

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN  
FEDERATION  
Federal State Autonomous  
educational institution of higher education  
NORTH CAUCASIAN FEDERAL UNIVERSITY

Methodological guidelines  
to practical work  
by discipline  
Intelligent Terminal Operating System / Интеллектуальная терминальная  
операционная система  
for master's degree students  
02.04.01 Mathematics and Computer Science

Stavropol, 2026

## Table of contents

|                                                                          |     |
|--------------------------------------------------------------------------|-----|
| Introduction .....                                                       | 3   |
| 1 Obtaining the Astra Linux .....                                        | 4   |
| 2 Installing the Astra Linux operating system on a virtual machine ..... | 9   |
| 3 Initial setup of Astra Linux .....                                     | 22  |
| 4 Working with the Fly .....                                             | 27  |
| 5 Working with the Terminal and Bash Shell .....                         | 42  |
| 6 Working with help systems .....                                        | 61  |
| 7 Working with files and directories .....                               | 77  |
| 8 Working with text information .....                                    | 107 |
| Literature .....                                                         | 138 |

## **Introduction**

**The aim of the course** "Intelligent Terminal Operating System / Интеллектуальная терминальная операционная система" is to develop students' theoretical knowledge and practical skills in the application of modern operating systems in information systems in various fields of professional activity (industrial production, economics, business, etc.).

**The main objective** of studying the discipline is to familiarize students with the main trends in the development of operating systems and to acquire practical skills in using certain operating systems in various areas of industrial and commercial activity.

**As a result of studying the discipline, students**

**They will recognize:**

1. Mechanisms for managing processes, memory, and input/output.
2. Definition, purpose, composition and functions of operating systems.
3. Classification of operating systems.

**They will be able to:**

1. Expand user capabilities by installing applications.
2. Perform basic settings, change the operating system configuration.
3. Use the user interface tools of operating systems.

**Learn to master:**

1. Skills in installing, configuring and working with human-machine interface software.
2. Skills in installing and configuring operating systems.
3. Skills for obtaining information about operating system parameters.

This tutorial is intended for students studying operating systems.

### **Equipment and materials**

The laboratory practical training is carried out in a regular computer lab equipped with personal computers running the AstraLinux operating system .

### **Safety instructions**

Safety guidelines for the standard computer lab have been developed and approved by the administration and are communicated to each student upon signature. There are no additional requirements.

# **1 Obtaining the Astra Linux operating system distribution**

## **1.1 Objectives of the work**

1. Learn about the recommended system requirements for installing Astra Linux OS .
2. Read a brief description of the Astra Linux operating system
3. Download the Astra Linux operating system distribution.

## **1.2 Theoretical justification**

### **1.2.1 Computer requirements for installing Astra Linux OS**

Minimum system requirements are the conditions under which your computer can operate reliably without the Fly desktop graphical interface, without applications that require Fly (web browser, office suite, etc.), and without additional services that are included in the distribution but not enabled by default.

The minimum system requirements for installing Astra Linux SE 1.7.4 OS are as follows:

- Hardware platform – processor with x86-64 architecture (AMD, Intel);
- RAM capacity – not less than 1.5 GB;
- Free disk space – at least 4 GB;
- DVD Readers;
- Standard SVGA monitor;
- Compatibility with equipment (standard, as provided in the documentation).

The recommended system requirements ensure stable operation of the operating system with the Fly desktop graphical interface and all programs included in the distribution. This does not apply to cases where special requirements are specified in the documentation.

The recommended system requirements for installing Astra Linux SE 1.7.4 OS are as follows:

- Hardware platform – processor with x86-64 architecture (AMD, Intel);
- RAM capacity – not less than 4 GB;
- Free disk space – at least 16 GB;
- DVD Readers;
- Standard SVGA monitor;
- Compatibility with equipment (standard, provided by documentation);

The OS functions only on compatible hardware recommended by the OS manufacturer, the list of which, as well as the compatibility certification regulations, are published on the website: <http://astralinux.ru/>.

### **1.2.2 Brief description of the Astra Linux operating system**

Astra Group PJSC is a leader in the Russian IT market, developing software and information security solutions. The company's portfolio includes two dozen products,

the core of which is the certified, special-purpose Astra Linux operating system with integrated information security tools (ISS) for the secure operation of infrastructures of any scale and data of any sensitivity level, certified by the Federal Service for Technical and Export Control (FSTEC), the Federal Security Service (FSB), and the Russian Ministry of Defense.

The Astra Linux operating system allows for secure use of the Debian codebase, including a large number of applications, and provides timely access to necessary security patches. The system is equipped with its own information security tools, including a mandatory access control module and a closed software environment integrity module. This is in addition to the original Fly graphical user interface, which will be familiar to most users and will simplify the import substitution process.

The Astra Linux operating system supports a wide range of modern hardware, and the company's employees are actively working to expand this list. More than 3,000 companies already use Astra Linux, saving time on selecting and testing software and components. You can find the current list of compatible hardware on the company's website: <https://astralinux.ru/ready-for-astra/>

### 1.2.3 Astra Linux versions

Astra Linux versioning differs from Windows. The version number consists of several numbers separated by periods (Figure 1.1).

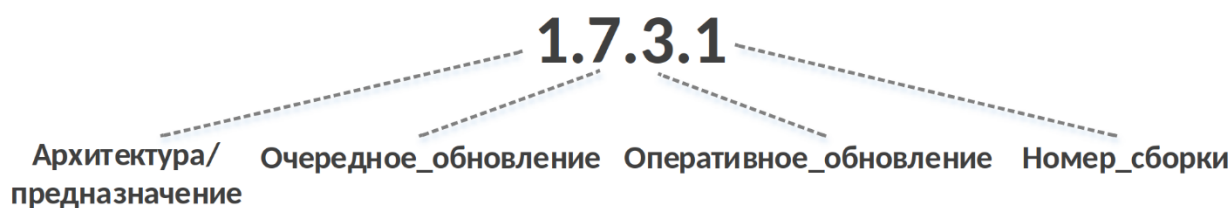


Figure 1.1 - Versioning Astra Linux OS

The first number (1) in the version number denotes the architecture or purpose. Possible values are:

- 1 - Astra Linux Special Edition (ALSE) x86-64 subarchitecture ;
- 2 - Astra Linux Common Edition (ALCE) x86-64 subarchitecture;
- 4 - ALSE ARM subarchitecture;
- 8 - ALSE subarchitecture Elbrus.

The second number (7) corresponds to the next update (OS version) number. The release of new updates is necessary for:

- implementation and improvement of functional capabilities;
- support of modern equipment;
- ensuring compliance with current information security requirements;
- improving usability, component management, and more.

Thus, versions 1.7 and 4.7 are the same version of the operating system, with the update number 7, but they target different architectures. Version 1.7 is designed for x86-64 processors, while version 4.7 will run on ARM processors.

The third number (3) corresponds to the numbers of operational updates, the

release of which allows for the elimination of critical and high-severity vulnerabilities. Such updates may be published not only as modified software component files, but also as instructions and guidelines for operating system configuration and operating features, containing information on compensating measures or limitations for operating the operating system.

In addition to regular and operational updates, the company releases so-called urgent operational updates, designated by the UU prefix (from "urgentupdate"). For example, two urgent updates, UU1 and UU2, were released for ALSE 1.7.3.

Licensees using the Astra Linux Special Edition operating system, certified according to the information security requirements of the Federal Service for Technical and Export Control of Russia (Certificate No. 2557), must apply the methods published in security bulletins and install software updates to neutralize the threats of vulnerabilities in information systems.

#### 1.2.4 Astra Linux Repositories

The official operating system repositories are available at <https://download.astralinux.ru>, but since indexing directories within the repository is prohibited, manually browsing these directories is impractical; they must be accessed through the apt package manager (Figure 1.2 ). Additionally, the Astra Linux OS distribution can be downloaded from the university file server [\\ storage 2](#) in the OS section .

| File Name ↓                                              | File Size ↓ | Date ↓            |
|----------------------------------------------------------|-------------|-------------------|
| <a href="#">Parent directory/</a>                        | -           | -                 |
| <a href="#">alce-2.12.46.6-17.04.2023_15.09.iso</a>      | 5625704448  | 2023-Apr-17 12:13 |
| <a href="#">alce-2.12.46.6-17.04.2023_15.09.iso.gost</a> | 102         | 2023-Apr-18 12:46 |
| <a href="#">alce-2.12.46.6-17.04.2023_15.09.iso.md5</a>  | 70          | 2023-Apr-18 12:45 |
| <a href="#">alce-current.iso</a>                         | 5625704448  | 2023-Apr-17 12:13 |
| <a href="#">alce-current.iso.gost</a>                    | 102         | 2023-Apr-18 12:46 |
| <a href="#">alce-current.iso.md5</a>                     | 70          | 2023-Apr-18 12:45 |
| <a href="#">orel-current.iso</a>                         | 5625704448  | 2023-Apr-17 12:13 |
| <a href="#">orel-stable.iso</a>                          | 5625704448  | 2023-Apr-17 12:13 |

Figure 1.2 - Astra Linux OS Repository

There are 4 repositories available for ALSE x.7:

- **main** is the operating system's primary repository. The program code from this repository implements all the product's functionality and security features.
- **update** - a repository with live updates of the main repository.
- **base** - the base repository, which combines main and update, and also contains packages with development tools.
- **Extended** - contains additional software that can run in the Astra Linux environment. This software is not updated to implement security features and

may be incompatible with the latest live updates of the main/base repository. This is documented in the live update bulletin and resolved as the extended repository is updated. Extended repository software is not subject to certification.

### 1.2.5 ALSE Security Levels

Starting with the latest ALSE 1.7 update, the operating system is supplied as a single distribution capable of functioning in one of three security modes (security levels) implemented by patented information security tools (ISM):

- **Basic security level "OREL"** . Recommended for processing publicly available information that does not require special protection.
- **Enhanced security level "Voronezh"** . Recommended for processing confidential information that does not contain state secrets.
- **Maximum security level "SMOLENSK"** . Recommended for processing information of any access category in government information systems.

A comparative table of Astra Linux Special Edition editions is given in Table 1.1.

Table 1.1. Comparison table of Astra Linux Special Edition editions

| Characteristic                                                                                     | Eagle | Voronezh | Smolensk |
|----------------------------------------------------------------------------------------------------|-------|----------|----------|
| FSTEC certificate                                                                                  | —     | v        | v        |
| FSB certificate                                                                                    | —     | —        | v        |
| MO certificate                                                                                     | —     | —        | v        |
| Mandatory integrity control                                                                        | —     | v        | v        |
| Role-based access control                                                                          | —     | v        | v        |
| Discretionary access control                                                                       | —     | v        | v        |
| Mandatory access control                                                                           | —     | —        | v        |
| DBMS and virtualization protection mechanisms                                                      | v     | v        | v        |
| Kiosk mode - a "white" list of applications allowed to run                                         | v     | v        | v        |
| Correct work with electronic signature                                                             | v     | v        | v        |
| Additional isolation of virtualization components and Docker containers (separate integrity level) | —     | v        | v        |
| Dynamic integrity control (closed software environment)                                            | —     | v        | v        |
| Security event logging                                                                             | v     | v        | v        |
| Control of connection of removable storage media                                                   | v     | v        | v        |

### 1.3 Assignment for independent completion

1. Download the required distribution and operating system for installing Astra Linux.

2. Take the test based on the lab materials  
<https://onlinetestpad.com/cozyheysdfaui>.

## 2 Installing the Astra Linux operating system on a virtual machine

### 2.1 Objectives of the work

1. Learn how to install the Astra Linux operating system on a virtual machine.

### 2.2 Theoretical justification

Installing Astra Linux involves several steps, some of which are performed automatically, meaning without your direct intervention. Let's look at each of these steps in turn.

#### 2.2.1 Step 1: Setting up the VirtualBox virtual machine

Launch the virtual machine. Specify a name for the virtual machine, specify the location of the Astra Linux operating system distribution file downloaded in the previous step (Figure 2.1), and start the machine.

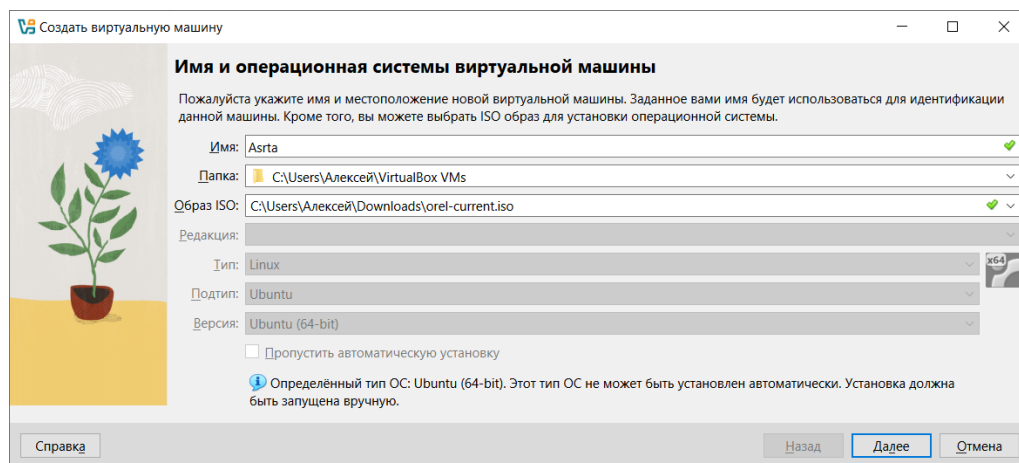


Figure 2.1 - Astra Linux OS Repository

#### 2.2.2 Step 2: Selecting the language and installation type

When booting from the Astra Linux installation disk, the GRUB bootloader first launches, prompting you to select installation options. Changing the language will switch the interface to English. The following installation modes are available:

- Graphical installation
- Installation using the text interface (Installation)
- Recovery mode (Rescuemode)

The graphical installation in Russian starts automatically after 5 seconds (Figure 2.2).

Let's install Astra Linux using the graphical interface, by selecting the appropriate option or waiting for it to launch automatically.

If the virtual machine has taken control of the mouse, you can regain control

using the hotkey combination: Left Ctrl + Right Ctrl.

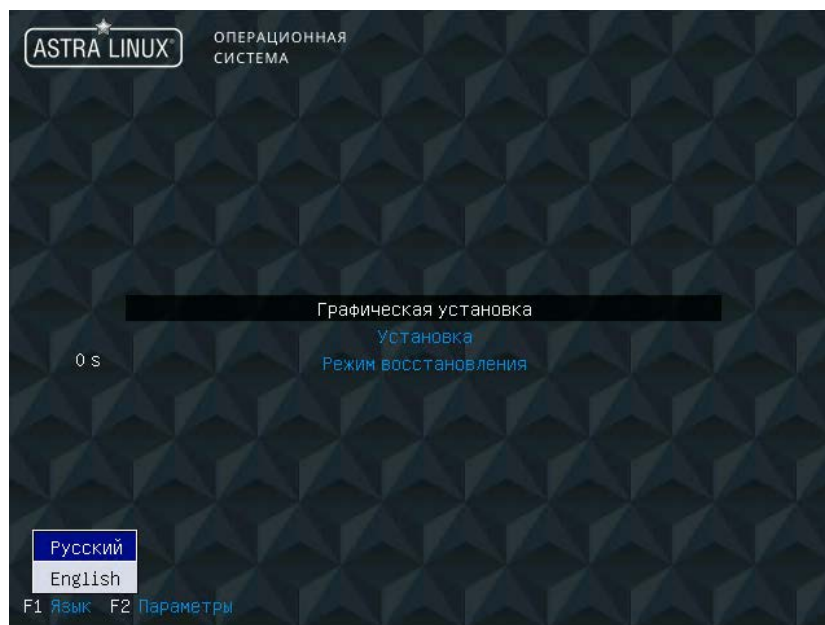


Figure 2.2 - Selecting the language and installation type of Astra Linux OS

### 2.2.3 Step 3. Reading and accepting the license agreement

Starting from this window (Figure 2.3), we're already working in Linux. If you're interested in checking this out, enable the virtual keyboard and switch to the second screen using the hotkey: ctrl + Alt + F2 .

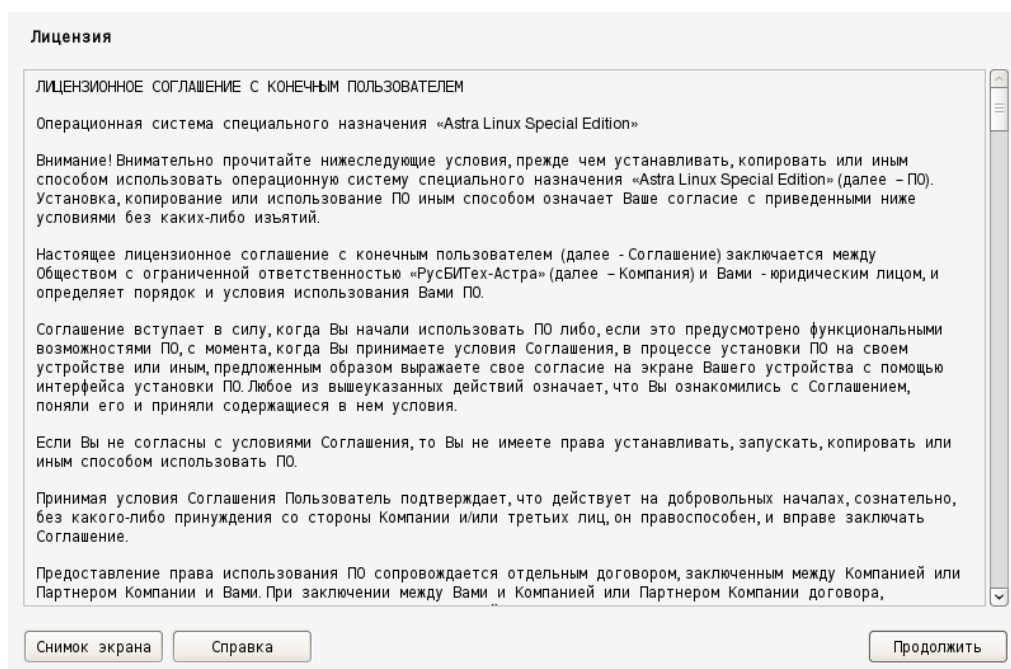


Figure 2.3 - Astra Linux OS License Agreement

To go back, use: ctrl + Alt + F5 .

If you don't have time to be distracted by trivial matters, read the license agreement and click Continue . Incidentally, residents of the former Soviet Union are

said to have a black belt in software license agreement reading speed.

#### 2.2.4 Step 4: Keyboard Setup

Choose the method that is convenient for you to switch the keyboard layout (by default it is Alt + Shift ).

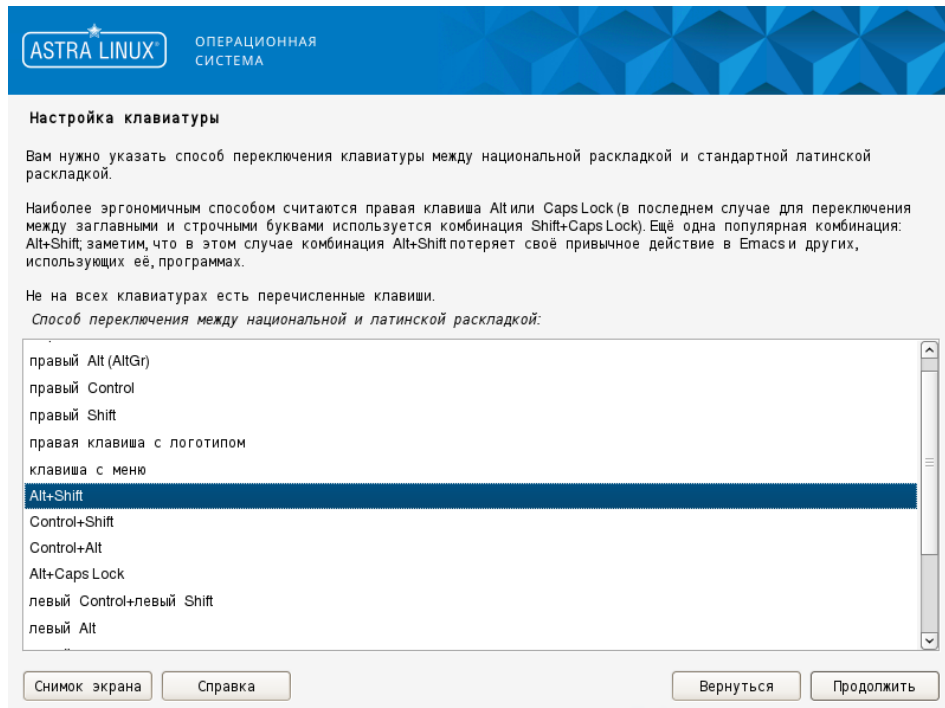


Figure 2.4 - Method for switching keyboard layouts

#### 2.2.5 Step 5. Find and mount the CD-ROM (automatically)



Figure 2.5 - Finding and mounting a CD-ROM

#### 2.2.6 Step 6. Loading the installation components from the CD (automatically)

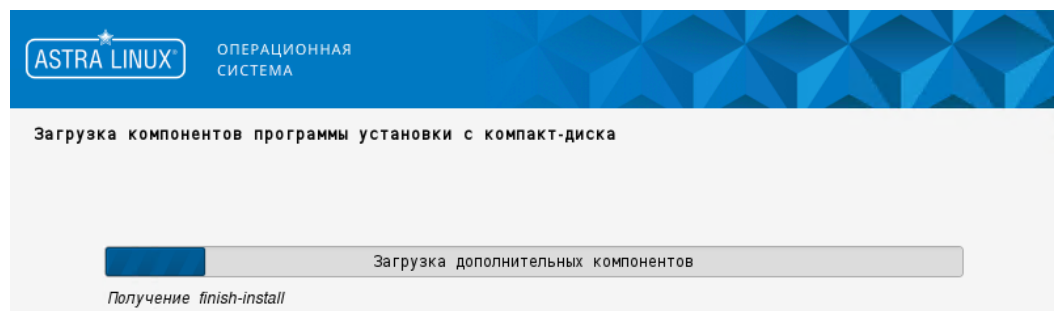


Figure 2.5 - Loading the installation program components

## 2.2.7 Step 7. Detecting the network card (automatically)



Figure 2.6 - Network card definition

## 2.2.8 Step 8. Network setup. Entering the computer name

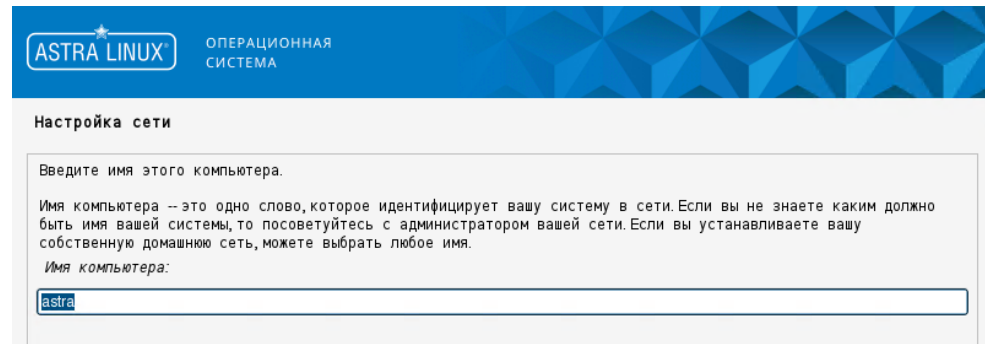


Figure 2.7 - Network setup

Enter the desired computer name (the default is "astra"). The computer name identifies the operating system on the network.

## 2.2.9 Step 9. Setting up user accounts and passwords

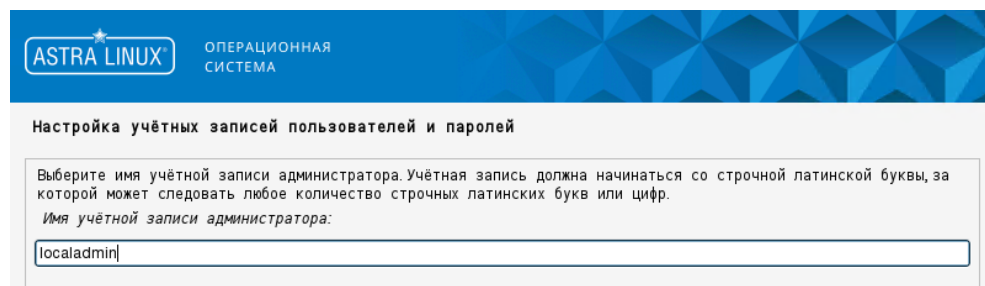


Figure 2.8 - Setting up user accounts and passwords

Enter the name of the local administrator for this computer (e.g. "localadmin")

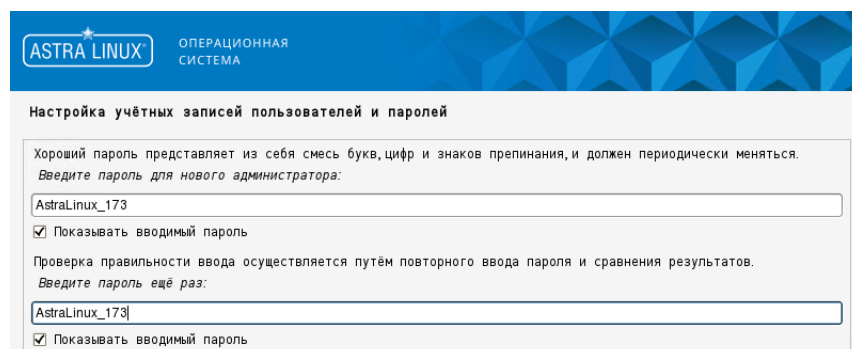


Figure 2.9 - Entering the local administrator name of this computer

Enter the account password and click "Continue." Just as a local administrator account with RID=500 is created during Windows installation, a local user account with POSIX UID=1000 is created in Astra Linux at this step. This account will automatically be included in the astra-admin group, which has the ability to elevate privileges to root using the sudo utility.

### 2.2.10 Step 10. Setting the time. Selecting a time zone

Select your time zone (Moscow by default) and click the "Continue" button.

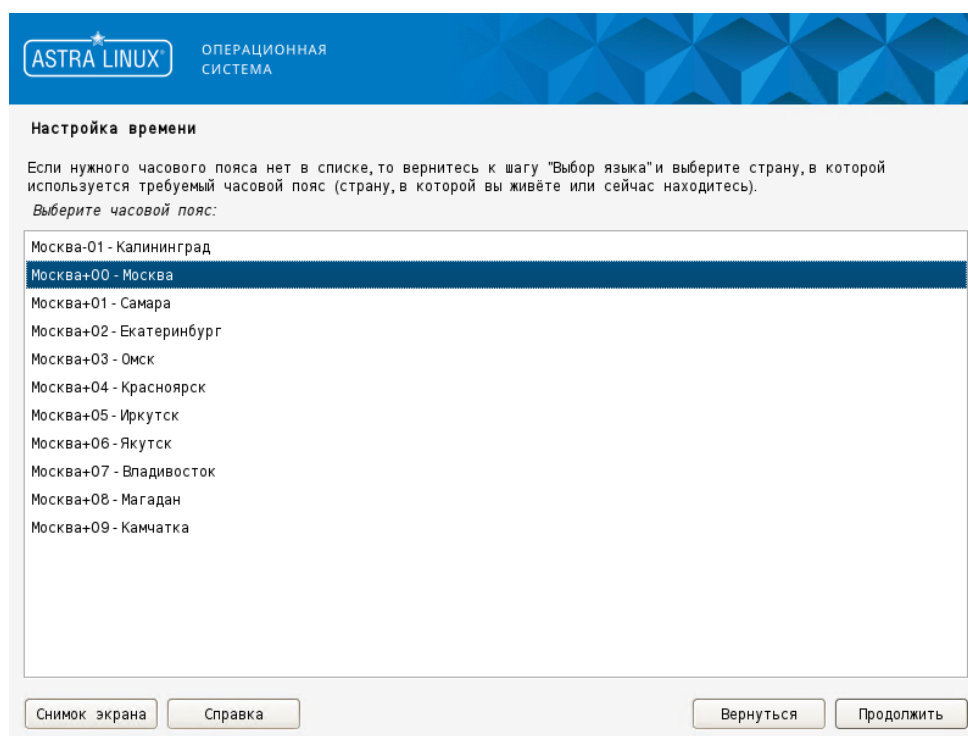


Figure 2.10 - Time setting

### 2.2.11 Step 1 1. Identifying disks

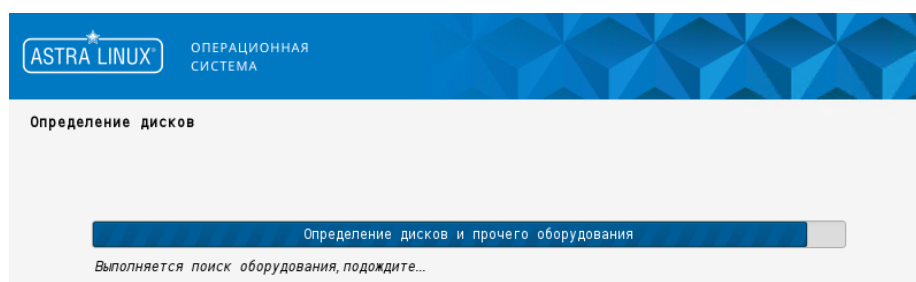


Figure 2.11 - Defining disks

### 2.2.12 Step 1 2. Partitioning disks

The installation wizard offers us several partitioning options to choose from:

- Auto – use entire disk
- Auto – use the entire disk and configure LVM

- Auto – use entire disk with protective conversion to LVM
- Manually

Disk partitioning is an important step in installing a Linux system. Due to the specifics of the file systems used in Linux, poor disk partitioning can cause performance issues on your computer, or even prevent it from creating new files on the disk even if there's free space (called inode exhaustion).

When partitioning a disk, it's important to consider the computer's intended purpose, the roles it will perform, and the planned workload and file system hierarchy. All these issues will be covered in detail in the modules on files and directories, file systems, and performance optimization.

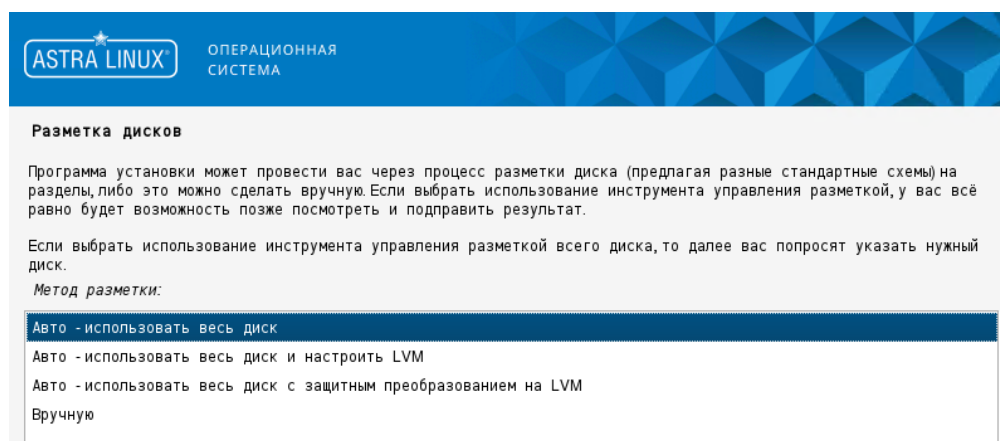


Figure 2.12 - Disk markings

For now, let's select the first partitioning option, "Auto – Use entire disk ." This automatic partitioning procedure consists of just five steps:

1. Choosing a drive. **If you have multiple drives, it's important to choose the right one!**

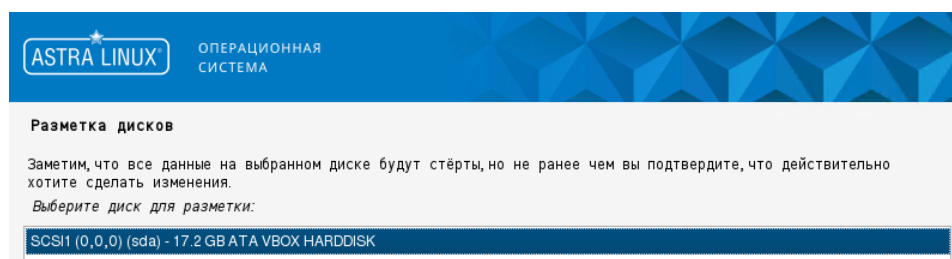


Figure 2.13 - Disk markings

2. Select a disk partitioning scheme. All files in one partition or a dedicated partition for /home . Select the All files in one partition option (recommended for beginners) .

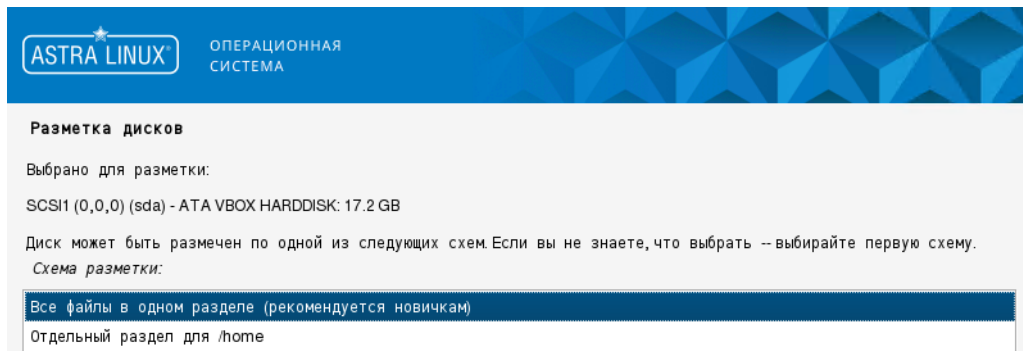


Figure 2.14 - Disk markings

### 3. Getting familiar with section settings

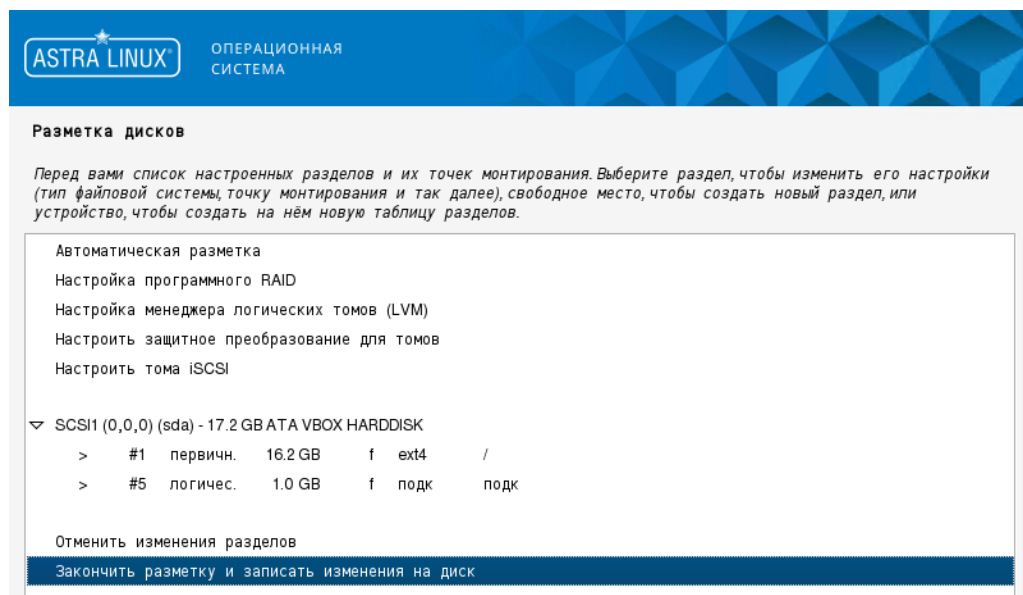


Figure 2.15 - Getting to know the section settings

4. Confirm the settings. After confirming, irreversible disk partitioning operations will be performed, and all existing data on the disk will be destroyed!

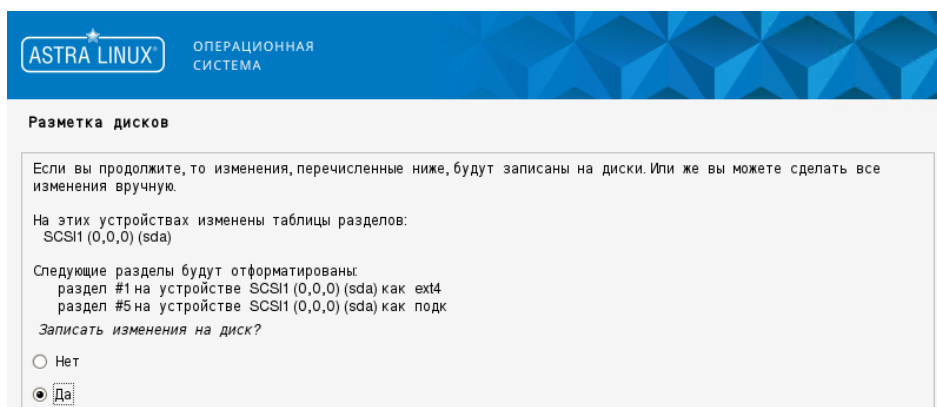


Figure 2.16 - Confirming the application of settings

5. Applying changes and writing them to disk.

## 2.2.13 Step 1 3. Installing the base system

### 1. Installing the base system: Starting the installation (automatically)

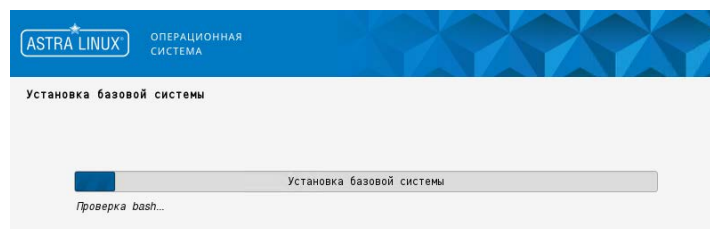


Figure 2.17 - Installing the base system

### 2. Installing the base system: choosing the OS kernel

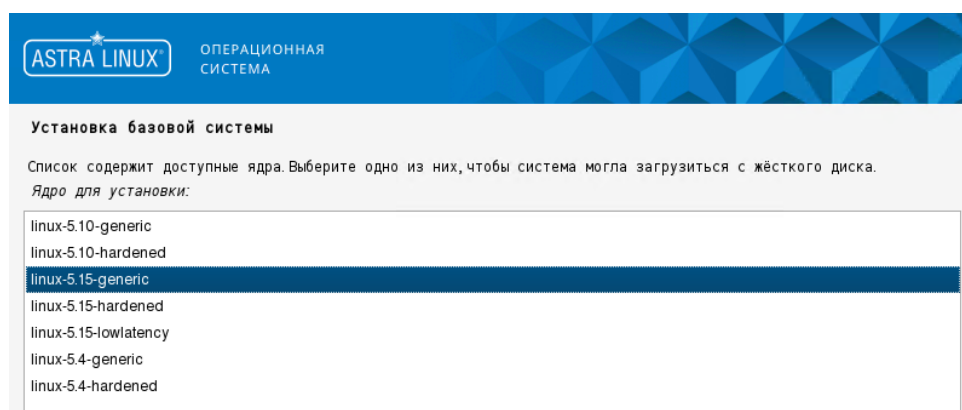


Figure 2.18 - Installing the base system

On this page you can select the Linux kernel version and build option:

- **generic** - general-purpose assembly;
- **hardened** - the assembly provides enhanced self-defense capabilities due to additional capabilities for cleaning residual information;
- **Lowlatency** – a build with a task switching frequency increased to 1000 Hz, which ensures faster system response but increases task switching overhead. Subjective improvement in computer performance comes at the cost of reduced useful performance and increased power consumption.

Please note that some products have limitations on the Linux kernel versions they can use, for example, ALD Pro cannot be installed on the hardened version.

### 3. Installing the base system: completing the installation (automatically)

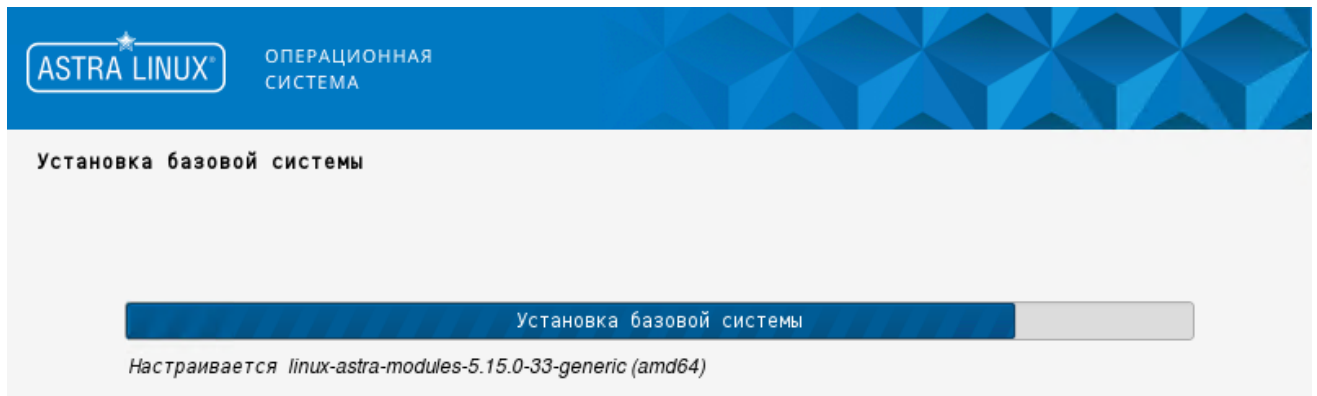


Figure 2.19 - Installing the base system

## 2.2.14 Step 14. Configure the package manager (automatically)

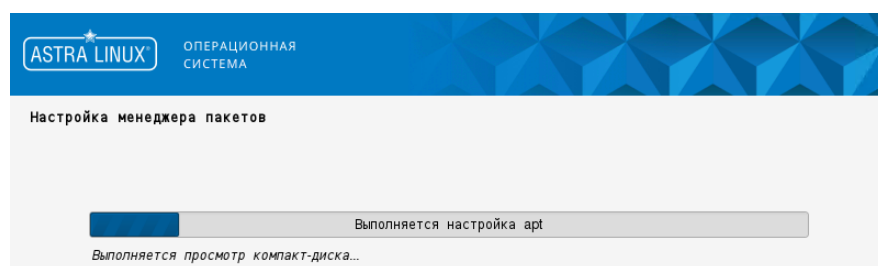


Figure 2.20 - Configuring the package manager

## 2.2.15 Step 15. Selecting and installing additional software

### 1. Selecting and installing additional software

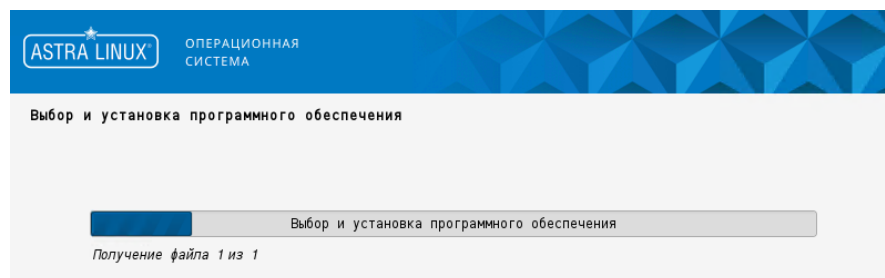


Figure 2.21 - Installing additional software

### 2. First, the master is prepared automatically.

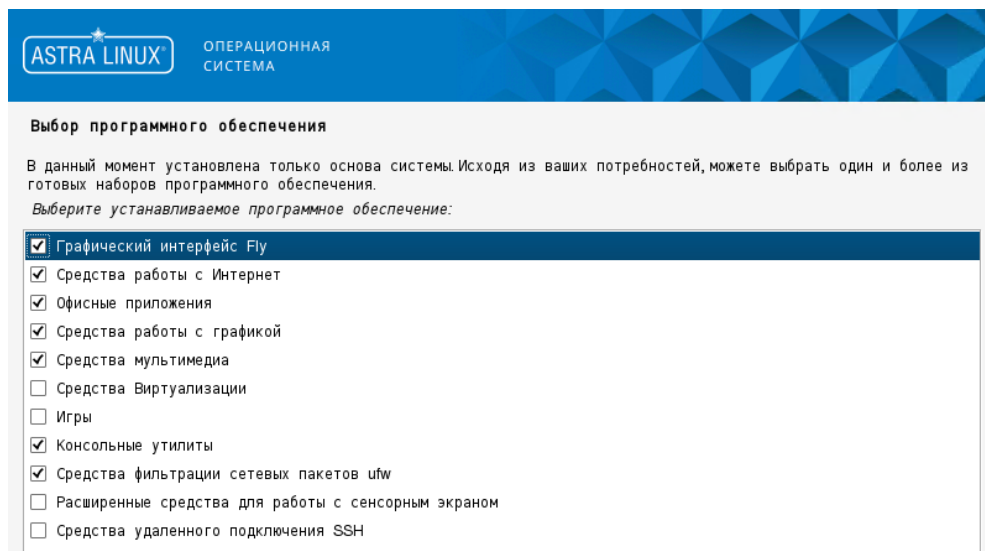


Figure 2.22 - Selecting software

Below is a short list of software included with the OS distribution (i.e., "out of the box"). For convenience, they are grouped into sections, allowing you to select a set of related programs at once. We'll leave the default software selection. In the Linux world, unlike Windows, software distributions are commonly referred to as packages.

