# Technology of assembly creation of an experimental version OS Linux kernels with quality assurance for applied and subject areas of knowledge

Lavrischeva E.M., Petrenko A.K., Kozin V.P.
Ivannikov Institute of System Programming of the Russian Academy of Sciences

***Abstract.*** *Discribed the technology of assembling an experimental version of the OS Linus kernel with reliable and safe operation in technical devices and application systems is proposed. The development of the core variant was carried out within the framework of RFBR projects No. 352 and No. 206 in the period 2016-2021. The technology of assembly creation of OS Linus variants and ensuring the safety and reliability of the core variant functioning in systems and devices is presented. The config, make operations of the configuration assembly of the IEEE 828 Configuration - 1996, 2012 of OS Linus kernel, means of verification, testing, security, protection and reliability assessment of individual elements and variants of the OS Linus kernel are described.*

***Keywords:*** *assembly, configuration, assembly operations, link, config, make, assembly, OS functional elements, modules, objects, components, libraries, internet.*

**Introduction**

Assembly, since the 1970s of the twentieth century, has been a means of combining individual elementary objects into a common product, system, complex for solving applied and technical problems within the military-industrial complex. The Prometheus, Ruza, Yauza complexes and a number of technical devices and devices for aviation, fleet and space were created (Lipaev V.V. Fragments of the history of the development of domestic programming for specialized computers in the 50-80s.- Synteg.Moscow-2003.-126s.). A domestic assembly method was formed [1-3], which provided linking by the link operation of multilingual modules and equivalent transformation of the data exchanged between them. The conversion primitives (64) were developed by us and included in the system-wide OS software (IBM, VS.MS, Intel, Unix, etc.). Functional software elements for modern fields of knowledge are accumulated in public Internet libraries in the form of reuses, services, artefacts, object, components, etc. [1-5].

Since 1996, ISO/IEC Life Cycle -2001, ISO/IEC 11404 GDM- 1996, IEEE 828 (Configuration) - 1996 and new types of assembly operations (build, assembly, config, make, cmake, waver) have appeared, which allow not only to collect ready-made library resources, but also to replace, delete and configure new versions of software and system-wide OS and Legacy Systems.

Within the framework of the RFBR 352 and RFBR 206 projects, the tasks of modeling high-quality applied, operating systems and assembling an experimental version of the OS Linux kernel for use in complexes, equipment and devices of the fields of knowledge were solved. To solve the problems of assembling the OS Linux variant, an analysis of the properties of the devices used was carried out from the standpoint of ensuring the reliable functioning of OS elements using the build operations (build, assembly, config, make, waver) of OS Linux functions into a new experimental core for the fields of knowledge.

Assembly operations are implemented in modern operating environments: make in BSD and GNU; build, assembly in JavaEE, Grid, Etics, IBMSphere, VS.MS, Intel, Unix, Linux, etc. The results of research and implementation of the tasks of modeling operating systems with quality assurance of OS Linux are presented in the works (see the list of publications on OS Linux.

**1. Technology for creating OS Linux kernel variants**

**Technology processes** includes:

1) *Preprocessing* – preprocessing of the OS kernel function, compilation to obtain the component or system version code and correction of detected errors.

2) *Compilation* of formally described functions to obtain output code with error checking in the syntax of their description.

3) *Search for functional elements of the kernel* and their representation in equivalence classes with verification, testing MC/DC, based on two concepts: solution and condition. The solution is given by a formula consisting of conditions and control transfers to the solution unit. A condition is a logical part of a solution that is related to other conditions. Each condition affects the final value of the decision on the selection of elements and their relationship.

4) *Linking (assembling, make)* individual elements of the Linux OS kernel using make, config statements with checking the interface of the linked functional elements and data exchange between them from external Internet libraries.

5) *Configuration of the structure kernel OS Linux*, using the Kconfig/make operation to obtain a system configuration file with checking for recursive dependencies and non-existent variables in the output code.

6) V used and the Linux OS variant assembled from them with error detection using LDV and CPAchecker and checking the availability of labels and the correctness of accessing functional elements.

7) Evaluation of the elements of the configured version of the Linux OS kernel for reliability, security and the formation of a certificate of the OS kernel variant.

**Functional components of OS Linux**

OS Linux contains more than 10,000 variables and a large number of functional system components that provide processing of various kinds of tasks for the functioning of systems and devices. The necessary OS components are selected from the base OS core, placed in the system characteristics model, including functional, interface elements and data. They are tested for correctness on sets of tests and linking operations with components of other elements of the model. Then the tested elements are assembled into a configuration structure using the standard config build operation to obtain a new OS variant. The standard config operation is performed for manual, automatic graphical assembly of Linux OS variants: oldconfig; defconfig; menuconfig; xconfig; gconfig, etc.

