Содержание

Administrator's Guide

General principles of VEOS operation

Processes and files

VEOS is a multi-user integrated system, i.e. it is designed for simultaneous operation of several users.

The user can either work in the system himself, executing some sequence of commands, or application processes can be executed on their behalf. The user interacts with the system through a command interpreter, for example, bash. A command interpreter is an application program that accepts commands or a commands set from the user and converts them into system calls to the system kernel. The interpreter allows the user to work with files, navigate through the file system tree, and run application processes. All UNIX command interpreters have a developed command language and allow to write complex programs that simplify the process of system administration and working with it.

OS functioning processes

All programs that are loaded into the server memory (i.e. executed) at the current time are called processes. The processes can be divided into two main classes: system processes and user processes. System processes are programs that solve internal OS tasks, for example, organizing virtual memory on disk or providing users with certain services (processes—services). User processes are processes launched by the user from the command interpreter to solve user tasks or control system processes.

VEOS was originally developed as a multitasking system. It uses technologies tried and tested by other UNIX implementations - VEOS predecessors. The process background mode is the mode when the program can work without user interaction. If interactive work with the user is needed (in general), the process will be "stopped" by the kernel, and its work continues only after it is transferred to the "normal" operation mode.

OS File System

VEOS uses the Linux file system, which is a single tree. The root of this tree is a directory called root and denoted by /. Parts of the file system tree can be physically located in different partitions of different disks or on other computers altogether — this is clear to the user. The process of attaching a partition file system to a tree is called mounting, and deleting is called unmounting. For example, the CD-ROM file system in the distribution is mounted by default in the /media/cdrom directory (the path in the distribution is indicated using /, but not \, as in Windows). The current directory is denoted by ".", the parent directory (of the above level) – by "..", for example, to go to the top-level directory:

```
> cd ..
```

Directory structure

Root directory /:

- /bin — command shells, basic utilities;
- /boot — contains the system kernel;
- /dev — pseudo files of devices that allow to work with devices directly. Files in /dev are created by the udev service;
- /etc — system-wide configuration files for most programs in the system;
- /etc/rc?.d, /etc/init.d, /etc/rc.boot, /etc/rc.d — directories where the command files are located that are executed when the system is started or when its operating mode is changed;
- /etc/passwd – a user database that contains information about the user name, their real name, personal directory, their encrypted password and other data;
- /etc/shadow — shadow user database. In this case, the information from the /etc/passwd file is moved to /etc/shadow, which is unreadable for everyone except the root user. If an alternative shadow password management scheme (TCB) is used, all shadow passwords for each user are located in the /etc/tcb/username/shadow directory;
- /home — users' home directories;
- /lib — contains files of dynamic libraries required for the operation of most applications, and loadable kernel modules;
- /lost+found — recovered files;
- /media — pluggable media (directories for mounting removable devices file systems);
- /mnt — temporary mount points;
- /opt — helper packets;
- /opt/vasexperts – VAS Experts parent product directory;
- /proc – a virtual file system stored in the computer memory when the OS is loaded. This directory contains the latest information about all processes running on the computer;
- /root — home directory of the system administrator;
- /run — application status files;
- /sbin — a set of programs for administrative work with the system (system utilities);
- /selinux — SELinux virtual file system;
- /srv — virtual service data;
- /sys — a file system containing information about the current state of the system;
- /tmp — temporary files;
- /usr — user binaries and read-only data (programs and libraries);
- /var — files for storing changing data (working program files, queues, logs).

/usr directory:

- /usr/bin — additional programs for all accounts;
- /usr/sbin — commands used during system administration and not intended to be placed in the root file system;
- /usr/local – a location where it is recommended to place files installed without using packet managers, the internal organization of directories is almost the same as the root directory;
- /usr/man — directory where the man reference manual files are stored;
- /usr/share – a directory for hosting public files of most applications.

/var directory:

- /var/log — the place where the system and application audit files are stored;
- /var/spool – a directory for storing files in the processing queue for a particular process (print

queues, unread or unsent emails, cron tasks, etc.).