### 3. Selecting and installing additional software: software installation (automatic)

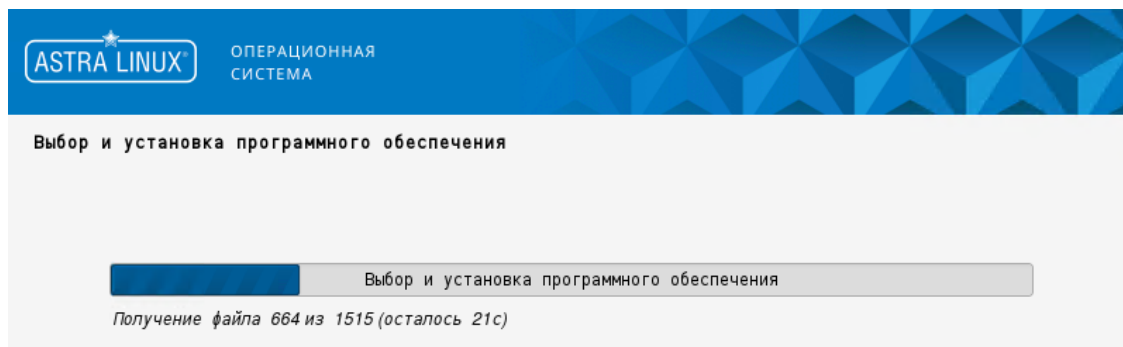


Figure 2.23 - Installing additional software

Depending on the software you choose, this step may take a long time, so please be patient.

## 2.2.16 Step 16. Additional OS settings

### 1. Additional OS settings: choosing the security level

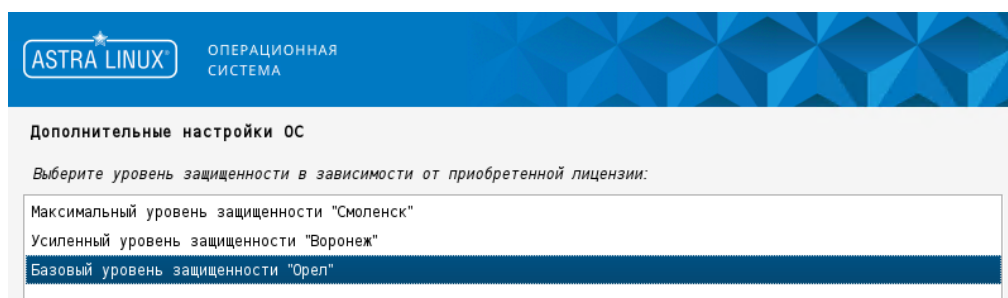


Figure 2.24 - Selecting the security level

The security level should be selected based on your information security needs and the type of license purchased. Let's choose the basic "Orel" level.

## 2. Additional OS settings: choosing security options

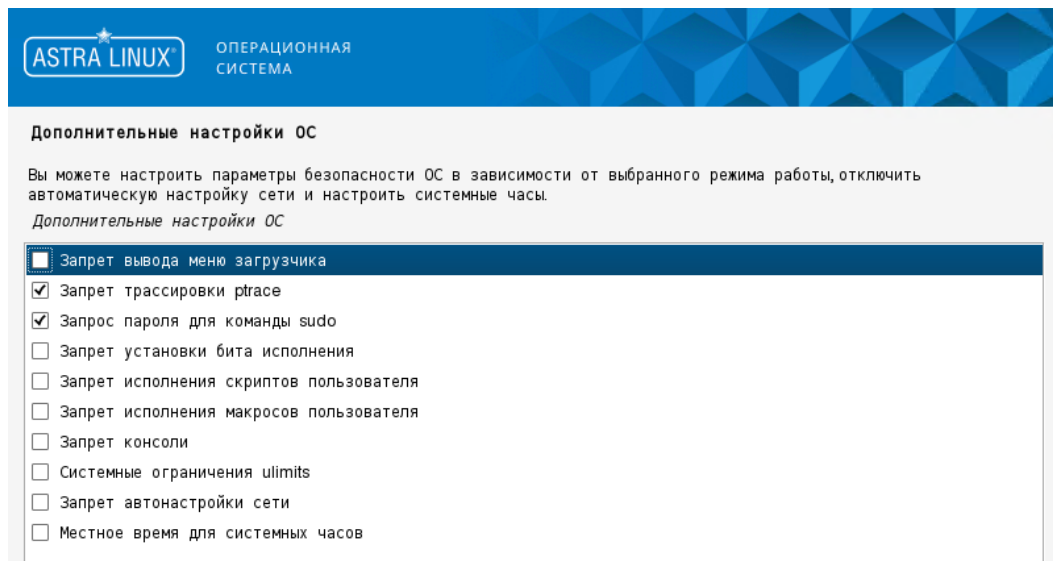


Figure 2.25 - Selecting security parameters

Let's leave the OS security settings at their default values.

## 2.2.17 Step 17. Installing the GRUB bootloader

**GRUB** (GNU Unified Bootloader) is an operating system boot loader from the GNU Project. GRUB allows a user to have multiple operating systems installed on a single computer and select one to boot from when turning on the computer.

### 1. Installing the GRUB bootloader: Preparing for installation (automatic)

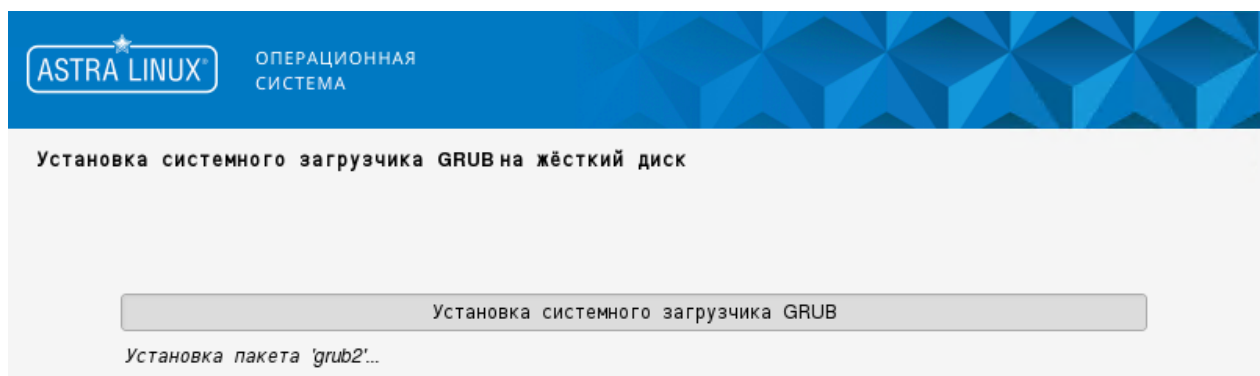


Figure 2.25 - Selecting security parameters

### 2. Installing the GRUB bootloader: Confirming the installation

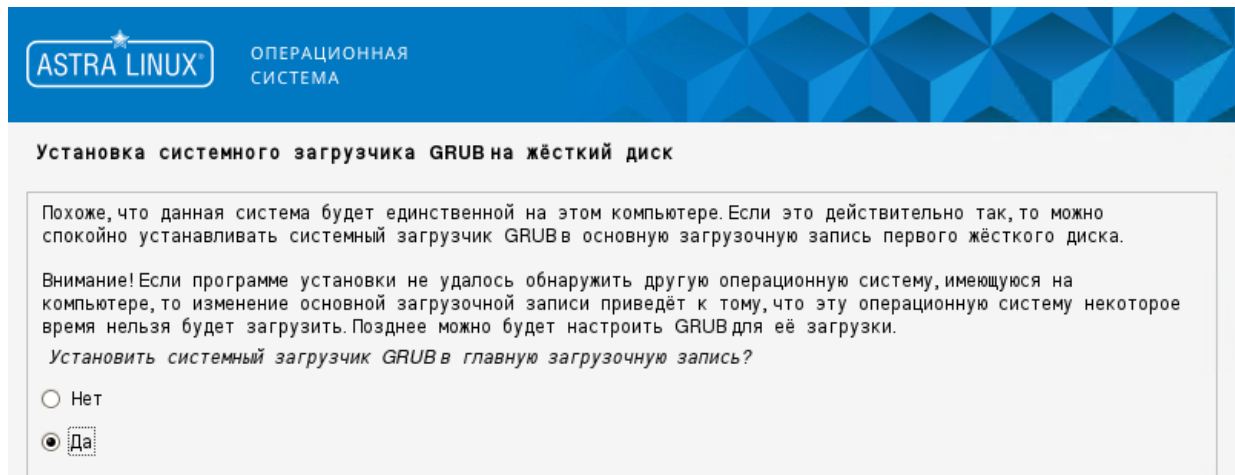


Figure 2.26 - Installing the GRUB bootloader

## 2.2.18 Step 17. Completing the installation

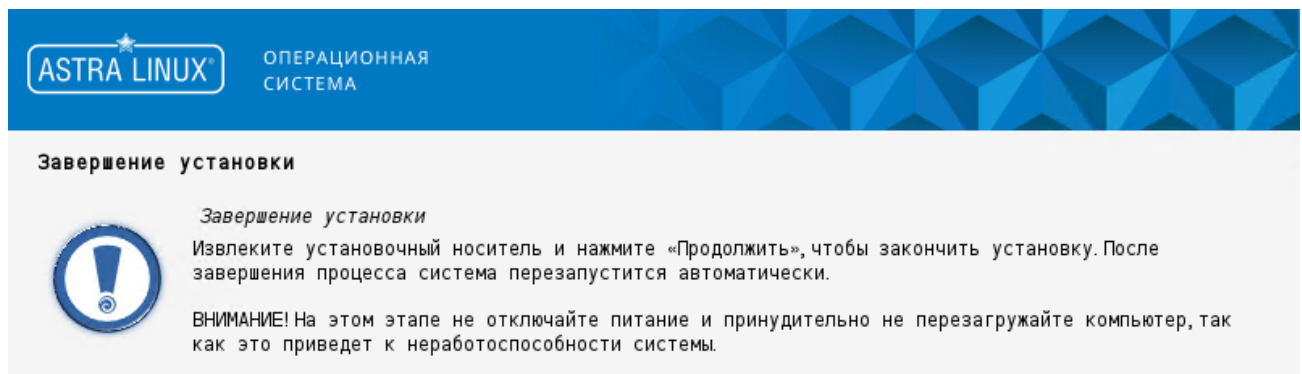


Figure 2.27 - Completing the installation

The system prompts us to remove the installation media and warns us that a forced reboot or power-off of the computer is not allowed. **Please wait for the automatic reboot!** Congratulations, you have installed Astra Linux OS on your computer!

After rebooting, you will be taken to the graphical interface and will see a prompt to enter your credentials.

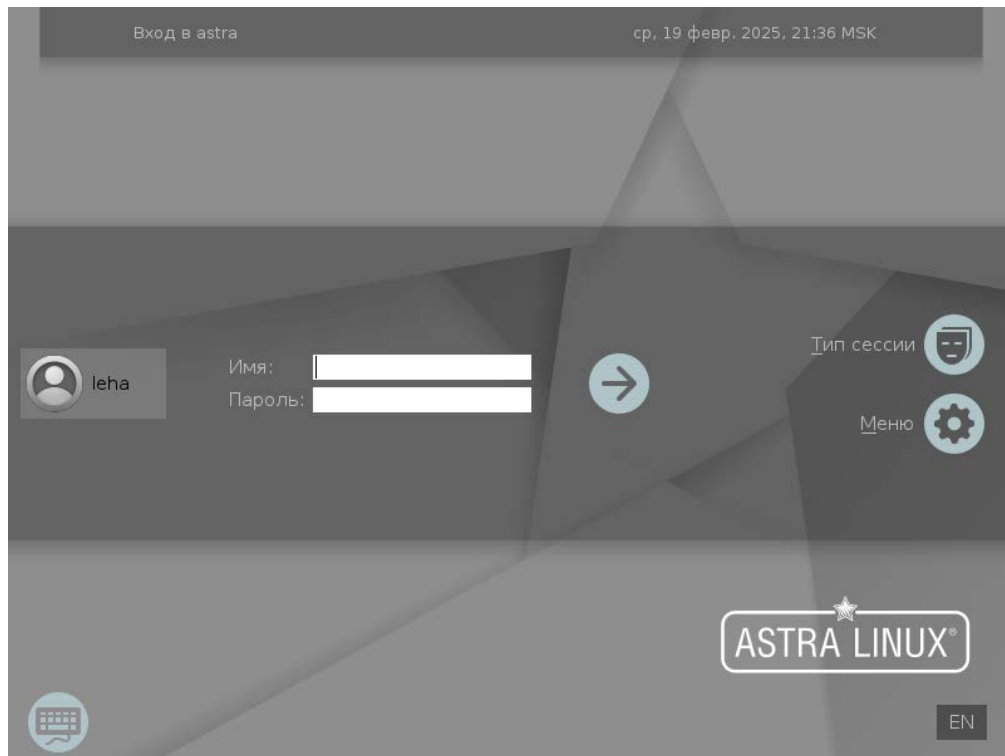


Figure 2.28 - Prompt to enter credentials

### 2.3 Assignment for independent completion

1. Install the Astra Linux operating system.

2. Take the test based on the lab materials

<https://onlinetestpad.com/dam4c2nvptoui>.

## 3 Initial setup of Astra Linux

### 3.1 Objectives of the work

1. Perform initial setup of Astra Linux OS.

### 3.2 Theoretical justification

In this lab, we won't focus on the "why" and "why" of it, but rather on the "how" of configuring a number of settings to ensure the system works properly on the network and is easy to use. To log in, enter your account name and password in the appropriate fields and click the "Login" button.

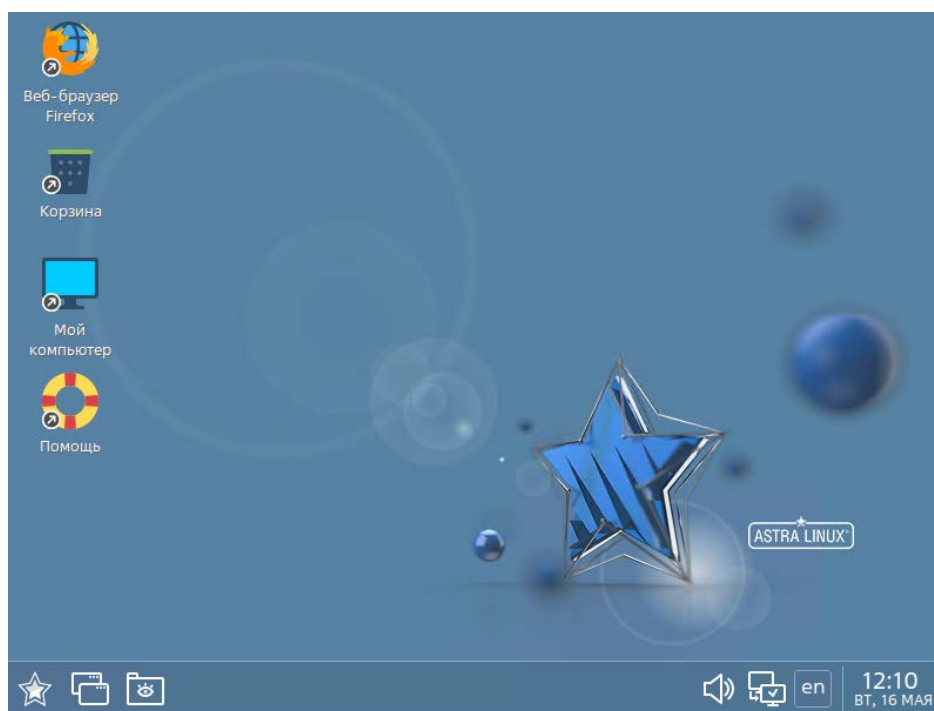


Figure 3.1 – Astra Linux OS desktop

#### 3.2.1 Setting screen resolution

1. Right-click on an empty area of the desktop and select Monitor Settings... from the context menu . The Monitor Settings window ( fly-admin-screen utility ) will open.
2. Select the desired screen resolution.
3. Accept the settings by clicking the "Yes" button, which is marked with an arrow (the button text may be hidden if the program window does not fit on the screen).

#### 3.2.2 Network settings

1. Right-click on the network icon (Network Manager applet) in the taskbar and select "Change connection...".



Figure 3.2 – Network settings

2. In the window that opens, select your connection and click on the gear button.

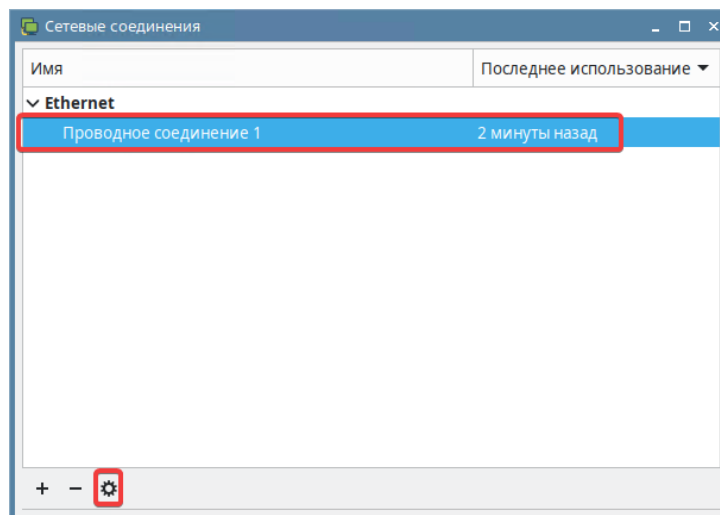


Figure 3.3 – Network connection

3. In the window that opens, go to the "IPv4 Settings" tab.

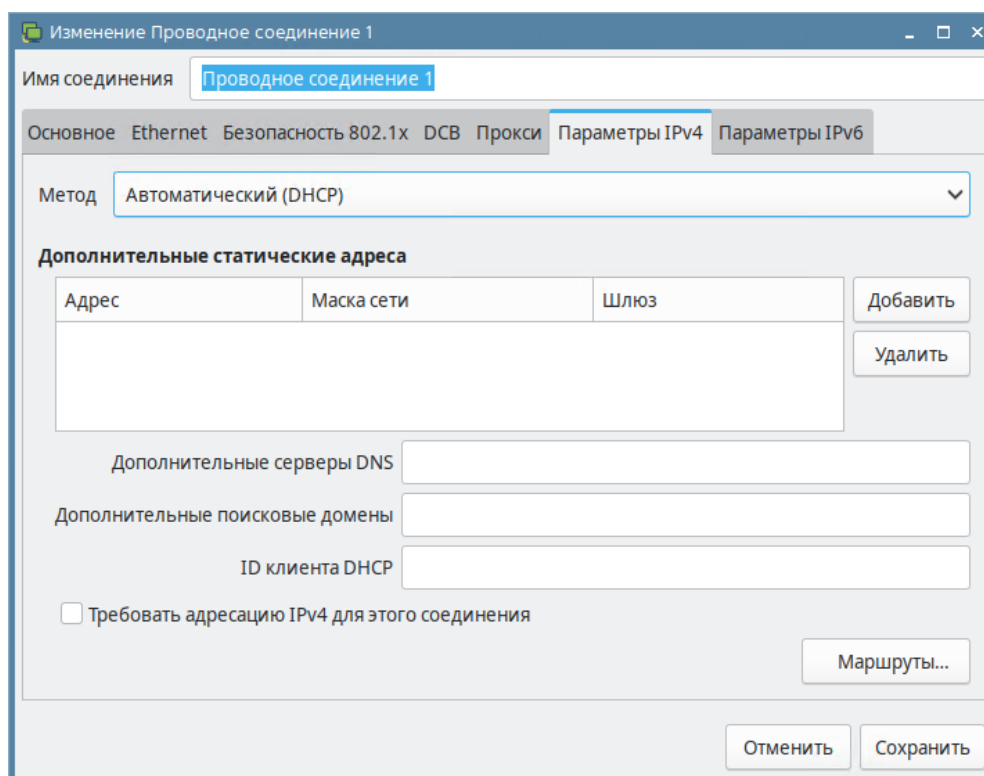


Figure 3.4 – IPv4 Parameters

4. The main settings here include:
- Method for obtaining an IP address

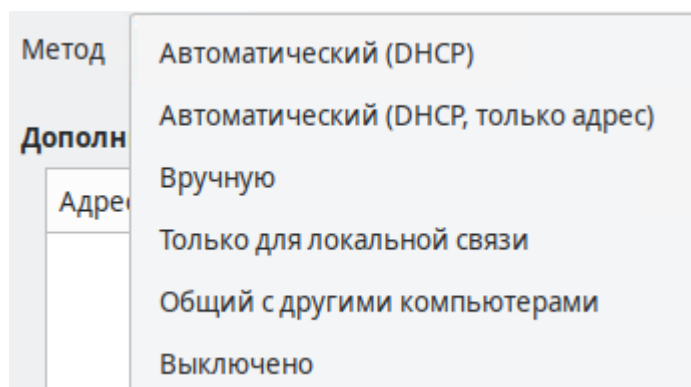


Figure 3.5 – Method for obtaining an IP address

- Static addresses
- DNS servers
- Search domains

### 3.2.3 Setting up a static address for a network connection

- Select the "Manual" IP address acquisition method.
- Click on the "Add" button.
- In the line that appears, in the “Addresses” section, enter:
  1. The IP address of this connection
  2. network mask
  3. Gateway IP address
- Specify the DNS server.
- Click the Save button.

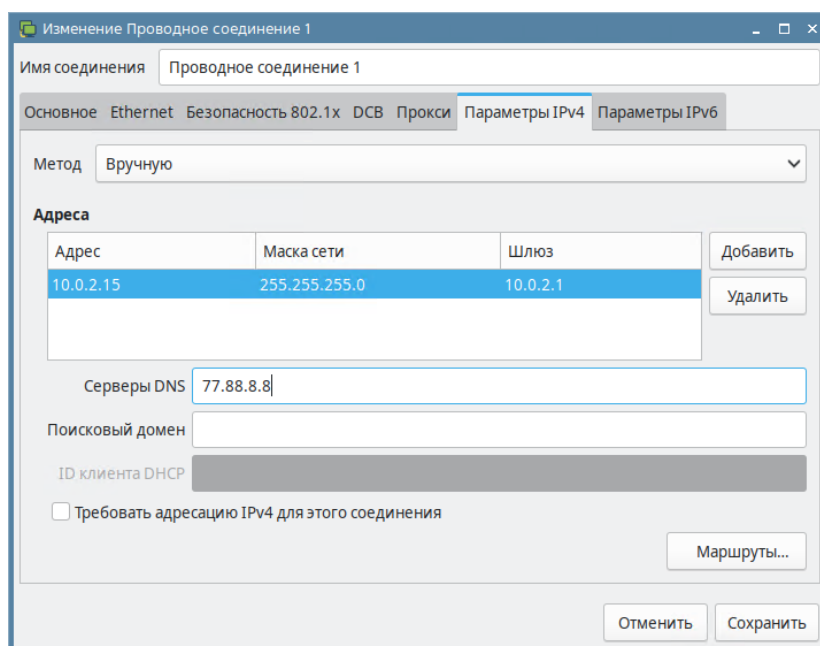


Figure 3.6 – Setting up a static address for a network connection

To restart the network connection and apply the settings, right-click the network icon (Network Manager applet) in the taskbar, select "Enable network support" and repeat this action.

If everything is ok, you will see a notification that the connection has been established successfully.

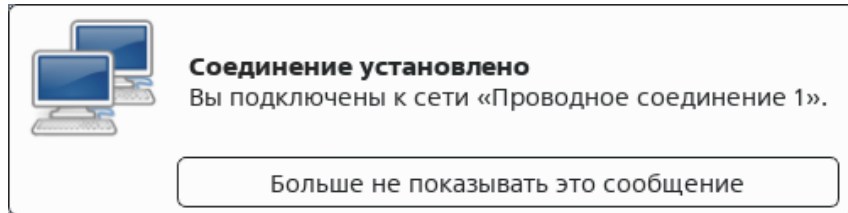


Figure 3.7 – Notification of a successfully established connection

To check the availability of resources from the internet, open the terminal using the Alt + T keyboard shortcut and enter the command `pingyandex.ru` . A successful ping will indicate the server is available and the name-to-IP address translation mechanism is working.

Press `ctrl + c` to complete the ping command .

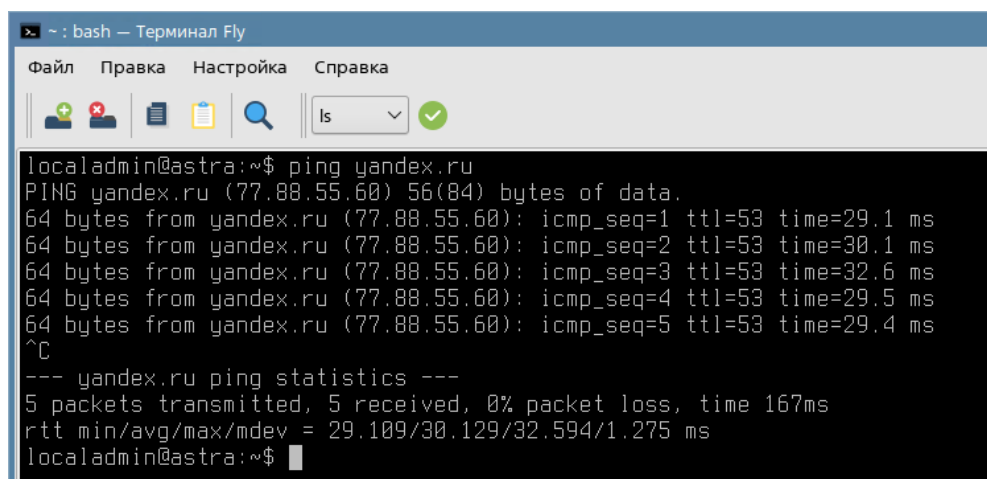


Figure 3.7 – Ping in the terminal

### 3.2.4 System update

Perform a system update, and to do this you need to go to the control panel (Figure 3.8):

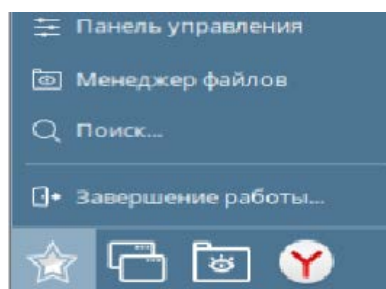


Figure 3.8 – Going to the control panel

In the control panel, go to System - > Check for updates (Figure 3.9).

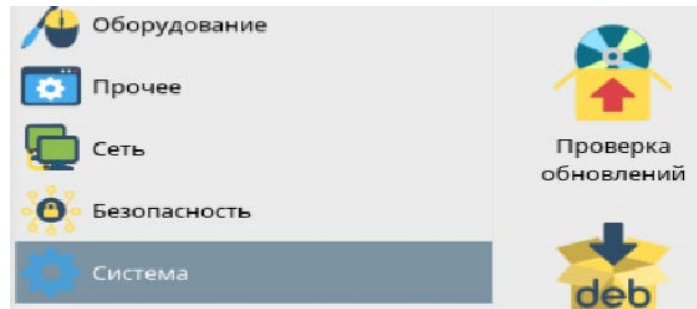


Figure 3.9 – Going to check for updates

We check for updates (Figure 3.10) and, if available, we update.

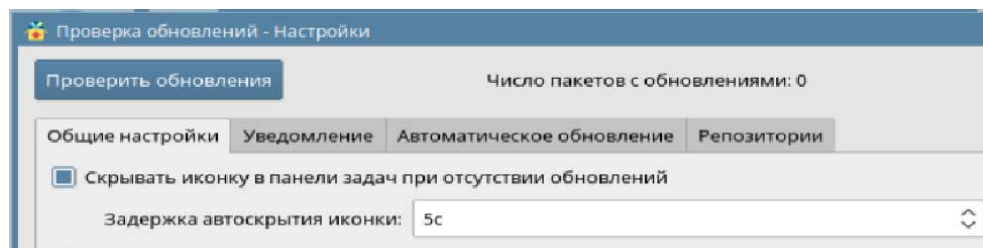


Figure 3.10 – Checking for updates

### 3.3 Assignment for independent completion

1. Configure the screen resolution, network connection and update the system in the Astra Linux operating system.

2. Take the test based on the lab materials <https://onlinetestpad.com/ijlmuw2awazhm>.

## 4 Working with the Fly graphical interface

### 4.1 Objectives of the work

Study:

1. Functioning of the graphics mode.
2. What is a desktop?
3. How to run Control Panel utilities from the command line and why they are needed.

### 4.2 Theoretical justification

#### 4.2.1 Login

After turning on your computer and loading the OS, you'll see the login window. While winlogon.exe is responsible for logging in Windows, its equivalent in Astra Linux is fly-dm\_greet, which is part of the fly-qdm package.

The Fly desktop environment is implemented on the basis of the X Window System (X11, X or simply "X", see Fig. 4.1), the terminology of which contradicts the generally accepted concept of client-server applications:

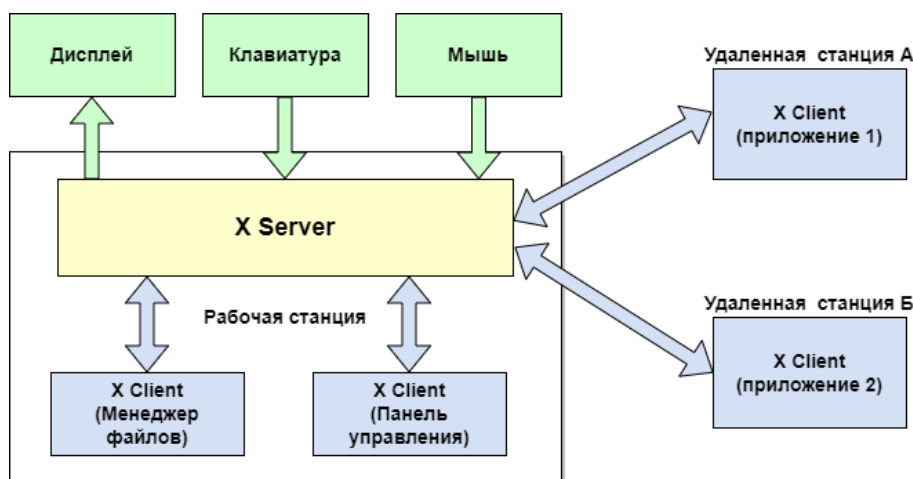


Figure 4.1 - X Window System Architecture

- **The "X Server"** (also known as the display server) is the first application launched when the graphics system starts. Its primary function is to interact with the graphics card, monitor, and various input devices. The server provides applications with a canvas of a specific size and transmits information about mouse movements, keyboard presses, and other events from input devices. In Astra Linux, the server functions are performed by the fly-dm (Fly Display Manager) application.
- **An "X Client"** is any program that uses a display server to display its interface on the screen. The program can be run locally (for example, fly-dm\_greet) or remotely, such as when administering a remote server from a graphics perspective.

The appearance of graphical elements and effects are determined not by the X

Server, but by the window manager, which in Astra Linux is called fly-wm (Window Manager for Fly Desktop). This application implements the desktop, the classic Start menu, the taskbar, the desktop switcher, and the screen locker.

The Astra Linux operating system allows you to change the appearance of the login window. This can be done using the fly-admin-dm utility ("Login"), which must be run as an administrator. A description of the utility is provided in the online help.

The online help can be accessed using the “Help” shortcut located on the desktop, or by pressing the hotkeys Alt + F1 or F1 in the active window of the graphical program.

Let's list the interface elements of the login window and their purpose, see Fig. 4.2:

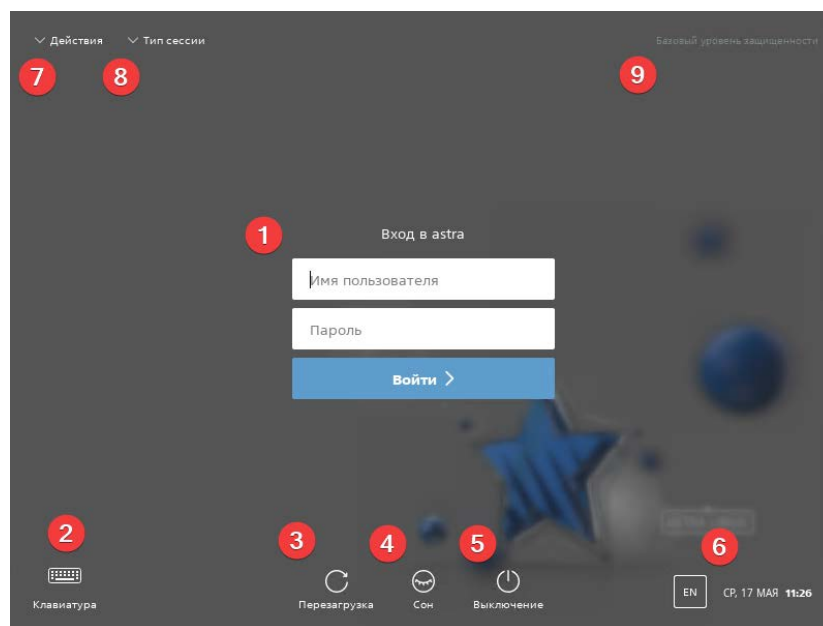


Figure 4.2 - Astra Linux login window

1. Username and password entry fields with a Login button .
2. On-screen keyboard call button.
3. Restart button – restarts the computer without asking for additional confirmation.
4. The Sleep button pauses the system after receiving confirmation. Data is stored in RAM, meaning the computer requires power in this mode.
5. Power button – turns off the computer without asking for additional confirmation.
6. Button for changing the language layout and the system time display area.
7. The actions menu contains the following commands:
  - **Switch session** – if you have multiple sessions running, you can use this action to switch to one of them.
  - **User Selection** – changes the login interface. Instead of a username entry field, a list of users who can log in is available (this list is configured using the fly-admin-dm utility – "login"). After performing the specified action, the name of this menu item changes to "Enter Username." Selecting this action switches the interface back to username and password entry mode (Figure 4.3).

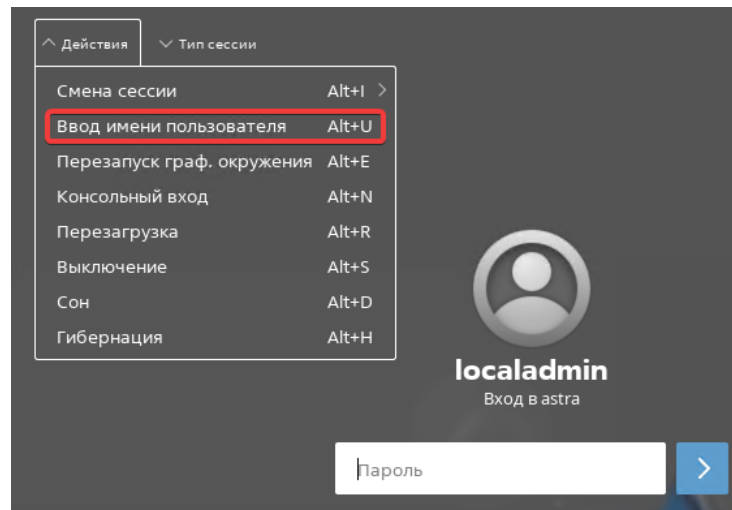


Figure 4.2 - User selection

- **Restarting the graphical environment** – restarts the Astra Linux graphical environment without a full system reboot (similar to rebooting Windows while holding down the **Shift** key ).
  - **Reboot** – reboots the computer without asking for additional confirmation.
  - **Shutdown** – turns off the computer without asking for additional confirmation.
  - **Sleep** – the system pauses after receiving confirmation. Data is stored in RAM, meaning the computer requires power in this mode.
  - **Hibernation** is the process of suspending the system after receiving confirmation. All data from RAM is saved to disk, so the computer does not require power in this state.
8. Session Type menu. By default, the system launches the most recently used session type. The following are available:
- **Recovery mode** – to restore the system via the terminal.
  - **Tablet** – adapted for devices equipped with a touch screen.
  - **Desktop** – for computers equipped with a keyboard and mouse.
9. Display area of the current OS security level.

Let's log in to the system using the account we created during the OS installation. Enter your username and password and click Login . If authentication is successful, you will be logged into the user's session.

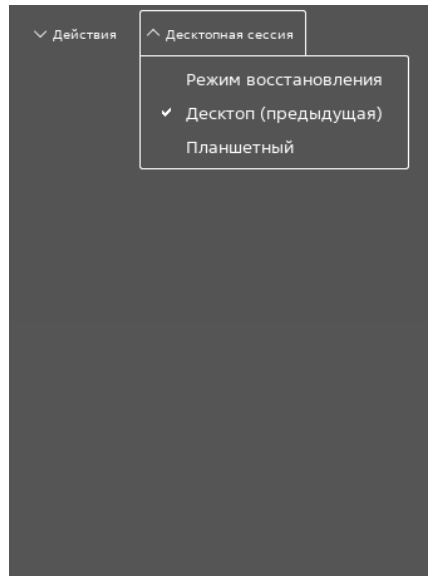


Figure 4.3 - Session Type

### 4.2.2 Desktop

While the Windows desktop is the explorer.exe application, in Astra Linux the Fly desktop functions are implemented in the fly-wm (Windows Manager) application, and the graphical management utilities are shells of the corresponding command-line utilities. The Astra Linux desktop interface is in many ways similar to Windows (Fig. 4.4): it has a start menu, a familiar taskbar format, and familiar context menus—all of which significantly simplifies user training when solving import substitution problems.

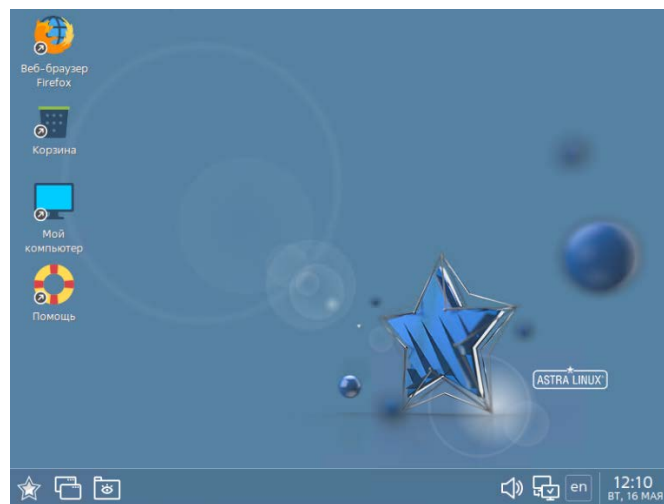


Figure 4.4 - Astra Linux desktop

First, let's set a comfortable desktop screen resolution. To do this:

1. Right-click an empty area of the desktop and select Monitor Settings... from the context menu . The Monitor Settings window (fly-admin-screen utility) will open, see Figure 4.5.
  2. Select the desired screen resolution.
- Apply the settings by clicking the Yes button , marked with an arrow in the

figure (the button text may be hidden if the program window does not fit on the screen), see Fig. 4.5.

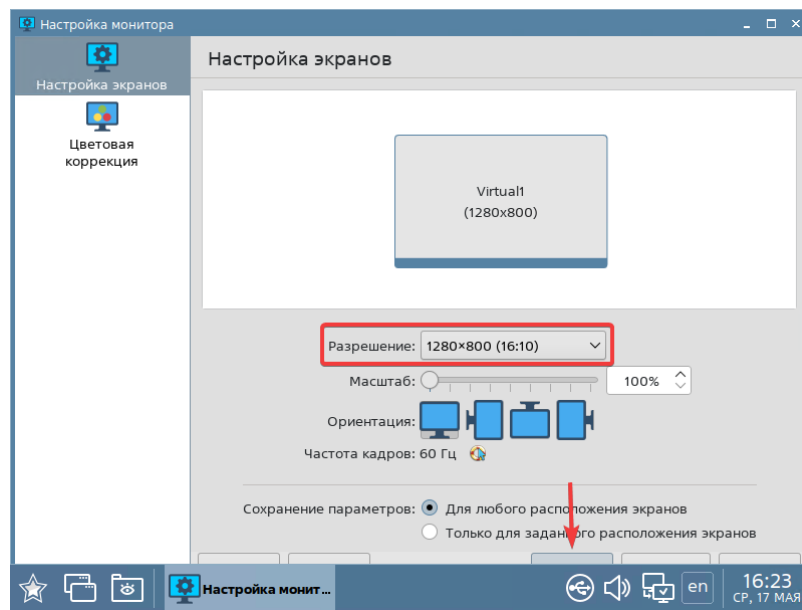


Figure 4.5 - Monitor settings

Now let's look at the purpose of the taskbar elements, see Fig. 4.5:



Figure 4.6 - Taskbar

1. Start menu button .
2. The Quick Launch bar. By default, it includes the window switcher and file manager.
3. Desktop switch button.
4. Task switching panel.
5. The notification area. By default, the KMix audio mixer, Network Manager applet, Keyboard layout, and Date and time are available here. When an external storage device is connected, a corresponding icon will appear.
6. The button that allows you to minimize all windows is located on the right side of the taskbar (marked with an arrow).

Start menu resembles the classic menu of the Windows operating system, see Figure 4.7 .

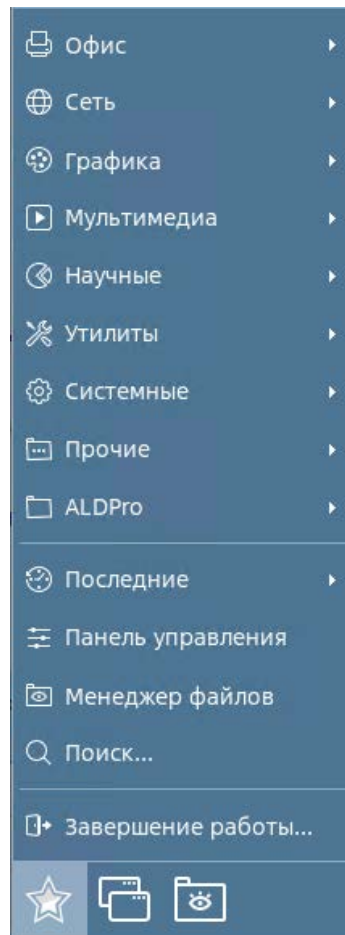


Figure 4.7 - Start Menu

Here we have access to thematic sections of applications and utilities, a section with recently launched programs, the ability to open the control panel, launch the file manager, search, and the shutdown menu. You can open the online help for the desktop by pressing **Alt+F1** , then select Documentation Desktop .

►►Graphics

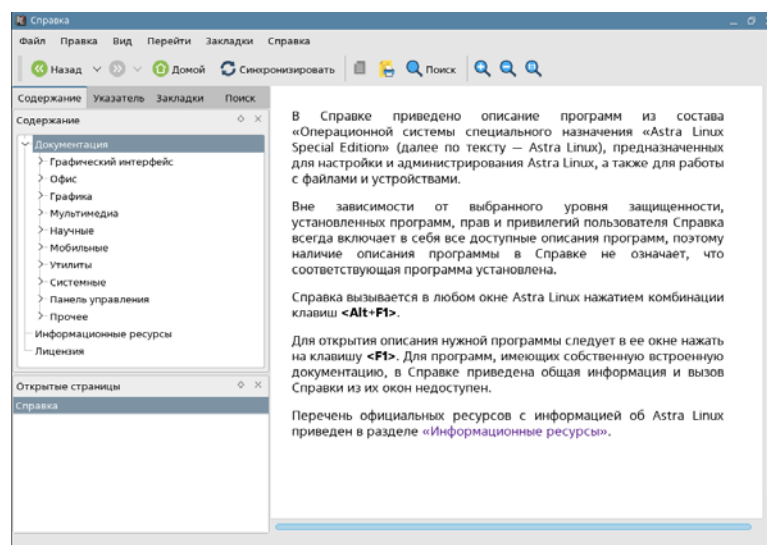


Figure 4.8 - Astra Linux Online Help

### 4.2.3 File Manager

The File Manager ( `fly-fm` ) program is designed for managing file system objects. It allows you to mount and unmount file systems of storage devices, such as local hard drives and their partitions, CD/DVD drives, and USB drives. It also allows you to access network SMB shares, work with archives, and manage disk encryption (protective encryption).

The Fly file manager allows you to work not only with the basic "User-Group-Other" permissions but also with extended POSIX ACLs, which is a significant advantage. The file manager window is similar to Windows Explorer, but note that the appearance settings are saved when you close the application, which can be confusing at first. To customize the appearance, follow these steps:

1. open only one window of the File Manager application;
2. Make any desired changes to the appearance settings, for example, enable the display of hidden files (menu View → Show hidden);
3. Close the application window to save the changes you made.

After completing these steps, you can continue working with the file manager, and in all subsequent windows, the display of hidden files will be enabled by default.

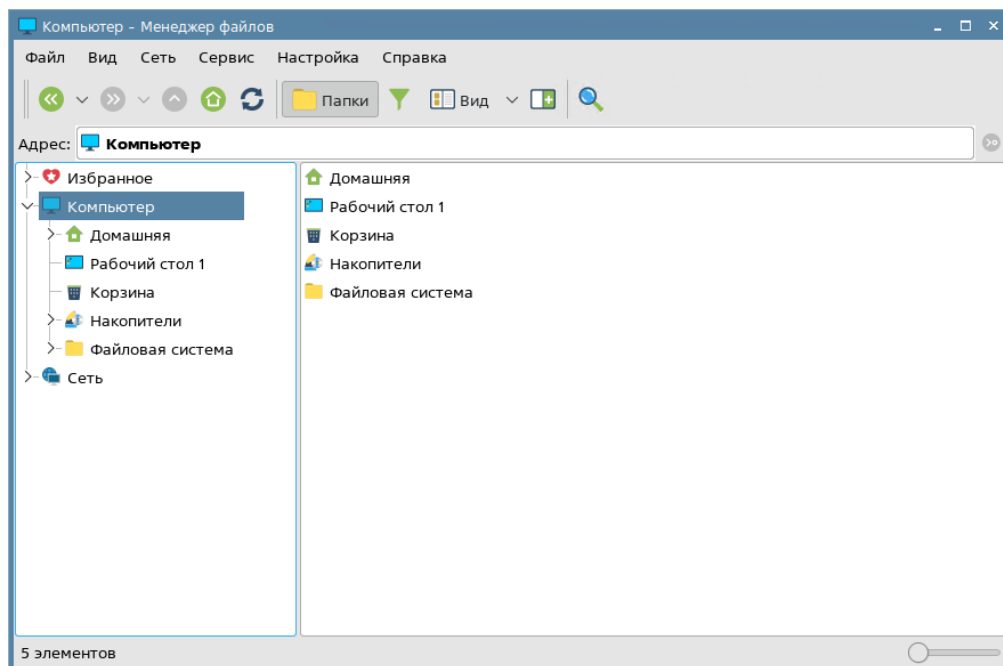


Figure 4.9 - File Manager program window

If a file or directory name begins with a period (.), it will be considered hidden in Linux. For example, a user's working directory contains files such as `.bashrc`, `.profile`, and others. To view hidden files from the command line, use the `ls` command with the `-a` switch: `ls -a .`

### 4.2.4 Desktop and Start Menu

Each user in the system has the ability to customize the display of the Start menu, its appearance, the arrangement of elements, and the features of working with

the keyboard and mouse.