OS kernel code common to all Linux processor architectures. Then the architecture-dependent code is located, forming a BSP (Board Support Package) for the architecture of the platform used in the implementation. The main components of the Linux OS kernel are:

- interfaces of system calls to kernel functions;
- processes, threads, scheduler, linker and management;
- physical, virtual memory and VFS file system with switching facilities;
- IP network Protocol (Internet Protocol), TCP transport protocol (Transmission Control Protocol);
- device drivers and specific hardware devices;
- a special hypervisor with which to build a version of Linux OS for application systems.

**1.2. Means of describing the structures of individual functions of systems and devices**

Hardware Abstraction layer (HAL) to support multiple hardware architectures (processes, semaphores, etc) includes:

API (Application Programming Interface) for describing interfaces;

IPC (Inter process Communication) for communication of SELinux technical and software security tools of the National Security Agency NSA, focused on the security of OS components;

means of checking files /etc /udev /rules.d/70-persistent - net.rules for names of network devices, equipment, drivers and their naming schemes using Udev;

description of symbolic links /dev/cdrom and /dev/dvd for CD-ROM or DVD-ROM devices for USB and FireWire devices;

the directory of the video devices section is sys / class or / sys / block and / sys / class / video4linux / videoX;

specification of interfaces using ifconfig.xyz network scripts etc/sysconfig/. specification of the source files of components in YAP (Fortran, Cobol, Ada, C, C++, Basic, Smalltalk, ...).

**1.3. Standard tools for creating and configuration assembly of variants OS**

POSIX (Portable Operating System Interface) -2012, including a set of standards describing the interfaces between the OS, application programs in the API library in PL (C, C++) and a set of application interfaces in a UNIX environment, OS and Intel OS;

FHS (File system Hierarchy Standard) Version 3.0 standard hierarchical file structure, unifying the location of the files and directories in UNIX OS and BD f file /etc/passwd;

LSB (Linux Standard Base) Version 5.0 (2015) -LSB Core: Bash, BC, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib, SB Runtime Languages, Perl.

**OS Linux packages**

**Bash-3.2, Binutils-2.25, Bison-2.7, Bzip2-1.0.4, Coreutils-6.9, Diffutils-2.8.1, Findutils-4.2.31, Gawk-4.0.1, GCC-4.9,** включая C++ compiler, g++, **Glibc-2.11, Grep-2.5.1a, Gzip-1.3.12, Linux Kernel-3.2, M4-1.4.10, Make-4.0, Patch-2.5.4, Perl-5.8.8, Sed-4.1.5, Tar-1.22, Texinfo-4.7, Xz-5.0.0;** compilation package - Glibc; linker - toolchain, configurable by Glibc.

When the chroot process is executed, the bash hashing function is disabled, and the bin toolchain directory is moved to the end of PATH.

**OS Linux System Tools**

Linux OS (about 6 GB gigabytes) contains a disk partition for storing all source archives and compiling packages. When configuring a hard disk to use Linux OS, the most convenient installation is to boot the host hard disk. If the host already has one IDE disk, which is treated as /dev/hda, the IDE disk is treated as hdb or hdc, depending on the host configuration. Then the disk is formatted and mounted. The only difference is that the target disk used on the host as a secondary disk /dev/hdb or /dev/hdc will be treated as an hda for compiling packages in the disk partition and swap-swap components of OS Linux workstations and servers by accessing large memory and emulating additional memory on the storage devices.

Most embedded storage devices, flash and DOC devices, have limited erase and write cycles. When creating a kernel, application memory is reduced to a minimum set of binary files. The disk is split by a special cfdisk or fdisk utility, on which a new /dev /sda partition will be created for the main Integrated Drive Electronics (IDE) disk. After creating this partition, Linux OS variant file system is formed using the following tools:

- ext2 for small partitions that are not updated often and the disk is divided into additional sections; - ext3 upgrade for ext2, which includes a log to restore the status of the partition after it is disabled;
- ext4 version of the ext file family, which includes several new features - nanosecond timestamps, creation and use of very large files (16 TB) and speed improvements.

When selecting functions from a set of functions and a list of data types stored in the file system and in the library, software for devices and systems of variable configuration in XML is generated. When creating a new Linux OS kernel, USB drivers (ehci_hcd, ohci_hcd and uhci_hcd) are introduced as modules loaded in the correct order, the ehci_hcd module loads ohci_hcd and uhci_hcd with error checking during boot. GRUB and write data to the first physical track of the hard disk with access to the GRUB modules in the boot partition /boot/grub/. The OS partition is about 100 MB in size and contains all the necessary information for the configuration assembly of the new version of the Linux OS kernel.