File structure organization

The system of users home directories helps to organize the safe work of users in a multi-user system. Outside of their home directory, the user has minimal rights (usually reading and executing files) and cannot damage the system, for example, by deleting or modifying a file. In addition to the files created by the user, their home directory usually contains personal configuration files of some programs. A route (path) is a sequence of directory names representing a path in the file system to a given file, where each subsequent name is separated from the previous one by a slash. If the route name starts with a slash, then the path to the desired file starts from the root directory of the entire system tree. In the opposite case, if the route name starts directly with the file name, then the path to the file being searched should start from the current directory (working directory). The file name can contain any characters except for the forward slash (/). However, you should avoid using most punctuation marks and non-printable characters in file names. When choosing file names, it is recommended to be limited to the following characters:

- lowercase and uppercase letters;
- underscore character (_);
- dot (.).

File and directory names are case-sensitive, i.e. touch test.txt and touch TEST.TXT commands will create two different files with the corresponding names

For convenience, the dot can be used to separate the file name from the file extension. This feature may be necessary for users or some programs, but it does not matter for the shell.

Disk and partition names

All physical devices of your computer are displayed in the /dev directory of the distribution file system (more on this below). Disks (including IDE/SATA/SCSI/SAS hard drives, USB drives) have names:

- /dev/sda — the first disk;
- /dev/sdb — the second disk;

etc. Disks are designated /dev/sdX, where X is a, b, c, d, e, ... depending on the serial number of the disk on the bus. A disk partition is indicated by a number after its name. For example, /dev/sdb4 is the fourth partition of the second disk.

Partitions required for VEOS operation

For the OS operation, at least two partitions shall be created on the hard disk(s): the root one (that is, the one that will contain the / directory) and the swap partition. The size of the latter, as a rule, ranges from one to two times the computer's RAM size. If there is a lot of free space on the disk, then

separate partitions for the /usr, /home, /var directories may be created.

Working with the most frequently used components

Virtual Console

VEOS provides access to virtual consoles from which several working sessions in the system may be carried out simultaneously. Switching between the first six virtual consoles is performed by pressing the keyboard shortcut Alt+F1 — Alt+F6 (Ctrl+Alt+F1 — Ctrl+Alt+F6).

Command shells

Command interpreters (shell) are used to control VEOS. After logging in, you will see a prompt — a string containing the "$" symbol (this symbol will further denote the command string). The program is waiting for your commands. The role of the command interpreter is to transmit your commands to the operating system. With the help of command interpreters, small script programs (scripts) may be written. The following command shells are available in VEOS:

- bash — Bourne Again Shell, the most common shell
- ksh — Korn Shell, a well-known shell in UNIX systems.

You can check which shell is currently being used by running the command:

```
$ echo $SHELL
```

The default shell is Bash — the most common shell under Linux, which keeps a command history and provides for their editability.

Bash Command Shell

Bash has several techniques for working with a command string. For example, the following combinations can be used:

- Ctrl+A — go to the beginning of the string;
- Ctrl+U — delete the current string;
- Ctrl+C — stop the current task.

To enter multiple commands in one string, the separator ";" can be used. You can navigate through the history of commands using the keys ↑ ("up") and ↓ ("down"). To find a specific command in the typed list without scrolling through the entire history, you can press Ctrl+R and start typing the characters of the previously entered command. To view the history of commands, you can use the history command. The commands present in the history are shown numbered in the list. To run a specific command, you need to enter:

```
> !command number
```

If you enter:

```
> !!
```

the last of the entered commands will start.

In Bash, command names may be substituted from the general list of commands, which greatly facilitates working with the system. When the Tab key is pressed, Bash completes the command, program or directory name, or displays several alternative options, suggesting the user to enter the next character and press the Tab key to clarify the choice. For example, to use the gunzip decompression program, you can enter the following command:

```
> gu
```

Then press the Tab key. Since there are several possible ways to complete the command in this case, you need to press the Tab key again to get a list of names starting with gu. In the proposed example, the following list can be got:

```
> gu
guile gunzip gupnp-binding-tool
```

If you type: n (gunzip is the only name, the third letter of which is "n"), and then press the Tab key, the shell will complement the name itself. To run the command, press Enter. Bash searches for programs called from the command string in directories defined in the $PATH system variable. By default, this directory list does not include the current directory, denoted by ./ (dot slash). Therefore, to run the program from the current directory, you shall use the command (in the example, the test command is run):

```
> ./test
```

Command

The simplest command consists of a single "word" and can be either built into the interpreter or be a separate command located in an executable binary file on disk. The text that follows the command is called parameters (or arguments) and they are entered to change the command behavior. In most cases, the first word is considered the command name, and the rest are its parameters. The behavior of commands can be controlled, for example, to change the output format of the ls command, you can use the -l key.

```
> ls
> ls -l
```

Such parameters are called keys or execution modifiers. The key belongs to this particular command and does not make sense by itself. This is how it differs from other parameters (for example, file names, numbers) that have their own meaning, independent of any command. Each command can recognize a certain set of keys and change its behavior accordingly. The same key can define completely different values for different commands.

