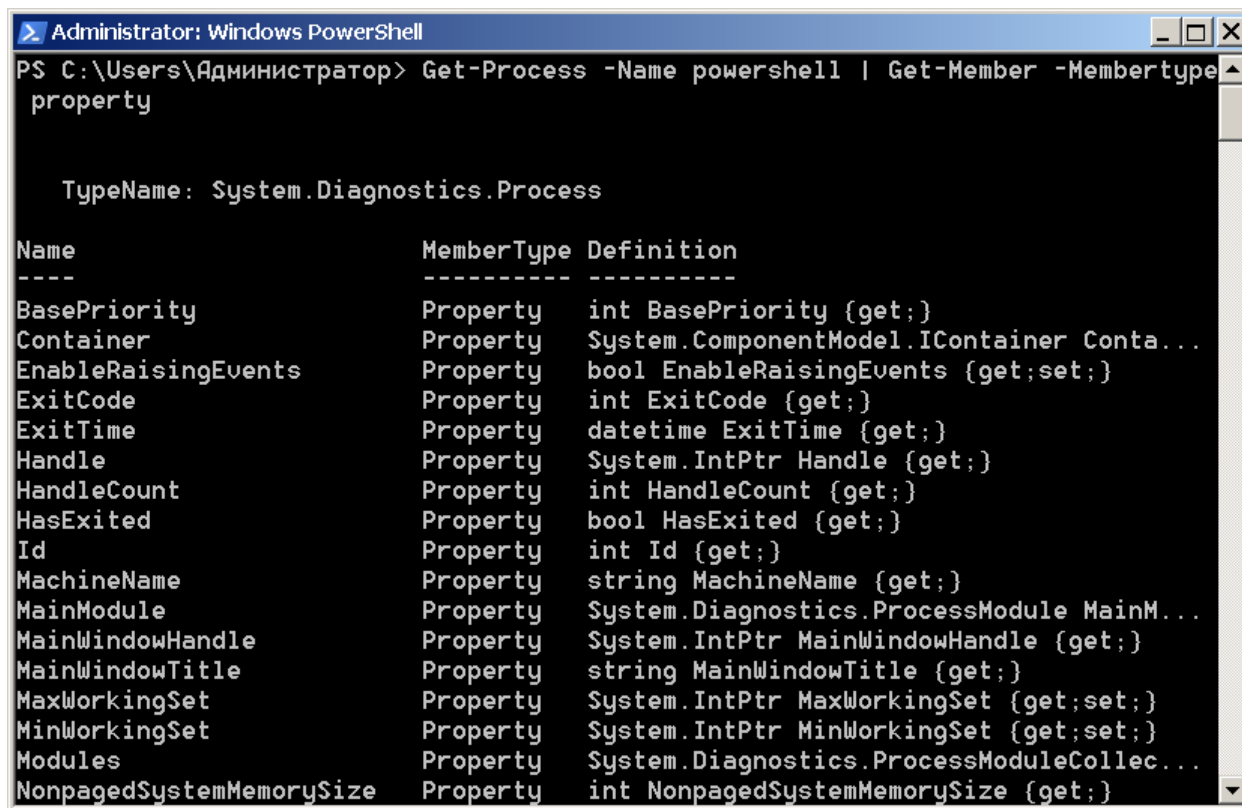


Figure 25 – Demonstration window displaying a list of all process properties

8. Let's show only the necessary properties. For example, let's leave only the name, process ID, path to the executable file, plug-ins and the time when the process was started. All this we list, using the following command: `Get-Process -Name powershell | Format-List name, id, path, modules, starttime`. Thanks to this, we learn about who started the process, how much it loads the system, where its file is located and a bunch of different, useful and not very information (Figure 26).

9. To stop the process via PowerShell, use a special `Stop-Process` command. Use your name or ID to specify the process you want to complete. For example, it is possible to stop work of firefox (Figure 27):

```
Get-Process | where {$_.name -match "firefox"} | Stop process
```

10. In order to highlight in yellow the processes that take up more than 300 MB of RAM, and all other processes in green, use the following code (Figure 28):

```

PS C:\Users\Administrator> get-process | foreach {if
($_.vm -gt 300000000) {write-host -foregroundcolor
yellow $_} else
{write-host -foregroundcolor green $_}}