**The Pakeges of kernel OS Linux**

For the OS kernel to function, the following packages must be installed:

Linux-4.18.5 API Headers, Man-pages-4.16, Glibc-2.28, Zlib-1.2.11, File-5.34, Readline-7.0, M4-1.4.18, Bc-1.07.1, Binutils-2.31.1, GMP-6.1.2, MPFR-4.0.1, MPC-1.1.0, Shadow-4.6, GCC-8.2.0, Bzip2-1.0.6, Pkg-config-0.29.2, Ncurses-6.1, Attr-2.4.48, Acl-2.2.53, Libcap-2.25, Sed-4.5, Psmisc-23.1, Iana-Etc-2.30, Bison-3.0.5, Flex-2.6.4, Grep-3.1, Bash-4.4.18, Libtool-2.4.6, GDBM-1.17, Gperf-3.1, Expat-2.2.6, netutils-1.9.4, Perl-5.28.0, XML::Parser-2.44, Intltool-0.51.0, Autoconf-2.69, Automake-1.16.1, Xz-5.2.4, Kmod-25, Gettext-0.19.8.1, Libelf-0.173, Libffi-3.2.1, OpenSSL-1.1.0i, Python-3.7.0, Ninja-1.8.2, Meson-0.47.1, Procps-ng-3.3.15, E2fsprogs-1.44.3, Coreutils-8.30, Check-0.12.0, Diffutils-3.6, Gawk-4.2.1, Findutils-4.6.0, Groff-1.22.3, GRUB-2.02, Less-530, Gzip-1.9, IPRoute2-4.18.0, Kbd-2.0.4, Libpipeline-1.5.0, Make-4.2.1, Patch-2.7.6, Sysklogd-1.5.1, Sysvinit-2.90, Eudev-3.2.5, Util-linux-2.32.1, Man-DB-2.8.4, Tar-1.30, Texinfo-6.5, Vim-8.1.

In the Linux 5.2 OS kernel, USB drivers (ehci_hcd, ohci_hcd and uhci_hcd) are defined, ehci_hcd, ohci_hcd and uhci_hcd boot modules are introduced to detect erroneous situations and correct them. GRUB writes data to the first physical track of the hard disk. Operating OS Linux software elements get access to GRUB modules in the boot partition /boot/grub/ track /boot/grub/grub.cfg.

**1.4. Configuration of experimental version of the OS Linux kernel**

The main OS Linux section is Binutils. It is installed first. Glibc performs various linker functional tests and determines which software features are enabled or disabled. The failure of the test suite allows you to identify errors when installing some functions of the Linux OS. The Binutils section installs its assembler, linker, Linux API headers and enables the Glibc standard library to interact with the included functional elements of the experimental Linux OS kernel. This option includes a compiler, binary tools, and kernel headers. Glibc uses a compiler related to the host parameter passed to the configure script (i686-lfs-linux-gnu-gcc compiler).

*Configure* is launched by the config /make operation of the glibc-build directory of the build mechanism. During the secondary Binutils installation, the — with-lib-path configuration switch is used to control the search for the ld library. During the secondary installation of GCC, changes are made to launch the dynamic linker from the built-in directory of the host system /lib and then exit the host. When managing the tools directory and files, LFS files /etc/passwd on the host system are added to the new Linux OS.

OS Linux contains several device nodes, console and null devices, the nodes of which are located on the hard disk. They are available to run udevd Linux with init =/bin/bash. Filling the /dev directory with devices by connecting the tmpfs virtual file system to the /dev directory. Devices and appliances are started during Udev boot and /dev and host directory monitoring link /dev are manually set and populated. Bind mount provides mounting of the directory mirror using mount points.

**Means of Creating and Configuration Assembly of OS Linux the experimental kernel**

POSIX (Portable Operating System Interface) -2012, which includes a set of standards describing interfaces between the OS, application programs in the API, a library in the YAP (C, C++, Basic, Ruby...) and a set of applications with interfaces in UNIX OS and Intel OS Linux.

*FHS* (File system Hierarchy Standard) Version 3.0 is a hierarchical file structure standard that unifies the location of files and directories in UNIX OS and the BD file f /etc/passwd.

LSB (Linux Standard Base) Version 5.0 (2015) -LSB Core: Bash, BC, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib, SB Runtime Languages, Perl. Publications on the experimental OS Linux kernel

- Kozin S.V., Mutilin V.S. Static verification core configuration Linux. Proceedings of ISP RAS. -M.: 2018, Volume 29. Issue 4.-pp.217-230: Kozin S.V., Mutilin V.S. Static Verification of Linux Kernel Configurations. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 4, 2017, pp. 217-230.DOI: 10.15514/ISPRAS-2017-29(4)-14.
- Kozin S.V., Configuration assembly of the Linux OS kernel for application systems. Proceedings of the ISP RAS. -M.: 2018, Volume 29. Issue 4.-pp.217-230.-M.: 2018, Volume 30. Issue 6.-pp.161- 170.