There is no rigid standard for the key format, but there are agreements:

- If the key starts with -, then this is a simple key. - is usually followed by a single character, most often a letter denoting the action or property that this key gives to the command. This makes it easier to distinguish keys from other parameters.
- If the key starts with –, then it is called an extended key. The extended key format begins with two –characters, followed by the full name of the content denoted by this key.

Some keys have both a simple and an extended format, and some have only one of the formats. Information about the resources of each command can be obtained using the –help or -h key. For example, you can get a hint about what the touch command does by typing touch –help in the terminal.

Docking commands in the VEOS system

Standard input and standard output

Многие команды системы имеют так называемые стандартный ввод (standard input) и стандартный вывод (standard output), часто сокращаемые до stdin и stdout. Ввод и вывод — это входная и выходная информация для данной команды. Программная оболочка делает так, что стандартным вводом является клавиатура, а стандартным выводом — экран монитора. Пример с использованием команды cat. По умолчанию команда cat читает данные из всех файлов, которые указаны в командной строке, и посылает эту информацию непосредственно в стандартный вывод (stdout). Следовательно, команда: cat /etc/os-release выведет на экран сначала содержимое файла. Если имя файла не указано, программа cat читает входные данные из stdin и возвращает их в stdout. Пример: cat Привет. Привет. Пока. Пока. Ctrl-D Каждую строку, вводимую с клавиатуры, программа cat возвращает на экран. При вводе информации со стандартного ввода конец текста сигнализируется вводом специальной комбинации клавиш, как правило, Ctrl+D. Сокращённое название сигнала конца текста — EOT (end of text).

Many system commands have so-called standard input and standard output, often abbreviated to stdin and stdout. Input and output are the input and output information for a given command. The software shell makes it so that the standard input is the keyboard, and the standard output is the monitor screen. Example using the cat command. By default, the cat command reads data from all files that are specified on the command string and sends this information directly to standard output (stdout). Therefore, the cat /etc/os-release command will first display the file contents. If no file name is specified, the cat program reads the input data from stdin and returns it to stdout. Example: cat
Hello.
Hello.
Bye.
Bye.
Ctrl-D

The cat program returns each string entered from the keyboard to the screen. When entering information from standard input, the end of the text is signaled by entering a special key combination, usually Ctrl+D. The abbreviated name of the text end signal is EOT (end of text).

Redirecting input and output

If necessary, the standard output may be redirected using the > symbol and standard input using the < symbol. Filter is a program that reads data from standard input, processes it in some way, and directs the result to standard output. When redirection is applied, files can act as standard input and output. As mentioned above, by default, stdin and stdout refer to the keyboard and the screen, respectively. The sort program is a simple filter — it sorts the input data and sends the result to the standard output. The cat program is a very simple filter — it does nothing with the input data, but simply sends them to the output.

Using docked commands

Command docking (pipelines) is carried out by the command shell, which directs the stdout of the first command to the stdin of the second command. The | symbol is used for docking. Send the ls command stdout to the sort command stdin:

```
> ls -la /etc| sort -r
-rw-r--r--.  1 root root        9 июн  7  2013 host.conf
-rw-r--r--.  1 root root      970 авг  8  2019 yum.conf
-rw-r--r--.  1 root root      966 авг  9  2019 rwtab
-rw-r--r--.  1 root root       94 мар 24  2017 GREP_COLORS
```

Output a list of files in parts:

```
> ls /usr/bin | more
```

If you need to display the last alphabetically named file in the current directory, the following command can be used:

```
> ls | sort -r | head -1 notes
```

where the head -1 command displays the first string of the input string stream it receives (in the example, the stream consists of data from the ls command) sorted in reverse alphabetical order.

Redirecting output in Append mode

The effect of using the > symbol to redirect the file output is destructive; i.e., the command

```
> ls > file-list
```

will destroy the contents of the file-list file, if this file previously existed, and create a new file in its place. If the redirection is done using the » characters instead, the output will be assigned to the end of the specified file, while the original contents of the file will not be destroyed.