```

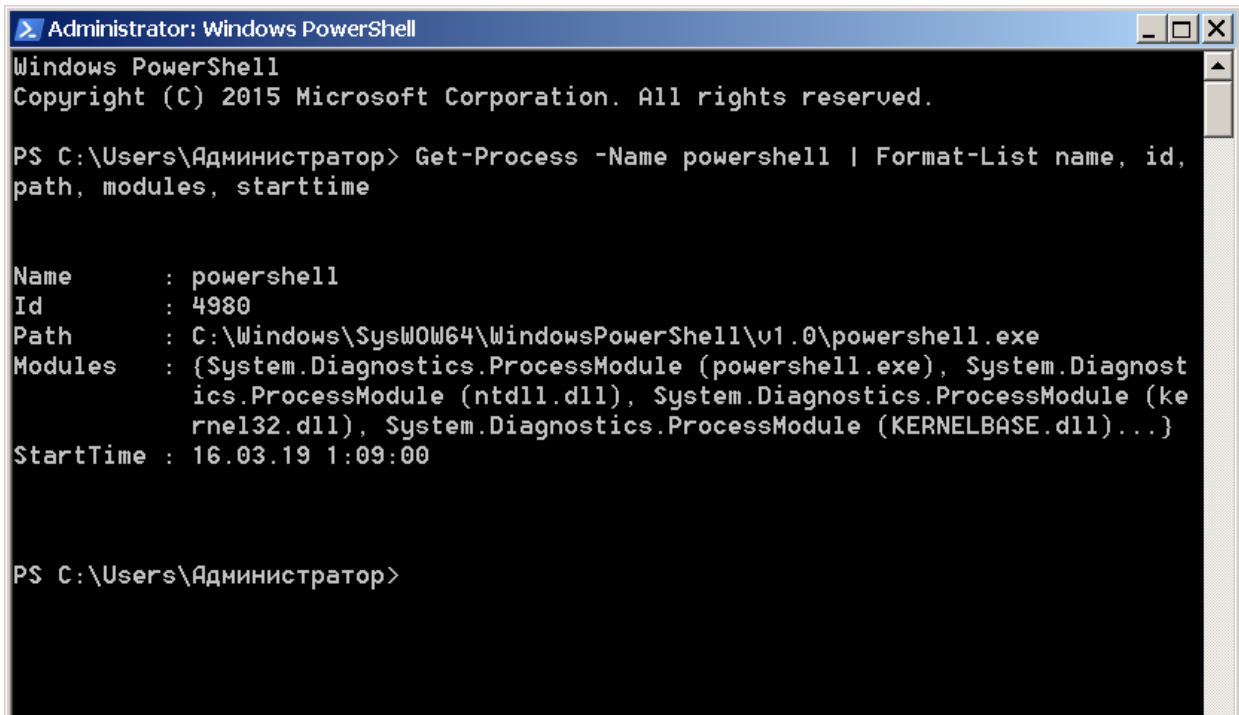


Figure 26 – Demonstration window of the property output command

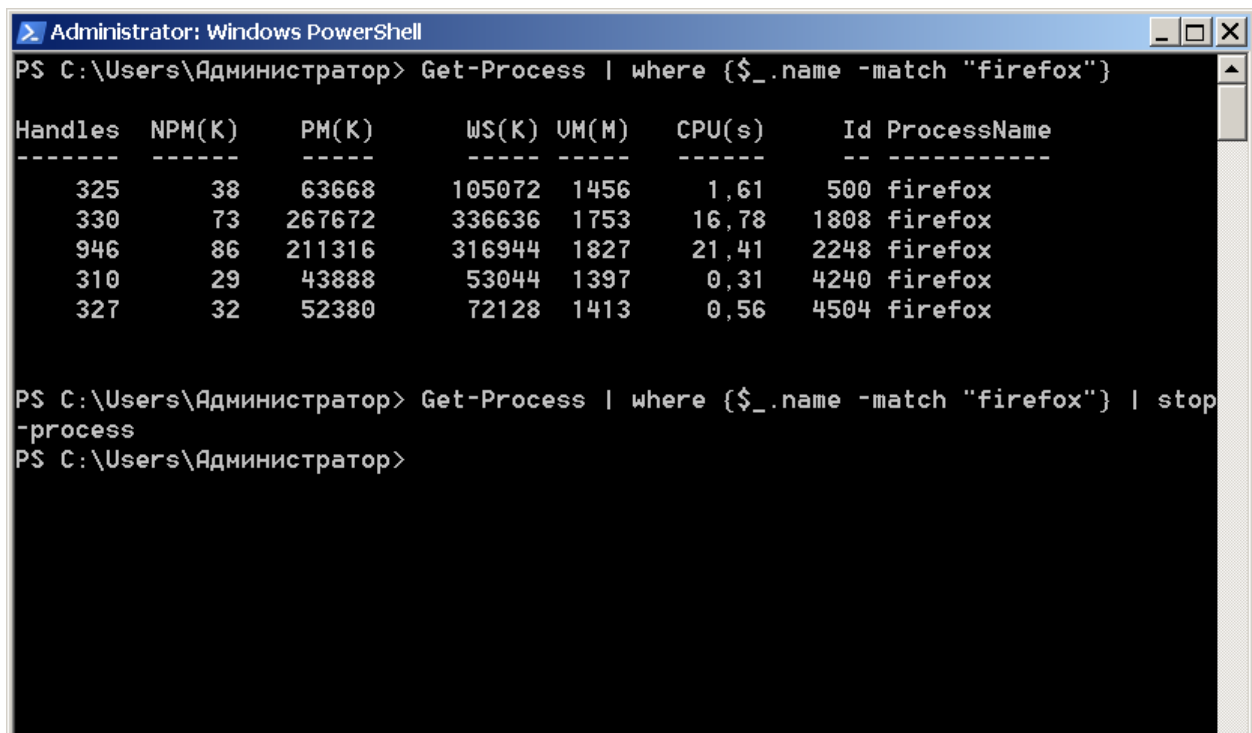
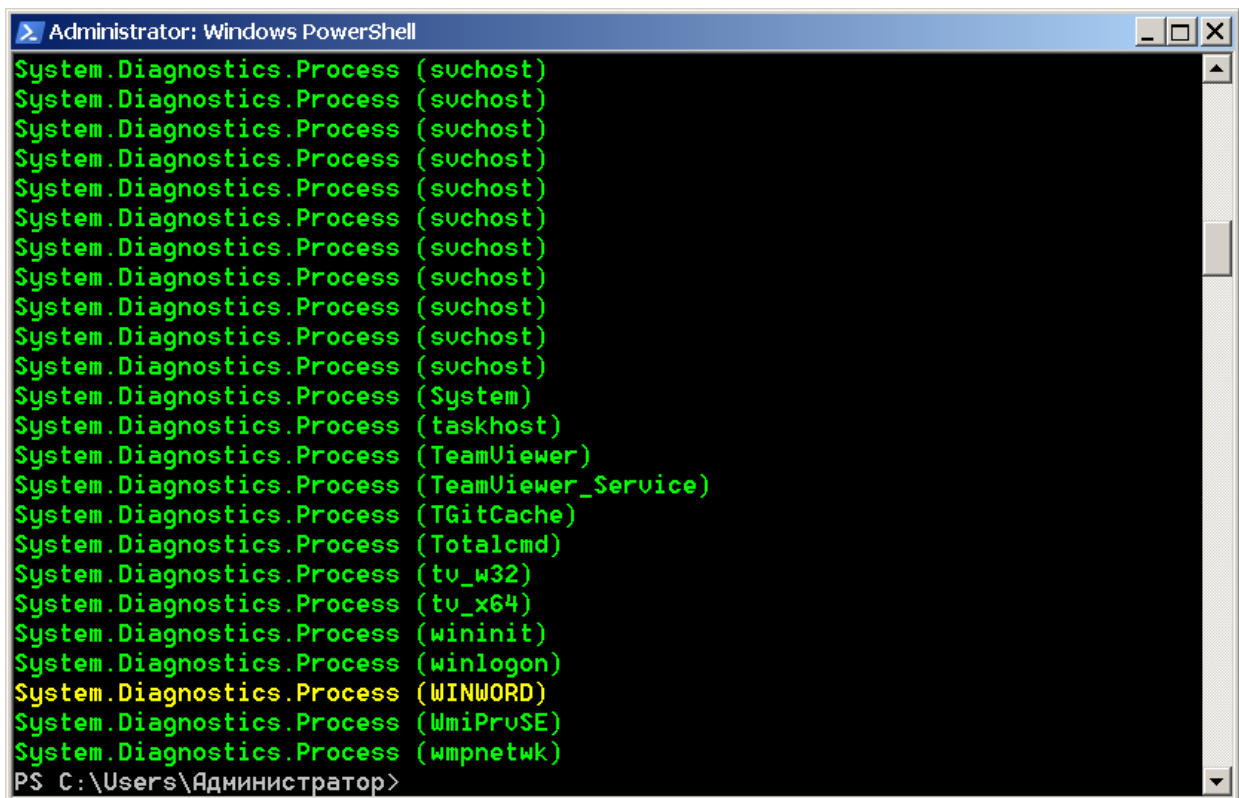


Figure 27 – Demonstration window of the process stop command



```
Administrator: Windows PowerShell
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (suchost)
System.Diagnostics.Process (System)
System.Diagnostics.Process (taskhost)
System.Diagnostics.Process (TeamViewer)
System.Diagnostics.Process (TeamViewer_Service)
System.Diagnostics.Process (TGitCache)
System.Diagnostics.Process (Totalcmd)
System.Diagnostics.Process (tu_w32)
System.Diagnostics.Process (tu_x64)
System.Diagnostics.Process (wininit)
System.Diagnostics.Process (winlogon)
System.Diagnostics.Process (WINWORD)
System.Diagnostics.Process (WmiPrvSE)
System.Diagnostics.Process (wmpnetwk)
PS C:\Users\Администратор>
```

Figure 28 – Yellow color demonstration window of the process

11. The following example shows the names of processes with the amount of memory:

```
PS C:\Users\Administrator> get-process | select-object name, vm | foreach-object {if ($_.vm -gt 300000000) {write-host -foregroundcolor yellow $_} else {write-host -foregroundcolor green $_}}
```

12. Write a report on the work. Contents of the report on laboratory work: topic; goal; setting objectives; the order of execution; research results; conclusion. Copies of screens are provided with the report in accordance with the order of research.

Additional information: There are many different utilities and applications for managing Windows processes. Typically, the command line is rarely used for these purposes. But in some cases, this is the only option and other means are not available. This can happen when you block standard dispatchers with various viruses and other bad programs. Therefore, it will be very useful to know and be able to work with processes through the PowerShell command line. To manage processes using command-line capabilities, the system has two special programs: tasklist and taskkill. From the name you can guess their purpose – the first can display a list of all running processes on this computer or on a remote machine, and the second can stop them.