**To start the configuration build of the version of the LFS system variant**

you log in to the chroot environment of the directory with access 755. At the same time, two changes are made one to the root directory, and the other to temporary files. The first change ensures guaranteed entry into /root, the second change is made to the /tmp and /var/tmp directories without deleting the files of another user (the so-called "sticky bit"). The directory tree formed using FHS contains directories - /usr/local/games and /usr/share/games, using symbolic links to replace real files and a list of mounted file systems in the /etc/mtab file.

An experimental version of the OS Linux kernel was created using this technology. Version of the kernel includes the following standard elements of the basic Linux kernel:

- data compression functions for arithmetic calculations of rational floating-point numbers;
- functions for reading audio files, compilers in C, C++, Basic, Puby languages, password protection system, interfaces for installing pkg packages, text editor, network operations and information retrieval in the system;
- database library, hash function generator, XML parser, OpenSSL and some other auxiliary functions;
- the make BSD and cmake GNU/VS.Vision build operator for calling executable files from the library for PTS or operation systems;
- the config/make standard IEEE 828 – configuration: 1996. 2007 for calling elements of kernel OS Lunux in enwironment JavaEE.

**Make in system GNU**

```
edit : main.o kbd.o command.o display.o \
insert.o search.o files.o utils.o
```

```
cc -o edit main.o kbd.o command.o display.o \
insert.o search.o files.o utils.o
main.o : main.c defs.h
cc -c main.c
kbd.o : kbd.c defs.h command.h
cc -c kbd.c
command.o : command.c defs.h command.h
cc -c command.c
display.o : display.c defs.h buffer.h
cc -c display.c
insert.o : insert.c defs.h buffer.h
cc -c insert.c
search.o : search.c defs.h buffer.h
cc -c search.c
files.o : files.c defs.h buffer.h command.h
cc -c files.c
utils.o : utils.c defs.h
cc -c utils.c
clean :
rm edit main.o kbd.o command.o display.o \
insert.o search.o files.o utils.o
```

Fig. 1. **Example Makefile in GNU**

**Cmake for assembly variant of kernel OS Linux in JavaEE**

Generation CMakeLists.txt to Makefile в MS.NET/JavaEE includes the following operation:

- add_executable(exec_name source1 source2 ...) # criate file exec_name from files of the source code source1, source2, и т.д.

- add_library(lib_name souт rce1 source2 ...) # crieat library lib_name from files kernel of OS Linux to files source1, source2 и т.д.

- target_include_directories (target PUBLIC dir1, dir2 ...) to directorias dir1, dir2, ... of files under assembly of files library.

- target_link_libraries(target lib1 lib2 ...) # exect cmakes from library lib1, lib2. cmake exect files myExec cmake_minimum_required (VERSION 2.6), MyProject) variant of OS Linux.

-add_executable (myExec source1.cpp source2.cpp)/

Fig. 2. **Implementation cmake in Java EE/ Visual Studio MS**

The operations shown in Fig. 1, 2 provide the configuration assembly of components in the simulated OS Linux in operating environment of wide Internet environments, which are recorded in different NPS (C, C++, Basic, Java, Phyton, Ada, etc.). The elements of the system model were checked for the correctness of their functionality during cmake/config and data assembly operations according to the IEEE 828-1996-2010 (Configuration) standard with version files in the output application system. The final version of the OS Linux kernel is presented in the *Appendix* and in the article*. This variant of kernel OS Linux was verificating, testing and reability.

*Kozin S.V., Configuration assembly of the OS Linux kernel for application systems. Proceedings of the ISP RAS. M.: 2018, Volume 29. Issue 4.-pp.217-230.-M.: 2018. This variant of kernel OS Linux was verification, testing and rehabilite by other specialists (Look APPLICATION).

**2. Ensuring the security and reliability of the OS Linux kernel variant**

To achieve the qualitative indicators of the core elements and the experimental version of the OS Linux kernel, the processes were carried out according to the requirements of the RFBR 206 project.:

- verification and validation (V&V) of individual functional elements of the OS on the life cycle processes (LC) and evaluation of the achievability of individual quality indicators (performance, efficiency, etc.);

- testing of the finished version of OS Linux and collecting data on failures, defects and errors and their corrections;

- evaluation of quality indicators according to ISO/IEC 9126 -1996 reliability standard, which ensures a high level of functioning and a low probability of failures during the execution of programs of the new experimental version of the OS Linux kernel.