Input and output redirection and command docking are performed by command shells that support the use of >, » and | characters. The commands themselves are not able to perceive and interpret these symbols.

Superuser Mode

VEOS is a multi—user system, and therefore the user is a key concept for organizing the entire access system. The files of all users in VEOS are stored separately, each user has their own home directory in which they can store their data. Other users' access to the user home directory may be restricted. A superuser in VEOS is a dedicated system user who is not subject to access restrictions. It is the superuser who has the ability to arbitrarily change the owner and group of the file. They have read and write access to any system file or directory. Among the VEOS accounts, there is always a superuser account — root. Therefore, instead of "superuser", they often say "root". A lot of system files belong to root, a lot of files are only available to it for reading or writing. The password of this account is one of the most valuable things about the system. It is with its help that system administrators perform the most responsible work.

System utilities require superuser privileges for their work, because they make changes to system files. When they are launched, a warning is displayed about insufficient rights of the current user.

For experienced users who know how to work with the command string, there are two different ways to get superuser rights. The first is to log in as root. The second way is to use a special su (shell of user) utility, which allows to execute one or more commands on behalf of another user. By default, this utility executes the sh command from the root user, that is, it starts the command interpreter. The difference from the previous method is that it is always known who exactly launched su, which means it is clear who performed a certain administrative action. In some cases, it is more convenient to use not su, but the sudo utility, which allows to execute only preset commands.

To use the su and sudo commands, you shall be a member of the wheel group. The user created during the system installation is already included in this group by default.

To switch to superuser mode, enter the su - command in the terminal.

If you use the su command without a key, the command interpreter with root rights is called.

At the same time, the value of the environment variables, in particular $PATH, remains the same as the user's: there will be no /sbin, /usr/sbin directories in the $PATH variable, the route, shutdown, mkswap and other commands will be unavailable without specifying the full name. Moreover, the $HOME variable will indicate the user's directory, all programs running in superuser mode will save their settings with root rights in the user's directory, which may cause problems in the future. To avoid this, you should use su -. In this mode, su will launch the command interpreter as a login shell, and it will behave exactly as if root had registered in the system.

User Management

Users and groups within the system are designated by digital identifiers — UID and GID, respectively. A user can be a member of one or more groups. By default, they belong to the group that matches their name. To find out which other groups the user belongs to, enter the id command, its output can be something like this:

```
uid=500(test) gid=500(test) группы=500(test),16(rpm)
```

Such an entry means that the user test (digital identifier 500) is a member of the test and rpm groups. Different groups may have different access levels to certain directories; the more groups a user enters, the more rights they have in the system.

Due to the fact that most of the privileged system utilities in VEOS have not SUID-, but SGID-bits, be extremely careful and cautious in reassigning group rights to system directories.

The passwd command

The passwd command supports the traditional options of passwd and shadow utilities. Syntax:

```
passwd [PARAMETERS...] <user>
-k, --keep-tokens   save outdated authorization data (passwords)
-d, --delete             delete user password (root only)
-l, --lock                  lock user password (root only)
-u, --unlock            unlock user password (root only)
-e, --expire              expire user password (root only)
-f, --force                 forced execution
-x, --maximum=DAYS   maximum password validity period (root only)
-n, --minimum=DAYS   minimum password validity period (root only)
-w, --warning=DAYS    how many days before the password expires it is
required to start warning the user (root only)
-i, --inactive=DAYS       how many days after the password expires the
account shall be locked (root only)
-S, --status                    report the password status for the user
(root only)
--stdin                      get a new value from stdin (root only)
```

Exit code: Upon successful completion, passwd terminates with exit code 0. Exit code 1 means that an error has occurred. The text description of the error is output to the standard error stream. The user can change their password at any time. The only thing required to change the password is to know the current password. Only the superuser can update another user's password.

Adding a new user

To add a new user, use the useradd and passwd commands:

```
> useradd test
> passwd test
The test user password is changed.
New password :
```

As a result, a test user with a specified password has appeared in the system. If the password turned out to be too weak from the system point of view, it will warn you about it. The user can later change their password using the passwd command — but if they try to put a weak password, the system will

refuse to change it (unlike root). The useradd program has many parameters that allow to change its default behavior. For example, you can force to specify which UID will be or which group the user will belong to.

User records modification

The usermod utility is used to modify user records:

```
> usermod -G wheel,test test
```

Such a command will change the list of groups that the test user belongs to — now it is wheel, test.

```
> usermod -l test1 test
```

The user name will be changed from test to test1.