7 WINDOWS OPERATING SYSTEM REGISTER

The registry plays a key role in configuring and managing Windows systems. It is a repository for both system-wide settings and settings for each user. Although many consider the registry to be some kind of static data stored on the hard drive, the registry is also a window into various memory structures that are maintained by the Windows executable system and kernel.

These configurations are read in the following four main cases:

1. During the boot process, the boot loader reads the configuration data and the list of bootable device drivers to boot into memory before initializing the kernel. Because the bootable configuration database – Boot Configuration Database (BCD) is actually stored in the registry bush.

2. During the kernel boot process, the kernel reads the settings that determine which device drivers need to be booted and how the various system elements, such as memory manager and process manager, are configured by themselves, while configuring system behavior.

3. As you log on, Explorer and other Windows components read a user's preferences, including network drive names, desktop wallpapers, screensavers, menu behavior, icon placement, and, more importantly, which programs to run and before which files have been accessed recently.

4. During its launch, the application reads system-wide settings, such as a list of optional components and licensing data, as well as individual user settings, which may include menu and toolbar layouts, and a list of documents it has recently accessed.

Most often, changes to the register are made in the following cases:

- although this is not considered a change, the initial structure of the registry and many default installations are determined by the prototype version of the registry that comes on the media used to install Windows and copied to the new installation;

- application installation utilities create default program settings and those settings that display the configuration settings selected during installation;

- during the installation of the device driver, the PnP device maintenance system creates registry settings that tell the I/O Manager how to run the driver, and creates other settings that determine the configuration of the driver's operations;

- when changing the settings of a program or system through interfaces, these changes are often stored in the registry.

The registry is a database with a structure similar to the structure of a disk volume. The registry contains partitions, similar to disk directories, and settings that can be compared to files on disk. A partition is a container that consists of other partitions (subsections) or parameters, the parameters store data.

Both the partitions and the parameters borrowed the agreement on their names from the file system. Therefore, you can uniquely identify the mark

parameter stored in the section by using the mark entry. The only exception to this naming scheme is the anonymous parameter of each section. The Regedit registry editor shows anonymous settings labeled (Default).

The root partition of HKEY_CURRENT_USER (HKCU) contains data related to personal settings and software configuration locally logged in to the user's system. It indicates the user profile of the currently logged on user located on the hard disk in the file \Users\\Ntuser.dat. When it load a user profile (for example, when it log on or when the service process starts in the context of a specific user name), a HKCU partition is created to display on the user partition in the HKEY_USERS (HKU) section.

The HKU partition contains a subsection for each downloaded user profile and uses a database of classes registered in the system.

It also contains the HKU\DEFAULT section associated with the system profile (used by processes running under the local system account.

This profile is used by Winlogon, for example, so that changes to the desktop background settings set in this profile are applied to the login screen. When a user logs in for the first time and their account is independent of the domain profile being moved (the user profile is taken from the central network location towards the domain controller), the system creates a profile for his account based on the saved profile in the %SystemDrive%\Users\Default directory.

The location of HKEY_USERS where the system stores profiles is determined by the HKLM\Software\Microsoft\WindowsNT\CurrentVersion\registry setting ProfileList\ProfilesDirectory, which defaults to %SystemDrive%\Users. The ProfileList section also stores a list of profiles available on the system.

The information for each profile is located in the section whose name reflects the security identifier – security identifier (SID) – the account that corresponds to the profile. The data stored in the profile partition includes the time the profile was last loaded, the binary representation of the account SID in the SID parameter, and the path to the profile bush on disk in the ProfileImagePath directory.

The HKEY_CLASSES_ROOT (HKCR) section consists of three types of information:

- associations with file extensions;
- registration of COM-classes;
- virtualize the root part of the system registry to manage user accounts – User Account Control (UAC).

Each registered file name extension has its own partition. Most partitions contain a parameter of type REG_SZ, which points to another partition in HKCR that contains information related to the file class that represents this extension.

For example, HKCR\.xls will point to information about Microsoft Office Excel files in a section such as HKCU\.xls\Excel.Sheet.8. Other sections

provide configuration details for COM objects registered in the system. Virtualize the UAC registry is located in the VirtualStore section, which has nothing to do with other types of data stored in HKCR.

The data stored in the HKCR section comes from two sources:

- from the registration data of classes for an individual user, located in the section HKCU\SOFTWARE\Classes (which is displayed on the file on the hard disk \Users\\AppData\Local\Microsoft\Windows\Usrclass.dat);
- from system-wide class registration data in the HKLM\SOFTWARE\Classes section. The reason for separating the credentials that apply to each user from the credentials that apply to the entire system is that these settings may be contained in the roaming profiles. This also closes a hole in the security system: an unprivileged user cannot modify or delete partitions in the system-wide version of HKCR and cannot affect the functionality of such programs in the system.

HKEY_LOCAL_MACHINE (HKLM) is the root partition, which contains system-wide configuration units: BCD00000000, COMPONENTS (loaded in dynamic mode as needed), HARDWARE, SAM, SECURITY, SOFTWARE and SYSTEM.

The HKLM\BCD00000000 section contains information from the Boot Configuration Database (BCD). This database replaced the Boot.ini file that was used before the release of Windows Vista, and added more flexibility and isolation to the boot configuration data for a single installation. Each entry in the BCD, such as Windows installation or command line configuration for installation, is stored in Objects.

Most of these simple elements appear in the BCD directory in the MSDN library. These items define various command line settings or boot options. The parameter associated with each subdivision of the element corresponds to the parameter for the corresponding command line key or boot parameter.

The BCDEdit command-line utility allows you to modify the BCD using symbolic names for elements and objects. It also provides comprehensive help on available download settings, but unfortunately this utility only works locally. Because the registry can be opened remotely as well as imported from the bush file, you can modify or read the BCD of the remote computer using Registry Editor.

The HKLM\COMPONENTS section contains information related to the Component Based Servicing (CBS) stack.

This stack contains various files and resources that are part of the Windows installation image (used by the Automated Installation Kit) or the OEM Preinstallation Kit.

The CBS maintenance APIs use the information in this section to identify installed components and obtain configuration information.

HKLM\HARDWARE stores descriptions of system-inherited hardware and some hardware mapping to drivers. On modern systems, it is likely to find only some peripherals, such as keyboard, mouse and ACPI BIOS data.

The Device Manager tool (launched by clicking Device Manager in the System window of the Control Panel) allows you to view information about the hardware registered in the registry by simply reading the parameters in the HARDWARE section (tree HKLM\SYSTEM\CurrentControlSet\Enum).

HKLM\SAM stores information about local accounts and groups, such as passwords, group definitions, and domain connections.

Windows Server systems that act as domain controllers store domain accounts and groups in Active Directory, a database that stores domain-wide settings and information.

By default, the security descriptor in the SAM partition is configured in such a way that it cannot be accessed even using an administrator account.

HKLM\SECURITY maintains a system-wide security policy and assignment of user rights. The HKLM\SAM unit is associated with the SECURITY unit in the HKLM\SECURITY\SAM unit. By default, you cannot view the contents of HKLM\SECURITY or HKLM\SAM, as the security settings of these sections allow access only with a System account. To allow administrators access, you can change the security descriptor or use PsExec to run Regedit with a local system account if you want to look inside.

In HKLM\SOFTWARE, Windows stores system-wide configuration information that is not required to boot the system. In addition, general installations, such as file paths and application directories, licensing and expiration information, are stored here by third-party programs.