The tasks of ensuring the reliability of the OS Linux functional elements are presented in the reports and articles of the RFBR 206 project participants and were used when checking the version of the OS Linux variant:

- Report at the OS DAY-2018 conference. Ekaterina Lavrischeva, ISPRASOPEN-2018.- http://0x1.tv/20180517F Analysis of methods for assessing the reliability of equipment and systems (Ekaterina Lavrishcheva, Nikolay Pakulin). https://videonauka.ru/stati/30-metodika-prepodavaniya-tekhnicheskikh-distsiplin/238-analiz-metodov-otsenki-nadezhnosti-oborudovaniya-i-sistem-i-praktika-primeneniya-etikh-metodov;

- Lavrischeva E.M., Zelenov S.V., Ryzhov A.G. Theory modeling of software and hardware systems with security and reliability in the Internet Web-services environment. Proceedings of the ISP RAS, Volume 30.No. 6.-2018.- pp.156-187.

**2.1. The main aspects of ensuring the reliability of PTS**

The reliability of software in the scientific literature is defined as the probability that the components of the system being created from ready-made resources function flawlessly for a given period of time in a given operating environment/environment. In the quality model, reliability is set on a set of attributes:

q2 = {a21, a22, a23, a24}, which determine the ability of the system to convert raw data into results under conditions depending on the lifetime of the system (wear and aging are not taken into account).

The decrease in the reliability of components is due to design errors. Failures and errors can appear at a given time interval of the component/system functioning.

According to the ISO/IEC 9000(1-4) standard quality reliability attributes include:

Reliability as the property of programs and equipment to function without failures. If the component contains a *defect*, then in the set $D=\{De|e\in L\}$ of all defects, a subset $E\subseteq D$ is allocated for which the results do not correspond to the function Ft specified in the development requirements. The probability p of execution of the component on D is equal to:

p = 1- card {E}/ card {D}.

*Failure* indicates a deviation of the system behavior from the prescribed performance of functions. The occurrence of a *failure* may be the cause of the error (*fault/error*) causing it. If the *error* is made by the developer, then the term mistake is used. When the distinction between fault and failure is not critical, the term defect is used, meaning either fault (cause) or failure (action). The relationship between these concepts is as follows: fault $\rightarrow$ error $\rightarrow$ failure. The cause of failures can be physical, structural and unforeseen situations during operation and interaction.

*The time to failure* determines the average time of occurrence of threats that violate security, and determines a measurable assessment of the damage caused by *threats*, *attacks*, etc.

The *security of operating*, software and technical systems (PTS) does not depend on the physical location of objects in the network, as well as on hardware and operating platforms. Security provides protection when threats appear in the operation of a certain software element or system on the network and in addition, illegal persons or unforeseen situations protect information from interference.

To combat different types of threats, security approaches have developed:

- standardization of cryptography (Criptographic Application Program Interface - CAPI), which includes standard means of controlling keys, information and data access;

- system data protection tools in operating environments and in a set of security support tools in general-purpose systems, which include a security system (SB) focused on countering security threats and ensuring the required level of information protection in a computer network.

**2.2. Tasks of the security system of applied and subject areas**

Security tasks include:

- authorized access to PTS facilities;
- successful reflection of threats (external or internal) arising in the network;
- control of the passage of requests on the Internet;
- evaluation of the effectiveness of the security system, etc.

The *security system model* is characterized by the following properties:

- authorization (AUTH), which consists in typing users or their groups to obtain permission to share individual PTS resources;
- authentication (OUT), establishing the authenticity of the person sending the message to the network to access the required resource;
- password protection (PZ);
- restriction and control of access (CD) to information, programs, servers, devices.

*The means of protection* and control of programs and information include:

- blocking of services (BU) to categories of users with a ban on entering the network;
- role protection based on the establishment of roles (powers) for individual components and/or users;
- audit (AUDIT), which provides control of events occurring in the network;
- confidentiality (CONF) to prohibit access to information of clients with unauthorized permissions;
- integrity (INTEGRITY), which consists in the inadmissibility of modifying information to unauthorized users;
- DOST, provides access to information of users with authority or privileges;
- the observability of the NAB, which consists in the control by the network management bodies (for example, by the SB administrator) of actions related to information security issues in the CS. System-wide anti-threat tools in modern operating Internet environments are divided into two groups of tools: analysis and control of system directories and lists of programs; control of data and messages, ensuring the integrity and correctness of data in the database, database servers, etc.

In the modern Internet, these types of security and correctness of programs and data are managed by the administrator (Security Service Administrator).

**2.3. Standard means of ensuring the protection and security of programs and data**

The means of *protection* on the Internet include:

- X.500 standard for specifying public keys and global directories;
- IDUP-GSS-API (Independent Data Unit Protection Generic Security Application Program Interface) for generating and certifying (Sertification Authenticity) public keys of the mail system in accordance with the X.500 standard;

JAVA - RMI (Remote Methods Invocation) for organizing the invocation of objects, their integration and ensuring the security of passing a request over the network;

PAC (Privilege Attribute Certificate) for identification of privilege attributes and secure access of SQL transactions to network databases, etc.