The Start menu supports two display modes: the classic menu and the menu bar. By default, the classic menu is configured after OS installation. A full description of the Start menu display modes and how to switch between them is provided in the online help.

Some desktop settings are configured by the administrator and are not accessible to the average user. Some customization options can be implemented using the configuration utilities from the Start menu. The "Control Panel" menu category in Control Panel includes graphical utilities that can be used to customize the desktop. A full description of the Fly desktop, its operating modes and capabilities, as well as a description of the graphical programs, is provided in the online help.

### 4.2.5 System Monitor

Using the hotkey `ctrl + Alt + del`, you can open the System Monitor ( `ksysguard` ) application, which provides operational information about processes and the overall system load: CPU, memory, swap partition and network.

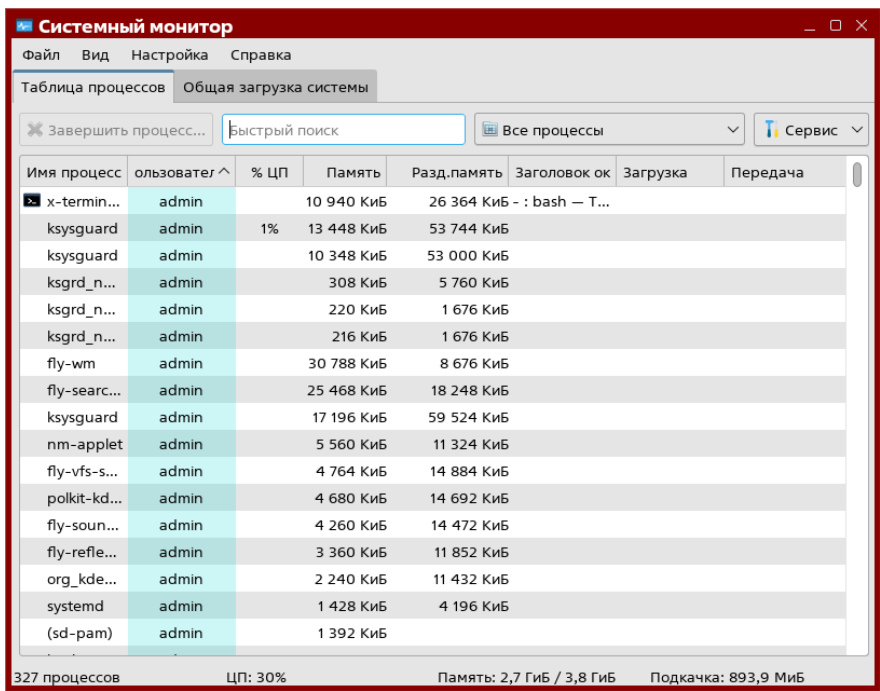


Figure 4.10 - Process Table tab

The application's interface is very similar to the Windows Task Manager. By default, two tabs open with preset names and update intervals: "Process Table" ( Figure 4.10 ) and "Overall System Load" ( Figure 4.11 ). However, you can create your own tabs with graphs using any available process and system load data. See File ▶ **New Tab** .

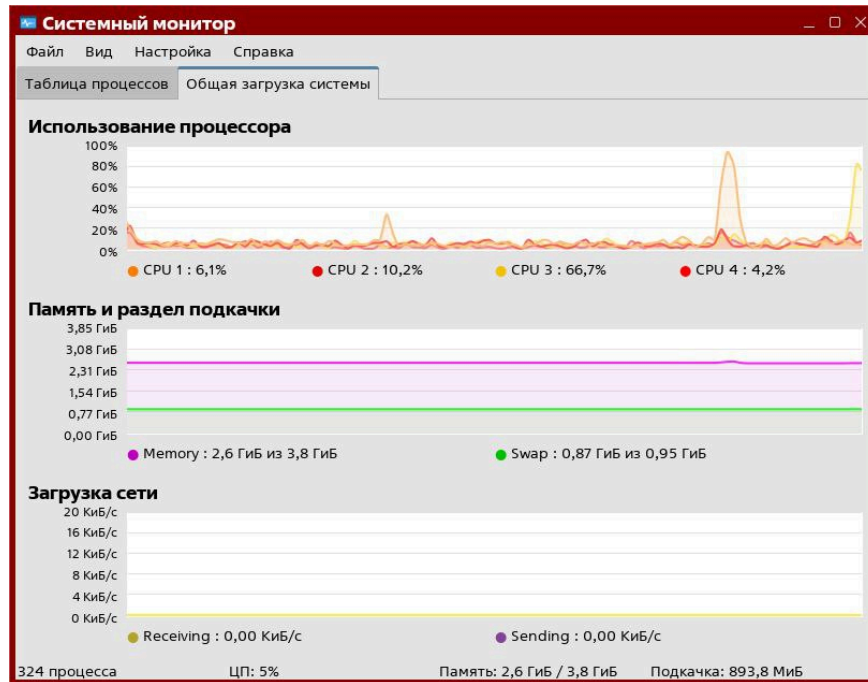


Figure 4.11 - The "General System Load" tab

## 4.2.6 Control Panel

The Control Panel ( `fly-admin-center` ) is accessible via the Start menu and is a graphical tool for navigating through utilities for managing users, devices, and other system settings, including security aspects, see Figure 4.12 .

► Control Panel

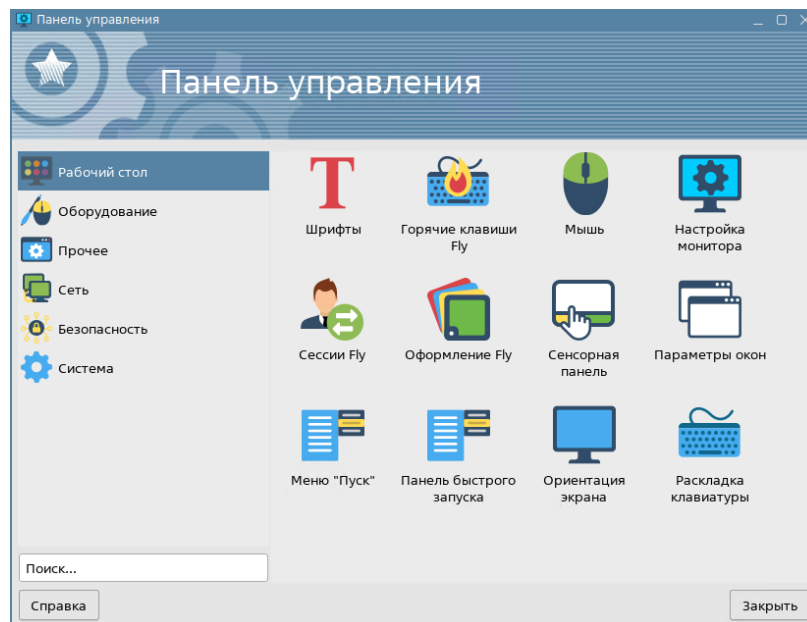


Figure 4.12 - Control Panel window

Control Panel utility shortcuts are `*.desktop` files located in the `/usr/share/applications/` folder.

- Desktop
  - Fonts ( `fly-admin-fonts` )
  - Hotkeys ( `fly-admin-hotkeys` )

- Mouse ( fly-admin-mouse )
- Monitor settings ( fly-admin-screen )
- Sessions ( fly-admin-session )
- design ( fly-admin-theme )
- Touchpad ( fly-admin-touchpad )
- Window options ( fly-admin-winprops )
- Start Menu ( fly-menuedit~/.fly/ startmenu )
- Quick Launch Toolbar ( fly-menuedit~/.fly/toolbar )
- Screen orientation ( fly-orientation )
- Keyboard layout ( fly-xkbmap )
- Equipment
  - Device Manager (fly-admin-device-manager)
  - Formatting external media (fly-admin-format)
  - Power supply (fly-admin-power)
  - Printers (fly-admin-printer)
  - Hot Plug Handling (fly-admin-reflex)
  - Remote USB Drive Service (fly-admin-usbip)
  - Adjusting Fly Brightness
- Other
  - Repository editor (fly-admin-repo)
  - TeXdoctk (texdoctk)
- Net
  - Network Settings (fly-admin-network)
  - Configuring a firewall (gufw)
  - Network Connections (nm-connection-editor)
- Safety
  - Setting up system event logging (fly-admin-events)
  - System Integrity Check (fly-admin-int-check)
  - System kiosk (fly-admin-kiosk)
  - PolicyKit-1 Sanctions (fly-admin-policykit-1)
  - Security Policy (fly-admin-smc)
  - System Event Viewer (fly-event-viewer)
  - Audit log (ksystemlog)
  - Audit Configuration (system-config-audit)
  - Change Password (fly-passwd) is a very useful graphical password change utility that works for both local users and FreeIPA/Samba AD domain users. If you install the system with default settings, you may need to additionally install the fly-passwd package.
- System
  - Manage Drivers (fly-kcmshell5 --desktopfile driver driver)
  - System Alternatives (fly-admin-alternatives)
  - Autostart (fly-admin-autostart)
  - Task Scheduler (fly-admin-cron)
  - Date and time (fly-admin-date)
  - Login (fly-admin-dm)

- o Environment variables (fly-admin-env)
- o bootloader (fly-admin-grub2)
- o Multi-terminal mode (fly-admin-multiseat)
- o Time synchronization (fly-admin-time)
- o Installing updates (fly-astra-update)
- o File Type Apps (fly-mimeapps)
- o System Reporting Center (fly-sosreport)
- o System initialization (systemdgenie)
- o Synaptic Package Manager (synaptic-pkexec)

#### 4.2.7 Other utilities

Below is a complete list of packages starting with `fly-*`. Please note that not all of the listed utilities are installed by default during OS installation.

A description of the utilities is available in the online help. You can access the online help by pressing `F1` in the active graphical utility window.

`fly-admin-autostart` – «Autostart»

Managing autostart of programs, automatic opening of files and directories when entering a session

`fly-admin-dm` – "Login"

Setting up a graphical login session

`fly-admin-date` – «Date and time»

View time and calendar, customize date and time display format

`fly-admin-time` – "Time Synchronization"

Setting up time synchronization

`fly-admin-grub2` – "GRUB2 Bootloader"

Graphical utility for configuring the GRUB2 OS bootloader

`systemdgenie` – System Initialization

Managing the Systemd system init service

`synaptic` – "Synaptic Package Manager"

Synaptic graphical package management utility

`gufw` – "Configure Firewall"

Configuring the UFW (Uncomplicated Firewall) firewall

`fly-admin-reflex` – "Hot Plug Handling"

Configuring actions performed when devices are connected

`fly-admin-env` – "Environment Variables"

Adding, changing, and removing environment variables

`fly-admin-cron` – Task Scheduler

Setting up a task schedule to run in the background, setting up the task execution environment (environment variables), allowing or disabling the execution of an already set task

`fly-admin-smc` – "Security Policy"

Managing local security policies. To import and export local non-personalized security policy settings (not related to user or group settings), use the `astra-adminctl` command-line tool (for more information, see the `astra-adminctl -h` help file).

fly-mimeapps - "File Type Apps"

View available apps and set the default app for file types

fly-admin-printer - «Printers»

Managing printers, setting up printing, and managing print jobs

fly-admin-policykit-1 - « PolicyKit-1 Sanctions »

View, grant, and revoke permissions to perform privileged actions managed using PolicyKit-1

PolicyKit is a tool for defining and enforcing rules for how unprivileged processes can access privileged ones, which is opposed to using utilities like sudo, but instead of granting administrative rights to a user's process, it allows precise control over what is and is not allowed.

fly-admin-session - « Fly Sessions »

Configuring login and logout settings for user sessions

nm - connection - editor - "Network connections"

Setting up network connections

fly-admin-network - « Parameters networks »

Managing network service startup

fly-admin-alternatives - "System Alternatives"

Managing the alternatives system allows several programs performing similar functions to register as alternative applications, one of which will be used by default. For example, when running the visudo utility to securely edit the sudoers file, the program specified in the editor alternative is used. By default, this is nano, but you can install vim.basic if you prefer that editor.

fly-admin-kiosk - "System kiosk"

Managing user environment restrictions. Using the Fly kiosk, you can configure the operating system to launch a single application (such as Firefox) without a desktop at boot, allowing you to create self-service kiosks.

hp-setup - "Installing HP Printers, Faxes, and Scanners"

Installing new Hewlett-Packard devices

hp-plugin "Installing an additional HP plugin"

Installing Hewlett-Packard drivers

fly-admin-power - "Power Supply"

Setting up and managing power and energy saving settings

fly-admin-service - «Services»

Status and configuration of services, their start and stop, automated configuration of services for operation with authentication in EPP and PAM mode

kssystemlog - "View Kssystemlog system logs"

Viewing the Extended Logging System log

system-config-audit - "Audit Configuration"

System audit settings

fly-sosreport - System Reporting Center

Collection of data on the system configuration and operation of subsystems for subsequent diagnostics

fly-run - "Run an application"

Launching a program or accessing a resource from the command line, including on behalf of a different user and/or with different capability attributes

**sudoafick-tk - File Integrity Checker**  
 A graphical utility for monitoring changes in the integrity of file system objects, afick

**fly-admin-device-manager - « Device Manager »**  
 View available devices, configure their drivers and parameters

**fly-admin-usbip - "Remote USB Storage Service"**  
 Providing remote access to USB drives and tokens using USB-over-IP

**fly-admin-format - "Formatting external storage"**  
 Deleting data and formatting external storage and its partitions

**fly-admin-iso - "Burn ISO image to USB drive"**  
 Program for burning ISO image to USB drive

**fly-admin-int-check - "System Integrity Check"**  
 System Integrity Check for Fly Desktop

**fly - admin - marker - "Marker Editor"**  
 View and change print marking settings, edit marker templates

**fly-print-station - Document Printing Management**  
 Marking and printing of restricted documents with the ability to move jobs to another printer

**sudogparted "Gparted Partition Editor"**  
 Creating, redistributing, or deleting OS system partitions

**ksysguard - System Monitor**  
 View information about processes and overall system load: CPU, memory, swap space, and network

**fly-term - "Fly Terminal"**  
 A terminal emulator that allows you to interact with the console.

**mc - " Midnight File Manager" commander »**  
 Viewing folders and FS elements, performing basic file management functions, accessing network resources, and working with archives

**kgpg "KGpg Key Management"**  
 GPG (GNU Privacy Guard) key management. GnuPG is a free program for encrypting information and creating digital signatures.

**fly-admin-ald-server "Domain Security Policy"**  
 Setting up the ALD domain controller service (the company is currently developing the ALD Pro product based on the FreeIPA directory service, and the ALD application is not recommended for deployment)

**fly-admin-ald-client - « Configuring the Fly ALD client »**  
 Adding a client computer to the ALD domain (the company is currently developing the ALD Pro product based on the FreeIPA directory service, and the ALD application is not recommended for deployment)

**fly-admin-ad-client - « Fly Active Directory Client Configuration »**  
 Joining a Client Computer to a Windows AD Domain

**fly-admin-ad-server « Configuring the Fly Active Directory**

Server »

AD Domain Controller service

fly-admin-ad-sssd-client - « Configuring the SSSD Fly Client »

Adding a client computer to an existing Windows AD domain, which will involve the SSSD (System Security Services Daemon) authentication and authorization management service.

fly-admin-freeipa-server - « Fly FreeIPA server setup »

Installing and configuring the FreeIPA server

fly-admin-freeipa-client - « Configuring the FreeIPA client Fly»

Adding a client computer to an existing FreeIPA domain

fly-admin-multiseat - "Multi-terminal mode"

Preparing your computer for simultaneous use by multiple users

fly-admin-dhcp - "Configure a DHCP server"

Setting up a DHCP server

fly-admin-openvpn-server - "Setting up the Fly OpenVPN server"

Setting up a VPN server

fly-admin-ftp - "FTP server"

Setting up an FTP server

fly-admin-ntp - "Time synchronization (NTP)"

Setting up an NTP time server

fly-admin-samba - "Shared Folders (Samba)"

Managing Samba Shares

fly-passwd - Change Password

Allows you to change the password, and this can be either a local user or a domain user (ALD Pro, ALD, FreeIPA, SambaAD)

fly-su - "User substitution"

Executing a command on behalf of another user is a graphical wrapper over the su utility.

fly-astra-update - "Installing updates"

Update installer

fly-admin-repo - "Repository Editor"

Creating and managing repositories

fly-admin-driver - "Driver Management"

Installing and selecting graphics drivers

### 4.3 Assignment for independent completion

1. Enable display of hidden objects in File Manager.
2. Change the system administrator password through the control panel.
3. Using help, find information about the ksysguard package, run this application.
4. Explore the contents of the taskbar yourself.
5. Take the test based on the lab materials [at https://onlinetestpad.com/bake2unqj4hyu](https://onlinetestpad.com/bake2unqj4hyu).



## 5 Working with the terminal and the Bash shell

### 5.1 Objectives of the work

Learn the order of entering commands for Astra Linux OS in terminal mode .

### 5.2 Theoretical justification

#### 5.2.1 System and virtual consoles

The user interface in Linux is called the console. Immediately after turning on the computer, the console displays the boot progress, and after the kernel initializes, the console begins providing access to several virtual consoles.

To switch between them, use the hotkey `ctrl + Alt + F<N>` , where `<N>` is an integer from 1 to 8, for example: `ctrl + Alt + F1` . Virtual consoles allow you to run multiple tasks in parallel and monitor system operation, which was one of Linux's killer features.

In Astra Linux, seven virtual consoles are initialized by default: `tty1 ... tty7`. Consoles one through six are text-based, while console seven runs graphics (Figure 5.1).

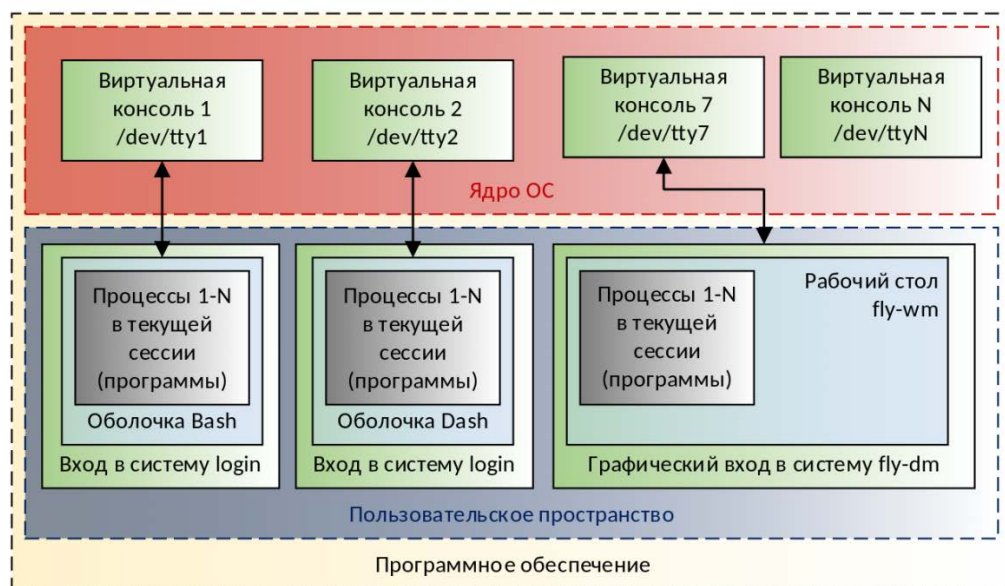


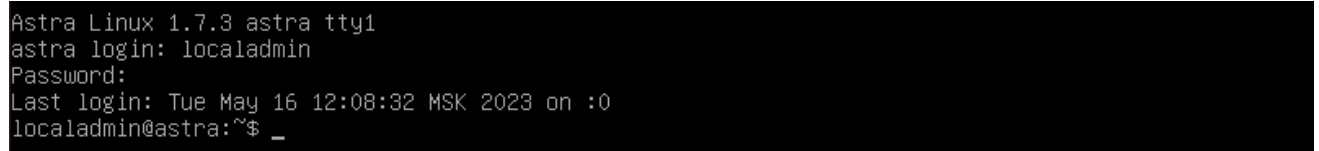
Figure 5.1 – Connecting virtual consoles to the kernel

Each virtual console in the system corresponds to a symbol file of the same name for that device in the `/dev` directory . For example, the `tty1` console corresponds to the file `/dev/tty1` . The `/dev/console` device is automatically bound to the active virtual console. The number of virtual consoles in the system can be controlled (up to a maximum of 63) using the `NAutoVTs` parameter in the `/etc/systemd/logind.conf` configuration file .

There's another way to quickly switch between virtual consoles. The keyboard

shortcut `Alt + <left arrow>` Or `Alt + <right arrow>` allows you to cycle through consoles in order. However, once you switch to the graphics console, this shortcut stops working, and to switch back to the text console, you must use `ctrl + Alt + <F1-F6>` again .

To operate virtual consoles on a Linux system, multiple instances of the `agetty` application are launched, each connecting to its own `tty` and providing a login prompt. After the user enters an account name, `agetty` launches the login program and passes it the entered value. The login application prompts the user for a password, verifies the credentials through the PAM stack, and, if authentication is successful, launches a Bash command shell for that user (Figure 5.2).

A screenshot of a terminal window showing a login prompt. The text displayed is: 'Astra Linux 1.7.3 astra tty1', 'astra login: localadmin', 'Password:', 'Last login: Tue May 16 12:08:32 MSK 2023 on :0', and 'localadmin@astra:~\$ \_'. The prompt is on a black background with white text.

```
Astra Linux 1.7.3 astra tty1
astra login: localadmin
Password:
Last login: Tue May 16 12:08:32 MSK 2023 on :0
localadmin@astra:~$ _
```

Figure 5.2 - Virtual Console

**A shell**, or command line interpreter, is a system application whose main purpose is to allow the user to run other applications.

There are a large number of different shells, the most popular of which is Bash (the default on Astra Linux). Other examples include Dash (a lightweight shell, accessible by running the binary at `/bin/sh` ) and Zsh. All shells are quite similar in syntax, so it's not always clear at first glance which shell is running.

Don't confuse the concepts of a virtual console (or simply console) and a shell. The console is implemented by kernel functions, while the shell is a regular user-space program. The following diagram shows the logic of interaction between virtual consoles, login utilities, shells, and application software (some simplifications are allowed).

Let's go to the virtual console `tty1` using the hotkey combination `ctrl + Alt + F1` and enter the credentials of the user that was created during the installation of the system (hereinafter, by "admin" we will mean this account).

`nanoscript` command , which will open the nano text editor (Figure 5.3). Enter the following text into the editor:

```
#!/bin/bash
pwd
whoami
```

Save the file by pressing `ctrl + o` and `Enter` .



Figure 5.3 - Nano text editor

Let's go to the virtual console tty2 by pressing `ctrl + Alt + F2` and log in as administrator.

Let's see what files and directories we have in the current directory using the `ls` command (Figure 5.4).

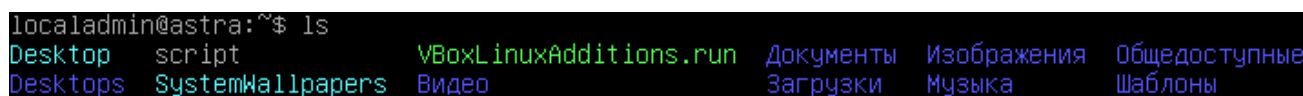


Figure 5.4 – Result of executing the ls command

As we can see, we now have a script file. Let's try running it by running the `./script` command, but... it won't work. The system will tell us we don't have permission. To run the script, we'll run `sudo chmod u+x script`, which will grant permission to execute the file, and then `./script`, which will actually run the script.

The script has run successfully, and two messages will be displayed on the screen (Figure 5.5):

1. current directory (in which it was launched)
2. and the account name (from which the script was run).

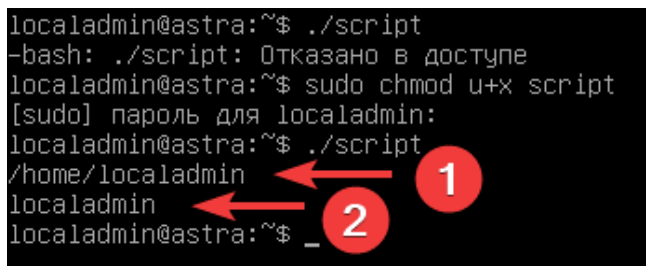
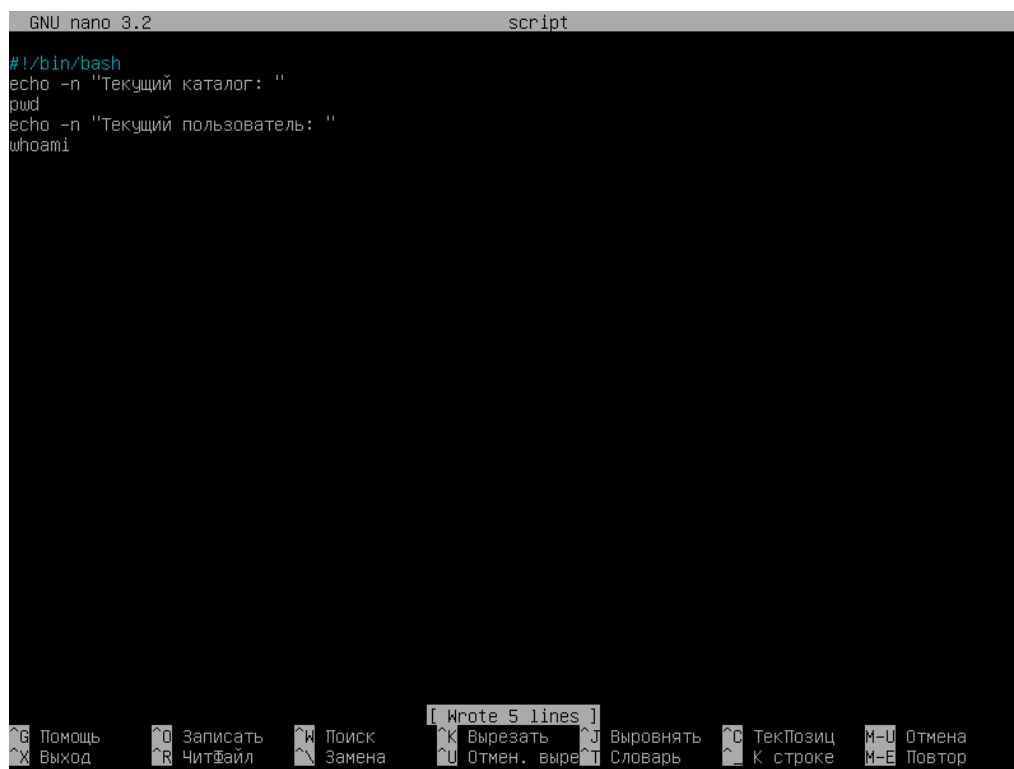


Figure 5.5 – Result of script execution

Let's switch to the first console tty1 using the combination Alt + <left arrow> , and change our script a little (Figure 5.6):

```
#!/bin/bash
echo -n "Current directory:"
pwd
echo -n "Current user:"
whoami
```

Save the file by pressing ctrl + o and Enter .



```
GNU nano 3.2 script
#!/bin/bash
echo -n "Текущий каталог: "
pwd
echo -n "Текущий пользователь: "
whoami
```

Wrote 5 lines

Помощь Выход Записать Читайл Поиск Замена Вырезать Отмен. выре Выводить Словарь ТекПозиц К строке Отмена Повтор

Figure 5.6 – Script in the first terminal

Let's switch to the second console tty2 using the combination Alt + <right arrow> , and run the script again by executing the command . /script (Figure 5.7):

```
localadmin@astra:~$ ./script
Текущий каталог: /home/localadmin
Текущий пользователь: localadmin
```

Figure 5.7 – Script in the second terminal

As you can see, the script output has changed as expected. This means that switching between consoles allows you to work effectively on Linux computers even without a graphical shell.

A user process can modify the behavior of any TTY device by manipulating the corresponding file in the /dev directory . Essentially, the tty[1-7] symbol files are not data files stored on disk; they are special device files that point to their drivers.

Therefore, when a process accesses the tty[1-7] file, it is essentially accessing the device driver. Naturally, this requires write access to the file. When a user logs in and connects to a specific TTY, that user must become the owner of the file corresponding to that TTY.

The login program changes the access rights to the terminal file, which in turn has the necessary privileges, since it runs as the superuser.

Let's go to the virtual console tty1 as administrator (Figure 5.8).

```
Astra Linux 1.7.3 astra tty1
astra login: localadmin
Password:
Last login: Wed May 24 20:45:35 MSK 2023 on tty2
localadmin@astra:~$ _
```

Figure 5.8 – Virtual console tty1 as administrator

Let's open the nano text editor (just type nano and press Enter). Go to the tty2 virtual console and log in as administrator (Figure 5.9).

```
Astra Linux 1.7.3 astra tty2
astra login: localadmin
Password:
Last login: Wed May 24 21:11:24 MSK 2023 on tty1
localadmin@astra:~$ _
```

Figure 5.9 – Virtual console tty2 as administrator

Let's run the command `ls -l /dev/tty[1-7]`, which will display detailed information about the files `/dev/tty1`, `/dev/tty2...`/`/dev/tty7`. We'll cover the meaning of this output in detail in separate modules of this course, but for now, we're interested in the highlighted column. It contains information about the owner of this file.

As you can see, the first two virtual consoles are owned by localadmin, the third through sixth are owned by the superuser root, and the owner of the seventh console, which runs the graphical interface, is the service user fly-dm (Figure 5.10).

```
localadmin@astra:~$ ls -l /dev/tty[1-7]
crw----- 1 localadmin tty 4, 1 мая 24 21:11 /dev/tty1
crw----- 1 localadmin tty 4, 2 мая 24 2023 /dev/tty2
crw--w---- 1 root      tty 4, 3 мая 20 08:58 /dev/tty3
crw--w---- 1 root      tty 4, 4 мая 20 08:58 /dev/tty4
crw--w---- 1 root      tty 4, 5 мая 20 08:58 /dev/tty5
crw--w---- 1 root      tty 4, 6 мая 24 20:38 /dev/tty6
crw--w---- 1 fly-dm    tty 4, 7 мая 20 08:58 /dev/tty7
```

Figure 5.10 – Information about files `/dev/tty1`, `/dev/tty2...`/`/dev/tty7`

Now let's do the magic trick. Staying in the tty2 console, enter the command (Figures 5.11-5.12):

```
echo "Hello from console tty2">/dev/tty1
```

```
localadmin@astra:~$ echo "Привет из консоли tty2" > /dev/tty1
localadmin@astra:~$
```

Figure 5.11 – Entering a command in the tty1 console as administrator

Let's go to the virtual console tty1 and...

```
GNU nano 3.2                                Новый буфер
Привет из консоли tty2
-
```

Figure 5.12 – Result of executing the command in the tty2 console

We will see the text we transmitted.

The command we entered redirected the text "Hello from console tty2" to console tty1 via the console file `/dev/tty1` . However, if we switch back to console tty2 and try to send text to console tty3, we'll get a message saying we don't have permission to do so (Figure 5.13) because we're not the owner of the file `/dev/tty3` .

```
localadmin@astra:~$ echo "Привет из консоли tty2" > /dev/tty3
-bash: /dev/tty3: Отказано в доступе
```

Figure 5.13 – Message in the tty3 console

### 5.2.2 Terminal emulator

As the years passed, the first graphical shells for Linux began to appear in the late 1990s. To ensure backward compatibility and the ability to run console applications from a graphical shell, specialized applications— **terminal emulators** (fly-term, xterm, ssh, telnet)—were developed.

Their purpose is exactly the same as that of virtual consoles: to deliver user input to a running program and display the program's output on the screen. However, the operating mechanism is fundamentally different, as the terminal emulator itself operates in user space, while backward compatibility of applications required kernel-level mechanisms. For this purpose, a software device called a **pseudo-terminal** (PTY, acronym for *Pseudo-TTY* ) was invented. Now that hardware terminals no longer exist, the term "terminal" refers specifically to a pseudo-terminal.

Linux doesn't limit the number of pseudo-terminals in the system and creates them upon request from terminal emulators. A pseudo-terminal can be launched within another pseudo-terminal (for example, by the screen utility or the ssh client). Each pseudo-terminal in the system is assigned a dynamic number, and a file with the same name is created in the `/dev/pts` directory . For example, the file `/dev/pts/2` is the file for pseudo-terminal number 2.

The terminal emulator can be launched from the graphical interface using the `Alt + T` keyboard shortcut or from the Start

can create multiple tabs in a single Fly terminal window, each with its own separate pseudo-terminal (Figure 5.14). You can create a tab using the corresponding button in the application window or by pressing the `ctrl + T` keyboard shortcut . You can increase or decrease the font size using the `ctrl + plus/minus` keys or by scrolling the mouse wheel while holding down the `ctrl` key .

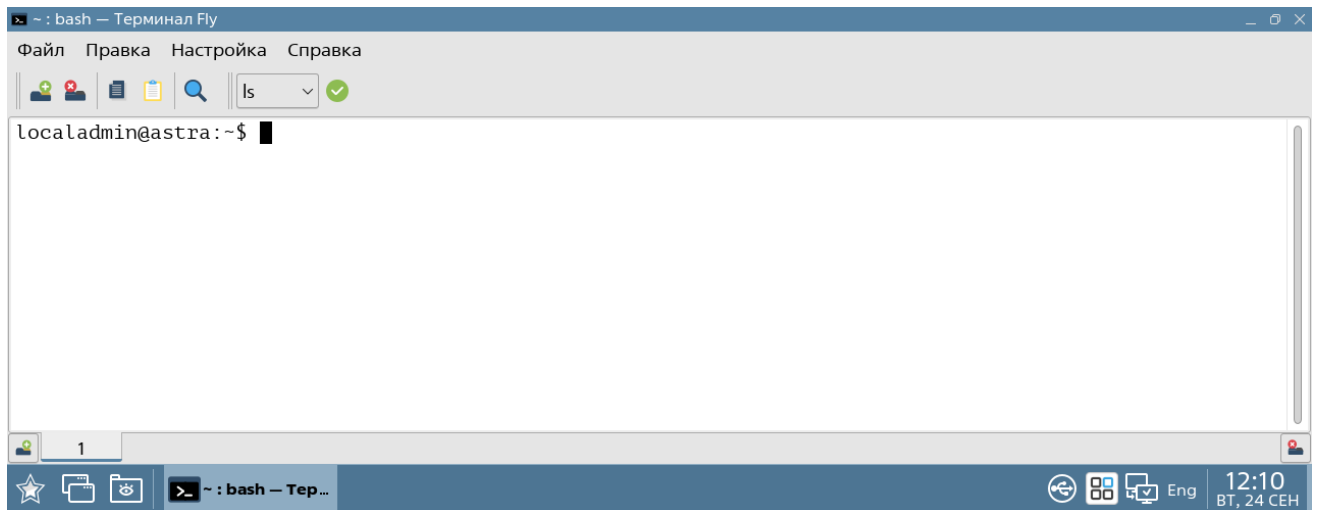


Figure 5.14 – Pseudo-terminal number 2

You can view and change the terminal display settings through the Settings menu . For example, to view and change hotkeys, select Settings Shortcuts...

► Keyb

From now on, the term "**terminal**" will refer to the fly-term terminal emulator window .

The diagram (Figure 5.15) below shows the logic of interaction between programs and shells when working in a terminal emulator (via a pseudo-terminal).

To find out the number of the pseudo-terminal that your terminal is working with, simply enter the `tty` command .

```
tty
/dev/pts/0
```

There are several special commands that the terminal handles in a unique way. Anyone who has worked in Windows is familiar with the `ctrl + c` shortcut , which terminates a program. In Linux, it works exactly the same way.

Commands that are called by holding down the `ctrl` key are usually denoted by the `^` symbol , followed by the name of the key, that is, the hotkey combination `ctrl + c` is the same as `^c` .

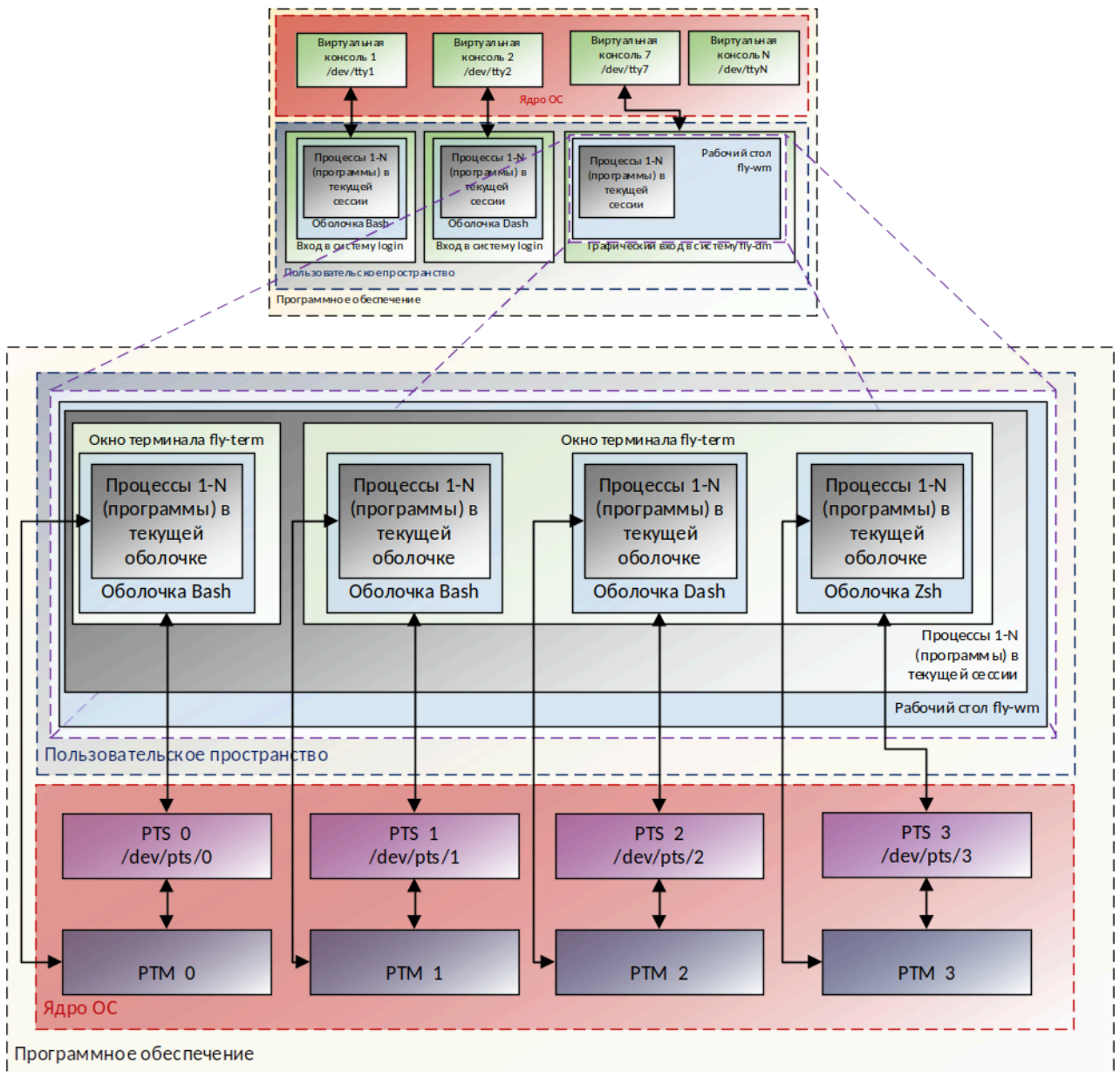


Figure 5.15 – Supplemented diagram showing the operation of terminal emulators and pseudo-terminals

**Example:** Let's enter the command `du/` , which displays the size of all directories in the file system. This is a slow process, so if we don't want to wait for it to complete, simply press `ctrl + c` , and the `SIGINT` signal will terminate the command.

```
du/
...
0 /proc/391/task/391/fd
0 /proc/391/task/391/fdinfo
du: unable to read directory '/proc/391/task/391/ns': Permission denied
0 /proc/391/task/391/ns
0 /proc/391/task/391/net/stat
^C
```

It's also useful to know what the `Ctrl + Z` keyboard shortcut does , as it's often used out of habit to undo the last action in Windows. If you use this keyboard shortcut while editing text in the nano editor, the application will disappear from the screen,

and you'll see the message *"Use 'fg' to return to nano . "*

In fact, the `ctrl + z` shortcut sends the process to the background, and to bring it back to the foreground, you just need to enter the `fg` command . If there are multiple background applications, `fg` will bring the last one back to life, but you can also call `fg` with an argument and explicitly specify the number of the desired application.

**Example:** Enter the command `nanotest.txt` and write something like "Hello world!" in the editor (Figure 5.16).

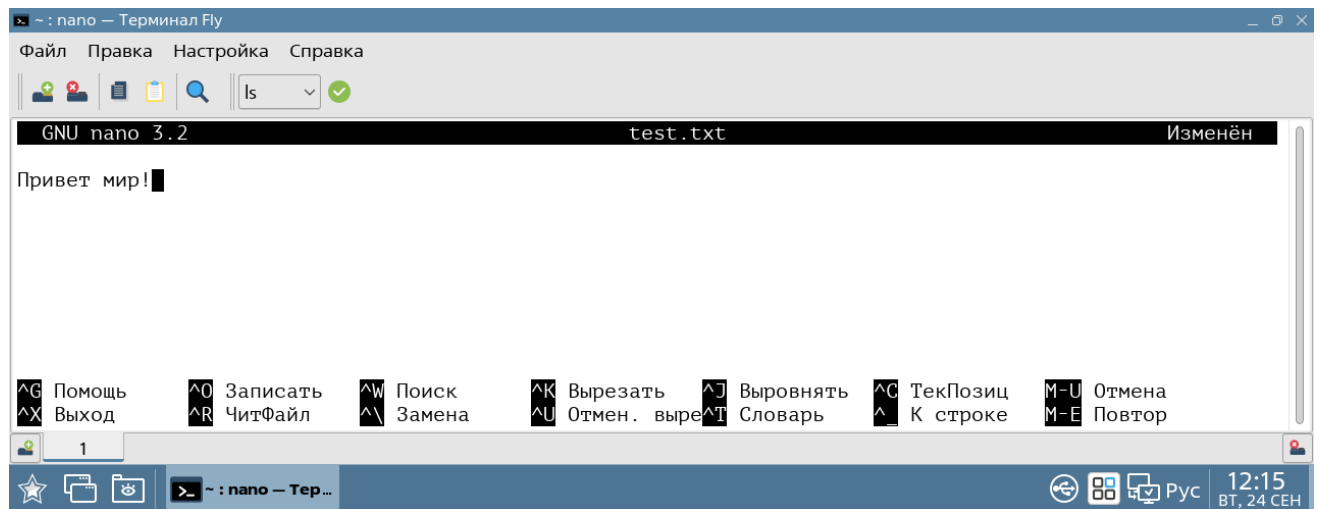


Figure 5.16 – “Hello world!” in nano editor

Now we press the key combination `ctrl + z` and we will get a message that the nano process has been stopped.

```
nanotest.txt
Use "fg" to return to nano
[1]+ Stopped nano test.txt
```

Using the `fg` command , we'll bring the nano process, previously sent to the background, back into interactive mode. As we can see, the text we entered is still there, meaning it's the same process.

### 5.2.3 Bash shell

When you open a terminal, the command shell specified in the `/etc/passwd` file is launched . In modern distributions, the default shell is `/bin/bash` . A list of all shells installed on the system can be found in the `/etc/shells` file .

```
cat /etc/shells
/etc/shells:valid login shells:
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/bin/dash
/usr/bin/dash
```

The values given correspond to the following command shells:

- **/bin/sh** – Bourne Shell, the oldest UNIX shell, created by Steve Bourne in 1979, which is now rarely used. The file `/bin/sh` is a symbolic link to `/bin/bash` or `/bin/dash` , depending on the distribution (on Astra Linux , `/bin/sh` is a

symbolic link to `/bin/bash` ). Invoking `/bin/sh` starts bash in POSIX-compatible mode (with the `--posix` option ).

- **`/usr/bin/bash`** – BourneAgain Shell, an enhanced version of the GNU `sh` command interpreter, which is currently one of the most widely used shells on Linux. This interpreter is used by default on Astra Linux.
- **`/bin/rbash`** is bash, but with some limitations. It is run as `rbash` or `bash` with the `-r` parameter . In this shell, changing directories is not allowed, for example, and changing the `SHELL`, `PATH`, `ENV`, or `BASH_ENV` environment variables is not possible. This mode is described in more detail in the bash man page (section "Restricted Command Interpreter").
- **`/bin/dash`** is a POSIX-compliant implementation of the Bourne Shell. It takes up only 112 KB of disk space, compared to bash's 1.1 MB. Compared to bash, it has fewer features for interactive work on the command line and is more often used in scripts due to its smaller size and faster execution speed.

When we open the terminal, we see a command line with an input prompt.

Here's what the invitation line can tell us:

| Part       | Meaning                                                                           |
|------------|-----------------------------------------------------------------------------------|
| localadmin | the current user under which the command shell is running                         |
| @          | separator, as in email                                                            |
| astra      | computer name                                                                     |
| :          | name and working directory separator                                              |
| ~          | current directory (the ~ symbol denotes the user's home directory)                |
| \$         | operating mode:<br>\$ - with simple user rights<br># - with root superuser rights |

The command line prompt text can be customized. It is generated based on the settings in the user profile.

In command shells, we can execute commands, most of which are called as external utilities, but some commands are still built into the shell itself.

For example, the `echo` command is a built-in command by default, so when we execute `typeecho` we will see "echo is a bash built-in command".

But if you disable this built-in command `enable-necho` , then the `type` command will output "echo is /usr/bin/echo", i.e. an external utility will be called to execute `echo` , see example:

```
typeecho
echo - is a built-in bash command
enable - n echo
typeecho
echo is /usr/bin/echo
```

A list of all built-in commands available in the shell can be viewed using the `help` command .