The HKLM\SYSTEM section contains system-wide configuration information required to boot the system, such as which device drivers to load and which jobs to run. Because this information is essential for starting the system, Windows also contains a copy of this information in this section under the name of the last successful configuration.

The HKEY_CURRENT_CONFIG section is just a reference to the current hardware profile that was saved in

HKLM\SYSTEM\CurrentControlSet\HardwareProfiles\Current.

Hardware profiles are no longer supported in Windows, but the section still exists to support heritage applications that may depend on their presence.

The registry is a mechanism used in Windows to access performance counter values that apply to either the OS or server applications. One of the concomitant benefits of providing access to performance counters through the registry is that remote performance tracking works "for free" because remote access to the registry is easily provided using the usual API features of the registry.

You can directly access the information of the performance counters in the registry by opening a special section HKEY_PERFORMANCE_DATA and requesting the parameters in it.

When viewing the registry in the registry editor, you will not find this section, it is available only by software, through the registry functions available in Windows, such as RegQueryValueEx.

Performance information is not actually stored in the registry, and registry functions use this section to search for information from performance data providers.

You can also access performance counter information by using the Performance Data Helper (PDH) assistant functions available through the Pdh.dll API.

A registry disk is not an ordinary large file, but is a set of individual files called bushes. Each bush contains a registry tree that has a partition that serves as its root or starting point for the tree. Subdivisions and their parameters are below the root.

The track names of all bushes, except those used for user profiles, are encoded in Configuration Manager. As the configuration manager loads the bushes, including system profiles, it writes the path to each bush in the HKLM\SYSTEM\CurrentControlSet\Control\Hivelist settings, deleting the path when the bush is unloaded.

You may notice that some of these bushes may change and do not have associated files. The system creates these bushes and manages them entirely in memory, so such bushes are temporary. The system creates intermittent bushes with each load.

As an example of a non-permanent bush we can cite HKLM\HARDWARE, which stores information about physical devices and resources allocated to these devices. Resource allocation and identification of installed equipment are performed each time the system boots, so it would be illogical to store this data on disk.

In some cases, the size of the bush is limited. For example, Windows imposes a limit on the size of the HKLM\SYSTEM bush. It comes this way because Winload reads the entire HKLM\SYSTEM bush into physical memory almost immediately after the boot process starts, when partitioning into virtual memory is not yet enabled.

Winload also loads Ntoskrnl and boot device drivers into physical memory, so it must limit the physical memory allocated by HKLM\SYSTEM.

On 32-bit systems, Winload allows the bush to be up to 400 MB in size or half the size of the system's physical memory, whichever is smaller.

A special type of partition, known as symbolic registry references, allows Configuration Manager to associate partitions to organize the registry.

A symbolic link is a partition that redirects the configuration manager to another partition. Thus, the HKLM\SAM partition is a symbolic reference to the partition at the root of the SAM bush.

Symbolic links are created by specifying the RegCreateKey function or the RegCreateKeyEx function of the REG_CREATE_LINK parameter. Inside, the configuration manager will create a REG_LINK parameter called SymbolicLinkValue, which will contain the path to the target partition.

Configuration Manager logically divides the bush into distributed units, called blocks, in a manner similar to how a file system divides a disk into clusters.

The register block size is 4096 bytes (4 KB). When a bush expands with new data, it always expands by increasing the number of blocks. The first block of the bush is called the base block. The basic block includes global information about the bushes, which includes:

- signature regf, which identifies the file as a bush;
- updated serial numbers;
- time stamp, which shows when the last time the recording operation was applied to the bush;
- information on repair or restoration of the register made by Winload;
- version number of the bush format;
- checksum;
- internal file name of the bush file (for example, \Device\HarddiskVolume1\WINDOWS\SYSTEM32\CONFIG\SAM).

The version number of the bush format determines the data format inside the bush. Configuration Manager uses the version 1.3 bush format for all bushes, except System and Software, for compatibility with moving profiles from Windows 2000.

For the System and Software bushes, it uses version 1.5 due to later format optimizations for bulk parameters (more than 1 MB) and search (instead of caching the first four characters of the name, the full name hash is used to reduce conflict situations).

Windows organizes the registry data that the bush stores in containers called cells.

Laboratory work № 7

WINDOWS OPERATING SYSTEM REGISTER

The purpose of the work is to study the Windows OS register.

Problem statement:

1. Explore the capabilities of Windows system utilities designed to manage the registry.
2. Examine the structure of the Windows registry.
3. Learn the system utility REGEDIT. Analyze and compare with alternative tools for working with the Windows registry.
4. Analyze the structure of the Windows registry.
5. Get acquainted with the *.REG file format.
6. Create the *.REG file.
7. Required workstation with the Windows family OS installed.

Execution of work:

1. Backup the registry (Figure 29).

Run the utility REGEDIT. In the window that opens, enter the command regedit and press Enter.

The registry editor starts from Explorer. We create a backup of the registry (Figure 30).

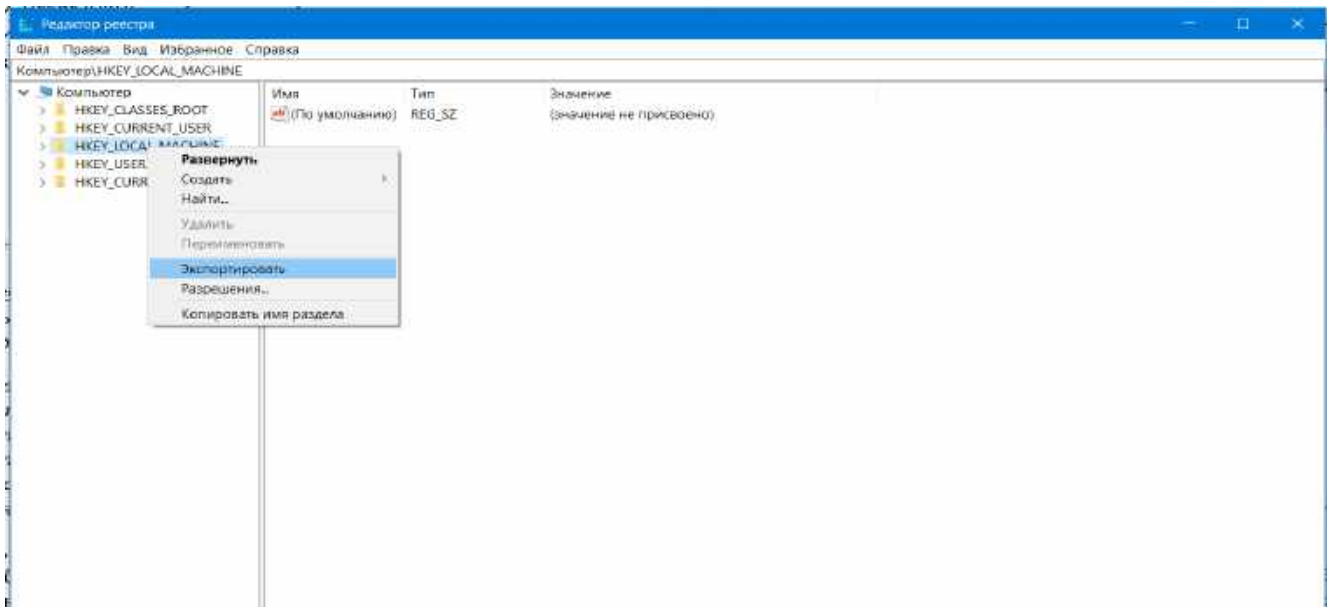


Figure 29 – Backup of the registry

Swap the keys to change the input language from Alt + Shift to Ctrl + Shift on the registry key HKEY_CURRENT_USER\Keyboard Layout\Toggle, as well as the keys Hotkey, Language Hotkey and Layout Hotkey, which have a default value of 1, 1 and 2, respectively.