*Standard security measures include*:

- Bull ISM/Open Master, ISM Access Master server for authorization, auditing and identification of access privileges to the server;
- HP Open View, Presidium /Authorization Server, Praesidium/ Security Service for authorization and protection of information using public keys and smart cards (Philips) in a heterogeneous environment;
- ORACLE Enterprise Manager for setting role protection and database access;
- OSF DCE/Keberos and SENAME/Public key for setting public and public keys;
- IBM Tivoli Global Enterprise Manager for network user identification;
- Secure Dynamic Access Master for centralization of remote user authorization services, etc.

*Mathematical methods of ensuring security* include the Markov method and methods of ensuring functional reliability, presented in reports and articles on project RFPI:

- Report "Methods and means of ensuring the reliability of hardware and software. The practice of applying methods. 5- Scientific and Practical

Conference - OS DAY, Moscow, May 17-18, 2018. http://0x1.tv/20181122AF, http://0x1.tv/20180517F. These methods were worked out by Pakulin N.A. and Ryzhov A.V. in a real-time system and an article on this topic was published on eng. in the Proceedings of the ISP RAS, Volume 30. No. 6. - p. 99-120. DOI: 10.15514 ISPRAS-2018-30(3).

- Lavrischeva E.M. Theory of graph modeling of complex systems from modular elements for applied fields.- Austria-science. Part 1 No.28/2019.- p.12-30. http://austria-science.info .

Since 2012, the main direction of development of reliability tools has been the guarantee [19, 20], which provides fault tolerance, high reliability, guaranteed safety of the operation of systems and OS software without catastrophic consequences; confidentiality, survivability of systems, devices and reliability of calculations. Studies have been conducted on attacks, accidents that occur in the software environment of Internet systems and proposed means to combat them (see the report of S.V. Zelenov at the ISP RAS-2019 conference).

In our country, the work on ensuring protection, safety and quality was carried out within the framework of the military-industrial complex and worked out under the leadership of V.V.Lipaev in the period 1960-2015. A particularly large set of tools has been developed abroad, more than 100 reliability models. Many articles are analyzed by three categories and properties (see articles below*) and have been using practically during verification and testing of experimental versions of OS Linux kernel.

*Avizienis A. Reliable Computing: From Concepts to Applications / A. Avizienis, J.-S. Lapri// IEEE Trans. On Computers. - 1986. - N 74 (5). - p.629 - 638.

- Basic concepts and taxonomy of reliable and secure computing / A. Avizienis, J.-S. Lapri, B. Randell [et al.]// IEEE Trans. on reliable and secure computing. -2004. - Vol. 1, N 1. - p.11 - 33.

- Lipaev V.V. Software quality.-Syntag.-2011. IX International Conference "Quality Strategy in Industry and Education". - Varna, Bulgaria, 2013. - Vol. 1. - 396 p.

- E.M. Lavrishcheva, S.V. Zelenev. A model approach to ensuring the security and reliability of Web services. Proceedings of the ISP RAS 2020, Volume 32 No. 5. pp.151-163.

**Conclusion.** Approaches and methods of configuration of the experimental version of the Linux OS are considered and an experimental version of the Linux OS kernel is configured for a class of subject areas of knowledge (medicine, physics, mathematics, etc.), which is given in the Appendix. This option has been checked for the correctness of the description of the functional elements of the experimental version of the Linux OS kernel, verification, validation and testing. The configured Linux OS kernel has been tested for reliability, security and quality [1-3, 6-19]. It can be offered to customers of subject areas of knowledge for use (Application: OS Linux version).

The resulting version of OS Linux has all the standard functions of the Linux kernel:

the data compression feature, the file system, command line, the functions of the arithmetic of rational numbers (including floating point); to read audio files, compiler with C++, Basic, Ruby, Java, Smalltalk, system password, setup interface pkg packages, the access control list, text editor, the ability to network, search for objects on the Internet; Library of database functions, a hash function generator, an XML parser, OpenSSL and some other auxiliary functions.

The version of OS Linux in the Application can be used for a class of applied systems and subject areas of knowledge (biology, medicine, chemistry, physics, etc.) –Application.

**APPLICATION. Experimental Version of kernel variant OS Linux**

boot/grub/grub.cfg

```
# Begin /boot/grub/grub.cfg
```

```
set default=0
set timeout=5
insmod ext2
set root=(hd0,2)
menuentry "GNU/Linux, Linux 4.18.5-lfs-8.3" {
linux /boot/vmlinuz-4.18.5-lfs-8.3 root=/dev/sda2 ro }
```

etc/modprobe.d/usb.conf

```
# Begin /etc/modprobe.d/usb.conf
install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true
# End /etc/modprobe.d/usb.conf
```

etc/fstab

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab
# file system mount-point type options dump fsck
# order
/dev/<xxx> / <fff> defaults 1 1
```