The usermod -L test1 and usermod -U test1 commands, respectively, temporarily block the ability to log in to the test1 user and return everything to their places. The changes will take effect only when the user logs in the next time. If changing or setting passwords for an entire group of users noninteractively, use the chpasswd utility. At the standard input, it should submit a list, each line of which will look like name:password.

Deleting users

To delete users, use userdel.

The userdel test1 command will remove test1 user from the system. If the -r parameter is additionally set, the user home directory will also be destroyed. A user cannot be deleted if they are still working in the system at the moment

The systemd initialization system

Starting the operating system

The algorithm for starting the computer is approximately as follows:

1. The BIOS of the computer.
2. The system loader (for example, LILO, GRUB or another). In the loader, you can set the system startup parameters or select the system to run.
3. The Linux kernel is loaded.
4. The first process in the system is started for execution — init.
5. The kernel runs the very first program in the init system. Its task is to launch new processes and restart the completed ones. You can see where init is located in your system process hierarchy by entering the pstree command.

The init configuration determines which initialization system will be used. The initialization system is a

set of scripts that will be executed when the system starts.

The systemd initialization system is the main VEOS initialization system, which has absorbed the advantages of the classic System V init and more modern launchd (OS X), SMF (Solaris) and Upstart (Ubuntu, Fedora), but at the same time free of many of their disadvantages. It was developed to provide a better expression of dependencies between services, which allows to do more work at the same time at the system boot, and reduce the system boot time.

systemd (system daemon) implements a fundamentally new approach to system initialization and operation control. One of the key innovations of this approach is the high degree of services launch parallelization during system initialization, which in the future allows to achieve a much higher speed than the traditional approach with sequential launch of interdependent services. Another important point is the control over mount points (non-vital file systems can be mounted only when they are accessed for the first time, without spending time on it during system initialization) and devices (certain services may be launched or stopped when the specified devices appear or are deleted). To track process groups, the cgroups mechanism is used, which can also be used to limit the system resources consumed by them. The convenience of systemd is especially noticeable on computers for home use — when users turn on and restart the computer daily. Unlike sysvinit, hanging when starting one service will not stop the entire boot process.

Examples of service control commands, a systemd log

The service and chkconfig commands will continue to work in the systemd world with virtually no changes. However, this table shows how to perform the same actions using the built-in systemctl utilities.

Sysvinit commands	**Systemd commands**	**Notes**
service fastdpi start	systemctl start fastdpi.service	Used to start the service (does not restart the constant ones)
service fastdpi stop	systemctl stop fastdpi.service	Used to stop the service (does not restart the constant ones)
service fastdpi restart	systemctl restart fastdpi.service	Used to stop and then start the service
service fastdpi reload	systemctl reload fastdpi.service	If supported, reloads configuration files without interrupting unfinished operations
service fastdpi condrestart	systemctl condrestart fastdpi.service	Restarts the service if it is already running
service fastdpi status	systemctl status fastdpi.service	Notifies whether the service is already running
ls /etc/rc.d/init.d/	systemctl list-unit-files –type=service (preferred)	Used to display a list of services that can be started or stopped.
chkconfig fastdpi on	systemctl enable fastdpi.service	Enables the service during the next reboot, or any other trigger

Sysvinit commands	Systemd commands	Notes
chkconfig fastdpi off	systemctl disable fastdpi.service	Shuts down the service during the next reboot, or any other trigger
chkconfig fastdpi	systemctl is-enabled fastdpi.service	Used to check whether the service is configured to run in the current environment
chkconfig –list	systemctl list-unit-files –type=service(preferred)	Outputs a table of services. It shows at which loading levels they (don't) run
chkconfig fastdpi –list	ls /etc/systemd/system/*.wants/fastdpi.service	Used to display at which levels the service (doesn't) run
chkconfig fastdpi –add	systemctl daemon-reload	Used when you create a new service or modify any configuration

The systemd log

Systemd includes the ability to maintain a system log. To read the log, use the journalctl command. By default, the syslog service no longer needs to be started. You can run journalctl with different keys:

```
> journalctl -b – will show messages only from the current download;
> journalctl -f – will show only the latest messages.
```

You can also view the messages of a certain process:

```
> journalctl _PID=1 – will show the messages of the initial process (init).
```

To get acquainted with other features, read the journalctl manual. To do this, use the man journalctl command.