To change the keyboard shortcut we will need to assign them new values "Hotkey" = "2", "Language Hotkey" = "2", "Layout Hotkey" = "3"

So, to create a reg-file, open a text editor, such as Windows Notepad. Paste the following code into the editor window:

```
Windows Registry Editor Version 5.00

;Switching the language to the left Ctrl + Shift
[HKEY_CURRENT_USER\Keyboard Layout\Toggle]
"Hotkey"="2"
"Language Hotkey"="2"
"Layout Hotkey"="3"
```

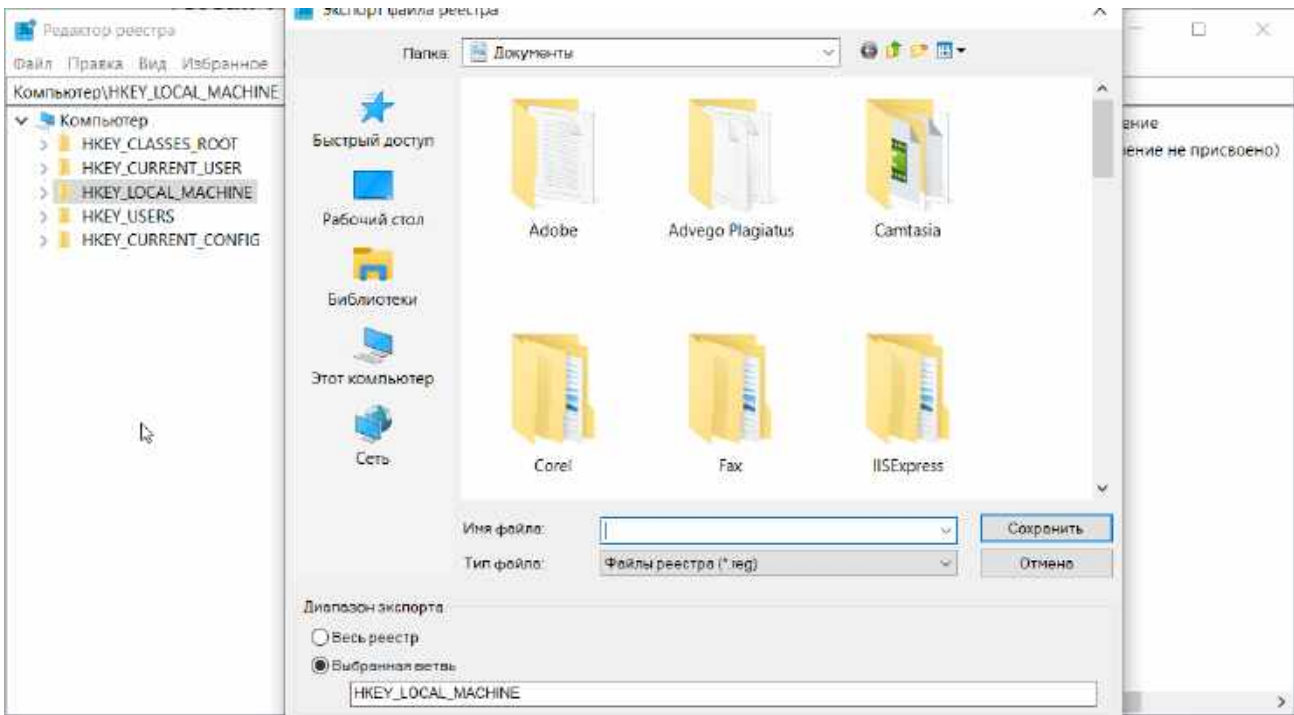


Figure 30 – The registry editor starts from Explorer

In the Windows registry editor in the left part of the window, go to the branch (Figure 31):
 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System

Next, in the EnableLUA key, in the "Value" field, change 1 to 0 (Figure 32).