Let's look at the syntax of Bash commands:

|                |                             |                    |
|----------------|-----------------------------|--------------------|
| ls             | -l                          | /home/localadmin   |
| ключевое слово | [набор параметров (ключей)] | [набор аргументов] |

Figure 5.17 – Bash command syntax

- **The keyword** may contain the path to the utility's binary file. For example, `ls` and `/usr/bin/ls` are the same. If the full path is not specified, the shell will search the system directories specified in the `PATH` variable.
- **Parameters and arguments** are optional and may be omitted altogether. Three different syntaxes are used to specify keys in Linux:
  - **POSIX syntax** : a single hyphen precedes each switch, and the switch itself is a lowercase or uppercase Latin letter. Switches can be grouped by specifying multiple switches without spaces after a single hyphen. For example, `ps -aux`.
  - **BSD syntax** – no hyphen before the key. For example, `psaux`.
  - **GNU syntax** uses long switches preceded by a double hyphen. For example, `ps--sortcmd` . Long switches may duplicate short switches, but there are often no equivalents between them.
  - **But there are exceptions** to these rules, for example, the `find` command uses the following keys: `-name` , `-type` , `-size` , etc.
- **Switches and arguments** are case-sensitive, meaning "-a" and "-A" are two completely different switches. This is where Linux differs significantly from Windows, which can be quite confusing for many at first.
- **Extra spaces** in commands are ignored.

The `$PATH` environment variable contains a list of directories in which the system will search for an executable file. This mechanism works similarly in Windows.

```
echo " $PATH "
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Let's take a look at some of the most commonly used commands:

- `whoami` (Who am I?) - the command displays the current user under which the terminal is running:

```
whoami
localadmin
```

- `echo` (`echo`) - a command to display a line of text:

```
echo "Hello"
Hello
```

- `pwd` (Print Working Directory) is a command to display the current directory:

```
pwd
/home/localadmin
```

- `cd` (Change Directory) is a command for navigating through a directory tree. We'll use `cd/` to navigate to the root of the file system, and `cd-` to return to the directory we were in:

```
cd /
pwd
/
```

```
cd -  
/home/localadmin
```

When launched, the shell initializes *the user's environment* - a memory area that stores a set of variables used to control programs.

To initialize the environment, the bash shell looks for and runs commands from certain files located in the user's home directory and in the system directory /etc.

- **Interactive login shell mode** . When a user connects to a remote machine via SSH, logs in to the machine through a local console, or runs the `su-` command , after entering the login and password, the bash shell is launched, which examines instructions from files in the following order: `/etc/profile` , `~/.bash_profile` , `~/.bash_login` , `~/.profile` .  
`~/.bash_profile` and `~/.bash_login` files are missing immediately after installation .

- **Interactive non -login shell**. If the user is already in the graphical shell or text console, opening a new terminal window starts the bash shell, which reads and executes instructions from the `~/.bashrc` file .
- **Non-interactive login shell mode** . When running a bash script, the shell looks at the `BASH_ENV` environment variable and executes the instructions found in the specified file.
- **Non-interactive mode without login** (non-interactive non-login shell). If you run the `bash -c command` command , in this mode, instructions from the file specified by the `BASH_ENV` environment variable are also executed.

Variables in Bash can be divided into three groups:

1. **environment variables** (environment variables, global variables)
2. **user variables**
3. **special variables** (positional)

Environment variables are set in the files `/etc/.profile` , `~/.profile` , `~/.bashrc` , `~/.bash_profile` and are initialized when the shell starts.

Environment variables are typically written in capital letters. To view all environment variables available in the current shell, you can use the `env` or `printenv` commands .

Here are some of these variables:

- **\$BASH** is the full path to the Bash shell executable.
- **\$BASH\_VERSION** - Bash version.
- **\$CDPATH** - stores directory search paths (used when entering the command `cd directory_name` without a slash).
- **\$CLASSPATH** - Contains a list of directories to search for Java class files and Java archives.
- **\$HOME** - the current user's home directory.
- **\$HOSTNAME** - stores the computer name.
- **\$HISTSIZE** - the number of events stored in history for 1 session.
- **\$HISTFILE** - location of the event history file.
- **\$HISTFILESIZE** - The number of events stored in history between sessions.
- **\$IFS** - stores characters that are separators between commands and parameters (by default - space, tab, and newline).

- **\$LANG** - The current localization setting, which allows you to configure the shell for use in different countries and languages.
- **\$OSTYPE** - contains a description of the operating system.
- **\$PATH** is a list of directories to search for commands and applications when the full path to the file is not specified. When the user enters a command, the system will check the directories in the order specified here when searching for the executable file.
- **\$PS1** - PS1 is used as the main prompt string (in our case *[localadmin@astra~]\$* ).
- **\$PS2** - PS2 is used as a secondary prompt string.
- **\$PROMPT\_COMMAND** - This command must be executed before the Bash prompt is displayed.
- **\$PWD** is the full path to the current working directory.
- **\$OLDPWD** is the previous working directory. This information is saved by the shell to enable `cd -` to change to the previous directory .
- **\$SHELL** - the full path to the current command shell.
- **\$USER** - the name of the current user.

When the shell starts, there are no user variables. A user variable appears only if the user declares it and assigns a value to it.

A user variable will only be available within the shell process in which it was declared, but it can be turned into an environment variable with the `export` command , making it available to child processes started by the shell.

When calling a command or script with arguments, the command name and its arguments are positional variables. They are called positional variables because they are referenced within the script by their position on the command line. We'll cover positional variables in the module on Bash scripting.

Let's look at some basic operations with variables.

The syntax for declaring a variable or assigning a value to it is:

```
var1="Variable1"
```

If the variable value does not contain spaces, then the quotation marks can be omitted, otherwise they are required:

```
var1 = Variable1
var2 = "Variable 2"
var3 = Variable 3
bash: 3: command not found
```

Spaces between the variable name, the equal sign, and its value are not allowed:

```
var3 = "Variable 3"
bash: var3: command not found
var3 = "Variable 3"
bash: Variable 3: command not found
```

The value of one variable can be assigned to another

```
var3=$var1 :
var1 "
```

When displaying a variable, you must use not the name of the variable, but a reference to its value `echo $var1` (the **\$** symbol before it):

```
echo " $ var 1 "
Variable1
echo var 1
var 1
```

If the value of a variable contains more than one space between words, its value must be wrapped in quotation marks, otherwise the spaces will be lost in the output:

```
echo " $var2 "  
Variable 2  
var2  
Variable 2
```

Within single quotes, as well as when escaping the \$ symbol, variable values are not substituted:

```
echo '$var1'  
var1  
echo \$ var1  
var1
```

To reset the value of a variable, simply assign it an empty value:

```
var1 =
```

However, this will not delete the variable itself. To delete a variable, use the `unset<variablename>` command :

```
unset var1
```

To turn a user variable, which is only accessible within the current shell process, into an environment variable that is also accessible in child processes, use the `export` command :

```
env | grepvar2  
export var2  
env | grepvar2  
var2= Variable 2
```

If we now open a child bash process and try to print the values of the variables, we will see that the variable `var2` is available to us, but the variable `var3` is not:

```
echo " $var2 "  
Variable 2  
echo " $var3 "  
Variable 3  
bash  
echo " $var2 "  
Variable 2  
echo " $var3 "  
exit
```

Bash aliases are nothing more than keyboard shortcuts, a means of avoiding long command line typing. For example, if you add `alias lm="ls -l|more"` to your `~/.bashrc` file , you can save time and effort by simply typing `lm` . Setting `alias rm="rm -i"` (interactive file deletion mode) can save you a lot of trouble, as it reduces the likelihood of accidentally deleting important files.

To create an alias, use the command:

```
alias alias_name="value"  
alias ll = " ls - la "  
ll  
total 6632  
drwx----- 34 localadminlocaladmin 4096 May 24 12:19 .  
drwxr-xr- x 8 root root 4096 Jul 26 10:38 ..  
- rw ----- 1 localadminlocaladmin 26916 May 24 12:18 .bash_history  
- rw- r--r-- 1 localadminlocaladmin 220 May 16 12:06 .bash_logout  
- rw- r--r-- 1 localadminlocaladmin 3526 May 16 12:06 .bashrc
```

An alias created this way will only work in the current console. To make it permanent, add this command to your `~/.bash_profile` or `~/.bashrc` file .

To view all aliases, you need to use the `-p` switch :

```
alias -p  
alias -p  
alias ll='ls -ll'
```

```
alias ls='ls --color=auto'
```

To remove an alias, there is a special command `unalias<aliasname>` :

```
unalias ll
```

```
alias -p
```

```
alias ls='ls --color=auto'
```

In the Fly graphical shell, this key combination is assigned to open a new terminal window.

- `Alt + Backspace` - deletes the word in front of the cursor (to the left), or part of the word if the cursor is in the middle of it.
- `Alt + D` - deletes the word after the cursor (to the right), or part of the word if the cursor is in the middle of it.

For quick and easy navigation through command line text (and commands and their sequences can be quite long), arrow keys may not be enough. There are several useful shortcuts for this:

- `Ctrl + A` - moves to the beginning of the line. Similar to the Home key .
- `Ctrl + E` - moves to the end of the line. Similar to the End key .
- `Ctrl + B` - moves the cursor back (to the left) one character, but it's easier to use arrow keys.
- `Ctrl + F` - moves the cursor forward (to the right) one character, but it's easier to use arrow keys.
- `Alt + B` - move the cursor back (to the left) one word.
- `Alt + F` - move the cursor forward (to the right) one word.
- `Ctrl + xx` – perhaps for the most dedicated Linux fans. It moves the cursor between two points, cyclically. Typically, these are the beginning and end of a line. However, if after moving to the beginning of a line, some text of length *m* is typed, the cursor will move between the character on the line numbered *m* and *nm*, where *m* is the current line length.

All commands you type in the terminal window are saved in the command history. If you type a command and press the up arrow, the last executed command will appear in the input line, allowing you to re-execute it immediately or after editing it. This works the same way in Windows command shells.

The command history is stored both in RAM and in the `.bash_history` file , which is located in the user's home directory. In Astra Linux, by default, the command history includes commands from all sessions and terminals, which is convenient.

The number of lines stored in the history is determined by the `HISTSIZE` and `HISTFILESIZE` variables. The former determines the number of lines stored in memory, while the latter determines the number of lines stored in a file. The default values of these variables are set in the user profile `~/.bashrc` and are 1000 and 2000, respectively.

The profile also specifies shell options that can be controlled using the `shopt` command . For example, to enable saving history from all sessions and terminals for a user, the profile specifies the `shopt-shistappend` command . Shell options can also be controlled using the `set` command ; a list of these options is available in the `SHELLOPTS` variable.

If you've ever worked with `cmd` or `.bat` files in Windows, you've probably encountered the `"2>"` construct. And everyone's certainly familiar with the `">"` and

">>" constructs. In Linux, everything works the same way.

In Linux, there are 3 standard streams, which are denoted by a positive integer:

- 0 – standard input (stdin) is used to transfer data from the user (usually keyboard input or input from a file) to the program.
- 1 – Standard output (stdout) is used to transmit data generated during program execution. By default, data is output to the screen, but can be redirected.
- 2 – Error output (stderr) for any errors that may occur during program execution. Errors are output to the screen by a separate stream by default, but can be redirected.

A program can only read data from the standard input stream, while the other two streams are used only for writing. Data is displayed on the screen and read from the keyboard, as the standard streams are associated with the user's terminal. Streams can be connected to files, programs, and even devices. In Bash, this operation is called redirection.

There are several ways to redirect:

- <file – use a file as a data source for the standard input stream.
- file – directs standard output to a file. If the file doesn't exist, it will be created; if it does, it will be overwritten.
- >>file – directs standard output to a file. If the file doesn't exist, it will be created; if it does, the data will be appended to the end of the file.
- 2>file – direct the standard error stream to a file.
- 2>>file – directs standard error to a file. If the file doesn't exist, it will be created; if it does, the data will be appended to it.
- &>file or >&file – direct standard output and standard error to a file.

Another form of writing: >file2>&1 .

Splitting streams allows, for example, to write program output to one file and errors to another.

### **Examples:**

- redirecting the command output to a file.

```
echo " Hello !" >hello.txt
cat hello.txt
Hello!
```

The example uses the `cat` command, which copies standard input to standard output. It is often used to view the contents of files, merge multiple files into one, and write short text to a file.

- redirecting the command output to the end of the file.

```
echo " How are you ?" >>hello.txt
cat hello.txt
Hello!
How are you?
```

- passing content to a command as arguments.

```
cat <hello.txt
Hello!
How are you?
```

A pipeline is a very powerful tool that allows you to pass the output of one command as arguments to another. This is accomplished using the `|` (vertical bar)

operator. Anyone who has worked with PowerShell should be familiar with this tool.

When used, the standard output of the command on the left is redirected to the standard input of the command on the right. The number of commands and redirections is unlimited, allowing for long chains of commands, such as `cmd1|cmd2|...|cmdN` . The pipeline also enables asynchronous command execution using I/O buffering. This means we don't have to wait for the first command to complete before the second one can begin execution—the first command will pass its results to the second command as they are ready.

**Example:**

`ls` is a command that displays the contents of the current directory. Let's run this command:

```
ls
Desktop Script Video Images Templates
Desktops, System, Wallpapers, Documents, Music
hello.txt VBoxLinuxAdditions.run Downloads Public
```

If we want to display only files (or directories) that contain "hello" in their name, we can use the `grep` command , which filters lines based on a given pattern, by piping the output of the `ls` command to it .

```
Let's run the command ls|grep"hello" :
ls | grep "hello"
hello.txt
```

In the terminal and, more importantly, in the console, we can work with the clipboard. Here are the shortcuts for working with the clipboard:

- `ctrl + w` - cuts the word in front of the cursor (on the left) and copies it to the clipboard.
- `ctrl + u` - cuts text from the beginning of the line to the cursor position and places it on the clipboard.
- `ctrl + k` - cuts and copies to the clipboard all characters in the line after the cursor (on the right).
- `ctrl + y` - pastes text from the clipboard before the cursor.

In Linux, unlike Windows, case is very important – file and folder names, commands, and switches are case-sensitive. To simplify working with case, you can use combinations:

- `Alt + u` - Changes the case of all characters after the cursor (to the right) to uppercase until the end of the current word.
- `Alt + l` - Changes the case of all characters after the cursor (to the right) to lower case until the end of the current word.
- `Alt + c` - changes the case of the character under the cursor to uppercase and moves the cursor to the end of the word.
- `ctrl + L` - clears the terminal of text, an alternative to the `clear` command , similar to `cls` in Windows.
- `ctrl + s` - pauses programs outputting data to the terminal window. The program continues to run.

This feature is disabled by default. To enable it, you can run the `sttyixon` command , but doing so will disable the reverse command history search assigned to the same key combination.

- `ctrl + q` - cancel pausing the output of data to the terminal window (cancel `ctrl + s` ).

### 5.3 Assignment for independent completion

#### 5.3.1 Task 1.

1. Determine the `tty` file of the current pseudo-terminal session.
2. Send an echo to the pseudo-terminal file. To do this, open a new terminal and send a greeting.
3. Display the `$COLUMNS` and `$LINES` variables.
4. Run `sleep9000` and after a while press `ctrl + c` to interrupt the command.
5. Open the `mc` utility , look at the file hierarchy in `mc` and try to close it by pressing `ctrl + c` .
6. Why doesn't `mc` close with `ctrl + c` ?

#### 5.3.2 Task 2.

1. Before you start working, find out what working directory you are in.
2. Change the current directory to `etc`.  
What has changed in the command line?
3. Please check your working directory again.
4. List all objects using the `ls` command.
5. Look at the `hostname`, print `hostname` to the screen using the `cat` command.  
Answer the question: did the file change after executing `cat`?  
Where did the `cat` command output the contents of the file?  
Is the `cat` command safe?
6. Make a copy of the `hostname` before changing the `STDOUT` redirection.  
`cat hostname > ~/hostname.old`  
Where is the `hostname.old` file saved?
7. Check how the backup was saved:  
`cat < ~/hostname.old`
8. Change the `hostname` using the `nano` editor:  
`nano /etc/hostname`  
Answer the question: why is it highlighted in red?  
[ File „`/etc/hostname`“ is unwritable ]?  
Close the editor with `ctrl + x` . If the editor prompts you to save changes when exiting `nano`, press `N` and then `Enter` .
9. Repeat the command with elevated rights.  
`sudo!!`
10. Change the text to `dc-1` (future host name).
11. We wrote something incorrectly and automatically pressed `Ctrl + Z`. Windows has a habit of undoing entered text with this combination.  
Press `ctrl + z` .  
With this action we paused the process using the control sequence `ctrl + z` .
12. Bring the process back from the background with the `fg` command.
13. Save the text we need `dc-1` by pressing `Ctrl + O` and `Enter`.

14. Close the nano editor with Ctrl + X.
15. To change the hostname, please reboot.
16. After changing the hostname, problems with displaying sudo will occur, and for this you need to change the host in /etc/hosts.

### **5.3.3 Task 3.**

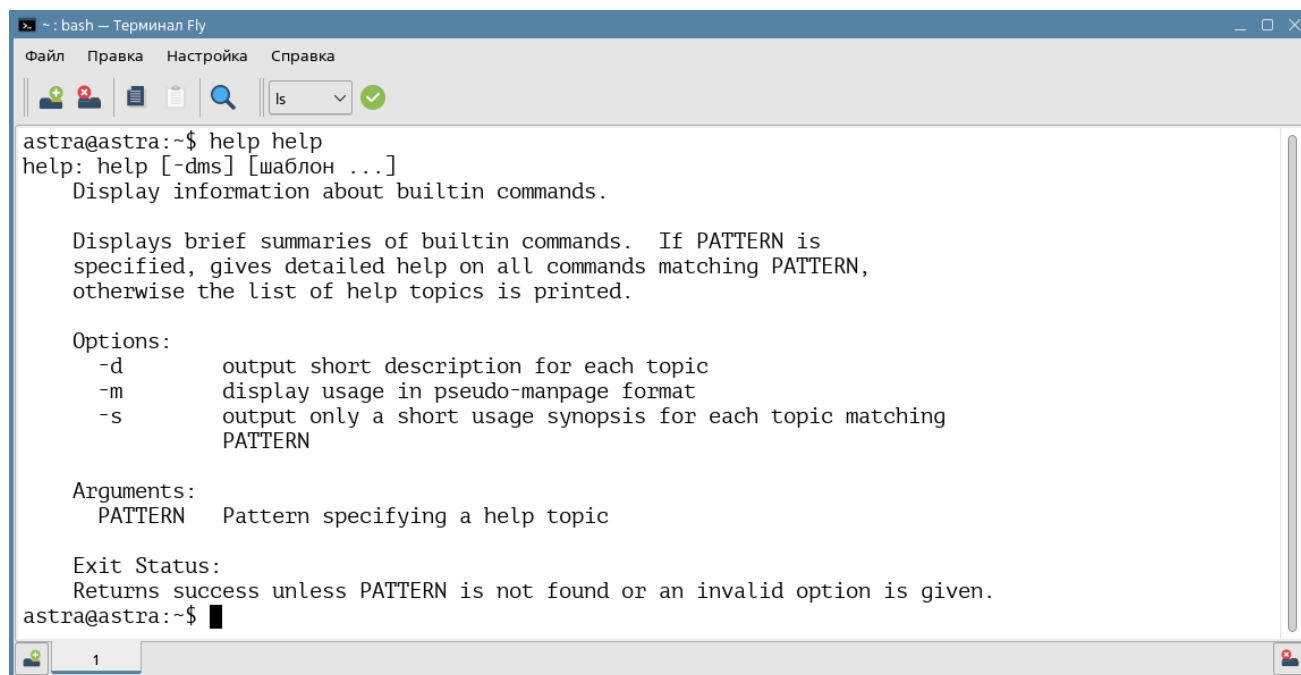
1. Display a greeting to the current user. Where can the greeting and the \$USER variable be used?
2. See what common global environment variables there are.
3. Print all variables of the current session.
4. Use the grep filter to find only LINES or COLUMNS in the pipelined text data.
5. Try restarting your PC as the current non-root user.  
Please answer why it didn't work by looking at the \$PATH environment variable:
  1. See what binaries a standard user can run.
  2. See which files the root user can run. To do this, log in as the root user.

### **5.3.4 Task 4.**

1. Take a short test <https://onlinetestpad.com/6v4rkwqq76fcg>.

You may notice that the very first paragraph contains a reference to other help systems that can be accessed using the `man` and `info` utilities . Enter the `help`

command to get help about itself using the help utility, see Figure 6.2:

A terminal window titled '~: bash — Терминал Fly' with a menu bar (Файл, Правка, Настройка, Справка) and a toolbar. The terminal shows the command 'astra@astra:~\$ help help' and its output. The output describes the 'help' command, its options (-d, -m, -s), its arguments (PATTERN), and its exit status. The prompt 'astra@astra:~\$' is followed by a cursor.

```
astra@astra:~$ help help
help: help [-dms] [шаблон ...]
    Display information about builtin commands.

    Displays brief summaries of builtin commands.  If PATTERN is
    specified, gives detailed help on all commands matching PATTERN,
    otherwise the list of help topics is printed.

    Options:
      -d      output short description for each topic
      -m      display usage in pseudo-manpage format
      -s      output only a short usage synopsis for each topic matching
              PATTERN

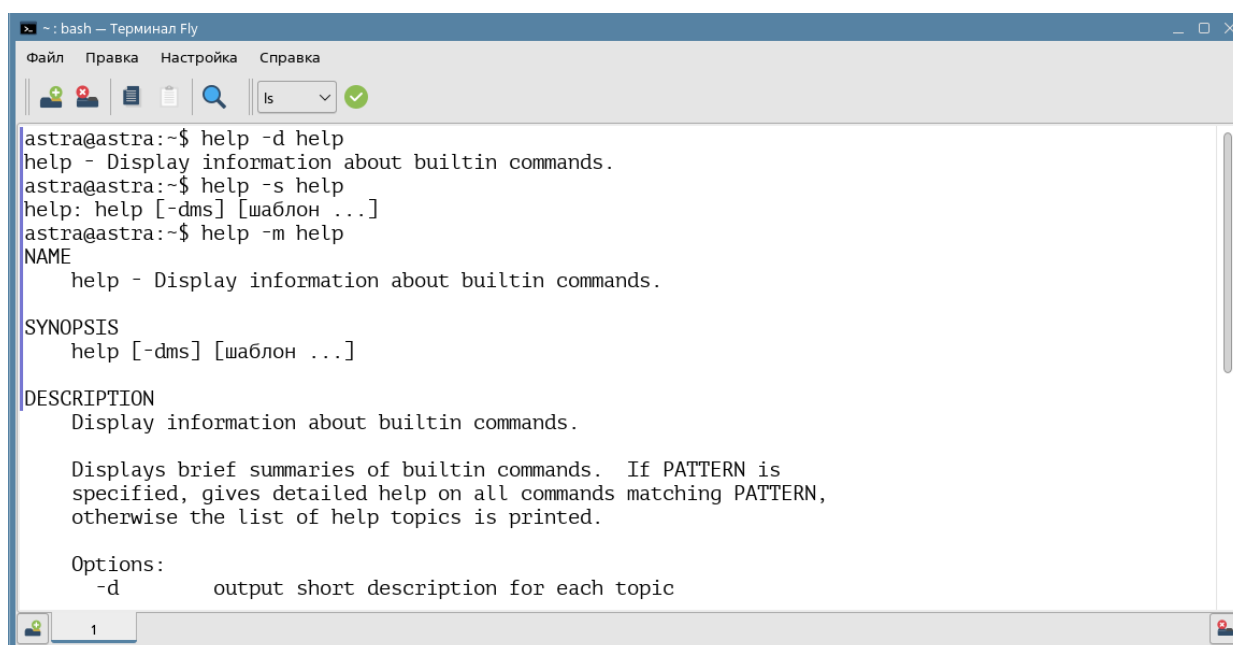
    Arguments:
      PATTERN  Pattern specifying a help topic

    Exit Status:
      Returns success unless PATTERN is not found or an invalid option is given.
astra@astra:~$
```

Figure 6.2 - Help for the help command

As you can see, the help command has only three keys, see Figure 6.3:

- -d - displays a short description of the command;
- -m - outputs help in pseudo-manpage format;
- -s - displays a list of keys in a short format without their description.

A terminal window titled '~: bash — Терминал Fly' with a menu bar and toolbar. The terminal shows three commands: 'astra@astra:~\$ help -d help', 'astra@astra:~\$ help -s help', and 'astra@astra:~\$ help -m help'. The output for 'help -d help' is shown, displaying the name, synopsis, description, and options for the 'help' command. The prompt 'astra@astra:~\$' is followed by a cursor.

```
astra@astra:~$ help -d help
help - Display information about builtin commands.
astra@astra:~$ help -s help
help: help [-dms] [шаблон ...]
astra@astra:~$ help -m help
NAME
    help - Display information about builtin commands.

SYNOPSIS
    help [-dms] [шаблон ...]

DESCRIPTION
    Display information about builtin commands.

    Displays brief summaries of builtin commands.  If PATTERN is
    specified, gives detailed help on all commands matching PATTERN,
    otherwise the list of help topics is printed.

    Options:
      -d      output short description for each topic
```

Figure 6.3 . Outputting information on help command keys

We can look at the help for the `helpenable` command, which we used to disable the built-in command `enable-necho`. As you may recall, this switches the shell to the utility of the same name, which works similarly. To re- enable `echo`, you need to call

the `enable` command without the `enableecho` switch , and calling the command without any parameters will display a full list of all active shell commands.

### 6.2.3 Manpages

Just a couple of years after Unix's release, the first manual pages appeared, although their format was fundamentally different from modern help. A help system that could dynamically update its contents and allow centralized search was urgently needed. Manpages, a portmanteau of "manualpages," became such a system. The manpages system contains help on individual commands, special device files, configuration files, system and library calls, and other aspects of the system, typically presented in eight standard sections.

In the modern version of manpages, manual pages are written in the `troff` markup language and are `gzip`-compressed files. English manual pages are stored in the `/usr/share/man/man<N>` directories , where `<N>` is the manual section number. Localized pages are located in the `/usr/share/man/<locale>/man<N>` directories , where `<locale>` is the language code. For example, the full path to the compressed file containing the manual page for the `man` utility itself can be found using the `man-wman` command :

```
man-wman
/usr/share/man/ru/man1/man.1.gz
```

We can view manpages using the `man` application (part of the `man-db` package), which uses several auxiliary utilities. First, the `man` utility searches for the requested page, then it unpacks the corresponding file using `gzip` , formats the text in `roff` or `groff` markup using the `troff` processor , and finally, outputs the result to the terminal window using the `less` utility . The `troff` processor allows text to be adapted to any screen width and performs syntax highlighting using terminal escape sequences. The `less` utility allows you to scroll through long manual pages and search the text.

`less` utility's capabilities for page navigation. Enter the `manman` command in the terminal to display help for the `man` command itself , see Figure 6.4:

```
manman
```

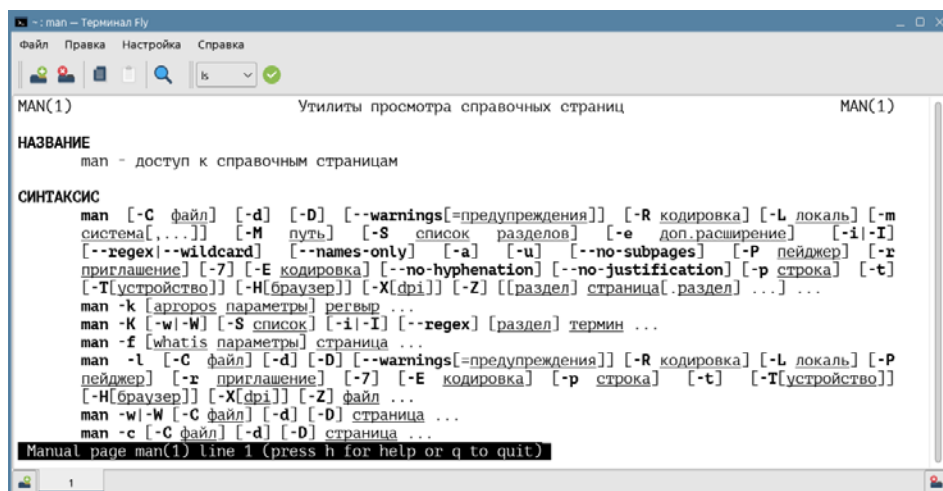


Figure 6.4 - Open less view of the `man` command's manual page

This is the help page for the `man` utility from the first section of the documentation, as indicated by the MAN(1) label in the upper left corner of the page. As mentioned earlier, the page is viewed using the `less` utility, which allows you to scroll up and down the text and perform searches.

Let's look at some techniques for working with the `less` program :

- Line-by-line movement:
  - `<down arrow>` keys, `Enter` .
  - Up - mouse wheel up, `<up arrow>` key .
- Page-by-page navigation:
  - Down - keys `<space>` , `PgDown` .
  - Up - `PgUp` key .
- Go to the top of the page - Home key.
- Go to the end of the page - End key.
- Detailed help on commands - key `h`.
- Exit - `q` key.

A detailed study of the `man(1)` manual page would take too much time, so we'll only cover the most important points and basic techniques. In many manuals, you'll encounter references to manual pages in the format `login(1)` or `login(5)`, where the numbers in parentheses indicate the section of the documentation where you should look for the information you need. The `login` utility, of course, remains the same, but the information about it in these sections can vary significantly in both depth and topical focus.

Manpages uses eight main and one non-standard manual page section:

1. Basic commands (General commands).
2. System calls.
3. Library functions.
4. Special files, such as devices from `/dev` .
5. File formats and conventions, such as `passwd` , *cron table* .
6. Games.
7. Miscellaneous.
8. System administration commands.
9. Kernel Routines [*non-standard section*].

Each help section has an intro page that describes its purpose.

To view this page, enter the command `man<N>intro` , where `<N>` indicates the desired help section (see above `/usr/share/man/man<N>` ).

For example, to display the header page for the first section, you would enter the command, see Figure 6.5: `manintro`

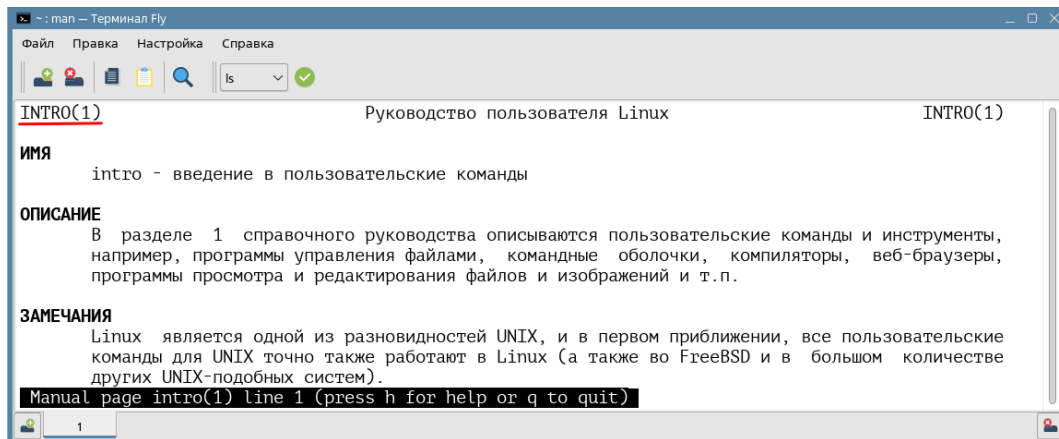


Figure 6.6 - Intro page of the man command help section

To display help for the fourth section, use the command: `man 4 intro`

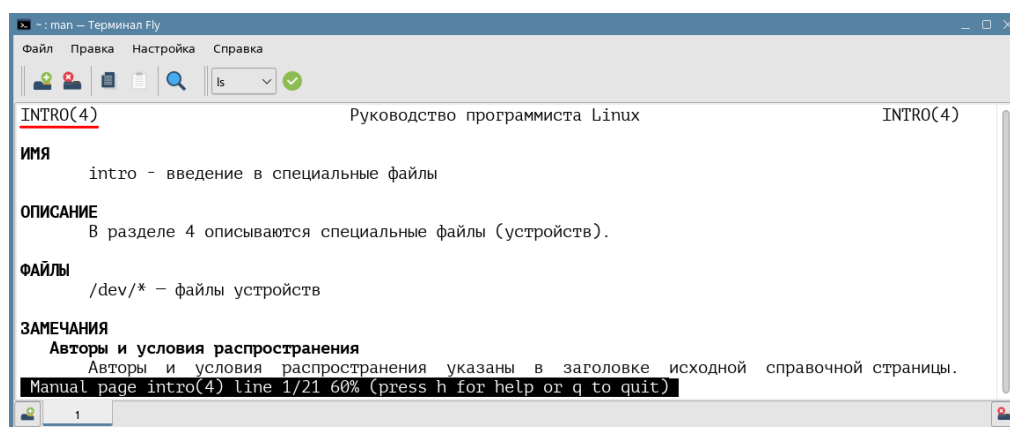


Figure 6.7 - Introduction to Special Files Section 4

The help for the `man` utility specifies the order in which to search for information by section (1 nl 8 3 2 3posix 3pm 3perl 3am 5 4 9 6 7), which should be taken into account when calling the utility without specifying a specific section.

When compiling manual pages, we try to adhere to general rules, so most pages contain the following sections:

- **Name** - The name of the command that this help page describes.
- **Synopsis** - a brief description of the command and its syntax.
- **Configuration** - This section is present on the special files pages (4), as it describes the configuration of the devices.
- **Description** - description of the main purpose of the program.
- **Options (Keys)** - options that the command accepts.
- **Examples** - one or more examples of how to use the command.
- **Authors** - information about the developers who participated in the creation of the utility and are engaged in its further development.

To get help information for a specific command, call `man<command>` , for example, `manwhereis` :

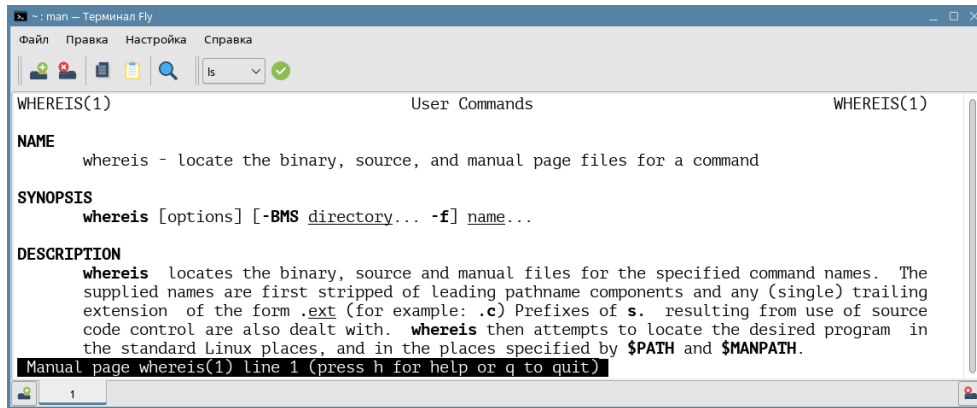


Figure 6.8 - Man pages for the whereis command

There are also manual pages for most utilities and system configuration files, so you can substitute a relative or full path to a configuration file for the command name, for example, `manhosts` :

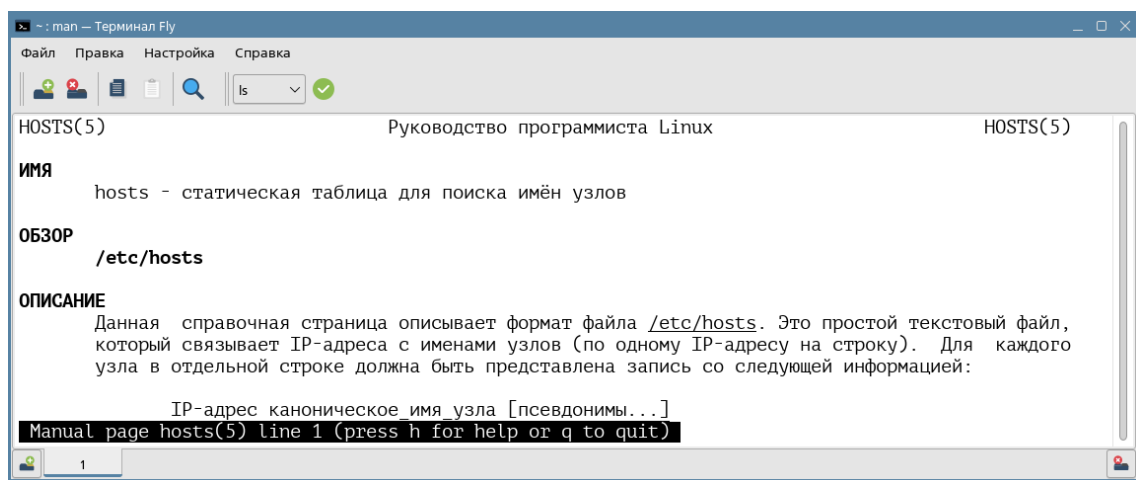


Figure 6.9 - Hosts configuration file manual pages

By scrolling to the end using the `End` key , you can see the "SEE ALSO" section, which describes additional links to similar topics and help sections.

```
...
SEE ALSO
hostname( 1), resolver(3), host.conf(5), resolv.conf(5),
resolver( 5), hostname(7), named(8)
```

Internet RFC 952

Let's open section 5 of `resolv.conf` with the command:

```
man 5 resolv.conf
```

Similarly, you can get help on services (traditionally called daemons in Linux), such as `mansyslogd` :

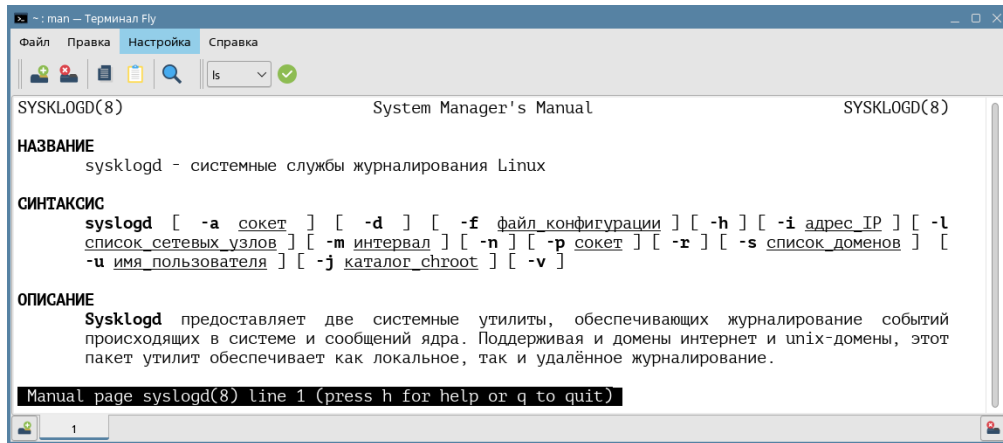


Figure 6.10 - Manual pages for the syslogd service

## 6.2.4 Search for manual pages

Without specifying a section, the `man` utility searches in the default order. To get information about a specific section, you can specify it explicitly. But how do you know which sections might contain the information you need without having to search through them one by one? For this purpose, there are the `man-f<searchstring>` and `whatis<searchstring>` commands —they will display a list of manual pages from all sections whose names match the entered string, for example, `man-fpasswd` :

```
man-fpasswd
passwd(1) - changes the user's password
passwd (5) - password file
passwd (1 ssl ) - computepasswordhashes
```

But you can search not only by page titles, but also by page descriptions. To do this, use the `man-k<regular expression>` OR `apropos<regular expression>` command , for example, `man-kpasswd` :

```
man-kpasswd
chpasswd (8) - updates passwords in batch mode
gpasswd (1) - manage /etc/group and /etc/gshadow
passwd(1) - changes the user's password
passwd (5) - password file
passwd2des (3) - RFS password encryption
update-passwd (8) - securely update files /etc/passwd, /etc/shadow ...
chpasswd (8) - update group passwords in batch mode
exim4_passwd (5) - Files in use by the Debian exim4 packages
exim4_passwd_client (5) - Files in use by the Debian exim4 packages
expect_mkpasswd (1 ) - generate new password, optionally apply it to a user
fgetpwent_r (3) - get passwd file entry reentrantly
getpwent_r (3) - get passwd file entry reentrantly
grub-mkpasswd-pbkdf2 (1) - generate hashed password for GRUB
openssl-passwd (1ssl) - compute password hashes
pam_localuser (8) - require users to be listed in /etc/passwd
passwd (1ssl) - compute password hashes
smbpasswd (5) - The Samba encrypted password file
smbpasswd (8) - change a user's SMB password
```

To ensure the fastest possible help search, utilities access the built-in index database, located in the `/var/cache/man` folder, rather than the `/usr/share/man` files directly. This database must be updated using the `mandb` command each time applications are installed or updated.