```
/dev/<yyy> swap swap pri=1 0 0
proc /proc proc nosuid,noexec,nodev 0 0
sysfs /sys sysfs nosuid,noexec,nodev 0 0
devpts /dev/pts devpts gid=5,mode=620 0 0
tmpfs /run tmpfs defaults 0 0
devtmpfs /dev devtmpfs mode=0755,nosuid 0 0
# End /etc/fstab
EOF
```

etc/shells

```
# Begin /etc/shells
/bin/sh
/bin/bash
# End /etc/shells
```

etc/inputrc

```
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>
# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off
# Enable 8bit input
set meta-flag On
set input-meta On
# Turns off 8th bit stripping
set convert-meta Off
# Keep the 8th bit for display
set output-meta On
# none, visible or audible
set bell-style none
# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word
# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert
# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line
# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line
# End /etc/inputrc
```

etc/sysconfig/clock

```
# Begin /etc/sysconfig/clock
UTC=1
# Set this to any options you might need to give to hwclock,
# such as machine hardware clock type for Alphas.
CLOCKPARAMS=
# End /etc/sysconfig/clock
```

etc/inittab

```
# Begin /etc/inittab
id:3:initdefault:
si::sysinit:/etc/rc.d/init.d/rc S
l0:0:wait:/etc/rc.d/init.d/rc 0
```

```
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
su:S016:once:/sbin/sulogin
1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600
# End /etc/inittab
```

etc/hosts

```
# Begin /etc/hosts
127.0.0.1 localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.1> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
# End /etc/hosts
```

*etc/sysconfig*

```
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
```

etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
daemon:x:6:6:Daemon User:/dev/null:/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/var/run/dbus:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
```

etc/group

```
root:x:0:
bin:x:1:daemon
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
```

```
systemd-journal:x:23:
input:x:24:
mail:x:34:
nogroup:x:99:
users:x:999:
```

*The work was supported by RFBR grant* No. 19-01-00206.

**References:**

1.Lavrischeva E.M., Grishchenko V.N. Connection of multilingual modules.- Moscow.- Finance and statistics.- 1982-- 136 p.

2. Lavrischeva E.M., Grishchenko V.N. Assembly programming. Fundamentals of the software products industry.- Nauk. Dumka, 2009.-371P. (www.twirpx.com).

3. Lipaev V.V., Pozin B.A., Shtrik A.A. Technology of assembly programming. Moscow: 1992. -284s.

4. Robert Love. Development of the Linux kernel. Novellyu- /Moscow-St. Petersburg-Kiev, 2006.-446c.

5. Lavrischeva E. Software Engineering. New disciplines and E-learning Them for Development of Application Systems.- Progressive Academic Publishing, *European Journal of Engineering and Technology*, ISSN 2056-5860.- p. 36-63. 2015. http://www.idpublications.org/ejet-vol-3-no-3-2015.

5. Lavrischeva Ekaterina. Ontological Approach to the Formal Specification of the Standard Life Cycle, Science and Information Conference-2015, Jule 28-30, London, UK, www.conference.thesai.org.- p.965-972.

6. Ekaterina M. Lavrischeva. Ontology of Domains. Ontological Description Software Engineering Domain—The Standard Life Cycle, Journal of Software Engineering and Applications, 2015, 8, p.1-15. Published Online July 2015 in SciRes.http://www.scirp.org/journal/jsea.

7. Ekaterina M. Lavrischeva. Assemblling Paradigms of Programming in Software Engineering.- 2016, 9, P.296-317, http://www.scrip.org/journal/jsea, http://dx.do.org/10.4236/jsea.96021.

8. Lavrishcheva E. M. The theory of object-component modeling of software systems. Preprint of the RAS, No. 29. 2016. - 48 p. ISBN 078-5-91474-025-9.13 (in rus).

9. Lavrischeva E.M., Petrenko A.K. Modeling of systems and their families. Proceedings of the ISP RAS, 2016, volume 28. issue 6. - pp. 180-190. DOI:10.15514/ISPRAS-2016-28(6)-4.

10. Kulyamin V.V., Lavrishcheva E.M., Mutilin V.S., Petrenko A.K. Verification and analysis of variable operating systems. Proceedings of the Institute of System Programming of the Russian Academy of Sciences, volume 28, issue 3, 2016, pp. 189-208. DOI:10.15514/ISPRAS-2016-28(3)-12.

11. E. M. Lavrischeva, L. E. Karpov, A. N. Tomilin. Approaches to the presentation of scientific knowledge in Internet science. XIX All-Russian Scientific Conference "Scientific service on the Internet", Novorossiysk, September 18-23, 2017.-pp. 310-326.

12. Lavrischeva E.M., Mutilin V.S., Ryzhov A.G. Designing variability models for software and operating systems. Proceedings of the ISP RAS..Volume 29, Issue 5.-2017.- c.93-110. DOI: 10.15514/ISPRAS- 2017(5).