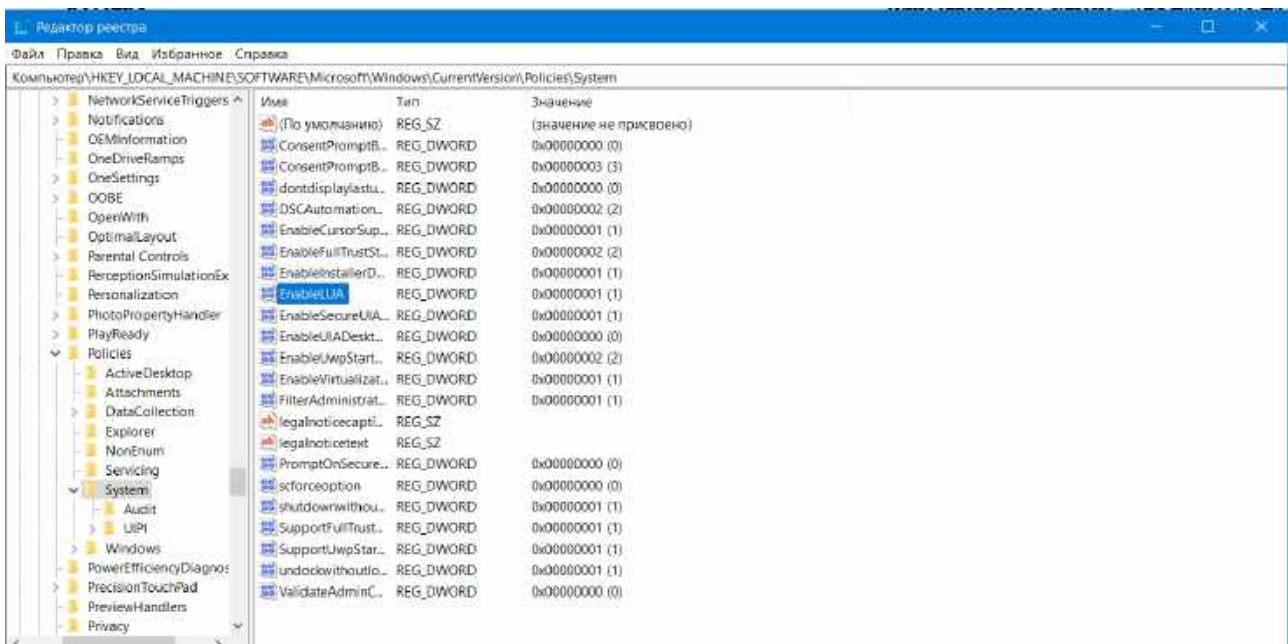


Figure 31 – EnableLUA key

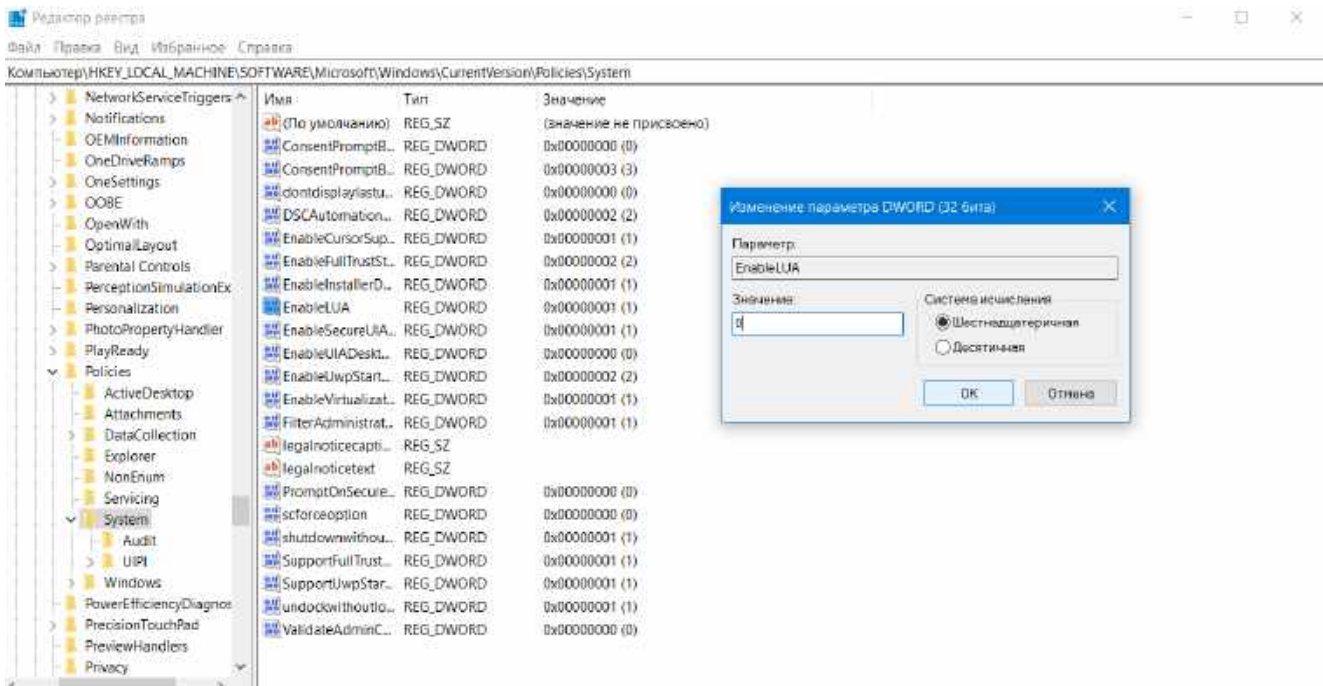


Figure 32 – Change of EnableLUA key

Windows Registry Editor Version 5.00 is a system that allows Windows to determine what type of file was running. Mandatory at the beginning of all reg-files, registered once.

;Switching the language to the left Ctrl + Shift – comment line. Any line of the registry change file that begins with a semicolon that will not be executed by the system is for explanatory entries. Comments are written as many times as you want.

[HKEY_CURRENT_USER\Keyboard Layout\Toggle] is the registry hive where changes will be made. We will note in detail about multiple records a little below.

"Hotkey" = "2", "Language Hotkey" = "2", "Layout Hotkey" = "3" – registry keys located in the processed branch, and their parameters are set. Each key from a new line.

There is created reg file and listing of program in Figure 33.

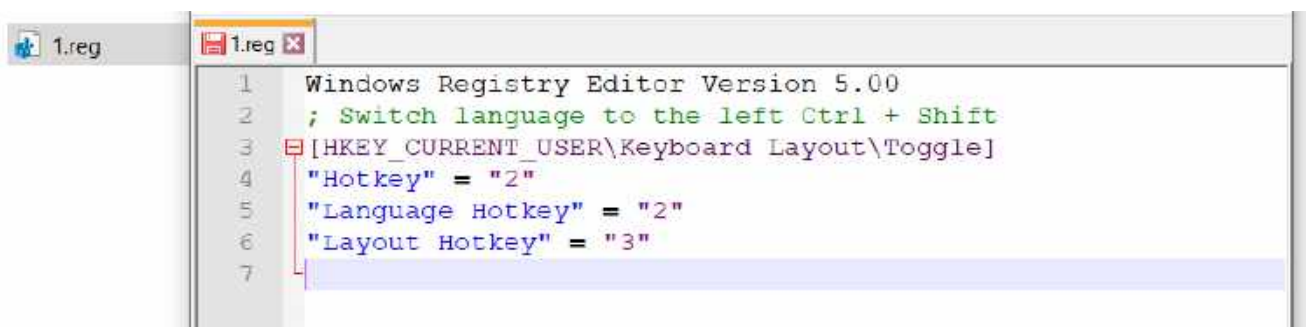


Figure 33 – Window of created reg file and listing of program for problem 1

After that, click on the File button in the text editor menu, select Save as, select All files in the File type line, and set the name in the File line, at the end of which we put a full stop and add the .reg extension (your last name in the title).

2. Write a reg-file that allows you to disable the automatic restart of the computer in the event of a blue BSOD screen, will look like this (Figure 34):

```
Windows Registry Editor Version 5.00

; Disable automatic restart in case BSOD
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\CrashControl]
"AutoReboot"=dword:00000000
```

where dword: is the data type of the added registry key.

After that, click on the File button in the text editor menu, select Save as, select All files in the File type line, and set the name in the File line, at the end of which we put a full stop and add the .reg extension (your last name in the title).

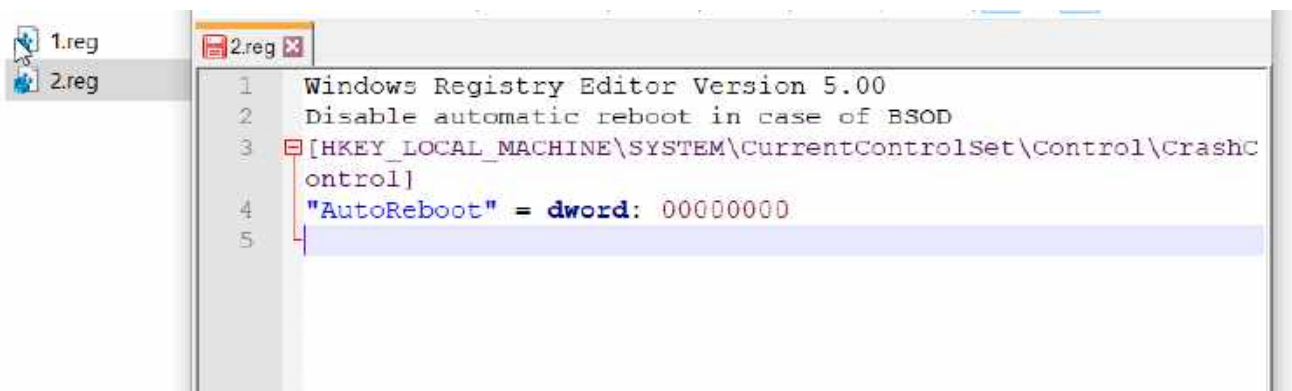


Figure 34 – Window of created reg file and listing of program for problem 2

3. Write a code that allows you to disable autorun from removable media and connecting devices (Figure 35):

```
Windows Registry Editor Version 5.00

; Disable autorun on Windows media and devices for all
users
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Current
Version\policies\Explorer]
"NoDriveTypeAutoRun"=dword:000000ff
```



```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Cdrom]
```

```
"AutoRun"=dword:00000000
```

```
; Disable autorun of Windows media and devices for the current user
```

```
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer]
```

```
"NoDriveTypeAutoRun"=dword:000000ff
```

```
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\AutoplayHandlers]
```

```
"DisableAutoplay"=dword:00000001
```



Figure 35 – Window of created reg file and listing of program for problem 3

Records of the current user's settings are added here for accuracy, if necessary, you can insert into the created reg-file only the necessary lines.