```
mandb
Processing manual pages in /home/astra/.local/share/man...
```

Updating index cache for path '/home/astra /.local /share/man/man1'. Please wait...completed.

Checking sideloads of cat in /home/astra/.local/share/man...

Processing man pages in /home/astra/.local/share/man/cat1...

The 1 man subdirectory contains newer manual pages.

Added 1 help page.

Added 0 side cat pages.

Cleaned 0 old database records.

less utility interface allows you to search through the text of a manual page, offering two methods: a regular search and a search that displays matching lines.

manman help again , then enter the forward slash symbol </> and type the word we want to find, for example, /section :

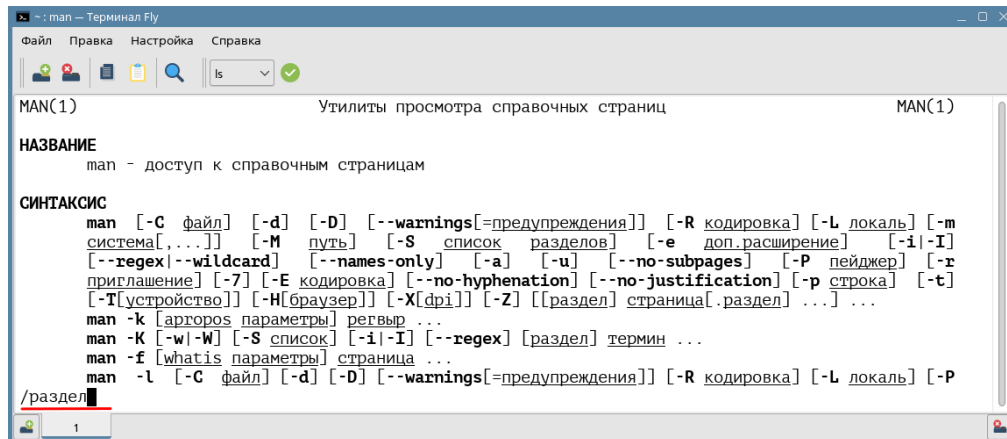


Figure 6.11 - Manual pages for the syslogd service

Next, press the Enter key to go to the line containing the first occurrence of this word:

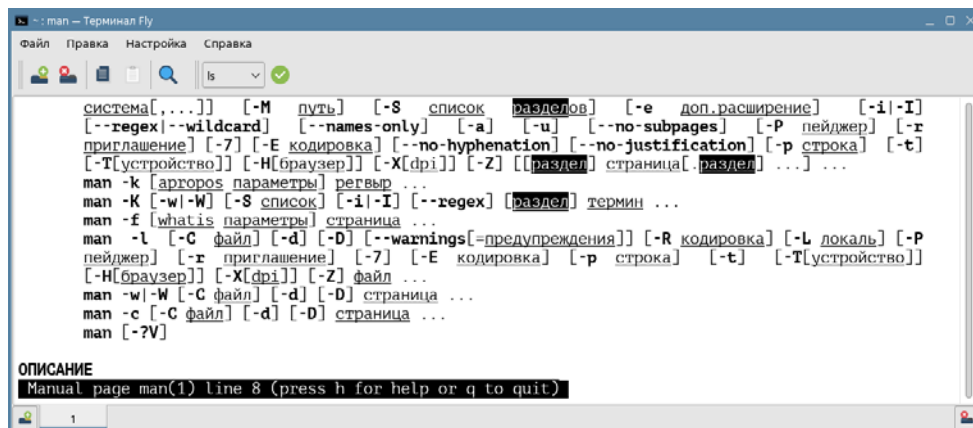


Figure 6.12 - Manual pages for the syslogd service

n and N keys allow you to move forward/backward through the entries, respectively.

All occurrences of this word will be highlighted in color, and to enable/disable highlighting of found occurrences, you can use the keyboard shortcut Esc + u .

To search backwards (if you are at the end of a page), use the question mark character <?> instead of the forward slash character , for example ? page .

To quickly review search results, you can display all occurrences on a single

page. When using this method, it is recommended to enable line numbers.

To do this, enter `-N` and the command will change to `"Constantlydisplaylinenumbers"` (press `RETURN`) » and then press `Enter` :

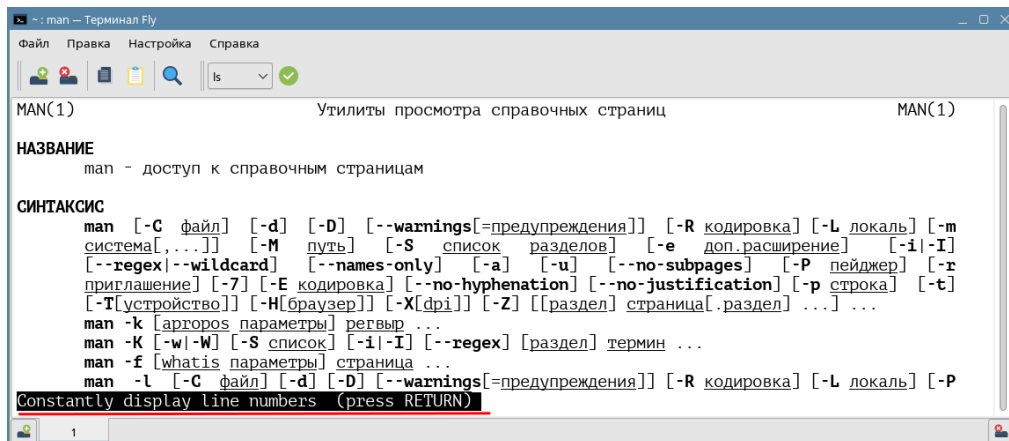


Figure 6.13 - Manual pages for the syslogd service

Line numbers will appear in the less interface.

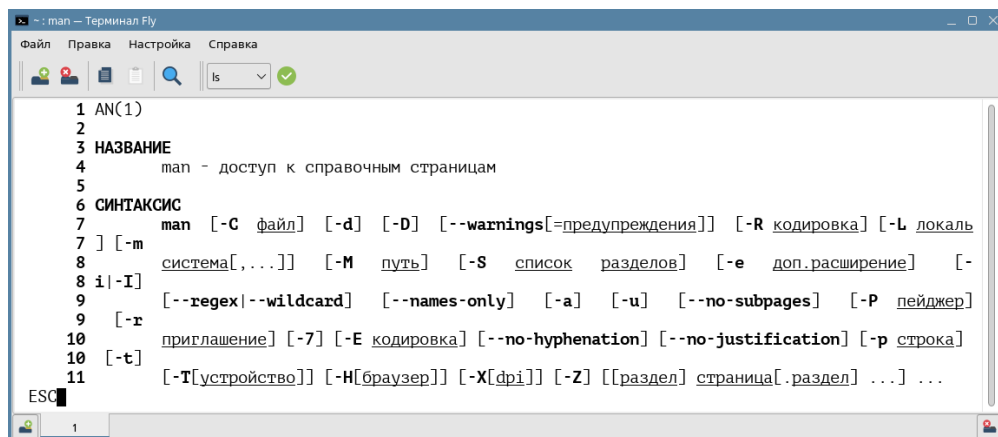


Figure 6.14 - Manual pages for the syslogd service

Now we enter `<&>` (the `/` symbol will appear automatically), type the search word `whatis` and press the `Enter` key :

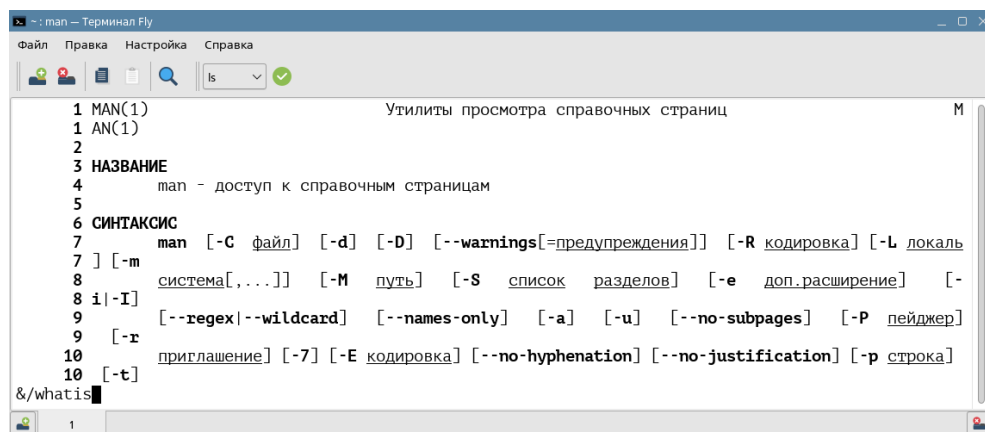
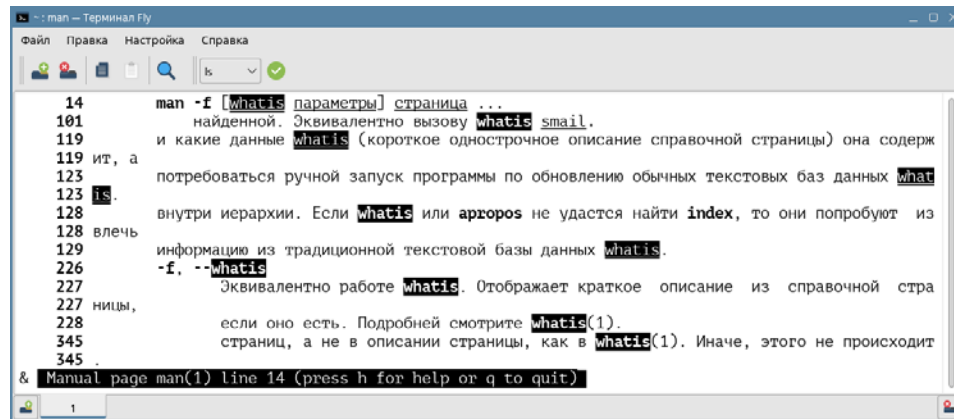


Figure 6.15 - Manual pages for the syslogd service

Only those lines in which this word is found will be displayed on the screen.

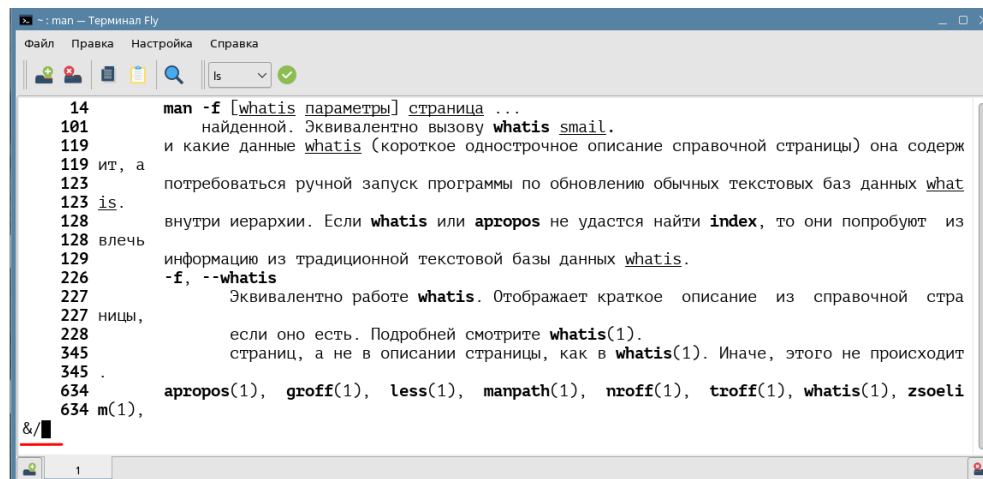


```
man -f [whatis параметры] страница ...
101 найденной. Эквивалентно вызову whatis smail.
119 и какие данные whatis (короткое однострочное описание справочной страницы) она содержит, а
123 потребоваться ручной запуск программы по обновлению обычных текстовых баз данных what
123 is.
128 внутри иерархии. Если whatis или apropos не удастся найти index, то они попробуют из
128 влечь
129 информацию из традиционной текстовой базы данных whatis.
226 -f, --whatis
227 Эквивалентно работе whatis. Отображает краткое описание из справочной стра
227 ницы,
228 если оно есть. Подробней смотрите whatis(1).
345 страниц, а не в описании страницы, как в whatis(1). Иначе, этого не происходит
345 .
& Manual page man(1) line 14 (press h for help or q to quit)
```

Figure 6.16 - Manual pages for the syslogd service

If you are interested in a specific line, you can jump to it by its number, for example 226.

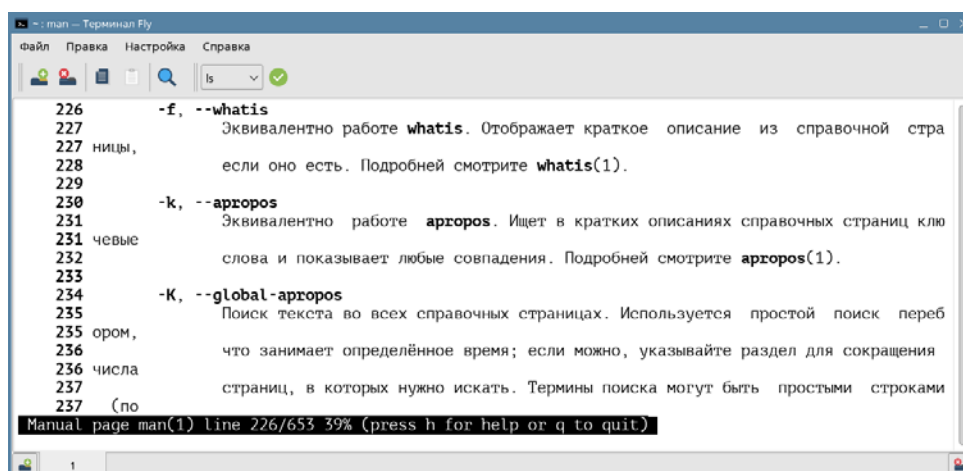
To do this, first return the display of all lines with the command & then Enter :



```
man -f [whatis параметры] страница ...
101 найденной. Эквивалентно вызову whatis smail.
119 и какие данные whatis (короткое однострочное описание справочной страницы) она содержит
119 ит, а
123 потребоваться ручной запуск программы по обновлению обычных текстовых баз данных what
123 is.
128 внутри иерархии. Если whatis или apropos не удастся найти index, то они попробуют из
128 влечь
129 информацию из традиционной текстовой базы данных whatis.
226 -f, --whatis
227 Эквивалентно работе whatis. Отображает краткое описание из справочной стра
227 ницы,
228 если оно есть. Подробней смотрите whatis(1).
345 страниц, а не в описании страницы, как в whatis(1). Иначе, этого не происходит
345 .
634 apropos(1), groff(1), less(1), manpath(1), nroff(1), troff(1), whatis(1), zsoeli
634 m(1),
&/
```

Figure 6.17 - Manual pages for the syslogd service

Then enter line number 226 and press g to jump.



```
-f, --whatis
227 Эквивалентно работе whatis. Отображает краткое описание из справочной стра
227 ницы,
228 если оно есть. Подробней смотрите whatis(1).
229
230 -k, --apropos
231 Эквивалентно работе apropos. Ищет в кратких описаниях справочных страниц ключ
231 евые
232 слова и показывает любые совпадения. Подробней смотрите apropos(1).
233
234 -K, --global-apropos
235 Поиск текста во всех справочных страницах. Используется простой поиск переб
235 ором,
236 что занимает определённое время; если можно, указывайте раздел для сокращения
236 числа
237 страниц, в которых нужно искать. Термины поиска могут быть простыми строками
237 (по
Manual page man(1) line 226/653 39% (press h for help or q to quit)
```

Figure 6.18 - Manual pages for the syslogd service

To disable line numbering, enter the `-N` command and press Enter . The `whereis` utility allows you to find the location of files on disk. The application searches in the same way as the command line shell, i.e., all directories listed in the `PATH` variable are searched. To see a complete list of these directories, you can use the `whereis -l` (lowercase L) command:

```
whereis-l
bin: /usr/bin
bin: /usr/sbin
bin: /usr/lib/x86_64-linux-gnu
bin: /usr/lib
```

`whereis<commandname>` command searches for all files related to the requested command:

```
whereisls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
```

The command with the `whereis-b<commandname>` switch searches only for binary files:

```
whereis-bls
ls: /usr/bin/ls
```

The command with the `whereis-m<commandname>` key searches only in the documentation:

```
whereis-mls
ls: /usr/share/man/man1/ls.1.gz
```

`which` utility displays the full path to the specified commands:

```
whichman
/usr/bin/man
```

By default, the command only displays the first value found, but using the `which-aman` switch you can get the entire list:

```
which-aman
/usr/bin/man
/bin/man
```

Another documentation system is `Texinfo`, which, unlike `man` , allows you to create large, detailed manuals.

The `texinfo` markup language allows you to format text, create links, tables, etc. Based on these source files, you can generate reference documents in various formats, so the markup commands have conditional text inclusion instructions for each of the output formats.

`info` utility operates on a set of pre-generated hypertext pages from the `/usr/share/info` directory . Each page has a hierarchical structure consisting of so-called "nodes," which can be compared to sections, subsections, chapters, or sections in books.

`info` command launches the help system, and `info<command>` , such as `infocat` , displays the man page for that command. `texinfo` has built-in help for commands; you can display it by pressing the **<H> key** . You can learn more about the system in the `maninfo` help file .

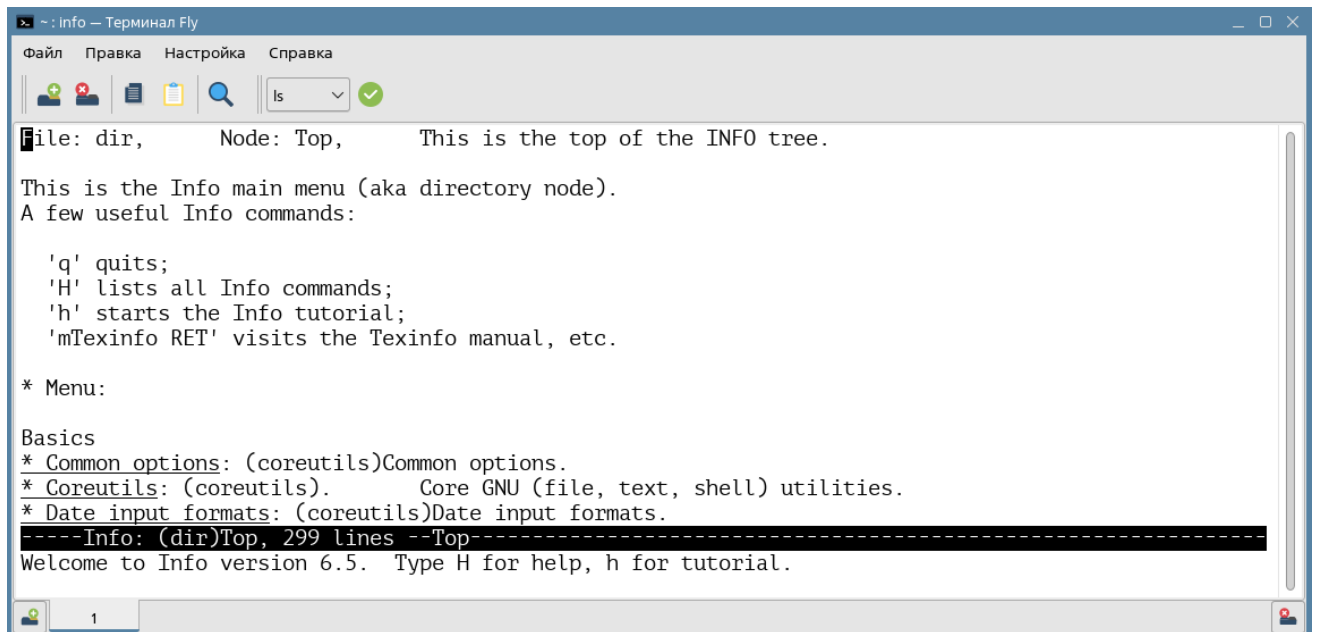


Figure 6.19 - Manual pages for the syslogd service

## 6.2.5 Astra Linux Help

Astra Linux's graphical interface also offers another help system, accessible by pressing **Alt + F1**. This launches the assistant application, which is passed the path to the `flyhelp.qhc` file as input:

```
assistant-collectionFile/usr/share/doc/fly/qthelp/ru/flyhelp.qhc
```

This help provides a detailed description of the graphical interface, utilities, settings, and programs included with the distribution. If you have questions about using the desktop, control panel, device manager, and other features, you can refer to this help system.

The help has an intuitive interface:

- In the left column there are four tabs: “contents”, “index”, “bookmarks” and “search”;

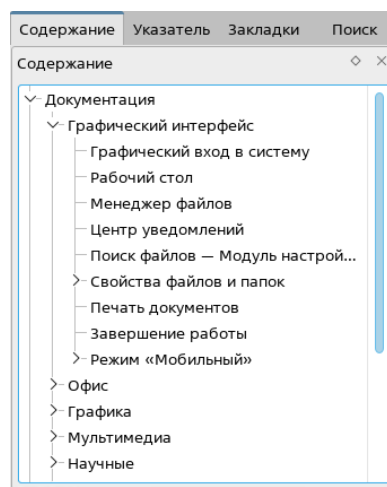


Figure 6.20 - Help Contents

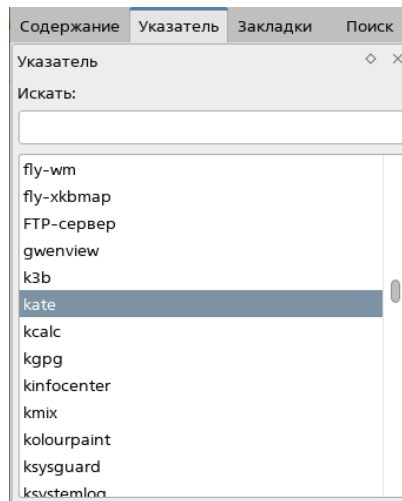


Figure 6.21 - Help Topics Index Tab

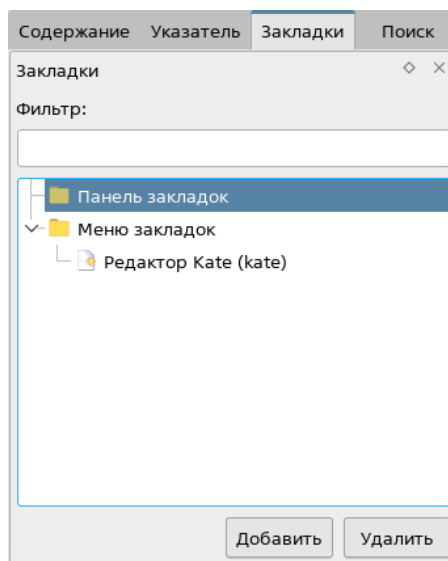


Figure 6.22 - Bookmarks panel for saving pages to bookmarks

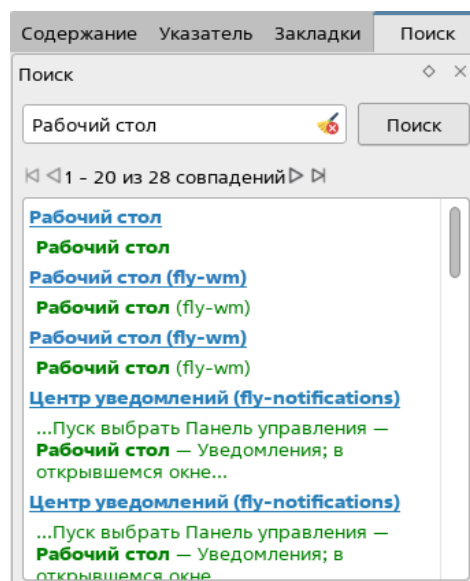


Figure 6.23 - Help Topic Search Tab

- Just below is a list of open pages;

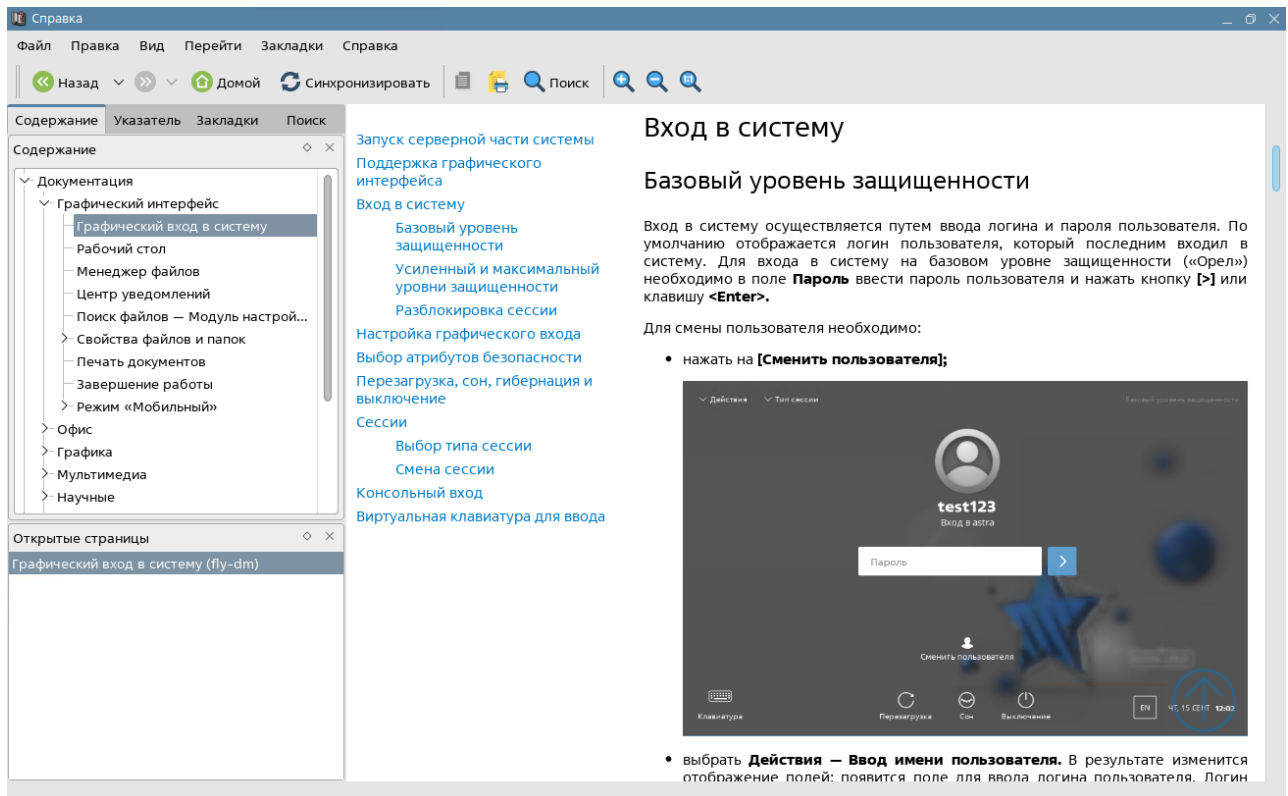


Figure 6.24 - Login Help

- On the right side of the window is the text of the page you are currently viewing.

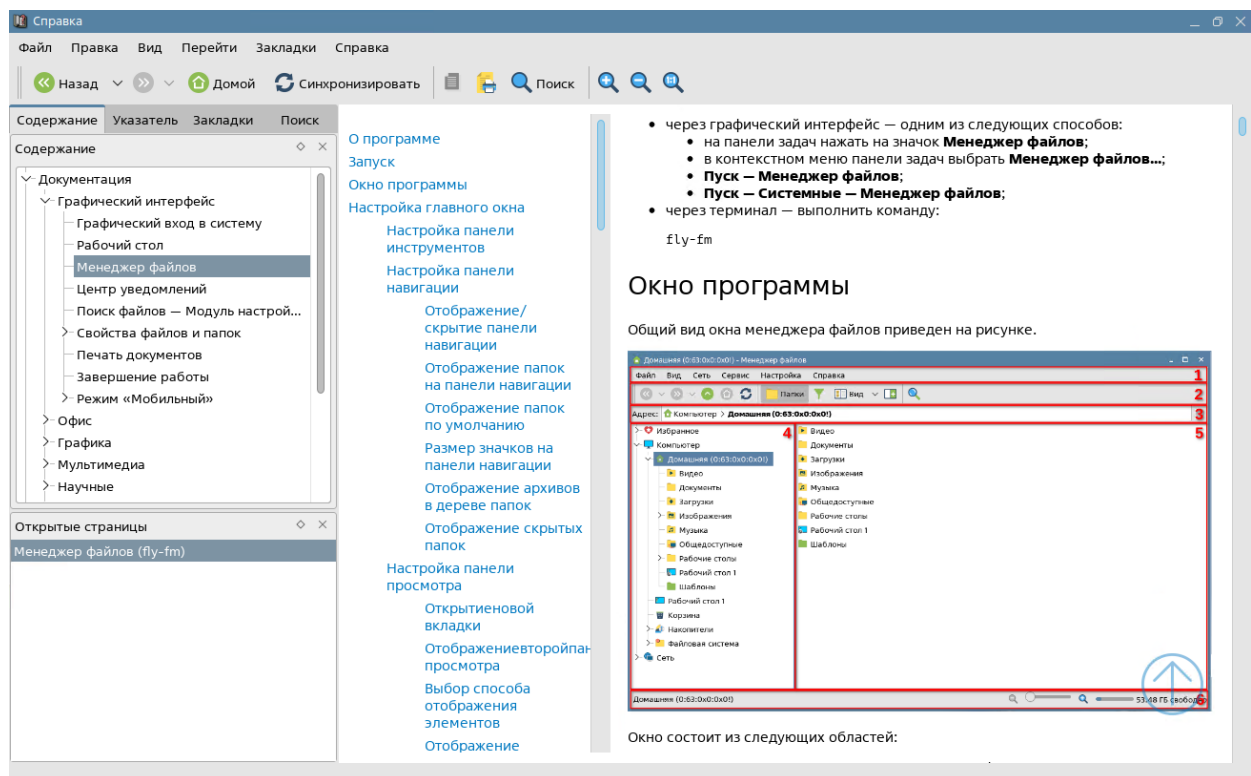


Figure 6.25 - Help for the fly-fm file manager

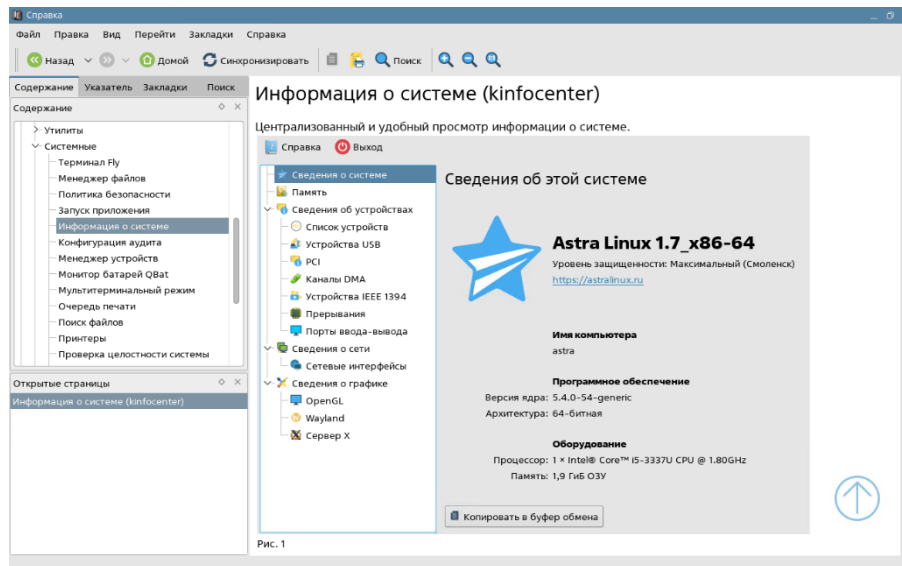


Figure 6.26 - Help for the System Information utility

## 6.3 Assignment for independent completion

### 6.3.1 Task 1.

1. Enter the `pwd` command and then the `!!` command . Verify that calling `!!` results in the `pwd` command being executed again .
2. By default, the `!147` and `!!` commands repeat the execution of the 147th and previous commands, respectively, but you can make the command text simply appear in the input line for further editing without running the command.
  - You will need to change the value of one of the bash shell parameters, which is done by a built-in command.
  - You can start searching for information with the phrase "history list substitution".
3. Change the settings and check that calling the `!!` command does not repeat the execution of the previous command, but places the text of this command in the input line.

### 6.3.2 Task 2.

1. Open the texinfo help system and study the command help.
2. From the main help menu, go to the "Coreutils" section, and from there to "Special filetypes" using the search and following links.
3. Explore the list of available commands and reference information for the `link` command.
4. Close the application and call texinfo help for the `ln` command.
5. Explain how the `link` and `ln` commands are similar and how they are different.

### 6.3.3 Task 3.

1. Explore the graphical help and its control interface.

2. Find the "Find Files" page using the Index and explore it.
3. Find the fly-admin-time help page and familiarize yourself with it.

#### **6.3.4 Task 4.**

1. Take a short test <https://onlinetestpad.com/rbte3ilummonk>.

## 7 Working with files and directories

### 7.1 Objectives of the work

1. Explore the Linux file system.

### 7.2 Theoretical justification

#### 7.2.1 Key differences in working with files

Let's recap the differences between Windows and Linux we discussed in the first module.

The first significant difference is that in Linux, files aren't what they seem. Files aren't just used to store information on disk. They're also used to implement things like directories and links. The file access interface is also used to interact with devices, pipes, and sockets.

This approach to providing access to all operating system capabilities through a unified file system interface is known as the "everything is a file" approach. This interface, of course, may not always be optimal for a given task, but it is always consistent. For example, consider the `/proc` directory . We've already mentioned that a virtual file system is mounted at this location, providing access to information about the system and its processes. Therefore, using the command `cat /proc/meminfo`, you can see memory usage parameters.

While Linux adheres to the concept of "everything is a file," Windows follows the concept of "everything is an object," meaning that all system resources—files, directories, hardware devices—have properties and methods, as in object-oriented programming.

The second difference from Windows is the unified file system namespace. While in Windows each block device is assigned its own letter and the absolute path to a local file is `c:\pagefile.sys` , in \*nix-like systems all drives are integrated into a single namespace, which begins with a slash `/` (the root).

For example, in the Astra Linux operating system, when an external USB drive is connected, its file system is mounted to `/run/user/<UID>/media/<device ID>` using the FUSE module ( *from filesysteminuserspace* ). Furthermore, any directories, such as `/home` or `/var` , can be physically located on other devices, including network devices, and mounted using the operating system.

The third difference is the approach to file storage. In Windows, operating system service files are located in the `c:\Windows` directory , while application files are located in the `C:\ProgramFiles` directory . Each application has its own separate folder, which stores all its files, organized in any way desired by the application developers.

Linux adheres to the Filesystem Hierarchy Standard (FHS) for file organization, which standardizes file placement based on their purpose. For example, application executables should be located in the `/bin` and `/sbin` directories ( *binary* ), *configuration files are located in the /etc folder ( etc ), and mutable files, such as logs, should be located in the /var directory ( variable )*.

The fourth difference is that Linux file systems are traditionally case-sensitive, meaning that the same directory can contain files `test.txt` , `Test.txt` , and `TEST.txt` , which will be considered completely different files.

If you mount a drive formatted with NTFS or exFAT in Linux, you'll only be able to create one of these files and access it using any of the suggested naming options, just like in Windows. This is a feature of the file system, not the operating system.

Let's look at the fifth difference, which is the use of links.

Although the `mklink` command for creating links has been available in Windows since the NT era, most Windows users limit themselves to using shortcuts, which are regular `*.lnk` files that contain settings for quick access to files and folders. `*.desktop` files are the complete equivalent of `*.lnk` files in Linux .

In Linux, hardlinks and softlinks are widely used, so changes to one file can be reflected in multiple places at once, which can be very confusing at first.

For example, using the commands `stat/run/cups/cups.sock` and `stat/var/run/cups/cups.sock` , you can verify that these files point to the same inode number. This is because the `/var/run` folder is a soft link to the `/run` folder , which can be verified with the command `ls -l/var/run` . We'll see later that this was done for backward compatibility.

It's even more interesting when multiple hard links point to the same file. For example, using the commands `stat/usr/bin/gunzip` and `stat/usr/bin/uncompress` , you can easily verify that these are indeed two different hard links pointing to the same inode. Therefore, until we remove both links, the file will physically remain on the disk.

Finally, let's remember that in Linux, the forward slash `/` (slash) is used as a separator between file system object names, for example, `/etc/hosts` . In Windows, you can use a slash, but the backslash `\` (backslash) is more traditional, for example, `C:\Windows\System32\drivers\etc\hosts` .

## **7.2.2 Linux file system structure**

In Linux, the system drive is mounted directly to the operating system, and the mount point is the forward slash, also known as the root. The system drive's file system is therefore called the root file system. It is to the folders of the root file system that the file systems of other devices are subsequently mounted.

To ensure we always know which directories to look for the files we need among the countless "net analogs" created on top of Linux, developers needed to streamline the root system structure and created a unified file system standard (Filesystem Hierarchy Standard, FHS). It's a very strange standard, one that almost everyone adheres to, but no one fully follows.

The reason for this state of affairs is that some of the FHS's ideas are already outdated, but the community is wary of making drastic changes and continues to move forward by inertia. Incidentally, the GoboLinux developers made a bold attempt to change this status quo, but their laudable efforts were met with ridicule and condemnation.

Therefore, when studying the FHS, some aspects are best accepted "as is," while others will only become clear in a strictly historical context. And if you think the Windows directory structure is free of such atavisms, you simply haven't looked at it through the eyes of the developers yet!

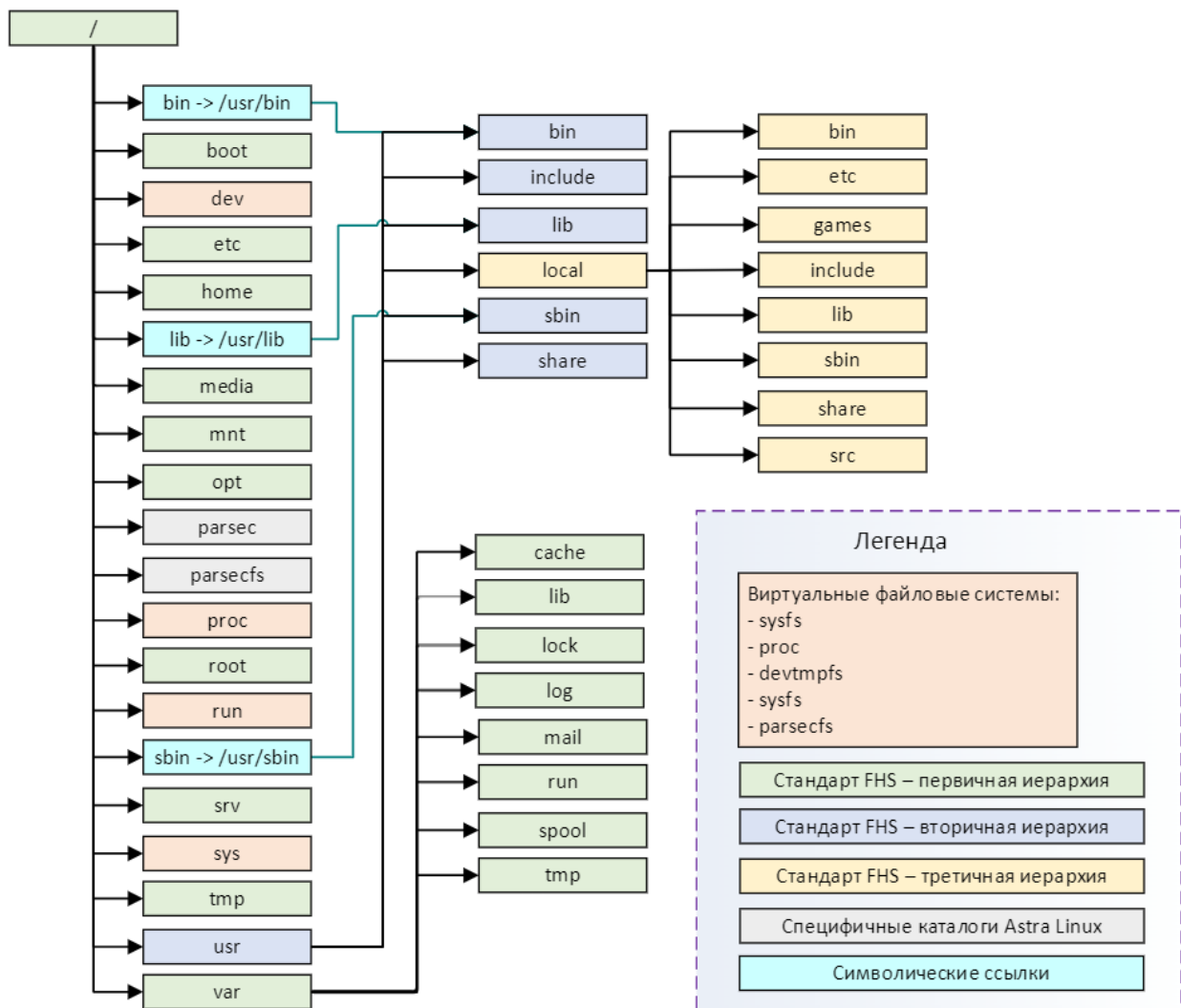


Figure 7.1 - Root file system in Astra Linux

Let's look at the main directories of the system, not in alphabetical order, so that our story is more complete:

- / is the root directory of the system, the system disk is mounted at this point.
- /boot – contains GRUB bootloader configuration files, system kernel images, and initrd temporary file system files used by the Linux kernel during initial boot.
- /dev is a directory in the devtmpfs pseudo-filesystem that contains device files . This directory exemplifies the "everything is a file" concept, so no such directory exists in Windows.

All devices can be divided into two groups:

- o **Character devices** provide a sequential input/output stream. For example, a mouse is a character device, and each button press results in a character being sent to the file /dev/input/mouse0 , which can be verified

by cat ing this file.

- o **Block devices** provide random access to data. For example, a hard drive is a block device because it allows reading and writing data in random order.
- /media – a point for automatically mounting removable media such as FDDs, CD-ROMs, and DVD-ROMs. Flash drives do not fall into this category – remember that external USB drives are mounted to /run/user/<UID>/media/ using the FUSE module (from filesystemuserspace).
- /mnt – a point for manually mounting external media (from the English word “mount” – to mount).
- /proc – the procfs virtual file system is mounted here. It can be used to obtain information about the operating system kernel state and running processes (from the English “process”). For example, the ps command obtains its data through procfs files.

Note that some procfs files can be manipulated to change the kernel configuration. For example, by writing a new value to the /proc/sys/kernel/hostname file , you can change the hostname on the fly.

- /sys - the virtual file system sysfs is mounted at this point, which is a new implementation of procfs and gradually replaces its functions ( *from the English “system”* ).
- /run – This directory contains process PID files, temporary socket files, and other data describing the system since boot. This directory is located in TMPFS, so it is automatically cleared upon system reboot.

Previously, the /var/run directory was used for the same purposes, which is still in the system for backward compatibility, but is now a symbolic link to the /run directory.

- /bin – contains binary files of standard Linux utilities that should be available to all users of the system ( *from the English word “binaries”* ). In Windows, such files have the \*.exe extension and are located in the C:\Windows\System32 directory. This directory is a soft link to the /usr/bin directory, as will be discussed later.
- /sbin – This directory contains a set of system utilities necessary for booting the system and restoring it in the event of a failure, such as init, iptables, and ifconfig ( *from systembinaries* ). Only root has permission to run these utilities. This directory is a soft link to /usr/sbin, as discussed below.
- /lib – This directory contains dynamic libraries needed to run applications in the /bin/ and /sbin directories ( *from the English “library”* ). This directory is a soft link to the /usr/lib directory, as will be discussed later in the text.
- /usr/bin, /usr/sbin, and /usr/lib are directories referenced by /bin, /sbin, and /lib. The reason for the existence of two copies of these directories can only be understood in the context of the development of computing technology.

Long ago, when hard drives were only a couple of megabytes in size, the developers of the Unix operating system quickly reached this limit and had to move some files from the bin,/sbin, and lib directories to the user drive (hence the name /usr , from “user”). The criterion for separating files into directories was simple: those files that weren't needed by the system before the user drive was mounted were moved to

the /usr directory .

Subsequently, disk sizes increased exponentially, making this solution obsolete. However, to ensure backward compatibility, the catalogs were retained and soft-linked. So, this atavism is now over 35 years old. And it's unlikely to change anytime soon.

- /etc – contains configuration files for the operating system and all its services (from the Latin "etcetera" – "and others," "and so on"). This directory can be compared to **the Windows registry** , which was developed to overcome the limitations of FAT16, such as the lack of access rights and slow access to a large number of small INI files containing application settings.
- /var– contains files that are subject to the most frequent changes ( *from the English variable* ):
  - /var/cache - This directory is used to store cache files whose generation requires significant computing resources and/or I/O operations. Deleting these files should not result in data loss.
  - /var/lib – contains files used to pass application state information between its invocations or to exchange information between multiple instances of the application running simultaneously. For example, this directory contains the SSSD service files after installation.
  - /var/lock – lock files indicating the use of devices and other resources. These files are necessary to allow multiple applications to use the same resource.
  - /var/log – directory contains log files.
  - /var/mail – This directory contains local users' mailboxes in the legacy mbox format, which is a plain text file. To send messages to local users, the sendmail utility must be installed on the system. If the pam\_mail module is configured, users will receive "youhavenewmail" notifications when logging into the terminal. Local mail is, to some extent, analogous to the Windows msg.exe utility.
  - /var/run is a legacy directory that is left for backward compatibility and is now a symbolic link to the /run directory.
  - /var/spool – tasks waiting to be processed (for example, a print queue or an incoming/outgoing mail queue).
  - /var/tmp – temporary files that should be saved between system reboots.
- /opt – the directory is intended for placing additional applications that deviate from the FHS standard and store all their files in one place ( *from the English "optional" - additional* ).

Each application is allocated a separate /opt/<package> directory , which may contain bin, lib, man, and other directories for storing executable files, libraries, and manual pages, respectively. Installation of such applications is as simple as copying a single folder and can be accomplished even without using a package manager. Thus, the /opt directory is the closest equivalent to the c:\Programfiles directory in Windows.

- /srv – the directory used to store network service files ( *from the English word "service"* ). For example, if the host is a web server, this directory might contain

a www folder containing files for the websites it serves.

- /tmp is the directory where temporary files are stored. Linux, unlike Windows, monitors temporary files and regularly cleans this directory.
- /home – contains the home directories of standard users, where personal files, program settings, and other items are stored. In Windows, the C:\Users directory serves the same purpose.
- /root is the superuser's home directory. At this point, an inquisitive comrade might wonder why the /root directory is located in the root directory and not in the /home folder. We hasten to authoritatively answer that it's another artifact from the past.

A long time ago, when hard drives were the size of... well, you get the idea. Generally, a single hard drive was barely enough to store operating system files, so the /home directory was almost always mounted from a separate hard drive. But to ensure that the superuser didn't lose access to their home directory even when the user drive wasn't mounted, their home directory was located separately in the system root. As the saying goes, "Your own flash drive is more useful."

- /usr – This directory is designed to store application programs and can be mounted over the network. It contains common files that can be shared by multiple computers at once, so it shouldn't contain any host-specific files, and users shouldn't have write access to this directory. In Windows, to achieve the same effect, we can mount a network folder containing an archive of useful applications that don't require installation.

In the root directory of Astra Linux, there are two more directories specific to this operating system: parsec and parsecfs. These are related to the PARSEC security subsystem, which is part of the OS kernel. However, these topics are beyond the scope of this course.

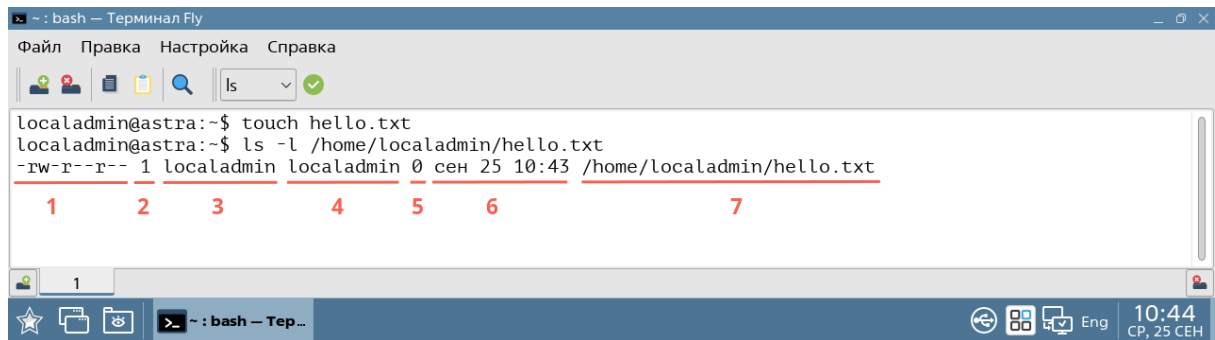
Windows and Linux file systems use slightly different approaches to organizing data storage on disk, understanding which will help you avoid problems in the future.

In Windows, the primary file system is NTFS. The disk is divided into clusters, and a special database called the Master File Table (MFT) is used to store metadata about files. Files correspond to rows in this table, or so-called records (MFT records), and the table columns store the values of specific file attributes.

As new files are created, the MFT size increases, and this database file can become fragmented. To avoid system performance degradation due to MFT fragmentation, the system reserves 12.5% of the disk space for the MFT when formatting the disk and retains it until all free disk space is exhausted.

*from English "information node"), which is analogous to MFT records, is used to store metadata about files .*

`ls -l /home/localadmin/hello.txt` command is based on the data in the inode:



```
localadmin@astra:~$ touch hello.txt
localadmin@astra:~$ ls -l /home/localadmin/hello.txt
-rw-r--r-- 1 localadmin localadmin 0 сен 25 10:43 /home/localadmin/hello.txt
```

1 2 3 4 5 6 7

Figure 7.2 - Output of the ls command

1. File permissions. These will be discussed in detail in a separate module of this course.
2. The number of hard links to a file – more on that later.
3. The user who owns the file.
4. The group that is the owner.
5. File size in bytes.
6. Date the file was last modified.
7. Full path to the file.

And using the `-i` switch, the `ls` utility can show the index descriptor number:

```
ls -i /home/localadmin/hello.txt
424373 /home/localadmin/hello.txt
```

Using the `stat` command, you can obtain a number of additional file attributes stored in its inode. One advantage of the `stat` command over `ls` is that it displays file permissions numerically. For example, in the example below, the file `hello.txt` has permissions set to `0644`:

```
stat /home/localadmin/hello.txt
File : hello.txt
Size: 14 Blocks: 8 Block I/O: 4096 regular file
Device: 801h/2049d Inode: 671808 Links: 1
Access : (0644/-rw-r--r-- ) Uid : (1000/localadmin) Gid: (1000/localadmin)
Access: 2024-09-25 11:05:15.816000000 +0300
Modified: 2024-09-25 11:05:22.532000000 +0300
Modified: 2024-09-25 11:05:22.532000000 +0300
Created: -
```

`stat` utility in current ALSE editions does not show the file creation date, but you can easily find it in the graphical File Manager utility.

By the way, you can open fly-fm in the current directory with the command: `fly-fm .`, where the period symbol is the current directory, which will be passed as the default argument. You can also open the current directory in VS Code with `code-n .`

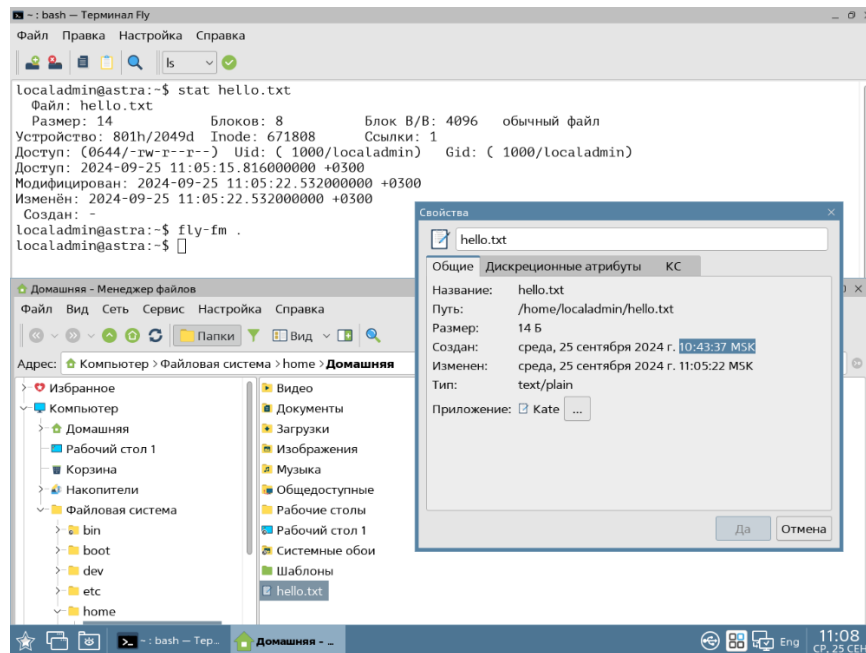


Figure 7.3 - File Manager

However, unlike the MFT, which can change its size dynamically, the number of inodes is static and is determined only when the disk is formatted. This is the cause of a common Linux problem: an application burns up all available inodes, creating a large number of small files, for example, in the log directory. In this case, there's plenty of free space left, but creating a new file won't work. Let's see what can be done about this.

If we connect an additional 32 GB disk to a virtual machine, then create a partition on the disk using the `fdisk` utility and format this partition as `ext4` using the `mkfs` utility with default parameters, we will get 8 million 4 KB blocks and 2 million index descriptors.