13. E.M. Lavrischeva1,2[0000-0002-1160-1077], A.K. Petrenko1,4 [0000-0001-7411-3831] and B.A.Pozin3,5 [0000-0002-0012-2230] Technology of Assembly of Intellectual and Information Resources Internet.- CEUR-WS.org/2019.-vol-2543/paper-22.pdf.-27p.

14. Ekaterina Lavrischeva. The theory graph modeling systems from quality modules of the application areas, Journal "Actual Problems of System and Software Engineering, 2019.- http://ceur-ws.org/ceur-author-agreement-ccby.pdf,Paper: 135.-p.235-247.

15. Kozin S.V., Configuration assembly of the Linux OS kernel for application systems. Proceedings of ISP RAS. M.: 2018, Volume 29. Vol.4.-pp. 217-230.-M.: 2018, Volume 29. Vol.5.-pp. 161 - 170.

16. Lavrishcheva E.M., Zelenov S.V., Ryzhov A.G.. Theory modeling of software and hardware systems with security and reliability in the Internet Web services environment. Proceedings of the Institute of System Programming of the Russian Academy of Sciences.2019. Volume 30. Vol.6 pp. 93-111.

17. E.M.Lavrishcheva, I.B.Petrov. Modeling of technical and mathematical problems of applied fields of knowledge on a computer. Proceedings of the ISP RAS 2020, Volume 32 No. 6 pp.167-182.

18. A model approach to ensuring the security and reliability of Web services. Ekaterina Mikhailovna Lavrischeva, Sergey Vadimovich Zelenov. Proceedings of ISP RAS 2020, Volume 32 No. 5 pp.151-163.

19. Lavrishcheva E.M. Theory of graph modeling of complex systems from modular elements for applied fields.- Austria-science Part 1 No. 28, 2019.- p.12-30. http://austria-science.info.

11. E. M. Lavrischeva, L. E. Karpov, A. N. Tomilin. Approaches to the presentation of scientific knowledge in Internet science. Collection of the XIX All-Russian Scientific Conference "Scientific service on the Internet", Novorossiysk, September 18-23, 2017.-pp.310-326.

12. Kozin S.V., Configuration assembly of the Linux OS kernel for application systems. Proceedings of ISP RAS. M.: 2018, Volume 29. Vol.4.-pp. 217-230.-M.: 2018, Volume 30. Vol.6.-pp. 161 - 170.

13. Lavrischeva Ekaterina. Ontological Approach to the Formal Specification of the Standard Life Cycle, Conference "Science and Information Conference-2015, July 28-30, London, UK, www.conference.thesai.org.-p.965-972.

14. Kozin S.V., Mutilin V.S. Static Verification of Linux Kernel Configurations. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 4, 2017, pp. 217-230.DOI: 10.15514/ISPRAS-2017-29(4)-14.

15. Lavrischeva E.M., A.G.Ryzhov. An approach to modeling systems and sites from ready-made resources.- XX All-Russian Conference, September 17-22, 2018, Novorossiysk.- IPM named after M.V.Keldysh.- Presentation of the report and publication in the collection.-pp. 321-345.

16. Lavrischeva E.M. Software Engineering. Paradimes.Technologies. Case. 2016.- www.urait.ru.-280p.

17. Lavrishcheva E.M., Pakulin N.V., Ryzhov A.G., Zelenov S.V. Analysis of methods for assessing the reliability of equipment and systems.- Proceedings of ISP RAS.-M.: 2018, Volume 30. Issue 3.-pp.99-120.

18. E. M. Lavrischeva. The Scientific Basis of System programming. Journal of Software Engineering and Applications, 2018, 11, p..408-434 http://www.scirp.org/journal/jsea ISSN Online: 1945-3124.

19. Lavrischeva E.M., Mutilin V.S., Kozin S.V., Ryzhov A.G. Creation of applied and information systems from ready-made Internet resources. Proceedings of ISP RAS.-M.: 2018, Volume 29.-p. 99-120, DOI: 10.15514 ISPRAS-2018-30(3).

20. E.M. Lavrischeva, A.K. Petrenko and B.A.Pozin. Technology of Assembly of Intellectual and Information Resources Internet.-2020.-CEUR-WS.org/vol-2543/ipaper-22.pdf.-27p.

21. Модельный подход к обеспечению безопасности и надежности Web-сервисов. Екатерина Михайловна ЛАВРИЩЕВА, Сергей Вадимович ЗЕЛЕНОВ. Труды ИСП РАН 2020, Том 32 № 5. С.151-163.

22. E.M.Lavrischeva, I.B.Petrov. Theory of modeling of technical and mathematical problems of subject areas of knowledge. Eurasian Scientific Association. 2021.№ 1- (1). - PP.35-43.