After that, click on the File button in the text editor menu, select Save as, select All files in the File type line, and set the name in the File line, at the end of which we put a full stop and add the .reg extension (your last name in the title).

4. Solve the task of deleting one of the extra folders from the Explorer window on the My Computer tab: Videos, Documents, Downloads, Images, Music, Desktop. Write a reg-file in which before each branch to delete put a minus (Figure 36):

```
Windows Registry Editor Version 5.00
```

```
;Delete the Music folder
```

```
[-
```

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersi
```

on\Explorer\MyComputer\NameSpace\{1CF1260C-4DD0-4ebb-811F-33C572699FDE}]

After that, click on the File button in the text editor menu, select Save as, select All files in the File type line, and set the name in the File line, at the end of which we put a full stop and add the .reg extension (your last name in the title).

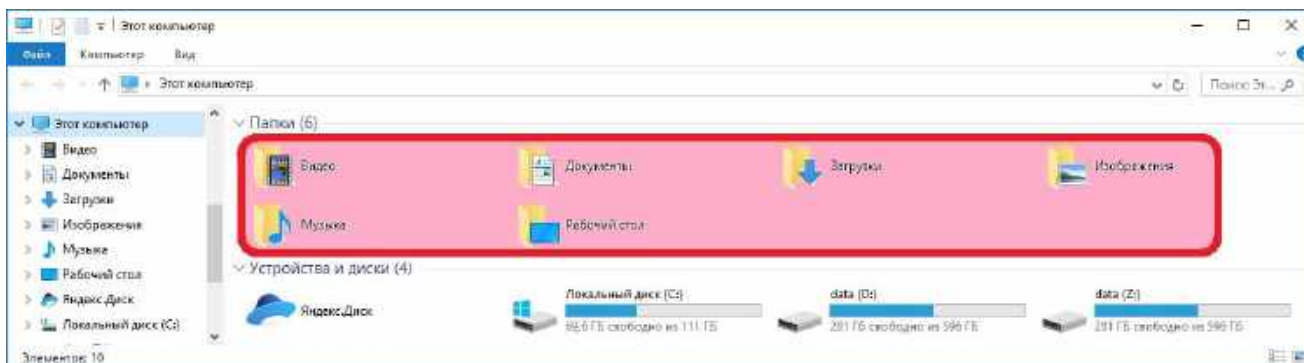


Figure 36 – Window of result for problem 4

5. Write a report on the work. Contents of the report on laboratory work: topic; goal; setting objectives; the order of execution; research results; conclusion. Copies of screens are provided with the report in accordance with the order of research.

Additional information: When configuring Windows, it could be situations where a setting cannot be changed through the user interface.

Most often, the solution lies in editing the Windows registry, which can affect a huge number of OS settings.

The Windows registry is a database of OS settings and parameters that has a tree-like structure.

That is, a huge number of custom and system settings of the OS are reflected in this virtual environment.

The virtual registry can be considered because it only organizes and structures the data that are physically stored in system files on the computer media or formed directly at the time of Windows startup.

Consider two main ways to run the standard registry editor utility:

1. Start the registry editor with the Run command:
 - run the Run utility by going to Start -> All Programs -> Standard (in Windows 10 the Run utility is in the Utilities directory), or by pressing the Start key on the keyboard (on some keyboards it is marked as Win) and R;
 - in the window, stopped the regedit command and press the Enter key.
2. Launch the registry editor from Explorer:
 - go to the directory C:\Windows;
 - run the executable file regedit.exe.

The display of information in the register has a certain structure (Figure 37).

In the right part of the window there are the sections and branches of the registry, each of which is also called the registry bush, in the left are the registry keys and their parameters.

Each section of the registry displays the information assigned to it. In modern versions of the OS from Microsoft there are five sections:

- 1) HKEY_CLASSES_ROOT (HKCR) – contains parameters for determining file types and objects;
- 2) HKEY_CURRENT_USER (HKCU) – settings of the current user (account);
- 3) HKEY_LOCAL_MACHINE (HKLM) – general computer settings that apply to all users;
- 4) HKEY_USERS (HKU) – displays information about users;
- 5) HKEY_CURRENT_CONFIG (HKCC) – displays the parameters of the equipment and connected devices of the computer.

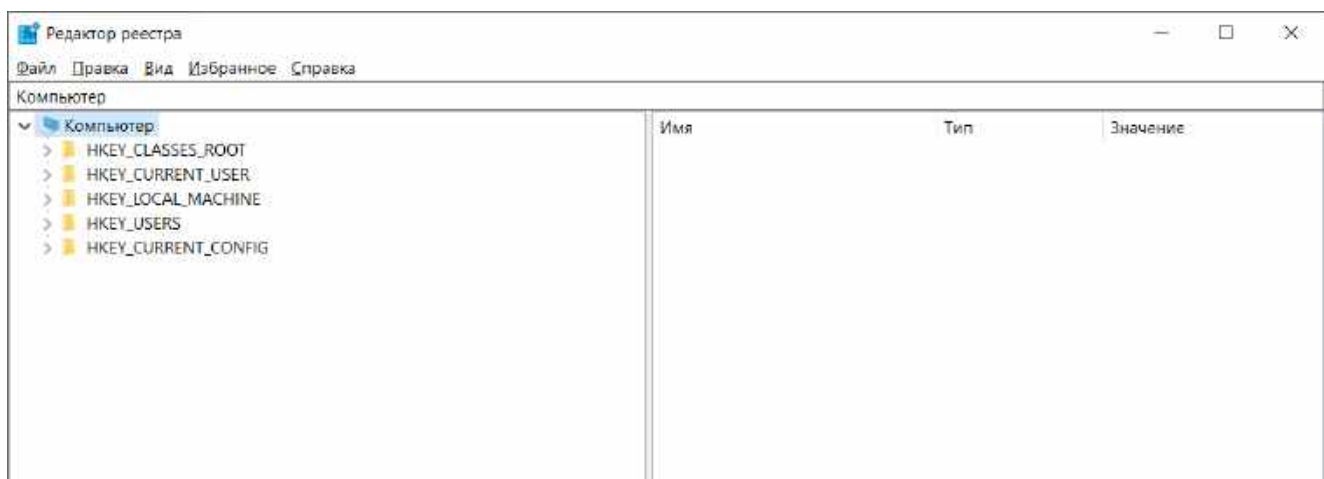


Figure 37 – Windows registry window

In earlier versions of Windows, there is another section: HKEY_DYN_DATA (HKDD) contains dynamically changing data on CPU usage, RAM usage and other current settings.

Registry keys in Windows can be of different types:

- term parameter;
- binary parameter;
- DWORD parameter (32 bits);
- QWORD parameter (64 bits);
- multiline parameter;
- extensible string parameter.

It is not possible to change the data type of the created key by standard means, if it was made a mistake when creating it, it needs to delete the incorrect record and enter a new key.

It is always advisable to back up a branch or partition variable before editing the registry.

Entering incorrect settings in the Windows registry can lead to unstable operation and system crash.

To create a backup of the registry partition, we will use the export function in the standard regedit utility:

1. Start the registry editor with the regedit command from the Run window or by running the executable file of the same name from Explorer (methods described above).

2. Right-click on the desired section and select Export.

3. In the window, select the directory to save the backup file, enter the file name and click Save.