```
lsblk
...
fdisk/dev/sdb
...
mkfs-text4/dev/sdb1
...
Creating filesystem with 8388352 4k blocks and 2097152 inodes
...
```

The default block size is 4 KB, and the number of inodes is determined at a rate of 1 inode per 16,384 bytes of disk space. See the `blocksize` and `inode_ratio` parameters in the `/etc/mke2fs.conf` configuration file. You can modify this configuration file or explicitly set the block size and desired number of inodes when formatting the disk using the `-b` and `-N` switches:

```
mkfs-text4-b 1024 -N 5000000 /dev/sdb1
...
Creating filesystem with 33553408 1k blocks and 5000000 inodes
...
```

The XFS (X File System) file system dynamically allocates inode blocks, eliminating the problem of inode exhaustion. If you're using the more traditional EXT4 file system, you should determine optimal disk parameters during the planning stage and periodically monitor inode usage statistics.

You can find out the number of occupied and free inodes using the `df-i` command

```
.
df-i
File System Inodes IUsed IFree IUsed% Mounted in
udev 240770 484 240286 1% /dev
tmpfs 253151 678 252473 1% /run
/dev/sda1 987360 226603 760757 23% /
tmpfs 253151 3 253148 1% /dev/shm
tmpfs 253151 4 253147 1% /run/lock
tmpfs 253151 19 253132 1% /sys/fs/cgroup
tmpfs 253151 10 253141 1% /run/user/107
tmpfs 253151 38 253113 1% /run/user/1000
```

Another difference is the way file names are stored. While in Windows the file name is part of the MFT record, in Linux the file name is stored using hard links.

The point is that an inode is a fixed-size data structure (256 bytes for ext4) that can only store a limited set of attributes: file type, size, creation/modification date, owner and group information, basic rwx permissions, and, of course, references to data blocks. Therefore, an inode is poorly suited for storing names of arbitrary length. However, special directory files are ideal for this purpose. Directory files contain so-called hard links, which associate a file name with an inode number. An inode can have several such links, but they are all equivalent, so a file will only be deleted from disk if all its hard links are deleted. To facilitate the decision-making process regarding file deletion, an inode has an attribute that the file system uses to keep track of the number of hard links. When a new link is created, the counter is incremented, when a link is deleted, it is decremented, and if the counter reaches zero, the file is deleted.

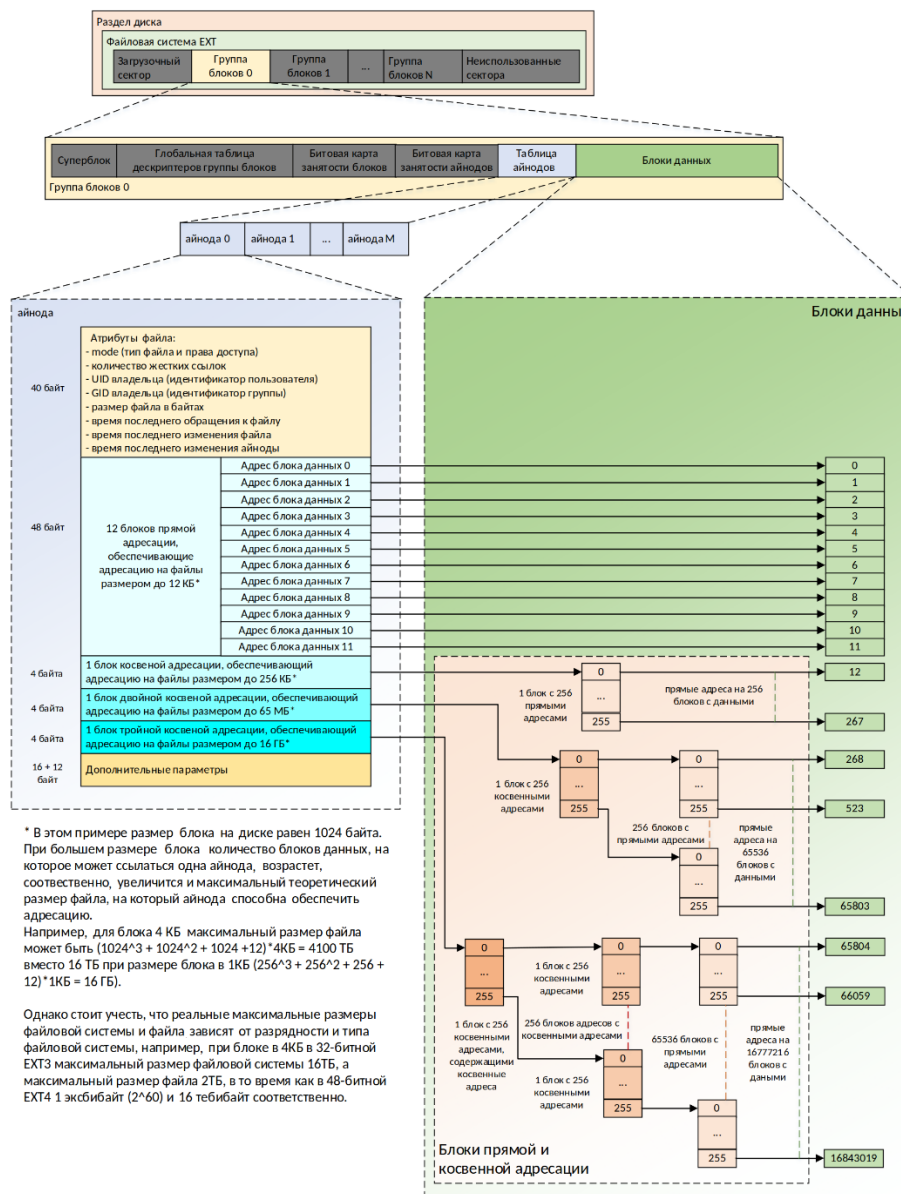


Figure 7.4 - Index descriptor structure and addressing scheme

Let's look at how the file system works with inodes (see Figure 7.4). If we want to read a file, the system takes the inode number from the hard link, reads the file's metadata, performs a permissions check, and returns the file's contents by reading them from disk using data block references. In cases where the file size fits within 12 blocks, the system will suffice with so-called direct references. For larger files, there are several indirect references that allow data blocks to be used to store references. This approach allows addressing files up to 16 terabytes in size with a standard block size of 4 KB. The structure of the inode and the data block addressing scheme are shown in the figure above.

There are seven types of inodes in Linux, five of which are classified as special files, see the table below.

| File type               | Designation in ls | Purpose                                                                                                                                        |
|-------------------------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Regular files</b>    | - rwxrwxrwx       | They contain various data, such as text, images, archives, scripts, libraries, executable files, etc.                                          |
| <b>Catalogs</b>         | d rwxrwxrwx       | Contain hard links that associate file names with inodes.                                                                                      |
| <b>Special files:</b>   |                   |                                                                                                                                                |
| <b>Symbolic links</b>   | l rwxrwxrwx       | Provides an alternative name for accessing files located anywhere in the file system, including on other media (similar to mklink in Windows). |
| <b>Block devices</b>    | b rwxrwxrwx       | Provide an interface for interacting with computer hardware.                                                                                   |
| <b>Symbolic devices</b> | c rwxrwxrwx       |                                                                                                                                                |
| <b>Channels</b>         | p rwxrwxrwx       | Provides interprocess communication mechanisms in Linux.                                                                                       |
| <b>Sockets</b>          | s rwxrwxrwx       |                                                                                                                                                |

Regular files include all files containing data, such as text, images, archives, scripts, executable files, libraries, etc. The shorthand for regular files is the hyphen symbol "-".

Let's open a terminal and enter the `ls -l` command, familiar from our terminal module. In that same module, we created two files—`script` and `hello.txt`:

```
ls -l
total 5872
lrwxrwxrwx 1 localadminlocaladmin 17 May 23 16:35 Desktop ->
Desktops/Desktop1
drwx----- 6 localadminlocaladmin 4096 May 16 12:08 Desktops
-rw-r--r-- 1 localadminlocaladmin 31 May 26 12:44 hello.txt
-rwxr--r-- 1 localadminlocaladmin 117 May 25 11:09 script
lrwxrwxrwx 1 localadminlocaladmin 21 May 16 12:08 SystemWallpapers ->
/usr/share/wallpapers
-r-xr-xr-x 1 localadminlocaladmin 5968312 May 20 08:48
VBoxLinuxAdditions.run
drwxr-xr-x 2 localadminlocaladmin 4096 May 16 12:08 Video
drwxr-xr-x 2 localadminlocaladmin 4096 May 16 12:08 Documents
drwxr-xr-x 2 localadminlocaladmin 4096 May 16 12:08 Downloads
drwxr-xr-x 3 localadminlocaladmin 4096 May 16 12:08 Images
drwxr-xr-x 2 localadminlocaladmin 4096 May 16 12:08 Music
drwxr-xr-x 2 localadminlocaladmin 4096 May 16 12:08 Public
drwxr-xr-x 2 localadminlocaladmin 4096 May 16 12:08 Templates
```

In the command output, we see several file types, which can be identified by the line color and the first character. As you can see, symbolic links are highlighted in turquoise, directories are blue, executable files are green, and regular files are displayed in the default color.

In Linux, a directory is a special file that stores a list of hard links linking a child file name to an inode number. Regular text editors can only view the contents of these files; direct editing of directory files is prohibited. Editing of these files is handled by the operating system kernel, which receives commands to create, delete, and rename files. To view the contents of a directory, you can use the `ls`, `dir`, or `tree` utilities.

`tree` utility displays the directory structure in a pseudo-graphical format similar

to the tree output in cmd. To install it, run `sudo apt install tree` .

A symbolic link, or symlink, is a special file that contains a reference to another file's address by its name, rather than its inode. Links will be discussed in more detail later in the text.

Device files are used to access computer hardware (drives, printers, terminals, etc.). When a device file is accessed, the operating system kernel passes the request to the device driver.

Character devices are accessed sequentially. A mouse is an example of a character device. Block devices allow data to be accessed in any order. A hard drive is an example of a block device.

Both channels and sockets are designed to facilitate communication between processes. The key difference between a channel and a socket is that a channel is unidirectional, while a socket allows data to be transferred in both directions simultaneously. An example of a socket is the print spooler file `/usr/lib/systemd/system/cups.socket` .

As we already know, Linux has two types of links: hardlinks *and* symbolic *links* . Furthermore, the Fly graphical shell uses `*.desktop` shortcuts, which are absolutely identical to `*.lnk shortcuts` in Windows.

It's worth noting that the NTFS file system supports hard and symbolic links, as well as junction points ( *directory hardlinks and junctionpoints* ). To create them, you need to use `fsutil` , `mklink` , or PowerShell. However, in Windows, such links are rarely used.

In Linux, a hard link is an entry in a file directory that associates a file name with its inode number.

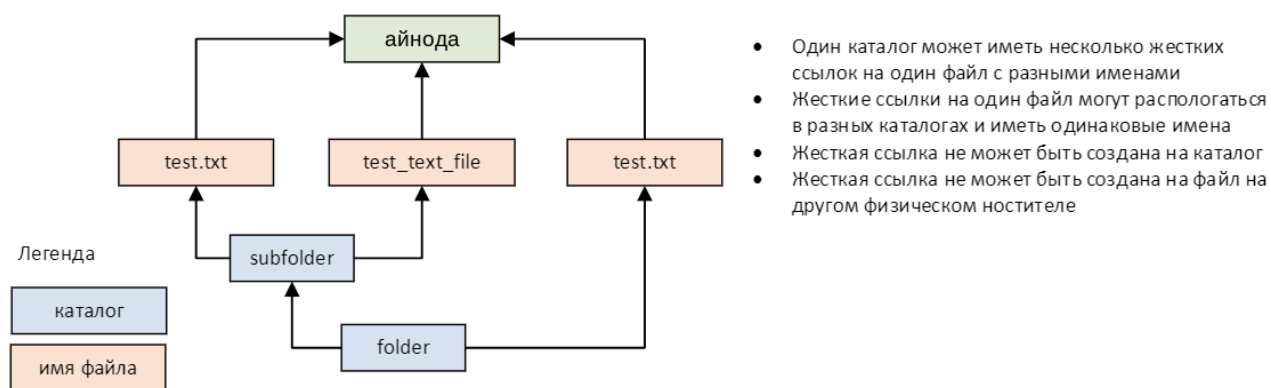


Figure 7.5 - Diagram showing the relationship between file names and the inode

Whenever a file or directory is created, a hard link is created for it, and the link count is set to one ( $0 + 1 = 1$ ). Let's run the `mkdir test` command to create the test folder, navigate to this folder with the `cd test` command , and run the following commands:

- `touch test1` to create the file `test1`
- `ls -il` to view the number of references to objects in the current directory and their inode numbers.

```
mkdir test
cd test
```

```
touchtest1
ls-il
total 0
453079 -rw-r--r-- 1 localadminlocaladmin 0 Jun 8 16:02 tes
```

Now let's create a hard link for this file with the command `ln test1 hardlink_test1`, and view the links again with `ls -il`:

```
ln test1 hardlink_test1
ls-il
total 0
453079 -rw-r--r-- 2 localadminlocaladmin 0 Jun 8 16:02 hardlink_test1
453079 -rw-r--r-- 2 localadminlocaladmin 0 Jun 8 16:02 test1
```

As you can see, the counter has increased by one. Let's create a subdirectory with the `mkdir subtest` command and create another hard link to this file in it with the command:

```
ln hardlink_test1 subtest/hardlink2_test1
```

When accessing a file via a hard link, we can use any name, since all links point to the same inode, and it's impossible to tell which hard link was created first and which was created later. You can verify this by running the command `ls -il subtest`.

Now let's update the file's last modification time using the `touch hardlink_test1` command and look again at the properties of the objects in test and subtest:

```
mkdirsubtest
lnhardlink_test1subtest/hardlink2_test1
ls-ilsubtest
total 0
453079 -rw-r--r-- 3 localadminlocaladmin 0 Jun 8 16:02 hardlink2_test1
touchhardlink_test1
ls-il
total 4
453079 -rw-r--r-- 3 localadminlocaladmin 0 Jun 8 16:11 hardlink_test1
453080 drwxr-xr-x 2 localadminlocaladmin 4096 Jun 8 16:08 subtest
453079 -rw-r--r-- 3 localadminlocaladmin 0 Jun 8 16:11 test1
ls-ilsubtest
total 0
453079 -rw-r--r-- 3 localadminlocaladmin 0 Jun 8 16:11 hardlink2_test1
```

When you delete a file with the `rm` command, the hard link to the file is removed and the hard link counter is decremented by one, but nothing happens to the data blocks. If the link counter reaches zero, the system deletes the file and, after a while, uses the freed data blocks to store other files.

Since a hard link is simply a reference to a file's inode, Linux prohibits creating hard links to inodes outside the device on which the inode resides, because the inode ID is unique only within the current filesystem.

The "." and ".." in relative paths are hard links to the current and parent directories, created by the operating system itself. Creating additional hard links to directories is prohibited, as this can lead to a loop where one directory links to a second directory, which in turn links to the first.

```
lnsubtesthardlink_subtest
ln: subtest: hard links to directories are not allowed
```

The `ln` command has a special switch `-d`, which allows you to create a hard link to a directory on some systems with superuser rights, but this operation is not allowed on Astra Linux:

```
sudo ln -d subtesthardlink_subtest
ln: Failed to create hard link 'hardlink_subtest' => 'subtest': Operation
```

not permitted

Unlike a hard link, a symbolic link allows you to reference files and directories on any device accessible by the root file system because it is a file that stores the full path to the file referenced by the link. Symbolic link inodes do not duplicate the attributes of the files they point to, so symbolic links can have their own access rights. The extended diagram of the relationship between file names and inodes illustrates the relationship between symbolic links and file names.

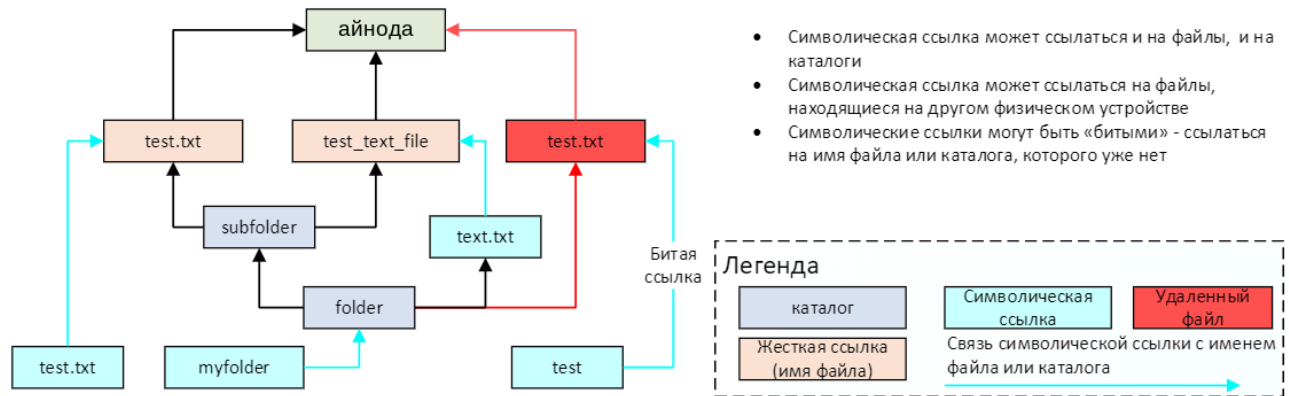


Figure 7.6 - Expanded diagram showing the relationship between symbolic links, file names, and the inode

A symbolic link refers to a path to a file or folder. If the object it refers to is deleted (moved), the link will point to a non-existent object, meaning it becomes a *broken link*.

Let's create several symbolic links to files and a folder using the `ln` command with the `-s` (`--symbolic`) switch:

- `ln -s ~/test/subtest ~/subtest` – creates a link `subtest` in the home directory to the `subtest` directory in the `test` directory.

```
cd ~/test
ln -s ~/test/subtest ~/subtest
ls -l ~/subtest
lrwxrwxrwx 1 localadminlocaladmin Jun 29 9 10:38 /home/localadmin/subtest -
>
/home/localadmin/test/subtest
ln -s test1 test1_link
ln -s hardlink_test1 hardlink_test1_link
ls -l
total 4
-rw-r--r-- 3 localadminlocaladmin 0 Jun 8 16:11 hardlink_test1
lrwxrwxrwx 1 localadminlocaladmin 14 Jun 9 10:41 hardlink_test1_link ->
hardlink_test1
drwxr-xr-x 2 localadminlocaladmin 4096 Jun 8 16:08 subtest
-rw-r--r-- 3 localadminlocaladmin 0 Jun 8 16:11 test1
lrwxrwxrwx 1 localadminlocaladmin 5 Jun 9 10:40 test1_link -> test1
```

Let's try accessing files and directories via symbolic links. Run the commands `ls ~/subtest`, `echo "testtext" >hardlink_test1_link`, and `cat test1_link`:

```
ls ~/subtest
hardlink2_test1
echo "test text" >hardlink_test1_link
cat test1_link
test text
cat test1
```

```
testtext
```

As we can see, you can work with links in exactly the same way as with regular files.

Let's delete the `hardlink_test1` hardlink using the `rm hardlink_test1` command and check access to the file through the hard and symbolic links using the `cat hardlink_test1_link`, `cat test1_link`, `cat test1` commands :

```
rmhardlink_test1
ls-l
total 8
lrwxrwxrwx 1 localadminlocaladmin 14 Jun 9 10:41 hardlink_test1_link ->
hardlink_test1
drwxr-xr-x 2 localadminlocaladmin 4096 Jun 8 16:08 subtest
- rw- r--r-- 2 localadminlocaladmin 10 Jun 9 10:50 test1
lrwxrwxrwx 1 localadminlocaladmin 5 Jun 9 10:40 test1_link -> test1
cathardlink_test1_link
cat: hardlink_test1_link: No such file or directory
cattest 1_ link
test text
cattest 1
testtext
```

As we can see, after deleting the hard link, the `ls` command output highlights the broken link in red, and when attempting to read the file, we receive a message that the file does not exist. However, the file is still accessible via a symbolic link to a different filename, as is the other hard link.

Finally, we note that when deleting a symbolic link, the hard link to the file and the data itself are not deleted.

The naming rules for files and directories (remember, a directory is also a file) are identical. A name cannot exceed 255 bytes, and the total path length (including all components) cannot exceed 4096 bytes.

The minimum safe character set to use in file names is:

- letters (both Latin and Cyrillic in any register);
- numbers;
- symbol "." (dot);
- the symbol "\_" (underscore);
- symbol "-" (dash).

If a file or directory name begins with a period (.), it will be considered hidden in Linux. For example, a user's working directory contains files such as `.bashrc`, `.profile`, and others that are not normally displayed. To see all files, including hidden ones, use the `ls` command with the `-a` switch.

Using special characters, such as spaces, the symbols "\*", "?", "\$", "'", "" is not recommended, since their presence in file names can cause some programs to work incorrectly, although it is permissible to use any characters except "/", for example, the command `touchtest\file.txt` will create a file in the current directory whose name will contain a space:

```
touchtest \ file.txt
ls
... 'test file.txt' ...
```

There are several ways to solve problems with special characters in file names when working in the command line:

- **Escaping special characters.** To escape special characters, precede them

with a backslash "\" or enclose the entire file name in single quotation marks, for example, `cat test\\file.txt` or `cat 'testfile.txt'` :

```
echo text>test \ file.txt
cat 'test file.txt'
text
```

- Tab completion **will complete** the path and automatically escape characters:

```
cat test\ file.txt
```

- **Working with files with a dash at the beginning of their name.** If a filename begins with a dash , the command interpreter will treat it as a command parameter. Standard escaping won't help. Instead, you can use either a relative path to the file or the special parameter `--` , which tells the command that this is a parameter, not a key.

```
echo text> '-test.txt'
```

```
cat-test.txt
cat: invalid key - "."
```

```
cat \- test.txt
cat: invalid key - "."
```

```
cat '-test.txt'
cat: invalid key - "."
```

```
cat---test.txt
text
```

```
cat ./- test .txt
text
```

Linux filenames, like Windows ones, can use extensions or so-called suffixes, such as `*.txt` or `*.doc` . However, while in Windows extensions play a key role in determining the file type, in Linux they are ignored, which is why it is sometimes said that Linux has no extensions.

A file's type in Linux is determined by its signature, a few bytes at the beginning of the file that uniquely identify the file type and thus serve as its signature. For this reason, renaming the image `logo.png` to the text file `logo.txt` will not fool the file utility , which will continue to treat it as a color PNG image:

```
wgethttps://astralinux.ru/logo.png
--2024-09-25 11:21:05 -- https://astralinux.ru/logo.png
Recognized as astralinux.ru (astralinux.ru)...84.201.172.233
Connecting to astralinux.ru (astralinux.ru)|84.201.172.233|:443...
connection established.
HTTP request sent. Waiting for response... 200 OK
Length : 35129 (34K) [image/png]
Save as: "logo.png"

logo.png 100% [=====>] 34.31K --.-KB/s for
0.001s
```

```
2024-09-25 11:21:05 (26.3 MB/s) - "logo.png" saved [35129/35129]
```

```
filelogo.png
logo.png: PNG image data, 2969 x 789, 8-bit/color RGBA, non-interlaced
mvlogo.pnglogo.txt
filelogo.txt
logo.txt: PNG image data, 2969 x 789, 8-bit/color RGBA, non-interlaced
```

However, it's worth noting that the Astra Linux file manager handles extensions

in a similar way to Windows Explorer, so it's recommended to stick to generally accepted conventions so that files can be opened from the graphic by double-clicking.

A file system path is a string of characters that indicates the location of a file or directory within the file system. A path can be absolute or relative. An absolute, or full, path points to the same location in the file system regardless of the current working directory or other circumstances, so a full path always starts with the root directory. A relative path is a path relative to the current working directory of the user or active applications.

Relative paths can use wildcards:

- The period "." at the beginning of a path indicates the current directory. For example, entering the command `ls ./Desktop` will display the contents of the directory `/home/localadmin/Desktop`. You can verify this by entering the command with the absolute path `ls /home/localadmin/Desktop`:

```
ls./Desktop
firefox.desktopfly-help.desktopmycomp.desktopmytrash.desktop
ls/home/localadmin/Desktop
firefox.desktop fly -help.desktopmycomp.desktopmytrash.desktop
```

- The two dots ".." indicate that the system needs to go up one level to find the file. For example, if these characters are at the very beginning of the path, they point to the parent directory. Entering the command `ls ..` will display the contents of the parent directory `/home/`:

```
ls..
localadmin
```

- The "~" symbol represents the full path to the user's home directory. We'll enter `cd /` to go to root, and then `cd ~` to go back to the home directory. We can use the `pwd` command to check which directory we're in, but this isn't necessary—we can see the current directory in the prompt (in this case, the "~" symbol was changed to "/" and then back to "~"):

```
cd /
pwd
/
cd ~
pwd
/home/localadmin
```

### 7.2.3 Working with files and directories

The main commands include the following:

- `pwd` - print the full path to the current directory ( *from the English printworkingdirectory - print the working directory* ).
- `cd [<directory>]` - go to the specified directory ( *from English changedirectory - change directory* ).
- `mkdir <directory>` - create a directory.
- `rmdir <directory>` - delete an empty directory ( *from the English remove - delete* ).
- `rm -rf <directory>` - delete a directory with all its contents.
- `du [<directory>]` - display the directory size ( *from English diskusage -*

*disk usage* ).

Let's look at them in more detail with examples.

### The pwd command

Displays the full path to the current directory, for example:

```
pwd
/home/localadmin
```

### The ls command

Lists the contents of a directory. The command has quite a few switches; a full list can be found in the `man ls` help file .

Let's look at the main ones:

- `ls -a` ( `ls --all` ) does not hide files starting with a dot "." when outputting.
- `ls -A` ( `ls --almost-all` ) lists all files (including hidden ones) except "." and ".."
- `ls -l` – in addition to the name of each file, additional information is displayed: file type, file permissions, number of links to the file, owner name, group name, file size in bytes, and timestamp (the time the file was last modified, unless otherwise specified).
- `ls -R` – displays information recursively from all subdirectories.

### **Examples of use :**

```
ls-a
. .. hardlink_test1_ link subtest test1 test1_link
ls-A
hardlink_test1_ link subtest test1 test1_link
ls-l
total 8
lrwxrwxrwx 1 localadminlocaladmin 14 Jun 9 10:41 hardlink_test1_link
hardlink_test1
drwxr-xr-x 2 localadminlocaladmin 4096 Jun 8 16:08 subtest
- rw- r--r-- 2 localadminlocaladmin 10 Jun 9 10:50 test1
lrwxrwxrwx 1 localadminlocaladmin 5 Jun 9 10:40 test1_link -> test1
ls-la
total 16
drwxr-xr- x 3localadminlocaladmin 4096 Jun 9 10:55 .
drwx----- 18 localadminlocaladmin 4096 Jun 9 10:38 ..
lrwxrwxrwx 1localadminlocaladmin 14 Jun 9 10:41 hardlink_test1_link
hardlink_test1
drwxr-xr- x 2localadminlocaladmin 4096 Jun 8 16:08 subtest
- rw- r--r-- 2localadminlocaladmin 10 Jun 9 10:50 test1
lrwxrwxrwx 1localadminlocaladmin 5 Jun 9 10:40 test1_link -> test1
ls-R
.:
hardlink_test1_ link subtest test1 test1_link
./subtest:
hardlink2_test1
```

### The cd command

Changes to the specified directory. It's convenient to use relative paths and the "-" symbol, which denotes the previous directory from which the change was made.

- `cd` – go to the user's home directory
- `cd ..` – go to the parent directory

### **Examples of use:**

```
cd
cd test
cd -
```

```
/home/localadmin
cd /
cd ~/test/subtest
cd ..
```

### The mkdir command

The mkdir command creates directories.

- `mkdir test2` – will create a directory test2 in the current directory.
- `mkdir -p test3/subtest/subsubtest` – will create the subsubtest directory and all missing parent directories in the path before it.

#### **Examples of use:**

```
ls-l | grep^d | grep test
drwxr-xr-x 3 localadminlocaladmin 4096 Jun 9 10:55 test
mkdirtest2
ls-l | grep^d | grep test
drwxr-xr-x 3 localadminlocaladmin 4096 Jun 9 10:55 test
drwxr-xr-x 2 localadminlocaladmin 4096 Jun 9 12:46 test2
mkdir~/test3/subtest/subsubtest
mkdir: cannot create directory '/home/localadmin/test3/subtest/subsubtest':
No such file or directory
mkdir-p~/test3/subtest/subsubtest
treetest3
test3
├─ subtest
└─ subsubtest
2 directories, 0 files
```

### rmdir command

Deletes directories if they are empty .

- `rmdir test2` – deletes the test2 folder if it's empty. Otherwise, it returns an error.

#### **Examples of use:**

```
rmdirtest2
rmdirtest3
rmdir: failed to remove 'test3': Directory is not empty
```

### The rm -rf command

`rm-rf<path>` command deletes a directory along with its contents. The `-r` switch specifies that subdirectories should be recursively deleted, and the `-f` switch specifies that no confirmation should be requested from the user. For example, the `rm-rftest3` command will delete the `test3` directory and all its contents without further confirmation.

#### **Examples of use:**

```
rm-rftest3
```

This is one of the most dangerous commands in Linux. If used carelessly, especially with root privileges, it can permanently delete some data or the entire root filesystem, including information on mounted drives ( `rm -rf /` ).

The basic commands for working with files include the following:

- `touch <file>` - create a file or change timestamps of an existing file.
- `realpath <file>` - find out the absolute path to a file.
- `file <file>` - define the file type.
- `stat <file>` - get information about a file (file size, file creation date, etc.) and check the file's existence.
- `cp <file1> <file2>` - copies file "file1" to file "file2" ( *from the English*

word "copy" ). If a file with the same name already exists, it will be replaced.

- mv <file1> <file2> - rename file "file1" to "file2" ( *from English move - move* ).
- mv <file> <directory> - move a file to a directory.
- rm <file> - delete a file ( *from English remove - delete* ).
- find <file> - find a file.
- sh <file> - run a file with a Bash script.
- ./file - run the executable file.

Let's look at them in more detail with examples.

### Touch command

The command's primary function is to update a file's timestamps. However, if you need to create a new, empty file, you can use this command as well.

- touch – updates the timestamps of an existing file or creates a new file.
- touch -a – changes only the file's last access time (the time the file was last read).
- touch -m – changes only the file's last modification time (the time the file's contents were last modified).
- touch -d"STRING" – can set the time, including in free form, for example, "nextWednesday", more details `man touch` .
- touch -r<referencefile> <newfile> – copies the date and time values from the specified file.

### **Examples of use:**

Preparing for a test catalog demonstration

```
mkdirtest 4
```

```
CD Test 4
```

```
ls
```

Creating a file

```
touchtest .txt
```

```
ls-l
```

```
total 0
```

```
-rw-r--r-- 1 localadminlocaladmin 0 Jun 13 18:37 test.txt
```

```
stattest .txt
```

```
File : test.txt
```

```
Size: 0 Blocks: 0 Block I/O: 4096 empty regular file
```

```
Device: 801h/2049d Inode: 442812 Links: 1
```

```
Access : (0644/-rw-r--r-- ) Uid : (1000/localadmin) Gid: (1000/localadmin)
```

```
Access: 2023-06-13 18:37:01.960076695 +0300
```

```
Modified: 2023-06-13 18:37:01.960076695 +0300
```

```
Modified: 2023-06-13 18:37:01.960076695 +0300
```

```
Created: -
```

Updating file timestamps:

```
touchtest.txt
```

```
stattest.txt
```

```
File : test.txt
```

```
Size: 0 Blocks: 0 Block I/O: 4096 empty regular file
```

```
Device: 801h/2049d Inode: 442812 Links: 1
```

```
Access : (0644/-rw-r--r-- ) Uid : (1000/localadmin) Gid: ( 1000/localadmin)
```

```
Access: 2023-06-13 18:38:35.536075019 +0300
```

```
Modified: 2023-06-13 18:38:35.536075019 +0300
```

```
Modified: 2023-06-13 18:38:35.536075019 +0300
```

```
Created: -
```

Updating last access timestamp:

touch-atest.txt

stattest.txt

File : test.txt

Size: 0 Blocks: 0 Block I/O: 4096 empty regular file

Device: 801h/2049d Inode: 442812 Links: 1

Access : (0644/-rw-r--r-- ) Uid : (1000/localadmin) Gid: (1000/localadmin)

Access: 2023-06-13 18:39:09.220074416 +0300

Modified: 2023-06-13 18:38:35.536075019 +0300

Modified: 2023-06-13 18:39:09.220074416 +0300

Created: -

Last modification timestamp update:

touch-mtest.txt

stattest.txt

File : test.txt

Size: 0 Blocks: 0 Block I/O: 4096 empty regular file

Device: 801h/2049d Inode: 442812 Links: 1

Access : (0644/-rw-r--r-- ) Uid : (1000/localadmin) Gid: (1000/localadmin)

Access: 2023-06-13 18:39:09.220074416 +0300

Modified: 2023-06-13 18:39:28.928074063 +0300

Modified: 2023-06-13 18:39:28.928074063 +0300

Created: -

Update the last modification timestamp with free-form date selection

touch-d "next Wednesday" test.txt

stattest .txt

File : test.txt

Size: 0 Blocks: 0 Block I/O: 4096 empty regular file

Device: 801h/2049d Inode: 442812 Links: 1

Access : (0644/-rw-r--r-- ) Uid : (1000/localadmin) Gid: (1000/localadmin)

Access: 2023-06-14 00:00:00.000000000 +0300

Modified: 2023-06-14 00:00:00.000000000 +0300

Modified: 2023-06-13 18:43:22.472069880 +0300

Created: -

One might expect that, similar to the `mkdir` command , there would be a dedicated `mkfile` command . Indeed, Solaris OS does have such a command, but Linux does not. Instead, the `touch` command is commonly used . An empty file can also be created, for example, using the `>` operator ( `>file`) or using a text editor.

ls

emptyfile hardlink \_test1\_link subtest test1 test1\_link

>emptyfile

ls-l

total 8

-rw-r--r-- 1 localadminlocaladmin 0 Jun 13 18:30 emptyfile

lrwxrwxrwx 1 localadminlocaladmin 14 Jun 9 10:41 hardlink\_test1\_link ->

hardlink\_test1

drwxr-xr-x 2 localadminlocaladmin 4096 Jun 8 16:08 subtest

-rw-r--r-- 2 localadminlocaladmin 10 Jun 9 10:50 test1

lrwxrwxrwx 1 localadminlocaladmin 5 Jun 9 10:40 test1\_link -> test1

Realpath command

`realpath<file>` command displays the full path to the file.

**Examples of use:**

Output the full path to the file:

ls

test.txt

realpathtest.txt

/home/localadmin/test4/test.txt

Team file

The file `< file >` command defines the file type.

### Examples of use:

```
filehello .txt
hello .txt : UTF - 8 Unicode text
```

```
filescrip
```

```
script: Bourne-Again shell script, UTF-8 Unicode text executable
```

```
fileVBoxLinuxAdditions.run
```

```
VBoxLinuxAdditions.run: POSIX shell script executable (binary data)
```

```
file/usr/bin/ls
```

```
/usr/bin/ls: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically ...
```

```
file~/ Images /Wallpapers/800x600/ fly-default#5682a3.png
```

```
/home/localadmin/ Images /Wallpapers/800x600/fly-default#5682a3.png: PNG
image ...
```

### The stat command

stat command displays the attributes of an inode or file system status.

- stat<file> – displays the contents of the index descriptor attributes.
- stat -f<file> – displays the status of the file system on which the file is located.

### Examples of use:

Outputting information from a file descriptor:

```
stat test .txt
```

```
File : test.txt
```

```
Size: 0 Blocks: 0 Block I/O: 4096 empty regular file
```

```
Device: 801h/2049d Inode: 442812 Links: 1
```

```
Access : (0644/-rw-r--r-- ) Uid : (1000/localadmin) Gid: (1000/localadmin)
```

```
Access: 2023-06-14 00:00:00.000000000 +0300
```

```
Modified: 2023-06-14 00:00:00.000000000 +0300
```

```
Modified: 2023-06-13 18:50:34.144062150 +0300
```

```
Created: -
```

Display information about the file system on which the file is located:

```
stat - ftest .txt
```

```
File : " test.txt "
```

```
ID: 383b368255943a03 Name Length: 255 Type: ext2/ext3
```

```
Block size: 4096 Base block size: 4096
```

```
Blocks: Total: 3,865,570 Free: 2,036,428 Available: 1,835,136
```

```
Inodes: Total: 987,360 Free: 760,820
```

### The cp command

cp<source><destination> command copies files and directories, where *source* or multiple <sources> are copied to the specified <destination> :

- cpfile1file2 – copies *file1* to *file2* .
- cpfile1destination/ – copies *file1* to the *destination* directory .
- cp-rsource/destination ( -r , --recursive ) - Copies the source directory and all subdirectories to the destination directory if the destination directory exists . If the directory does not exist, it creates it and copies **the contents** of the source directory.
- cp-rTsourcedestination/ ( -r , --recursive ; -T , --no-target-directory ) - copies the contents of the source directory and all subdirectories to the destination directory. If the destination directory does not exist, it is created.
- cp-rsource/\*destination/ – copies the contents of the source directory and all subdirectories to the destination directory, if the destination

directory exists. If the directory doesn't exist, an error is returned.

- `cp-ifile1destination/ ( -i , --interactive )` - copies file1 to a directory, prompting for confirmation of overwriting. The directory must exist.
- `cp-nfile1destination` - the `-n` switch prevents overwriting existing files.
- `cp-bfile1destination` - the `-b` switch creates a backup copy of the file.
- `cpslinkdestination` - copies the original file, not the slink symbolic link.
- `cphlinkdestination` - copies not the hlink hard link, but the original file.
- `cp-Pslinkdestination` - copies a symbolic link as a link.
- `cp--preserve=linksfile1_hlink1file1_hlink2_filedestination` - if file1\_hlink1 and file1\_hlink2 are hard links to the same inode, the utility will copy the source file and create two hard links for it in the destination directory.
- `cp-psource/\*destination` - copies the contents of the source directory to the destination directory, preserving the ownership and permissions information stored in the inode.
- `cp--preserve=timestampsfiledestination` - copies a file to the destination directory while preserving the original file's timestamps.
- `cp-asource/*destination` - copies the file system structure 1 to 1, including timestamps, symbolic and hard links.

### Examples of use:

Preparing to demonstrate a test directory with a set of files, links, and a subdirectory with files.

```
mkdir -p./test_cp/destination
cd test_cp
mkdir ./source
touch ./ file { 1 ..3 }
touch./source/file { 4 ..6 }
ln./file1./hlink_file1
ln-s $( realpathfile2 ) ./slink_file2
tree
.
```

```
├── destination
├── file1
├── file2
├── file3
├── hlink_file1
├── slink_file2 -> /home/localadmin/file2
├── source
│   ├── file4
│   ├── file5
│   └── file6
```

1 directory, 8 files

Copying a file to a file

[localadmin@astra](#):~/test\_cp\$ cp file1 file1\_1 [localadmin@astra](#):~/test\_cp\$ ls  
destination file1 file1\_1 file2 file3 hlink\_file1 slink\_file2

Copying a file to the destination directory. The directory must exist.

```
cpfile1destination
ls ./ destination /
file 1 file 4 file 5 file 6
```

Copying a directory to a directory must exist

```
cp-r source destination /
lsdestination /
```

file1 source

```
cp-r source destination2/  
lsdestination2  
file4 file5 file6
```

```
cp-r source destination2/  
lsdestination 2  
file 4 file 5 file 6 source
```

Copying the contents of a directory to a directory. The directory must exist.

Method 1.

```
cp-rT source destination/  
lsdestination/  
file1 file4 file5 file6 source
```

Deleting copied files

```
rm./destination/file { 4 ..6 }  
lsdestination/  
file1 source
```

Method 2.

```
cpsource/*destination/  
lsdestination /  
file1 file4 file5 file6 source
```

Copying a file to a directory, asking for confirmation to overwrite. The directory must exist.

```
cp-ifile1destination /  
cp : overwrite 'destination/file1'? y  
statdestination/file  
stat: failed to stat for 'destination/file': No such file or directory  
statdestination / file 1  
File: destination / file 1  
Size: 0 Blocks: 0 Block I/O: 4096 empty regular file  
Device: 801h/2049d Inode: 453120 Links: 1  
Access : (0644/-rw-r--r-- ) Uid : (1000/localadmin) Gid: (1000/localadmin)  
Access: 2023-06-13 22:29:16.231827158 +0300  
Modified: 2023-06-13 22:58:56.091795284 +0300  
Modified: 2023-06-13 22:58:56.091795284 +0300  
Created: -
```

Creating a backup copy of a file

```
cp-bfile1destination  
lsdestination  
file1 file1 ~ file4 file5 file6 source
```

Copying symbolic links

```
cp-slink_file2destination/  
ls-lislink_file2  
442359 lrwxrwxrwx 1 localadminlocaladmin 30 Jun 15 09:38  
slink_file2->/home/localadmin/test_cp/file2  
ls-lidestination/slink_file2  
442645 -rw-r--r-- 1 localadminlocaladmin 0 Jun 15 09:40  
destination/slink_file2  
cp-Pslink_file2destination/  
ls-lidestination/slink_file2  
442645 lrwxrwxrwx 1 localadminlocaladmin 30 Jun 15 09:41  
destination/slink_file2->/home/localadmin/test_cp/file2
```

Copying hard links

```
rm-rfddestination/file1  
ls-li*file1  
453106 -rw-r--r-- 2 localadminlocaladmin 0 Jun 13 22:05 file1  
453106 -rw-r--r-- 2 localadminlocaladmin 0 Jun 13 22:05 hlink_file1  
cpfile1hlink_file1destination/  
ls-lidestination/*file1  
442910 -rw-r--r-- 1 localadminlocaladmin 0 Jun 15 10:22 destination/file1
```

```

453100 -rw-r--r-- 1 localadminlocaladmin 0 Jun 15 10:22
destination/hlink_file1
rm-rfdestination/*file1
cp--preserve = linksfile1hlink_file1destination/
ls-ldestination/*file1
442910 -rw-r--r-- 2 localadminlocaladmin 0 Jun 15 10:23 destination/file1
442910 -rw-r--r-- 2 localadminlocaladmin 0 Jun 15 10:23
destination/hlink_file1

```

Preserving attributes (in this example, timestamps) when copying

```

cpfile2destination/file2_1
cp--preserve = timestampsfile2destination/file2_2
ls-lfile2
453107 -rw-r--r-- 1 localadminlocaladmin 0 Jun 13 22:05 file2
ls-ldestination/file2*
453100 -rw-r--r-- 1 localadminlocaladmin 0 Jun 15 10:25 destination/file2_1
453108 -rw-r--r-- 1 localadminlocaladmin 0 Jun 13 22:05 destination/file2_2

```

Full copy with preservation of all attributes

```

cp-a source destination/source_backup
ls-l source
total 0
-rw-r--r-- 1 localadminlocaladmin 0 Jun 13 22:05 file4
-rw-r--r-- 1 localadminlocaladmin 0 Jun 13 22:05 file5
-rw-r--r-- 1 localadminlocaladmin 0 Jun 13 22:05 file6
ls-ldestination/source_backup
total 0
-rw-r--r-- 1 localadminlocaladmin 0 Jun 13 22:05 file4
-rw-r--r-- 1 localadminlocaladmin 0 Jun 13 22:05 file5
-rw-r--r-- 1 localadminlocaladmin 0 Jun 13 22:05 file6

```

### The mv command

The mv command moves (renames) files and directories.

- mvfiledestination – moves the file file to the destination directory .
- mvdirectorydestination – Moves the directory directory to the destination directory .
- mvfile1file2 – renames file1 to file2 .
- mvdirectory1directory2 – renames directory1 to directory2.

### **Examples of use:**

Preparing to demonstrate a test directory with a set of files and subdirectories with files.

```

cd
mkdirtest_mv
cd test_mv
touch ./ file { 1 ..3 }
mkdirdestination
mkdir source
touch./source/file { 4 ..5 }
tree

```

```

.
├── destination
├── file1
├── file2
├── file3
├── source
├── file4
└── file5

```

2 directories, 5 files

Moving a file .