If for some reason it is necessary to restore the registry data from the backup, we will only need to run a backup file with the extension *.reg and agree to make changes to the registry. Some keys have default values (Figure 38). This means that even if a particular key is missing in its branch, the system will behave as if it has been assigned a default setting. From this we can conclude that the process of editing the registry is divided into several types:

- change the parameters of existing keys;
- adding keys to the registry with the desired value;
- removal of unnecessary keys or even bushes from the register.

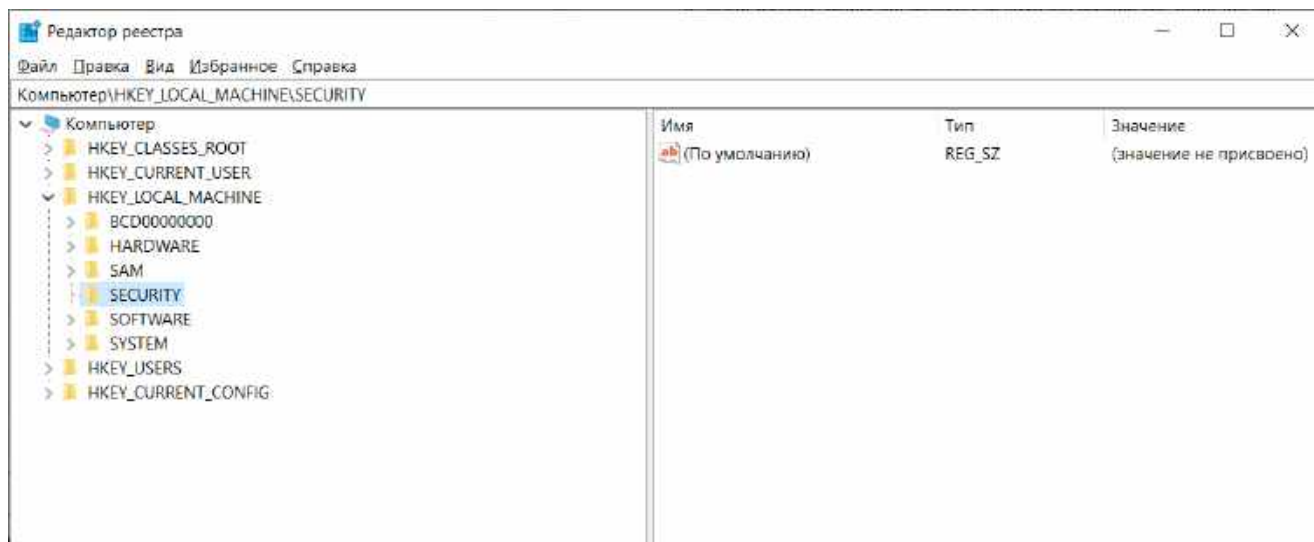


Figure 38 – Windows registry window with default value

If you need to change the keys located in different bushes of the Windows registry with one file, it is not necessary to create a separate file for each branch. For this purpose, it is enough for us after the description of parameters of the first branch to add adjustment of the following branch from a new line.

CONTROL QUESTIONS

1. History of OS.
2. Basic concepts and concepts of OS.
3. BIOS.
4. UEFI.
5. Platforms using the EFI.
6. Physical disk format.
7. Logical disk format.
8. FAT FS.
9. General information about files: file names and types.
10. Organize files and access to them.
11. Operations on files and directories.
12. Logical and physical organization of FS.
13. Classification of FS.
14. The structure of the NTFS volume. NTFS file structure.
15. Concepts and levels of RAID.
16. Hardware RAID controller.
17. Combined RAID levels.
18. Non-standard RAID levels.
19. JBOD disk array.
20. JBOD array characteristics.
21. General information about the Microsoft Management Console.
22. Microsoft Management Console user interface.
23. Create a new Microsoft Management Console. Set console options.
24. General principles of the internal structure of Windows.
25. OS model.
26. OS architecture.
27. The concept of the Windows registry.
28. Registry data types.
29. Logical structure of the register.
30. Root sections of the registry.

REFERENCES

Бондаренко, М. Ф. Операційні системи / М. Ф. Бондаренко, О. Г. Качко. – Київ : СМІТ, 2008. – 432 с.

Зибін, С. В. Операційні системи : метод. вказівки та завдання до виконання лаб. робіт з дисципліни «Операційні системи». В 2 ч. Ч. 1 / С. В. Зибін. – Київ : ДУІКТ, 2012. – 59 с.

Stallings, William. Operating Systems: Internals and Design Principles, 8th Edition, Pearson, 2014. – 800 p.

Tanenbaum, A., Bos, H. Modern Operating Systems, 4th Edition, Pearson, 2015. – 1120 p.

Третьяк, В. Ф. Основи операційних систем : навч. посіб. / В. Ф. Третьяк, Д. Ю. Голубничий, С. В. Кавун. – Харків : Вид-во ХНЕУ, 2005. – 228 с.

Solomon, D., Russinovich, M., Yosifovich, P., Ionescu, A. Windows Internals. Part 1. System architecture, processes, threads, memory management, and more, 7th Edition, Microsoft Press, 2017. – 800 p.

Шеховцов, В. А. Операційні системи / В. А. Шеховцов. – Київ : Вид. група BHV, 2005. – 576 с.

CONTENTS

Introduction.....	3
1 BASIC CONCEPTS OF OPERATING SYSTEMS. BIOS AND UEFI INTERFACES.....	4
Laboratory work № 1. BIOS AND UEFI TOOLS, AS WELL AS VIRTUALIZATION SOFTWARE.....	13
2 PHYSICAL AND LOGICAL STRUCTURE OF A HARD DRIVE.....	15
Laboratory work № 2. PHYSICAL AND LOGICAL STRUCTURE OF A HARD DRIVE.....	27
3 FILE SYSTEM.....	29
Laboratory work № 3. COMMAND ROW. WORKING WITH FILES.....	37
4 MICROSOFT MANAGEMENT CONSOLE WINDOWS.....	43
Laboratory work № 4. OPERATING SYSTEM ADMINISTRATION UTILITIES.....	46
5 STUDY OF THE INTERNAL DEVICE OF THE WINDOWS OPERATING SYSTEM.....	52
Laboratory work № 5. WINDOWS SCRIPT HOST PROGRAMMING.....	64
6 WINDOWS OPERATING SYSTEM ARCHITECTURE. PROCESSES AND FLOWS IN OPERATING SYSTEMS.....	73
Laboratory work № 6. PROCESS AND FLOW MANAGEMENT IN WINDOWS USING THE POWERSHELL.....	84
7 WINDOWS OPERATING SYSTEM REGISTER.....	91
Laboratory work № 7. WINDOWS OPERATING SYSTEM REGISTER.....	96
Control questions.....	105
References.....	106

Навчальне видання

**Морозова Ольга Ігорівна
Узун Дмитро Дмитрович**

ОПЕРАЦІЙНІ СИСТЕМИ

Частина 1

НАЛАШТУВАННЯ Й КОНФІГУРАЦІЯ
(Англійською мовою)

Редактор О. І. Морозова
Технічний редактор А. М. Ємленінова

Зв. план, 2021

Підписано до друку 20.05.2021

Формат 60x84 1/16. Папір офс. Офс. друк

Ум. друк. арк. 6,0. Обл.-вид. арк. 6,75. Наклад 50 пр.

Замовлення 124. Ціна вільна

Видавець і виготовлювач

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

61070, Харків-70, вул. Чкалова, 17

<http://www.khai.edu>

Видавничий центр «ХАІ»

61070, Харків-70, вул. Чкалова, 17

izdat@khai.edu

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції сер. ДК № 391 від 30.03.2001