```

ls-ilfile1
453139 -rw-r--r-- 1 localadminlocaladmin 0 Jun 15 10:47 file1
mvfile1destination/

```

```
ls-ildestination/file1
```

```
453139 -rw-r--r-- 1 localadminlocaladmin 0 Jun 15 10:47 destination/file1
```

### Moving a directory

```
mv source destination/  
tree
```

```
.  
├── destination  
│   ├── file1  
│   ├── source  
│   ├── file4  
│   └── file5  
├── file2  
└── file3
```

2 directories, 5 files

### Renaming a file

```
ls-lifile2
```

```
453140 -rw-r--r-- 1 localadminlocaladmin 0 Jun 15 10:47 file2
```

```
mvfile2file4
```

```
ls-lifile4
```

```
453140 -rw-r--r-- 1 localadminlocaladmin 0 Jun 15 10:47 file4
```

### The rm command

rm command deletes files and directories.

- rmfile – deletes the file file.
- rm-ddirectory – removes the directory directory if it is empty.
- rm-ifile – asks the user for confirmation before deleting the file.
- rm-Ifiles – asks for confirmation when deleting more than 3 files.
- rm-ffile – deletes file, ignoring the -i switch and any missing files and arguments. Never prompts for confirmation.
- rm-Irdirectory – Recursively deletes directories and their contents. Asks for confirmation.
- rm-rfdirectory – **recursively deletes directories and their contents. Never asks for confirmation!**

### Examples of use:

Preparing to demonstrate a test directory with a set of files, links, and a subdirectory with files.

```
cd  
mkdir-ptest_rm/directory1/subdirectory  
cd test_rm/  
mkdirdirectory2  
touchfile { 1 ..3 }  
touchdirectory1/ file { 4 ..5 }  
touchdirectory1/subdirectory/ file { 6 ..7 }  
tree
```

```
.  
├── directory1  
│   ├── file4  
│   ├── file5  
│   ├── subdirectory  
│   ├── file6  
│   └── file7  
├── directory2  
│   ├── file1  
│   ├── file2  
│   └── file3
```

3 directories, 7 files

### Deleting a simple file

rmfile1

Deleting a simple file with a confirmation prompt

rm-ifile2

rm: delete empty regular file 'file2'? y

rm-Ifile3

Deleting an empty directory

rmdirectory2

rm: cannot delete 'directory2': This is a directory

rm - ddirectory 2

Deleting a non-empty directory

rm - ddirectory 1/ subdirectory

rm: cannot delete 'subdirectory': Directory is not empty

rm-drddirectory1/subdirectory

tree

```
.
├── directory1
├── file4
│   └── file5
```

1 directory, 4 files

Deleting a non-empty directory with confirmation prompt

rm-drIdirectory1

rm: remove 1 argument recursively? n

rm-rIdirectory1

rm: remove 1 argument recursively? n

Deleting a non-empty directory without prompting for confirmation

rm-rfdirectory1

### The find command

find command searches for files in a directory hierarchy. This powerful tool offers a wide range of search criteria and the ability to perform a variety of actions on found files. For complete help, use the `manfind` command .

Command syntax:

find[directory][search criterion][action]

Where:

**the directory** is the starting point for searching,

**search criteria** using conditional operators:

#### **Main search criteria:**

- o -name – search by name, case sensitive
- o -iname – search by name, case-insensitive,
- o -type – search by file type (f, d, l, b, c, p, s),
- o -size – search by file size (“+” - more, “-” - less, unsigned - exact match, “c” - byte, “k” - KB, “M” - MB, “G” - GB), for example, “+1G”, “-1M”, “24K”.
- o -empty – search for empty files,
- o -cmin – search by modification time in minutes, for example, to search for files modified in the last hour “-60”, to search for all files except those modified in the last hour “+60”,
- o -atime – search for files by access time in days, for example, “+30” – search for files that have not been accessed in the last 30 days,
- o -user – search files by owner,
- o -perm – search for files by a set of access rights, for example, “755” or “777”.

#### **Basic search condition operators:**

- -and – to combine conditions with a logical “and”,
- -or – to combine conditions with a logical “or”,
- -not – to invert the condition.
- **Action** – what to do with each found file. Basic actions:
  - -delete – deletes objects matching the search results,
  - -ls – displays more detailed search results,
  - -print – used by default if no other action is specified, shows the full path to the found files,
  - -exec – executes the specified command on each line of search results:
  - -execcommand{ }\;

Where: command is a command, for example, cp , {} is a file from the search result, and \; is the end of the command.

### Using the command:

- find.-namefile – search for files in the current directory by name or file mask.
- find./directory-namefile – search for files in the directory directory by name or file mask.
- find.-size+1M – searches for files in the current directory and subdirectories larger than 1 MB.
- find.-size+1M-execdu{ }\; – searches for files in the current directory and subdirectories larger than 1 MB and displays the sizes of these files.
- find./directory-empty – searches for empty files and directories in the directory and its subdirectories.
- find./directory-empty-delete – searches for empty files and directories in the directory and its subdirectories and deletes them.
- find.-cmin-60-size+10k – searches for files and directories in the current directory and subdirectories that have been modified within the past 60 minutes and are larger than 10 KB.
- find.-namefile\\*-or-name\\*.txt – search for files and directories in the current directory and subdirectories using the mask “file\*” OR “\*.txt”.
- find.-namefile\\*-typed – search for directories in the current directory and subdirectories using the mask “file \*”.
- find/etc-typef-execgrep-iH'nameserver'{ }\; – searches for files in the /etc directory that contain the substring nameserver.

### Examples of use:

Search for files in the current directory and subdirectories using the mask

```
"test*.txt"
find.-nametest \* .txt
./test\*.txt
./test4/test3.txt
./test4/test2.txt
./test4/test.txt
./test.txt
```

Search for files in the test4 directory and subdirectories using the mask

```
"test*.txt"
find./test4-nametetest \* .txt
./test4/test3.txt
./test4/test2.txt
./test4/test.txt
```

Search for files in the current directory and subdirectories larger than 1MB.

```
find.-size+ 1M
./VBoxLinuxAdditions.run
./.local/share/flyhelp/ru/.flyhelp/fts
./.cache/icon-cache.kcache
./.cache/mesa_shader_cache/index
```

Search for files in the current directory and subdirectories larger than 1 MB and display the sizes of these files

```
find.-size+ 1M-execdu {} \;
5832 ./VBoxLinuxAdditions.run
3492 ./.local/share/flyhelp/ru/.flyhelp/fts
10304 ./.cache/icon-cache.kcache
68 ./.cache/mesa_shader_cache/index
```

Search for empty files and directories in the specified directory and its subdirectories

```
find ./test-empty
./test/emptyfile
```

Search for empty files and directories in the specified directory and its subdirectories and delete them

```
find ./test -empty -delete
find ./test -empty
```

Search for files and directories in the current directory and subdirectories that have been modified within the last 60 minutes

```
find.-cmin-60
./xsession-errors
./config/rusbitech
./config/rusbitech/fly-term.conf
./.local/share/baloo/index
./.local/share/baloo/index-lock
./ test4
./test4/test3.txt
./test4/test2.txt
./test4/test.txt
```

Search for files and directories in the current directory and subdirectories that were modified within the last 60 minutes and are larger than 10 KB.

```
find.-cmin-60-size+ 10k
./xsession-errors
./.local/share/baloo/index
```

Search for files and directories in the current directory and subdirectories using the mask "test\*" OR "\*txt"

```
find.-nametest \* -or-name \* txt
./ test
./test/test1
./test/test1_link
./Desktops/Desktop1/Newfile .txt
./-test.txt
./.local/share/Trash/files/test
./.local/share/Trash/files/.txt
./.local/share/Trash/info/test.trashinfo
./hello.txt
./ test1
./test\\.txt
./ test4
./test4/test3.txt
./test4/test2.txt
./test4/test.txt
./ test .txt
./ test 3
```

Search directories in the current directory and subdirectories using the mask

"test\*"

```
find.-nametest \* -typed
./test_cp
./ test
./test_cat
./ test5
./test5/test_cp
././local/share/Trash/files/test
./ test4
./test_tac
./ test3
./test_mv
./test_mv/test
```

### **7.3 Assignment for independent completion**

#### **7.3.1 Task 1.**

1. Find all files larger than 500 KB in the /usr/share directory.
2. Save the list obtained in the previous step to the file /tmp/search.list.
3. Find out the size of the /tmp/search.list file in kilobytes and save this value to the /tmp/search-size.txt file.
4. Find out the number of lines in the /tmp/search.list file and write the result to the /tmp/search-size.txt file, preserving the previous data.

#### **7.3.2 Task 2.**

1. Create the directory /tmp/block6/task/1/2/3 in one command and move the files /tmp/search.list and /tmp/search-size.txt into it in one command.
2. Create a hard link and a symbolic link to the file /tmp/block6/task/1/2/3/search-size.txt.
3. Display the contents of the directory /tmp/block6/task/1/2/3/, getting the inodes of all files.
4. Delete the created directory /tmp/block6/task/1/2/3/ along with its contents.

#### **7.3.3 Task 4.**

1. Take a short test <https://onlinetestpad.com/chopdwptgyrcm>.

## 8 Working with text information

### 8.1 Objectives of the work

1. Explore the capabilities of the Linux OS for working with text.

### 8.2 Theoretical justification

In Windows, most settings are stored in the registry, so it's crucial for administrators to know what regedit is. In Linux, however, everything is stored in the `/etc` folder as plain text files, so advanced text file handling skills are essential for Linux system administrators.

In this module, you'll learn how to edit files from the command line and which utilities can be used to find, replace, merge, and sort text. We'll discuss the importance of regular expressions and, of course, how to exit the vi/vim editor without a full computer reboot.

#### 8.2.1 Basic commands for viewing text files

The following utilities are most often used to view the contents of text files:

- `cat` - concatenates files and sends data to standard output;
- `tac` - works like the `cat` command, but prints lines in reverse order;
- `less` - opens a file in page-by-page mode;
- `head` - displays several lines from the beginning of the file;
- `tail` - prints multiple lines from the end of a file.

As a reminder, utilities are external executable files, while commands are built-in bash shell directives. To improve performance, some utilities are embedded within the shell as commands, such as `echo`. You can determine whether a command is a command or a utility using the `type` command.

Let's look at the syntax and examples of using these commands in more detail.

#### The `cat` utility

`cat` utility combines files into a single stream (from the English "con **cat** enate" - to combine, hence the pronunciation "kat" rather than "kat"). The `cat` utility is most commonly used in the following manner:

- `catfile1` - outputs the contents of file1 to the standard output stream, i.e. to the screen;
- `catfile1file2` - combines the contents of two files file1 and file2 into one stream sequentially one after another and displays it on the screen;
- `cat-nfile1` - displays the contents of file1 with automatic line numbering;
- `cat-sfile1` - displays the contents of file1, replacing several consecutive empty lines with one;
- `catfile1file2>file3` - redirects the contents of files file1 and file2 to file3.

However, some advanced users use the `cat` utility as a simple editor for creating small text files. If you execute the command `cat>file1`, all subsequent text will be saved to the specified file by pressing the hotkey `Ctrl + D`. The `Ctrl + C`

combination allows you to exit without saving, but the previous file contents will be lost.

Let's look at some examples of use:

Let's check the DNS settings by displaying the contents of the `resolv.conf` file

:

```
cat /etc/resolv.conf
Generated by Network Manager
nameserver 10.0.2.3
```

Based on the command execution result, we see that the file is managed by the NetworkManager service.

Let's merge the two logs into one stream, which may be useful in the future, for example, for searching two files at once:

```
sudocat /var/log/boot.log/var/log/kern.log
```

```
...
May 10 16:21:27 astra kernel: IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
May 10 18:14:26 astra kernel: e1000: eth0 NIC Link is Down
May 10 18:14:28 astra kernel: e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
May 10 20:53:13 astra kernel: 17:53:03.546136 timesyncvgsvcTimeSyncWorker: Radical host time change: 8 956 796 000 000ns (HostNow=1 716 227 583 546 000 000
May 11 00:00:12 astra kernel: 17:53:13.547092 timesyncvgsvcTimeSyncWorker: Radical guest time change: 9 518 866 935 000ns (GuestNow=1 716 227 593 547 082 00
May 11 00:00:12 astra kernel: traps: syslog- ng[ 467] general protection fault ip:7635a8f7f8a0 sp:7ffed1a22630 error:0 in libpython2.7.so.1.0[7635a8e5c000+17f000]
```

Let's redirect the merge result to a new file:

```
sudocat /var/log/boot.log/var/log/kern.log>~/logs/catalogs
```

```
...
May 10 16:21:27 astra kernel: IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
May 10 18:14:26 astra kernel: e1000: eth0 NIC Link is Down
May 10 18:14:28 astra kernel: e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
May 10 20:53:13 astra kernel: 17:53:03.546136 timesyncvgsvcTimeSyncWorker: Radical host time change: 8 956 796 000 000ns (HostNow=1 716 227 583 546 000 000
May 11 00:00:12 astra kernel: 17:53:13.547092 timesyncvgsvcTimeSyncWorker: Radical guest time change: 9 518 866 935 000ns (GuestNow=1 716 227 593 547 082 00
May 11 00:00:12 astra kernel: traps: syslog- ng[ 467] general protection fault ip:7635a8f7f8a0 sp:7ffed1a22630 error:0 in libpython2.7.so.1.0[7635a8e5c000+17f000]
```

The execution result will be similar, but the lines will now be output from the logs file .

Let's display the contents of the name service manager settings file with line numbers:

```
cat -n /etc/nsswitch.conf
...
7 passwd: files
Group 8: Files
9 shadow: files
10 gshadow: files
...
```

Based on the command execution result in lines 7-10, we see that the sources of users, groups, and passwords for the host are only local files databases.

`cat` utility as a text editor to add the ALD Pro product repository. To run this command, you'll need to log in as a superuser. Simply using `sudo` won't work:

```

cat>/etc/apt/sources.list.d/aldpro.list
cat>/etc/apt/sources.list.d/aldpro.list
deb https://dl.astralinux.ru/aldpro/frozen/01/2.3.0 1.7_x86-64 main base
<^D>
cat/etc/apt/sources.list.d/aldpro.list
deb https://dl.astralinux.ru/aldpro/frozen/01/2.3.0 1.7_x86-64 main base

```

The `cat` "editor" is very strict and unforgiving. You can, of course, delete the last character with the Backspace key , but you can't fully edit a line. Therefore, less experienced administrators use `cat` in combination with a special bash language construct, which is specified by two lesser characters " << " and is called " *here-documents* . "

Entering `cat>file1<<EOF` will open a multi-line text editor created by bash. You can edit the text and save it by typing the "EOF" delimiter. The delimiter can be any combination of characters, but it is commonly used as an abbreviation for "End of File . "

### **The tac utility**

`tac` command is a mirror image of the `cat` command —it also merges files into a single stream, but it reverses the order of the file's lines. Because it needs to reorder the lines, the utility's technical implementation is slightly different: `tac` first reads all the lines of a file, then sends them to the output stream in reverse order. The most common use of `tac` is as follows:

- `tacfile1file2` – prints the lines of `file1` in reverse order, then the lines of `file2` also in reverse order.

Let's look at an example of use:

When working with logs, it can be useful to invert the order of lines, which can be done using the `tac` utility :

```

sudotac/var/log/boot.log/var/log/kern.log
May 6 15:46:46 astra kernel: AMD AuthenticAMD
May 6 15:46:46 astra kernel: Intel GenuineIntel
May 6 15:46:46 astra kernel: KERNEL supported cpus:
May 6 15:46:46 astra kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-5.15.0-70-generic root=UUID=33f4245d-33e8-48a1-9be7-0841215e1f46 roparsec.max_ilev=63 quiet net.ifnames=0
May 16 15:46:46 astra kernel: Linux version 5.15.0-70-generic (builder@build5) (gcc (AstraLinuxSE 8.3.0-6) 8.3.0, GNU ld (GNU Binutils for AstraLinux) 2.31.1) #astra2 SMP Mon Apr 10 13:35:08 UTC 2023 (Astra 5.15.0-70.astra2-generic 5.15.92)

```

As a result of executing the command, at the end of the output we will see lines from the beginning of the file

### **Less utility**

`less` utility (from the English "less") implements the functions of a text viewer. It allows you to page through multi-page files and search them. Working with the `less` interface was covered in detail in Module 5, "Working with Help Systems," so we won't dwell on it here.

Most often, the utility is called like this:

- `lessfile1` – viewing file `file1` using the `less` utility ;
- `catfile|less` – outputs the contents of `file1` to the standard stream and passes it through the pipeline to the `less` utility .

### **head utility**

`head` utility displays the first `n` lines of a file (the header). The default is 10 lines.

- `headfile` – displays the first 10 lines of the file `file` on the screen;
- `head-n3file` – displays the first 3 lines of the file `file` on the screen.

Examples of use:

Display the first 10 lines of the `/etc/passwd` file

```
head/etc/passwd
root: x:0:0:root:/root:/bin/bash
daemon: x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin: x:2:2:bin:/bin:/usr/sbin/nologin
sys: x:3:3:sys:/dev:/usr/sbin/nologin
sync: x:4:65534:sync:/bin:/bin/sync
games: x:5:60:games:/usr/games:/usr/sbin/nologin
man: x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp: x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail: x:8:8:mail:/var/mail:/usr/sbin/nologin
news: x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

Display the first 3 lines of the `/etc/passwd` file

```
head-n 3 /etc/passwd
root: x:0:0:root:/root:/bin/bash
daemon: x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin: x:2:2:bin:/bin:/usr/sbin/nologin
```

### Tail utility

`tail` utility displays the last `n` lines of a file. The default is 10 lines. The utility can continue displaying new lines as they are added to the file.

- `tailfile` – outputs the last 10 lines of the file `file` to the standard output stream, i.e. to the screen;
- `tail-n3file` – displays the last 3 lines of the file `file` on the screen;
- `tail-f-n3file` – displays the last three lines of file and continues, displaying any new lines appended to the file. To exit the utility, use the keyboard shortcut `ctrl + c`.

Examples of use:

Output of the last 10 lines of the `/etc/passwd` file :

```
tail/etc/passwd
fly-dm: x:107:118::/var/lib/fly-dm:/usr/sbin/nologin
ntp: x:108:121::/nonexistent:/usr/sbin/nologin
Debian-exim :x:109:122 :/var/spool/exim4:/usr/sbin/nologin
nm-openvpn: x:110:124:NetworkManager
OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip: x:111:7:HPLIP system user,,,:/run/hplip:/bin/false
logcheck: x:112:125:logcheck system
account,,,:/var/lib/logcheck:/usr/sbin/nologin
avahi: x:113:126:Avahi mDNS daemon,,,:/var/run/avahi-
daemon:/usr/sbin/nologin
localadmin: x:1000:1000:localadmin,,,:/home/localadmin:/bin/bash
systemd-coredump: x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
vboxadd: x:998:1:/:/var/run/vboxadd:/bin/false
```

Output of the last 3 lines of the `/etc/passwd` file :

```
tail -n 3 /etc/passwd
localadmin: x:1000:1000:localadmin,,,:/home/localadmin:/bin/bash
systemd-coredump: x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
vboxadd: x:998:1:/:/var/run/vboxadd:/bin/false
```

Display the last 3 lines of the `/var/log/auth.log` file and new lines in interactive mode:

```
sudo tail -fn 3 /var/log/auth.log
[sudo] password for localadmin:
Jun 14 10:30:01 astraCRON[ 2096]: pam_unix(cron:session): session opened for user
root by (uid=0)
```

```

Jun 14 10:30:01 astraCRON[ 2096]: pam_unix(cron:session): session closed for user root
Jun 14 10:53:42 astrasudo[ 2173]: pam_unix(sudo:auth): authentication failure; logname=localadminuid=1000 euid=0 tty=/dev/pts/0 ruser=localadminrhost= user=localadmin
Jun 14 10:53:49 astrasudo[ 2173]: localadmin : TTY=pts/0 ; PWD=/home/localadmin ; USER=root ; COMMAND=/usr/bin/tail -f -n 3 /var/log/auth.log
Jun 14 10:53:49 astrasudo[ 2173]: pam_kiosk2(sudo:session): need_continue: UID 0 detected, skipping. U
ser: root
Jun 14 10:53:49 astrasudo[ 2173]: pam_unix(sudo:session): session opened for user root by localadmin(uid=0)
^C

```

To exit interactive mode, use `ctrl + c` , indicated by "`^C`" on the last line.

## 8.2.2 Basic commands for editing text from the command line

Let's look at a few more frequently used commands:

- `wc[file]` - displays statistics on the number of lines, words, and bytes in the specified file;
- `nl[file]` - adds line numbers and displays text from the specified file. Empty lines are not numbered;
- `tee file` - redirects the input stream to both the file and the standard output stream. The utility can redirect the stream to multiple files at once;
- `sort` - sorts strings;
- `uniq` - removes duplicate lines;
- `cut` - cuts a fragment from each line of a file and outputs the results to the standard output stream, which allows you to display data in tabular form;
- `tr` - replaces a substring and passes the stream on;
- `xargs` - Executes the specified command, supplementing its arguments with information from lines it receives from standard input. The command can be executed multiple times, depending on the number of lines passed to it.

Let's look at the syntax and examples of using these commands in more detail.

### **wc utility**

Analyzes the contents of a file and sends statistics on the number of lines, words, and bytes (from the English wordcount – the number of words) to the standard output stream.

- `wc file` – displays the number of lines, words, and bytes in the file;
- `wc -l file` – displays the number of lines in the file;
- `wc -w file` – displays the number of words in the file;
- `wc -m file` – displays the number of characters in the file;
- `wc -c file` – displays the number of bytes in the file;
- `wc -l file1 file2 file3` – displays the number of lines in each of the files `file1`, `file2` and `file3`;
- `ls | wc -l` – displays the number of files and directories in the current directory, excluding hidden objects. Files and directories in Linux are

considered hidden if their names begin with a period (.)

- `ls-A|wc-l` – displays the number of all files and directories in the current directory, excluding hidden objects;
- `ps-e|wc-l` – displays the number of processes in the system. The actual number of processes is one less (the `wc` command counts all rows, including the table header).

Examples of use:

Prints the number of lines, words, and bytes in the `.bashrc` file :

```
cd
wc .bashrc
113 483 3526 .bashrc
```

Displays the number of lines in the `.bash_history` file :

```
wc-l .bash_history
113 .bash_history
```

Displays the number of files and directories in the current directory, excluding hidden ones:

```
ls | wc-l
27
```

Displays the number of all files and directories in the current directory, excluding hidden ones:

```
ls-A | wc-l
48
```

Displays the number of processes in the system. The actual number of processes is one less (the `wc` command counts all rows, including the table header):

```
ps-e | wc-l
151
```

## **nl utility**

Adds line numbers to the output stream.

- `nlfile` – displays the file `file`, adding line numbers and aligning the numbers to the right.
- `nl-nlnfile` – displays the file `file`, adding line numbers with left-aligned line numbers.

Examples of use:

Will display the file `file`, adding line numbers with the numbering aligned to the right.

```
nl/etc/passwd
1 root:x:0:0:root :/root:/bin/bash
2 daemon:x:1:1:daemon :/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin :/bin:/usr/sbin/nologin
4 sys:x:3:3:sys :/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync :/bin:/bin/sync
...
30 hplip:x:111:7:HPLIP system user,,,:/run/hplip:/bin/false
31 logcheck:x:112:125:logcheck system
account,,,:/var/lib/logcheck:/usr/sbin/nologin
32 avahi:x:113:126:Avahi mDNS daemon,,,:/var/run/avahi-
daemon:/usr/sbin/nologin
33 localadmin:x:1000:1000:localadmin ,,,:/home/localadmin:/bin/bash
34 systemd -coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
35 vboxadd:x:998:1 :/var/run/vboxadd:/bin/false
```

Displays the file `file`, adding line numbers with left-aligned line numbers.

```
nl-nl/etc/passwd
1 root :x:0:0:root :/root:/bin/bash
2 daemon :x:1:1:daemon :/usr/sbin:/usr/sbin/nologin
```

```

3 bin :x:2:2:bin :/bin:/usr/sbin/nologin
4 sys :x:3:3:sys :/dev:/usr/sbin/nologin
5 sync :x:4:65534:sync :/bin:/bin/sync
...
30 hplip :x:111:7:HPLIP system user,,,:/run/hplip:/bin/false
31 logcheck :x:112:125:logcheck system
account,,,:/var/lib/logcheck:/usr/sbin/nologin
32 avahi :x:113:126:Avahi mDNS daemon,,,:/var/run/avahi-
daemon:/usr/sbin/nologin
33 localadmin :x:1000:1000:localadmin ,,,:/home/localadmin:/bin/bash
34 systemd-coredump :x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
35 vboxadd :x:998:1 :/var/run/vboxadd:/bin/false

```

## Tee utility

This utility redirects input to both the specified files and the standard output stream. It's used as follows:

- `<command>|tee file1 file2` – the result of the command `command` will be displayed on the screen and written to files `file1` and `file2`;
- `<command><directory>-h-d1|tee file|nl` – the result of executing the command `command` will be written to the file “as is”, and will be displayed on the screen with line numbers.

Examples of use:

The result of the command `command` will be displayed on the screen and written to files `file1` and `file2`:

```

date | tee file1 file2
Ptijun 16 08:47:28 MSK 2023
cat file1
Ptijun 16 08:47:28 MSK 2023
cat file 2
Ptijun 16 08:47:28 MSK 2023

```

The result of executing the `du` command will be written to the file "as is", and will be displayed on the screen with line numbers:

```

du /usr -h -d1 | tee du.out | nl
1 4.0K /usr/libx32
2 4.0K /usr/games
3 4.0K /usr/lib64
4 4.1G /usr/lib
5 4.0K /usr/lib32
6 76K /usr/local
7 342M /usr/bin
8 1.2M /usr/libexec
9 33M /usr/sbin
10 12M /usr/include
11 189M /usr/src
12 2.0G /usr/share
13 8.0K /usr/etc
14 6.6G /usr
cat du.out
4.0K /usr/libx32
4.0K /usr/games
4.0K /usr/lib64
4.1G /usr/lib
4.0K /usr/lib32
76K /usr/local
342M /usr/bin
1.2M /usr/libexec
33M /usr/sbin
12M /usr/include
189M /usr/src
2.0G /usr/share

```

8.0K /usr/etc

6.6G /usr

## Sort utility

Sorts lines of text files. Sorting occurs in the following order:

- lines with numbers are placed above other lines;
- lines starting with lowercase letters are placed above;
- Sorting is done in alphabetical order;
- The lines are sorted first alphabetically, and then secondarily according to other rules.

The utility is used as follows:

- `sortfile` – displays the sorted file on the screen.
- `command|sort-r` – sorts the output of `command` in reverse order (-r).
- `sort-ufile` – displays the sorted file and removes duplicate lines from the output.
- `sort-k3file` – will display the contents of file, divided into columns (using space as a separator) and sorted by the third column (-k 3).
- `sort-k1,3file` – displays the contents of file divided into columns and sorted by columns 1 and 3 (-k 1,3).
- `cat/etc/passwd|sort-t:-k3-nr|head` – will output the first 10 lines (head) of the result of sorting the /etc/passwd file by column 3 (-k 1) with the column separator ":" (-t:) in reverse order (-r) for numeric sorting (-n).

Examples of use:

Let's create two test files to demonstrate the sort command :

```
echo -e "computer\nmouse\nLAPTOP\ndata\nAstraLinux\nlaptop\nLinux\nlaptop"
>test_sort1.txt
echo -e "acbd\nd bac\nc adb\nb da fg r\nc es we\nc ds nd ck"
>test_sort2.txt
```

echo command with the **-e switch** interprets special characters, in this case **n** is the newline character:

```
cattest_sort1.txt
computer
mouse
LAPTOP
data
AstraLinux
laptop
Linux
laptop
```

Let's sort the test\_sort1.txt file :

```
sorttest_sort1.txt
AstraLinux
computer
data
laptop
laptop
LAPTOP
Linux
mouse
```

Let's sort it out file test\_sort1.txt V reverse in order :

```
sort-rtest_sort1.txt
mouse
Linux
LAPTOP
```

```
laptop
laptop
data
computer
AstraLinux
```

test\_sort1.txt file and remove duplicates from the output:

```
sort-utest_sort1.txt
```

```
AstraLinux
computer
data
laptop
LAPTOP
Linux
mouse
```

Let's sort the file by the 2nd column (split by space by default):

```
cattest_sort2.txt
```

```
acbd
```

```
dbac
```

```
cadb
```

```
b da fg r
```

```
ces we
```

```
c ds nd ck
```

```
cattest_sort2.txt | sort-k 2
```

```
cadb
```

```
dbac
```

```
acbd
```

```
b da fg r
```

```
c ds nd ck
```

```
ces we
```

Let's sort the file by columns 1 and 3 (the value of c in column 1 shows that sorting by column 3 also works):

```
cattest_sort2.txt | sort-k 1 ,3
```

```
acbd
```

```
b da fg r
```

```
cadb
```

```
c ds nd ck
```

```
ces we
```

```
dbac
```

Let's output the first 10 lines of the result of sorting the /etc/passwd file by column 3 (-k 3) with a column separator: (-t:) in reverse order (-r) with a numeric sort (-n):

```
cat/etc/passwd | sort-t:-k 3 -nr | head
```

```
nobody: x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

```
localadmin: x:1000:1000:localadmin,,,:/home/localadmin:/bin/bash
```

```
systemd-coredump: x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
```

```
vboxadd: x:998:1::/var/run/vboxadd:/bin/false
```

```
avahi: x:113:126:Avahi mDNS daemon,,,:/var/run/avahi-
```

```
daemon:/usr/sbin/nologin
```

```
logcheck: x:112:125:logcheck
```

```
system
```

```
account,,,:/var/lib/logcheck:/usr/sbin/nologin
```

```
hplip: x:111:7:HPLIP system user,,,:/run/hplip:/bin/false
```

```
nm-openvpn: x:110:124:NetworkManager
```

```
OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
```

```
Debian-exim :x:109:122 ::/var/spool/exim4:/usr/sbin/nologin
```

```
ntp: x:108:121::/nonexistent:/usr/sbin/nologin
```

### The uniq utility

This utility allows you to find or remove duplicate lines. For proper filtering, the list must be pre-sorted. Keep in mind that the command's result may differ from what you expect; for example, the -i switch, which ignores case when comparing, does not

work with Cyrillic characters.

- `command|sort|uniq` – the result of the `command` will be sorted, and all lines will be displayed on the screen without repetitions.
- `uniq-d` – outputs only lines that contain duplicates. Only one line from each set of duplicates will be output.
- `uniq-u` – outputs only lines that do not have repetitions.
- `uniq-c` – outputs unique and non-unique lines, indicating how many times these lines appeared in the source text (see the number at the beginning of the line).
- `uniq-i` – does not handle case-sensitive string comparisons. Does not work with Cyrillic characters.

Examples of use:

Create two files for demonstration with a set of characters, including repeating ones:

```
echo -e "a\nb\nc\nA\nd\nc\nd\nc\nd " >test_ru_uniq.txt
echo -e "a\nb\nc\nA\nb\nd\nc\ne\nc" >test_en_uniq.txt
```

The first file contains Cyrillic characters. Filtering with the `uniq` command doesn't work with an unsorted list:

```
cat test_ru_uniq.txt | uniq
A
b
V
A
G
V
d
V
D
```

Output all unique lines:

```
cat test_ru_uniq.txt | sort | uniq
A
A
b
V
G
d
D
```

Output only those lines that do not have repetitions.

```
cat test_ru_uniq.txt | sort | uniq-u
A
A
b
G
d
D
```

Output only those lines that have duplicates. Only one line from each set of duplicates will be output:

```
cat test_ru_uniq.txt | sort | uniq-d
V
```

Output only unique strings, ignoring case (does not work with Cyrillic):

```
cat test_ru_uniq.txt | sort | uniq-i
A
A
b
V
```

G  
d  
D

The first file contains Latin characters. Filtering with the uniq command doesn't work with an unsorted list:

```
cattest_en_uniq.txt | sort | uniq
```

a  
A  
b  
c  
C  
d  
e

Output only unique strings, ignoring case:

```
cattest_en_uniq.txt | sort | uniq-i
```

a  
b  
c  
d  
e

Output of all unique lines and the number of repetitions (number at the beginning of the line):

```
cattest_en_uniq.txt | sort | uniq-c
```

1 a  
1 A  
2 b  
2 s  
1 C  
1 day  
1 e

Output all unique lines and the number of repetitions (the number at the beginning of the line), case-insensitive:

```
cattest_en_uniq.txt | sort | uniq-ic
```

2 a  
2 b  
3 s  
1 day  
1 e

Displaying the number of lines in the /var/log/user.log file :

```
sudocat/var/log/user.log | sort | wc-l
```

17

Displaying the number of unique lines in the /var/log/user.log file :

```
sudocat/var/log/user.log | sort | uniq | wc-l
```

10

Displays lines in the /var/log/user.log file that do not have duplicates:

```
sudocat/var/log/user.log | sort | uniq-u
```

Jun 15 09:24:44 astravboxadd- service.sh[ 1008]: Starting VirtualBox Guest Addition service.

Jun 15 09:24:44 astravboxadd- service.sh[ 1021]: VirtualBox Guest Addition service started.

Jun 15 09:33:55 astrapulseaudio[ 1638]: [alsa-source-Intel ICH] alsa-source.c: ALSA woke us up to read new data from the device, but there was actually nothing to read.

Jun 15 09:33:55 astrapulseaudio[ 1638]: [alsa-source-Intel ICH] alsa-source.c: Most likely this is a bug in the ALSA driver 'snd\_intel8x0'. Please report this issue to the ALSA developers.

Jun 15 09:33:55 astrapulseaudio[ 1638]: [alsa-source-Intel ICH] alsa-source.c: We were woken up with POLLIN set -- however a subsequent snd\_pcm\_avalil( ) returned 0 or another value <min\_avail.

Jun 15 09:33:55 astrapulseaudio[ 1680]: [pulseaudio] pid.c: Daemon already

running.

Print only duplicate lines in the /var/log/user.log file :

```
sudo cat /var/log/user.log | sort | uniq -D
```

```
Jun 15 09:33:54 astrapulseaudio[ 1638]: [pulseaudio] alsa-util.c: Disabling timer-based scheduling because running inside a VM.
```

```
Jun 15 09:33:54 astrapulseaudio[ 1638]: [pulseaudio] alsa-util.c: Disabling timer-based scheduling because running inside a VM.
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.backlighthelper[ 1706]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.backlighthelper[ 1706]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.backlighthelper[ 1706]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.chargethresholdhelper[ 1693]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.chargethresholdhelper[ 1693]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.chargethresholdhelper[ 1693]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.discreetepguhelper[ 1673]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.discreetepguhelper[ 1673]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.discreetepguhelper[ 1673]: QDBusArgument: read from a write-only object
```

Print only duplicate lines in the /var/log/user.log file , where only one line from each set of duplicate lines will be printed:

```
sudo cat /var/log/user.log | sort | uniq -d
```

```
Jun 15 09:33:54 astrapulseaudio[ 1638]: [pulseaudio] alsa-util.c: Disabling timer-based scheduling because running inside a VM.
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.backlighthelper[ 1706]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.chargethresholdhelper[ 1693]: QDBusArgument: read from a write-only object
```

```
Jun 15 09:33:55 astraorg.kde.powerdevil.discreetepguhelper[ 1673]: QDBusArgument: read from a write-only object
```

## Cut utility

The utility extracts a fragment from each line of a file and outputs the results to the standard output stream, allowing you to display the data in tabular form. Example usage:

- `cut -c2,3,5 file` – displays the text from the file line by line and only the 2nd, 3rd and 5th characters in each line;
- `cut -c3-6 file` – displays the text from the file line by line and only characters 3 through 6 on each line;
- `cut -c-5 file` – displays the text from the file line by line and only the first 5 characters in each line;
- `cut -c3- file` – displays the text from the file line by line and only characters from 3 to the last in each line;
- `cut -d: -f1,3 file` – displays the text from file line by line, and only the first and third columns. Rows are separated into columns by the ":" separator.

Examples of use:

Displays a list of all users on the current system:

```
cut -d: -f 1 /etc/passwd
```

```
root
```

```
daemon
```

```
bin
sys
sync
games
man
lp
mail
...
```

Displays a list of users logged into the current system:

```
who | cut -d ' ' -f1 | sort
localadmin
localadmin
```

Displays the name of the service using port 80:

```
grep -w 80 /etc/services | cut -f 1 | uniq
http
```

Displays the CPU type:

```
cat /proc/cpuinfo | grep "name" | cut -d: -f2 | uniq
Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
```

### **tr utility**

Performs transformation, substitution (replacement), reduction, and/or deletion of characters coming from the standard input stream, writing the results to the standard output device.

- `tr string1 string2 <file` – displays the contents of the file `file`, replacing the characters from `string1` with the characters from `string2`;
- `tr -c string1 string2 <file 1>file 2` – saves the contents of `file1` in `file2`, replacing the characters missing in `string1` with the characters from `string2`;
- `<command>|tr -d string1` – displays the result of the command `<command>`, removing from it the characters that appear in `string1`;
- `<command>|tr -s line1 line2` – displays the result of the command `<command>` on the screen, replacing the sequence of characters from `line1`;
- `<command>|tr -cs string1 string2` – displays the result of the command `<command>`, replacing the characters from `string1` with the characters from `string2` and compressing the character sequences from `string2` (removing duplicate characters).

Examples of use:

Will display text in uppercase:

```
echo "text" | tr '[:lower:]' '[:upper:]'
TEXT
```

```
echo "text" | tr 'az' 'AZ'
TEXT
```

Will display only numbers from the text:

```
echo "phone number 8-960-999-99-99" | tr -cd [ :digit: ]
896099999999
```

Will display text, replacing each sequence of r's with a single letter:

```
echo "Astrrrrrra" | tr -s 'r' 'r'
Astra
```

Will display only letters and numbers in the text, and replace all other characters with a newline character:

```
echo 'first second third' | tr -cs '[:alnum:]' '\n'
first
second
```

third

Will display text on the screen, removing all numbers from it:

```
echo "1 please 2 delete 3 all 4 digits" | tr -d [0-9]
```

Please delete all numbers

Displays the contents of the PATH variable line by line, replacing the ':' separator with a newline character:

```
PATH | tr ':' '\n'
/usr/local/bin
/usr/bin
/bin
/usr/local/games
/usr/games
```

### 8.2.3 xargs utility

Executes commands using data received from the standard input stream. Typically used in conjunction with other commands via the pipeline.

- `echo"file1file2file3"|xargstouch` – for each argument from standard input (file1, file2, and file3), the touch command will be executed and three files will be created (if they don't exist). The `touchfile1file2file3` command will do the same;
- `echo"file1file2file3"|xargs-ttouch` – will do the same, but will display the command on the screen before executing it;
- `echo"file1file2file3"|xargs-ptouch` – the user will be asked for confirmation before executing the command. To confirm, enter "y";
- `echo"file1file2file3"|xargs-n1-ttouch` – The -n switch limits the number of arguments passed to the command. In this case, three touch commands will be executed with one argument per command instead of one touch command with three arguments;
- `echo"file1;file2;file3"|xargs-d\;-ttouch` – the -d switch allows you to specify a separator (a single character or escape sequence) and split the text into separate arguments;
- `xargs-t-L1-aips.txtping-c1` – The -a switch allows you to specify a file to use instead of standard input. The -L switch allows you to specify the number of lines to read from the file at a time before passing them to the command as an argument.
- `find~/test*-typef-print0|xargs-0rm-f` – The print0 switch in the find command tells it to print the full path to the found file and to use the null character instead of a newline to correctly interpret files with newlines or other whitespace characters in their names. The "0" switch in the xargs command allows this output from find to be correctly interpreted.
- `echo"file1file2file3"|xargs-tI%sh-c'{touch%;ls-l%;}'` – The -I switch allows you to specify a character or string, each occurrence of which in the command will be replaced with arguments. This is used to run multiple commands (in this case, touch and ls – first the files will be created, and then information about them will be read and displayed), which are executed sequentially one after another.
- `echo192.168.1.{1..254}|tr"":|xargs-d:-L1-P100ping-c1` – The -P switch

allows parallelizing the command. The number specifies the maximum number of processes that will run simultaneously. In this example, you can ping all host addresses on the network in 3 iterations. Another example: downloading files from the internet in parallel using a pipeline of `wget` and `xargs` commands .

Examples of use:

For each argument from standard input (file1, file2 and file3) the `touch` command will be executed and 3 files will be created (if they do not exist):

```
echo " file1 file2 file3" | xargstouch
lsfile { 1 ..3 }
file1 file2 file3
```

Will do the same thing, but will print the command before executing it:

```
echo "file1 file2 file3" | xargs-ttouch
touch file1 file2 file3
```

Will ask user confirmation before execution:

```
echo "file1 file2 file3" | xargs-ptouch
touch file1 file2 file3 ?... y
```

`-n` switch limits the number of arguments passed to the command:

```
echo "file1 file2 file3" | xargs-n 1 -ttouch
touch file1
touch file2
touch file3
```

In this case, three `touch` commands with one argument each will be executed, instead of one `touch` command with three arguments.

`-d` switch allows you to select a separator to divide the text supplied to the command into separate arguments:

```
echo " file1;file2;file3" | xargs-d \; touch
```

Input text can be passed not only from standard input, but also read from a file.

The `-L` switch allows you to specify how many lines of text will be processed as a single argument:

```
echo -e "77.88.8.8\n77.88.8.1" >ips.txt
catips.txt
77.88.8.8
77.88.8.1
xargs -tL 1 -aips.txt ping -c 1
ping -c 1 77.88.8.8
PING 77.88.8.8 (77.88.8.8) 56(84) bytes of data.
64 bytes from 77.88.8.8: icmp_seq=1 ttl=246 time=36.2 ms
--- 77.88.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 36.214/36.214/36.214/0.000 ms
ping -c 1 77.88.8.1
PING 77.88.8.1 (77.88.8.1) 56(84) bytes of data.
64 bytes from 77.88.8.1: icmp_seq=1 ttl=52 time=50.7 ms
--- 77.88.8.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 50.739/50.739/50.739/0.000 ms
```

`-print0` switch in the `find` command tells it to print the full path to the found file and to use the null character instead of a newline to correctly interpret files with newlines or other whitespace characters in their names. The `-0` switch in the `xargs` command allows the following output from the `find` command to be correctly interpreted :

```
find~/test*-typef-print0 | xargs-0rm-f
```

`-I` switch allows you to specify a character or string, each occurrence of which

in the command will be replaced by arguments. This is used to run multiple commands (in this case, touch and ls – first, the files will be created, then information about them will be read and displayed), which are executed sequentially one after another:

```
echo " file1 file2 file3" | xargs-tI%sh-c '{ touch %; ls -l %; }'  
sh -c { touch file1 file2 file3; ls -l file1 file2 file3; }  
-rw-r--r-- 1 localadminlocaladmin 0 Jun 21 10:21 file1  
-rw-r--r-- 1 localadminlocaladmin 0 Jun 21 10:21 file2  
-rw-r--r-- 1 localadminlocaladmin 0 Jun 21 10:21 file3
```

-P switch allows parallel command execution; the number specifies the maximum number of processes that will run simultaneously. In this example, you can ping all host addresses on the network in three iterations:

```
echo 192.168.1.{1..254} | tr " " : | xargs -d : -L 1 -P 100 ping -c 1
```

## 8.2.4 grep, sed, and awk utilities and regular expressions

This section presents three additional tools for working with text:

- grep – a utility for searching strings by pattern;
- sed – in addition to searching, the utility allows you to transform text, for example, insert or replace text fragments;
- awk is the most powerful tool on this list. It's a complete scripting language for text processing and filtering, offering many features not found in the first two utilities.

These three utilities have one more thing in common. They all support regular expressions. Furthermore, using regular expressions significantly expands their capabilities. A detailed study of regular expressions is beyond the scope of this course, but basic techniques for working with them will be covered.

Let's look at each of them in more detail below.

### The grep utility

grep command stands for " **search** globally for lines matching the regular expression , **and** print them . "

grep command offers many options for filtering text: selecting specific lines from text files, filtering command output, and searching the file system for files containing specific strings. This utility is so popular that it's preinstalled on almost all distributions.

There are two main syntaxes for using the grep command :

- grep[<options>]<pattern>[/path/to/file/or/folder...]
- <command>|grep[<options>]<pattern>

Where:

- **Options** are additional parameters that can be used to specify various search and output settings, such as the number of lines or the inversion mode.
- **A pattern** is any string or regular expression that will be searched.
- **The file or folder name** is where the search will be performed. Grep allows you to search multiple files and even entire directories using recursive mode.

Main options of the grep utility :

- -E ( --extended-regexp ) - enable extended regular expression mode

(ERE);

- -F ( --fixed-strings ) - treat the search pattern as a regular string, not a regular expression;
- -G ( --basic-regexp ) - interpret the search pattern as a basic regular expression (BRE);
- -P ( --perl-regexp ) - treat the search pattern as a Perl regular expression;
- -e ( --regexp ) - an alternative way to specify a search pattern. This option can be used multiple times, allowing you to specify multiple patterns to search for files containing one of them;
- -f ( --file ) - read search pattern from file;
- -i ( --ignore-case ) - ignore the case of characters;
- -v ( --invert-match ) - output only those lines in which the search pattern is not found;
- -w ( --word-regexp ) - search for the pattern as a word separated by spaces or other punctuation marks;
- -x ( --line-regexp ) - search for the pattern as a whole line from the beginning to the newline character;
- -c - display the number of lines found;
- --color - enable color mode, available values: **never**, **always** and **auto** ;
- -L ( --files-without-match ) - output only file names. All files searched will be output;
- -l ( --files-with-match ) - similar to the previous one, but only files that contain at least one occurrence will be output;
- -m ( --max-count ) - stop searching after the specified number of lines have been found;
- -o ( --only-matching ) - display only the matched part instead of displaying the entire line;
- -h ( --no-filename ) - do not display the file name;
- -q ( --quiet ) - do not output anything;
- -s ( --no-messages ) - do not display file reading errors;
- -A ( --after-content ) - show the entry and n lines after it;
- -B ( --before-content ) - show the entry and n lines before it;
- -C - show n lines before and after the occurrence;
- -a ( --text ) - treat binary files as text;
- --exclude - skip files whose names match the regular expression;
- --exclude-dir - skip all files in the specified directory;
- -I - skip binary files;
- --include - search only in files whose names match the regular expression;
- -r - recursive search through all subfolders;
- -R - recursive search, including links.

Examples of use:

- grepstringfile – displays lines from file containing occurrences of string.

- `grep-istringfile` – the `-i` option ignores case when searching for matches. String, STRING, and string will be equivalent when searching.
- `grep-estring1-estring2file` – the `-e` option allows you to specify multiple search conditions.
- `grep-nstringfile` – the `-n` option allows you to print the line numbers where occurrences were found.
- `grep -F"specialsymbol"file` – The `-F` option allows you to explicitly tell the utility that the string is a string and not a regular expression. Otherwise, when using special characters (if the regular expression is written incorrectly), the command will return an error.
- `command|grepstring` – filtering command output – will display all lines from the output of the command `command` that contain the string `string`.
- `grep-A4stringfile` specifies the number of lines (in this case, 4) to display after the line containing the match. This is useful for understanding context, for example, when searching for errors in logs. The `-B` and `-C` options are similar in functionality: `-B` displays the specified number of lines before the line containing the match (e.g., `-B3`), while `-C` displays lines both before and after the line containing the match (e.g., `-C2`).
- `grep-rstringdirectory` – the `-r` option allows you to search for files not only in the specified directory, but also recursively - in all its subdirectories.
- `grep-rlstringdirectory` – The `-I` (uppercase i) option allows you to exclude binary files from the search results.
- `grep-rsstringdirectory` – the `-s` option allows you to hide file access errors if you do not have sufficient permissions to read them.
- `grep-r--include="string_file"stringdirectory` – The `--include` option allows you to search only for files that match the `string_file` search mask. The `--exclude` option is used similarly, but to exclude specific files from the search.
- `grep-wstringfile` – the `-w` option allows you to search for exactly matching words, rather than for occurrences of a string in words. For example, without the `-w` option, the search string `abc` will match the words `aabc`, `abc123`, etc., while with the `-w` option, it will only match the words `abc`.
- `grep-cstringfile` – the `-c` option allows you to display the number of lines containing occurrences of the string found.
- `grep-vstringfile` – the `-v` option allows you to print lines that do not contain an occurrence of string.
- `grep-lrstringdirectory` – The `l` (lowercase L) option allows you to list only the names of files that contain at least one occurrence of the string.
- `grep--color=always` – the `--color` option allows you to specify the highlighting of found occurrences in lines.

Regular expressions are specially written strings used to search for character patterns in text. They are somewhat similar to shell wildcards, but their capabilities are

much broader. Many Linux text-processing utilities and programming languages include a regular expression engine. There are many regular expression dialects, but most Linux utilities conform to the POSIX regular expression standard. We'll look at these.

There are two implementations of regular expressions in the POSIX standard: Basic RegularExpression (BRE) and ExtendedRegularExpression (ERE). BRE is supported by virtually all POSIX-compliant programs, while ERE allows for the creation of more complex regular expressions but is not supported by all utilities. However, `grep` supports both.

The following metacharacters are used in simple regular expressions: `^ $ . [ ] * \` -. Metacharacters are a category of characters used to create search criteria in regular expressions. Simple characters are commonly called literals, for example, the expression `"dir"` denotes a string that meets the following criteria: there are at least three characters in the string; the characters `"d"`, `"i"`, and `"r"` are present in the string, in that order; there are no other characters between them. Characters that match themselves (like `"d"`, `"i"`, and `"r"`) are literals. To use a metacharacter as a literal, it must be escaped with a backslash (`\`). Some metacharacters in regular expressions have special meaning in the shell, so when using them in regular expressions, the string containing them must be enclosed in quotation marks.

The metacharacter `"."` matches any character in that position (but not null or "empty"). Example usage: `ls/bin|grep'.dir'` .

The asterisk `"*"` metacharacter after another character in a pattern means that this character can be repeated any number of times, including none. Example usage: `ls/bin|grep'mk*dir'` .

The metacharacters `"^"` and `"$"` act as anchors. Anchors allow you to search for a pattern not just within a string, but to explicitly specify that the match should be at the beginning of the string (the caret character (`^`)) or at the end of the string (the dollar character (`$`)). The regular expression `"^$"` matches empty strings. Examples of usage: `ls/bin|grep'^dir'` ; `ls/bin|grep'dir$'` .

The square bracket metacharacters `"[]"` allow you to specify a set, any member of which will match the character in that position. For example, `ls/bin|grep'[km]dir'` .

All but two metacharacters lose their special meanings when enclosed in square brackets.

- The `"^"` metacharacter immediately after the `"["` adds a logical "not," meaning that the character in that position will match any character not included in the set specified in the square brackets. For example, `ls/bin|grep'^km]dir'` . If the `"^"` character is not at the beginning of the brackets, it is interpreted as a regular character.
- The hyphen `"-"` metacharacter defines character ranges. You can specify ranges of letters or numbers. For example, `ls/bin|grep'[oz]dir'` ; `ls/dev|grep'tty1[4-7]'` .

POSIX has several classes that describe different sets of characters.

- **[ :alnum: ]** - Alphanumeric characters; equivalent to the range `[A-Za-z0-9]` in ASCII.

- **[[:alpha:]]** - Alphabetical characters; equivalent to the ASCII range [A-Za-z].
- **[[:digit:]]** - Numbers from 0 to 9.
- **[[:lower:]]** and **[[:upper:]]** - Lowercase and uppercase characters, respectively.
- **[[:space:]]** - Whitespace characters, including space, tab, carriage return, line feed, vertical tab, and format feed.

The following metacharacters are used in extended regular expressions:

- Parentheses: ( ) ;
- Curly braces: { } ;
- Question mark: ?;
- Plus symbol: +;
- Vertical bar: |.

To use extended regular expressions with the `grep` utility, you must use the `-E` switch .

The vertical bar metacharacter "|" represents a logical OR. It allows you to specify two sets of characters in a pattern, each of which will be matched. For example:

- The command `echo"123"|grep -E'123|345'` will display the string 123, since the substring 123 is specified in the `-E` parameter ;
- The command `echo"345"|grep -E'123|345'` will print the string 345, since the substring 345 is specified in the `-E` parameter ;
- The command `echo"234"|grep-E'123|345'` will not display anything on the screen, since the substrings 123 and 345 are not in the line.

The parentheses metacharacters "()" are used to group conditions.

For example, `ls/dev|grep -E'tty1(1|2|3)'` will print all lines containing `tty11`, `tty12`, or `tty13`, while the same expression without parentheses, `ls/dev|grep -E'tty11|2|3'` , will print lines containing either `tty11`, 2, or 3.

The question mark "?" metacharacter after another character in a pattern matches that character zero or one time. In other words, either the character is present in the text or it might not be, for example:

- The command `echo"13"|grep-E'12?3'` will print the line 13, since the character 2 in the `-E` parameter is optional;
- The command `echo"123"|grep-E'12?3'` will print the string 123, since the character 2 is valid in the `-E` parameter;
- The command `echo"1223"|grep-E'12?3'` will not output anything, since the character 2 is allowed only once.

The plus sign (+) metacharacter after another character in a pattern matches that character one or more times. It's similar to the \* symbol, but \* allows zero matches, while + allows at least one match. Example usage: `ls/bin|grep -E'mk+dir'` .

The curly brace metacharacters "{}" allow you to specify the number of matches of the character they follow:

- **{n}** - Match if the previous element occurs exactly n times.
- **{n,m}** - Match if the previous element occurs at least n and no more than

m times.

- **{n,}** - Match if the previous element occurs n or more times.
- **{,m}** - Match if the previous element occurs no more than m times.

### Examples of use:

```
echo "13" | grep-E '12{1,3}3'
```

```
echo "123" | grep - E '12{1,3}3'  
123
```

```
echo "1223" | grep - E '12{1,3}3'  
1223
```

```
echo "12223" | grep - E '12{1,3}3'  
12223
```

```
echo "122223" | grep-E '12{1,3}3'
```

### The sed utility

sed is a stream editor ( *from the English word "streameditor"* ) that can perform a variety of file operations, such as search and replace, insert, or delete. However, it is most often used for search and replace.

sed allows you to edit files without opening them, which is significantly faster than using something like vi. Additionally, sed supports regular expressions, which allow for complex pattern matching.

Utility format: sed[<parameters>]'<selection\_rules><instructions>'[files]

Parameters:

- -E – Use extended regular expressions. In this case, the regular expression in the selection rules is enclosed in slashes /regular expression/.
- -e – used when multiple instructions need to be specified in a command. It is placed before each instruction.
- -i ( --in-place ) - make changes to the specified file.
- -n – prohibits the output of all lines (by default, all lines are output).

Selection rules:

- **In accordance with the line number – 1.5 – from the first to the 5th line, 5.\$ – from** the fifth to the last line.
- According to regular expression.

Instructions:

- n – prohibits the output of all lines (by default, all lines are output)
- p – output lines that match the selection criteria. Often used with -n
- d – delete lines
- i'text' – insert text before the line that meets the selection criteria
- a'text' – insert text after the line that meets the selection criteria
- c'text' – replacing a string that meets the selection criteria with text
- s/search\_string/replace\_string/ – search and replace text

Full information on the sed utility can be found in the manded help file .

### Examples of use:

Output of the first 5 lines from the /etc/passwd file :

```
cat/etc/passwd | sed-n '1.5 p'  
root: x:0:0:root:/root:/bin/bash  
daemon: x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

```

bin: x:2:2:bin:/bin:/usr/sbin/nologin
sys: x:3:3:sys:/dev:/usr/sbin/nologin
sync: x:4:65534:sync:/bin:/bin/sync
Print lines 1 and 2 (removing all others) from the /etc/passwd file .
cat/etc/passwd | sed '3,$ d'
root: x:0:0:root:/root:/bin/bash
daemon: x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
Output lines 990 to 999:
cat/etc/passwd | sed-nE '/99[0-9]/ p'
systemd-coredump: x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
vboxadd: x:998:1:/:var/run/vboxadd:/bin/false
Insert text before the found line:
cat /etc/passwd | sed-e "/root/i The following line contains the root
user's data:" | head-n5
The next line contains the root user data:
root: x:0:0:root:/root:/bin/bash
daemon: x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin: x:2:2:bin:/bin:/usr/sbin/nologin
sys: x:3:3:sys:/dev:/usr/sbin/nologin
Insert text after the found line:
cat /etc/passwd | sed-e "/root/aThe previous line contains data | for user
root" | head-n5
root: x:0:0:root:/root:/bin/bash
The previous line contains the root user data.
daemon: x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin: x:2:2:bin:/bin:/usr/sbin/nologin
sys: x:3:3:sys:/dev:/usr/sbin/nologin
Insert text before and after the found string:
cat/etc/passwd | sed-e "/root/i-----" -e "/root/a | -----
-----" | head-n5
-----
root: x:0:0:root:/root:/bin/bash
-----
daemon: x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin: x:2:2:bin:/bin:/usr/sbin/nologin
Replacing the found text in the line:
cat/etc/passwd | sed-n '/localadmin/p'
localadmin: x:1000:1000:localadmin,,,:/home/localadmin:/bin/bash
cat/etc/passwd | sed-e 's#/bash#/sh#' | greplocaladmin
localadmin: x:1000:1000:localadmin,,,:/home/localadmin:/bin/sh

```

## 8.2.5 awk utility

AWK is a scripting language useful for command-line work and widely used for text processing. Text processing occurs line by line.

Using awk, you can select data—one or more individual fragments of text—based on specified selection criteria. For example, awk can search for a specific character, word, or pattern within a fragment of text, as well as select a specific row or column. Awk's capabilities enable complex text processing, reporting, data analysis, and the creation of command texts for execution by the interpreter.

Utility format: `awk[<parameters>]'<selection_rules>{<actions>}'[<files>]`

### Parameters:

With awk, you can work with strings as fields in a table. To do this, you can use the `-F` parameter to specify a delimiter, for example, `-F':'`, and access the delimited columns using built-in variables: `$0` is the entire string, `$1` is the first field, `$2` is the second field, and so on. The `NF` variable stores the number of fields in the

string, and NR stores the string number.

**Actions:**

printf or print can be used as the action .

**The following selection rules can be used:**

Regular expressions are enclosed in slashes.

Comparison operations – the following construction is used: {field\_number\_variable } { operand } {value}, for example, \$1<10. The following are used as operands:

- Arithmetic (comparisons): - > , >=, <, <=, ==, !=
- String (matches): == , !=
- Regular expressions (matches): ~ , ! ~

Examples of use:

Display usernames with uid from 990 to 999:

```
cat/etc/passwd | awk-F ':' ' /99[0-9]/{print$1}'  
systemd - coredump  
vboxadd
```

Print names, uids, and shells of users with uids between 10 and 20:

```
cat/etc/passwd | awk-F ':' '$3>10 && $3 < 20 {print $1 , $ 3,$NF}'  
proxy 13 /usr/sbin/nologin
```

Output of id and command to start all processes owned by the user:

```
psaux | awk '$1~/localad /{ print $2,$11}'  
1084 /lib/systemd/systemd  
1085 (sd-pam)  
1107 fly-wm  
1172 /usr/bin/dbus-launch  
1173 /usr/bin/dbus-daemon  
1196 /usr/bin/kglobalaccel5  
1202 /usr/bin/VBoxClient  
...
```

Complete information on the awk utility can be found in the manawk help file

.

## 8.2.6 Text editors

### Vim text editor

**Vim** is a text editor that Windows users will find quite unusual. What's most confusing to new administrators is that it uses multiple input modes, the main ones being "command" and "text."

Here are brief instructions for use:

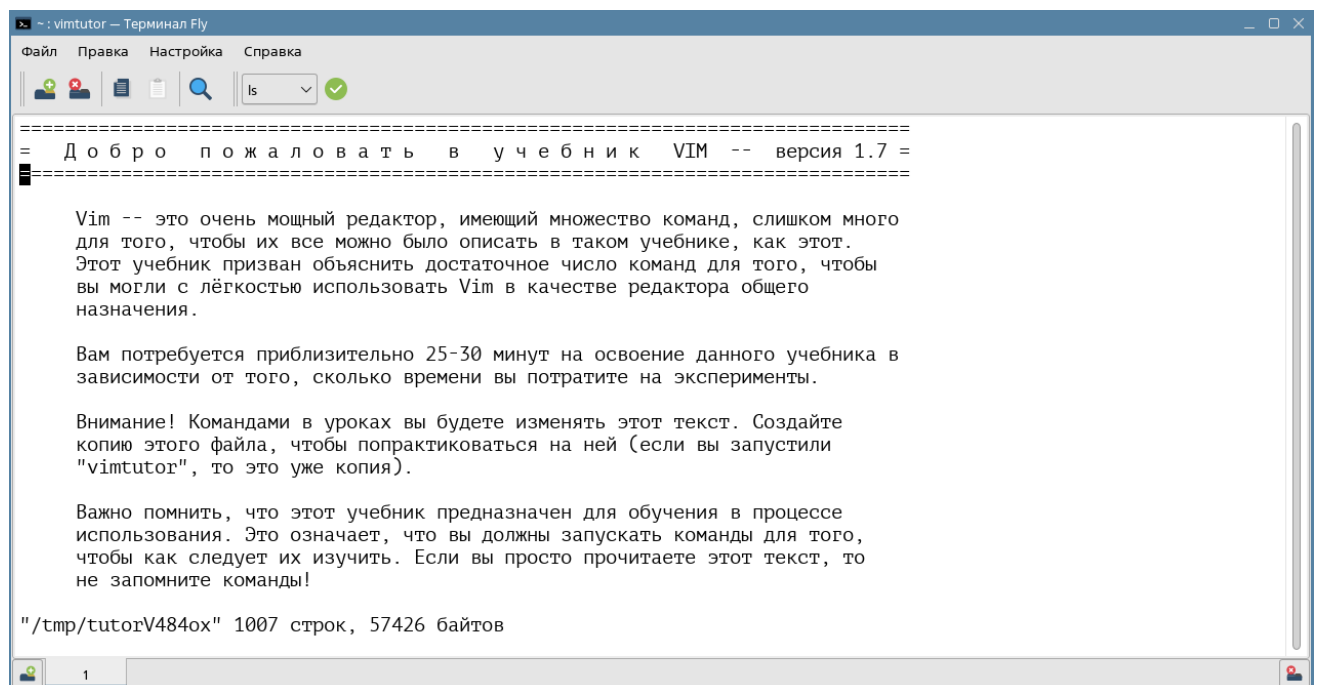
- Run the vitest.txt command to open the test.txt file for editing.
- Immediately after launching the application, you are in command mode and can, for example, type the exit command ``:q`` then press Enter to close the program.
- To edit a document, you'll need to switch to text mode using the Insert key . By pressing the Insert key repeatedly, you can select Insert or Overwrite mode.
- To copy text, simply select it with your mouse. To paste text from this buffer, hover the mouse over the desired location and press the mouse wheel. You can also use the more familiar clipboard, which is pasted

using the keyboard shortcut `ctrl + v`. To do this, hold down the `shift` key while selecting text. To copy, use the context menu commands available by right-clicking.

- To go back to command mode and save your changes, you need to press the `Esc` key and enter the command `:wq`.
- To exit without saving, use the command `:q!`.

Vim's command-line features significantly increase the speed of text processing, making it the choice of professionals who require high productivity when working from the console. However, mastering the necessary skills to the point of automatism can be challenging.

After installing the Vim editor, you have access to a built-in tutorial, which you can open with the `vimtutor` command. This opens a copy of the file, so you can practice without worrying about damaging the tutorial's content. It's highly recommended to start learning Vim by studying this tutorial.



Below we will cover the basic techniques for working with Vim.

Launching Vim is easy, but exiting it is. To launch it, simply run the command `vim[options][filename]`.

Below are some of the launch options:

- `-R` — open in read-only mode;
- `-g` — enable graphics mode;
- `-w` - save all actions to a file;
- `-b` — use binary mode to edit executable files;
- `-d` — find differences in files (you will need to specify multiple files to open);
- `-n` — disable the automatic saving function to restore the file in case of an error;
- `" +command" **` - this option is responsible for executing the command after

starting the editor;

- `+number` - moves the cursor to a user-defined line after launch;
- `+/pattern` - find the desired section in the template and move the cursor to its beginning.

Let's run `vimvim_test.txt` . We're now in command mode. To exit Vim, enter one of the following commands:

- `:q` – Exit the editor if no changes have been made. Otherwise, it won't let you exit this way.
- `:q!` – exit without saving. A faster and more convenient way to enter **ZQ** .
- `:qw` – save and exit. A faster and more convenient way to enter **ZZ** .

The `:w` command will simply save the current changes and you can continue editing the file.

Characters entered in this mode are treated as commands. You can repeat commands multiple times by entering a command and a number, or specify the scope of commands. For example, entering `d1` will delete the character under the cursor, `dw` will delete the entire word, and `d3w` will delete three words to the right of the cursor.

But you should understand that you must have the English keyboard layout enabled.

Text navigation commands:

- `h` , `l` , `j` or `k` – move the cursor left/right and up/down, but it will be easier to use the mouse arrows;
- `w` or `b` – move the cursor 1 word to the left or right;
- `H` – go to the top of the screen;
- `L` – go to the bottom of the screen;
- `G` – scroll to the end of the file;
- `zz` , `zt` , `zb` – move the line with the cursor on the screen to the middle/up/down;
- `Ctrl + u` or `Ctrl+d` – scroll the viewing area up or down by half a page.

Commands for pasting and copying:

- `p` or `P` – insert text after or before the cursor;
- `y` – copy character;
- `yy` or `Y` – copy the entire line.

Commands for undo and delete:

- `u` – undo the last action;
- `U` – undo the last action in the specified line where the cursor is located;
- `d` – delete character;
- `dd` – delete the entire line;
- `D` – delete everything from the cursor to the end of the line;
- `dt` – delete the portion of text that is located between the cursor location and the next occurrence of a specific character;
- `ciw` – delete a specific word and immediately switch to input mode;
- `c` – delete text from the cursor position to the end of the line, then automatically switch back to input mode.

Some more useful commands:

- `v` – select text;
- `ggvG=` – align text throughout the file (move to the beginning, switch to VISUAL mode, select the entire text of the document, adjust the indentation of the selected lines);
- `.` – repeat the last command again;
- `n` or `N` – continue searching forward or backward;
- `~` – switch the case of the character that is selected or on which the cursor is located;
- `/pattern` – search for occurrence;
- `%s/pattern/replace` – replace the first word with the second.

To switch to text mode, in which typed characters will be treated as typed text, press the `Insert` key. Pressing `Insert` again switches between the insert and replace text modes.

To return to command mode, simply press the `Esc` key.

Commands to switch to text mode:

- `i` (lowercase letter `i`) – switch to text mode and go to the beginning of the line;
- `A` – switch to text mode and go to the end of the line;
- `a` – switch to text mode and move to the next character;
- `r` – replace one character;
- `R` – replace several characters in a row;
- `O` – create a line below;
- `o` – create a line above.

In command line mode, entering a command begins with the `:` character. We already learned about this when we discussed quitting and saving a file. Below are some other commands in this mode:

- `:e file` – create a new file;
- `:r file` – insert the contents of this file into the current file;
- `:r!` – execute a shell command and insert the received response into Vim;
- `:buffers` – view open files;
- `:set variable=value` – set the value of a variable to control editor processes.

In this mode, you can use navigation commands to select part of the text (or all of it) for further editing. This is convenient because the user can see the selected section.

Using the visual mode comes down to performing three operations:

1. `v` hotkey to start selecting text.
2. Move your cursor to the end of the visual fragment to select text.
3. Enter a command operator that will perform an action on the selected text.  
For example, typing the `~` symbol will change the case of the characters and convert all text to uppercase.

Here are some useful command operators:

- `y` - copy text to clipboard

- p - paste text from the clipboard
- ~ - change the case of characters
- d - delete selected text

In addition to the character-by-character mode v, there is also a line-by-line mode V and a block mode Ctrl + V.

### Nano text editor

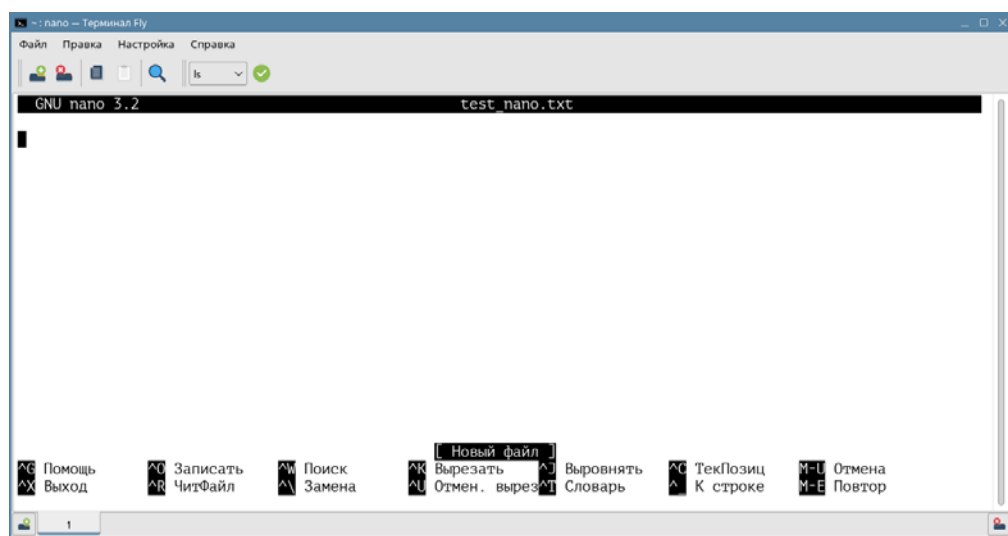
The nano text editor is part of the GNU suite and is the standard console editor for many Linux distributions. Nano's main advantage is its ease of use—it's as easy to use as notepad.exe, so you'll quickly master it to a high level for any task.

To launch it, simply call the command `nano[options][file_name]` .

Some of the options:

- +number - indicate the line in the file from which editing should begin;
- -B - create a backup copy of the file when saving;
- -c - backup folder;
- -D - display text in bold;
- -E - convert tabs to spaces;
- -I - do not read configuration from nanorc files;
- -P - remember and restore the cursor position during the last editing;
- -m - enable mouse support;
- -v - read-only mode, you cannot save the file.

Let's launch the text editor `nanotest_nano.txt` .



Unlike Vim, there's no special mode; you can navigate through the text and edit it directly. The bottom of the window displays a tooltip for commands, which can change depending on the current state. Combinations use Ctrl or Alt keys with other keys. As a reminder, the ^ symbol is shorthand for the Ctrl key .

The full list of commands is available by pressing F1 or Ctrl + G .

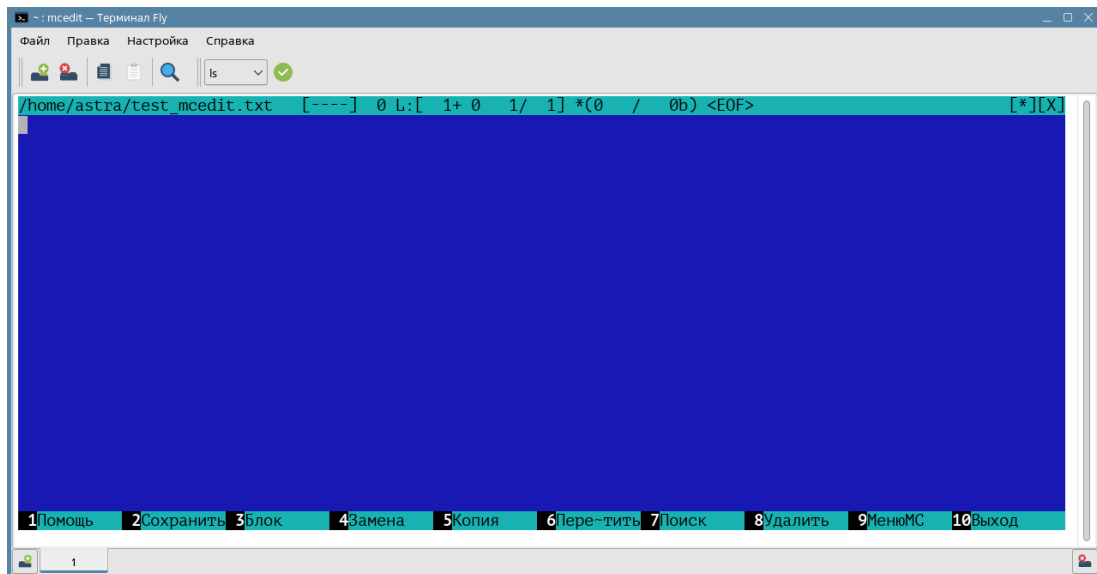
The nano editor can support regular expression searches. To do this, the `setregexp` parameter must be set in the user profile configuration file `~/.nanorc` .

You can enable syntax highlighting for popular programming languages in the editor. To do this, add highlighting configuration files located in `/usr/share/nano` to the `~/.nanorc` configuration file using the `include` directive . For example, to

highlight sh, add the line "include /usr/share/nano/sh.nanorc."

### **mcedit text editor**

mcedit editor is installed along with the Midnight Commander file manager and has a similar interface.



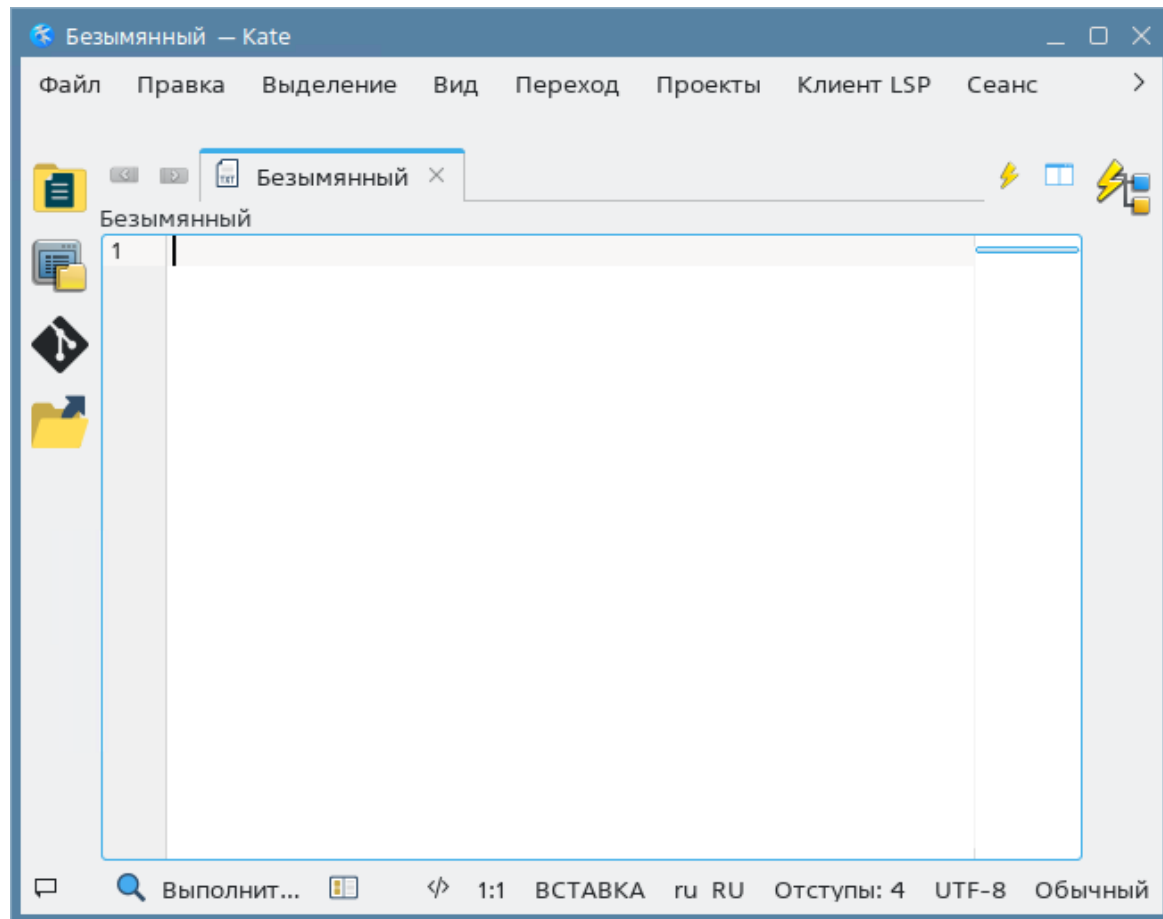
The bottom line lists the basic commands that are executed using the function keys F1-10 . For example, help is accessed by pressing F1 , and the main menu is accessed by pressing F9 .

Like nano, the mcedit editor supports regular expressions and syntax highlighting for popular programming languages and configuration files. To manually control the highlighting, use the Menu (F9)

► Options

### **Graphical text editor Kate**

The graphical text editor kate has an intuitive interface and offers extensive customization options. Its functionality is very similar to Notepad++, meaning it's much larger than Notepad but smaller than Visual Studio Code.



Like the editors mentioned above, kate supports regular expressions and syntax highlighting for programming languages and configuration files. It's recommended for beginners (and others) to use in graphical mode.

### 8.3 Assignment for independent completion

#### 8.3.1 Task 1.

Create a text file named sample with the following contents:  
Good afternoon, Petr Vasiliev!

The framework and location of personnel training represent an interesting experiment

for further development directions, see Fig. 10. Ideological considerations of the highest order, as well as consultation with a broad asset contribute preparation and implementation of further development directions according to the standard

ISO-9001. It should not be forgotten, however, that the framework and location of personnel training

contribute to the preparation and implementation of the development model. Everyday practice shows that strengthening and developing the structure require the definition and clarification of the personnel training system, which corresponds to urgent needs.

#### 8.3.2 Task 2.

Create another file "sign" with the signature:  
Write to us by email: info@company.lan

Mobile: +79881112230  
Office multi-channel: 8 800 2201100  
Website: [www.company.lan](http://www.company.lan)

### **8.3.3 Task 3.**

Output the text of the message to the standard output stream.

### **8.3.4 Task 4.**

Display the contents of two files on the screen in a row with one command.

### **8.3.5 Task 5.**

Find the number of lines in a file.

### **8.3.6 Task 6.**

Print the first line from the sample file.

### **8.3.7 Task 7.**

Find the name in the first sentence and remove the space from the name.

### **8.3.8 Task 8.**

Print the contents of the "sample" and "sign" files to the screen, numbering the lines in the output.

### **8.3.9 Task 9.**

Combine the contents of the "sample" and "sign" files and write it into a new file called "message".

### **8.3.10 Task 10.**

Print only the last line of the file "message" to standard output.

### **8.3.11 Task 11.**

Print the image number from the "message" file to standard output. Use the `grep` command and a regular expression to do this.

### **8.3.12 Task 12.**

Print the email address from the file "message" to standard output.

### **8.3.13 Task 13.**

8800 from the file "message" to standard output .

**8.3.14Task 14.**

Print international phone numbers from the file "message" to standard output.

**8.3.15Task 15.**

Remove two or more spaces from the "message" file.

**8.3.16Task 15.**

Take a short test <https://onlinetestpad.com/zsivbzz7mljrm>.

## Literature

1. Shirokov, A. I. Basics of working with the Astra Linux operating system: guidelines / A. I. Shirokov. - Moscow: Publishing House MISiS, 2022. - 68 p.
2. Fedotova, D. E. OS Windows & OS Linux: laboratory work on the course "Operating Systems" / D. E. Fedotova. - Moscow: Russian New University, 2009. - 224 p.
3. Krishchenko, V. A. Fundamentals of programming in the kernel of the GNU/Linux operating system: a tutorial / V. A. Krishchenko, N. Yu. Ryazanova. - Moscow: Bauman Moscow State Technical University, 2010. - 36 p.
4. Mamaylenko, S. N. Operating systems. Part 1. Linux operating system: practical course / S. N. Mamaylenko. - Novosibirsk: Siberian State University of Telecommunications and Informatics, 2008. - 119 p.
5. Shmakov, V. E. Open systems and Linux technologies: a tutorial / V. E. Shmakov, M. V. Khludova. - St. Petersburg: Peter the Great St. Petersburg Polytechnic University, 2018. - 58 p.
6. Gunko, A. V. System programming in the Linux environment: a tutorial / A. V. Gunko. - Novosibirsk: Novosibirsk State Technical University, 2020. - 235 p.
7. Voitov, N. M. Basics of working with Linux: a training course / N. M. Voitov. - Moscow: DMK Press, 2019. - 216 p.
8. Goncharuk, S. V. Linux OS Administration: a tutorial / S. V. Goncharuk. - 4th ed. - Moscow: Internet University of Information Technologies (INTUIT), IP R Media, 2024. - 163 p.
9. Morenkova, O. I. Operating Systems. Learning Linux: A Tutorial / O. I. Morenkova, A. Yu. - Novosibirsk: Siberian State University of Telecommunications and Informatics, 2022. - 107 p.
10. Akinin, M. V. System programming in Linux. Part 1. Process management: a tutorial / M. V. Akinin, N. V. Akinina, S. V. Zasorin. - Moscow: KURS, 2024. - 192 p.
11. Akinin, M. V. System programming in Linux. Part 2. File systems: a tutorial / M. V. Akinin, N. V. Akinina, S. V. Zasorin. - Moscow: KURS, 2024. - 124 p.

**MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION**  
**Federal State Autonomous Educational Institution**  
**higher professional education**  
**NORTH CAUCASIAN FEDERAL UNIVERSITY**

Methodological recommendations  
on organizing independent work  
by discipline  
Intelligent Terminal Operating System / Интеллектуальная терминальная  
операционная система  
for students in the field of study  
02.04.01 Mathematics and Computer Science

### **Guidelines for organizing independent work**

The educational process at a higher education institution relies heavily on independent student work, without which it is difficult to fully master complex course material and learn to continually improve acquired knowledge and skills. Independent work is outside of the classroom and is designed to allow students to independently familiarize themselves with specific sections of the course, using materials recommended by the instructor, and to prepare for individual course assignments. The purpose of independent student work is to:

- to teach students to work meaningfully and independently, first with educational material, then with scientific information, to lay the foundations of self-organization and self-education in order to instill the ability to continuously improve their qualifications in the future.
- consolidation, expansion and deepening of knowledge, skills and abilities acquired by students in classroom lessons under the guidance of teachers;
- students studying additional materials on the subjects they are studying and the ability to select the necessary material from various sources; developing in students independence, organization, self-discipline, creativity, the need to develop cognitive abilities and perseverance in achieving their goals.

The proposed approach to mastering the material increases motivation for classroom and extracurricular activities, which ensures the necessary level of knowledge in the studied disciplines and allows for increased student readiness to pass exams.

When beginning a new course, students should familiarize themselves with the curriculum, the academic, scientific, and methodological literature available in the library, obtain recommended textbooks and teaching aids from the library, and start a new notebook for taking notes from lectures and working with primary sources. During lectures, students should take notes on the course material. Pay attention to categories and formulations that reveal the essence of certain phenomena and processes, scientific conclusions, and practical recommendations. It is advisable to leave margins in their working notes for making notes from recommended literature that supplement the lecture material and highlight the particular importance of certain theoretical principles. Ask the instructor clarifying questions to clarify theoretical principles and resolve controversial situations. In preparation for seminars, study the primary literature, familiarize themselves with additional literature, and read new publications in periodicals (magazines, newspapers, etc.), taking into account the instructor's recommendations and the requirements of the curriculum. Refine your lecture notes, making relevant notes from the literature recommended by the instructor and stipulated by the curriculum. Prepare presentation outlines on all academic topics covered during the seminar. When preparing a report or abstract, seek guidance from the instructor. Create a plan for your presentation. Actively participate in the discussion of academic topics: present reports, abstracts, reviews of research articles, and individual periodicals related to the lesson topic. Use teaching aids, such as a blackboard and chalk, during your presentation. Ask the instructor questions to deepen your understanding of the material being studied. After summing up the seminar, address any deficiencies noted by the instructor. When preparing for the exam (at the end of the semester), review the material covered in strict accordance with the curriculum and the sample list of academic questions included in the syllabus. Use lecture notes and the literature recommended by the instructor. Pay particular attention to the topics of classes the student missed for various reasons. If necessary, seek advice and methodological assistance from the instructor.

#### **1. Methodological recommendations for students on certain forms of independent work**

The university education system requires students to have significantly greater independence in planning and organizing their activities. This can be quite challenging for a recent high school student: while at school, daily teacher supervision forced students to constantly and systematically prepare for classes, at university, the question of their knowledge level only really arises during exam periods. This situation leads some to the temptation to devote the entire semester to free time ("I'll learn it when I need to!"), and when exam time arrives, the material to be mastered is so overwhelming that no memory can cope with it in the remaining time.

#### **2. Rules and techniques for taking lecture notes**

- Record information according to the lecture outline, dividing the text into logical blocks. Use numbers to clearly understand the sequence of presentation. It's a good idea to include a line indentation in your notes.

- Capitalize key terms, highlighting them when writing the heading. For example, the phrase "intangible assets in the asset structure" could be written as "intangible assets in structure A." If any terms overlap, it makes sense to mark them with a circle or other identifying symbol;

- Core concepts that are used frequently in a given subject can be replaced with specific abbreviations that will be relevant to the specific notes. If you think you'll forget the abbreviations, don't be lazy and write them down at the end of your notebook;

- Abbreviating long, common words can speed up note-taking. For example, the word "state" can be replaced with the abbreviated "gos-vo," and so on.

- **Quick note-taking** is essential when using your own importance icons. Create your own or use the following:

- o "!" - important;

- o "?" - questionable;

- o "U" - look at the textbook;

- o "ZB" - for example;

- o "R" - double check.

- It makes sense to write the definitions of key concepts in a separate paragraph, highlighting them in any way possible. Underlining with a colored marker makes the memorization process easier;

- Speed note-taking requires the use of mathematical notations. Obviously, writing "+" instead of "plus" and " $\Sigma$ " instead of "sum" is much more productive;

- Don't neglect commonly used abbreviations. Replacing terms like "database management system" with "DBMS" is essential;

- Periodizations and classifications are best written in the form of diagrams and charts; These skills will help make your notes more understandable and significantly reduce the time it takes to record information.

These skills can be used not only at university, but also when attending continuing education courses, seminars, trainings, and various conferences. Knowing how to take notes effectively will help you easily digest large amounts of information, making it concise, simple, and understandable.

You can find increasingly productive methods for capturing information yourself. Remember, notes are a condensed transcript of a written or spoken text, created for personal use. For this reason, no two people can have identical notes. Don't completely imitate someone else's methods; create your own unique ones.

### 3. Working with a book

When working with books, it's important to select literature, learn how to read it correctly, and keep notes. Library catalogs use alphabetical and systematic indexes to select literature. It's important to remember that effective book-reading skills always save a lot of time and effort.

The instructor teaching the course recommends the appropriate selection of textbooks. Required literature may also be listed in the course's methodological guidelines. When studying the material in the textbook, students should move on to the next question only after they have thoroughly understood the previous one, writing down all calculations and computations (including those omitted from the textbook or given in the lecture for independent derivation). Independent, individual work plays a significant and important role in the study of any discipline.

Particular attention should be paid to defining the course's key concepts. Students should thoroughly analyze examples that illustrate these definitions and be able to construct similar examples independently. It's important to develop a clear understanding of what you're learning. Taking supporting notes is helpful. When studying the material from the textbook, it's helpful to supplement your lecture notes in the designated margins in your notebook. You should also note any questions you've identified for consultation with the instructor.

It's recommended to highlight the conclusions drawn from the study in your notes to help you remember them better when rereading your notes. Experience shows that many students find it

helpful to create a reference sheet containing the most important and frequently used formulas and concepts. Such a sheet helps you remember the formulas and key points of the lecture, and can also serve as a constant reference for the student.

#### **4. The main types of systematic recording of what has been read:**

1. Annotation is an extremely brief, coherent description of a book (article) viewed or read, its content, sources, nature and purpose;
2. Planning is a brief logical organization of the text, revealing the content and structure of the material being studied;
3. Thesis is a concise reproduction of the author's main assertions without using factual material;
4. Quoting is the verbatim copying of excerpts and extracts from the text that most significantly reflect a particular idea of the author;
5. Note-taking is a brief and sequential presentation of the content of what has been read.

A summary is a complex way of presenting the contents of a book or article in a logical sequence. A summary integrates previous notes and allows for a comprehensive understanding of the book or article's content. Therefore, the ability to create an outline, make notes, take excerpts, and other notes determines the technique of summary writing.

#### **6. Guidelines for writing notes:**

1. Read the text carefully. Look up any unfamiliar words in reference books. When taking notes, be sure to include reference information in the margins of your notes;
2. Highlight the main points, make a plan;
3. Briefly formulate the main points of the text, note the author's argumentation;
4. Take notes on the material, strictly following the outline. When taking notes, try to express your thoughts in your own words. Keep your notes clear and concise.
5. Write quotes correctly. When quoting, keep in mind the brevity and significance of the idea.

In the summary, it is advisable to present not only thesis statements but also their evidence. When formatting the summary, strive for the conciseness of each sentence. The author's thoughts should be presented concisely, with care taken to maintain style and expressiveness. The number of additional elements in the summary should be logically justified, and entries should be arranged in a specific sequence consistent with the logical structure of the work. Leave margins for clarification and addition.

Mastering note-taking skills requires a student to be determined and to work independently on a daily basis.

#### **7. When starting work, each student should take into account the following provisions.**

The discipline (module) is structured on a thematic principle, each topic is a logically complete section.

The lecture material is devoted to the examination of key, basic provisions of the courses and an explanation of the educational assignments assigned to students for independent work.

Laboratory work is carried out with the aim of consolidating the acquired information and acquiring skills for its application in solving practical problems in the relevant subject area.

Students' independent work is aimed at independent study of additional material, preparation for practical classes, and completion of all types of independent work.

To successfully master the discipline, it is necessary complete all types of independent work using the recommended sources of information.

#### **8. Educational, methodological and informational support of the discipline**

1. Shirokov, A. I. Basics of working with the Astra Linux operating system: guidelines / A. I. Shirokov. - Moscow: Publishing House MISiS, 2022. - 68 p.
2. Fedotova, D. E. OS Windows & OS Linux: laboratory work on the course "Operating Systems" / D. E. Fedotova. - Moscow: Russian New University, 2009. - 224 p.
3. Krishchenko, V. A. Fundamentals of programming in the kernel of the GNU/Linux operating system: a tutorial / V. A. Krishchenko, N. Yu. Ryazanova. - Moscow: Bauman Moscow State

- Technical University, 2010. - 36 p.
4. Mamaylenko, S. N. Operating systems. Part 1. Linux operating system: practical course / S. N. Mamaylenko. - Novosibirsk: Siberian State University of Telecommunications and Informatics, 2008. - 119 p.
  5. Shmakov, V. E. Open systems and Linux technologies: a tutorial / V. E. Shmakov, M. V. Khludova. - St. Petersburg: Peter the Great St. Petersburg Polytechnic University, 2018. - 58 p.
  6. Gunko, A. V. System programming in the Linux environment: a tutorial / A. V. Gunko. - Novosibirsk: Novosibirsk State Technical University, 2020. - 235 p.
  7. Voitov, N. M. Basics of working with Linux: a training course / N. M. Voitov. - Moscow: DMK Press, 2019. - 216 p.
  8. Goncharuk, S. V. Linux OS Administration : a tutorial / S. V. Goncharuk. - 4th ed. - Moscow: Internet University of Information Technologies (INTUIT), IP R Media, 2024. - 163 p.
  9. Morenkova, O. I. Operating Systems. Learning Linux: A Tutorial / O. I. Morenkova, A. Yu. - Novosibirsk: Siberian State University of Telecommunications and Informatics, 2022. - 107 p.
  10. Akinin, M. V. System programming in Linux. Part 1. Process management: a tutorial / M. V. Akinin, N. V. Akinina, S. V. Zasorin. - Moscow: KURS, 2024. - 192 p.
  11. Akinin, M. V. System programming in Linux. Part 2. File systems: a tutorial / M. V. Akinin, N. V. Akinina, S. V. Zasorin. - Moscow: KURS, 2024. - 124 p.