

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 100503 –

Специализация -

Информационная безопасность
автоматизированных систем
Защищенные автоматизированные
системы управления

ДИПЛОМНАЯ РАБОТА

на тему

БИБЛИОТЕКА ФУНКЦИЙ ДЛЯ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ
ИНФОРМАЦИИ НА АВТОМАТИЗИРОВАННЫХ РАБОЧИХ МЕСТАХ
КОМАНДНО-ШТАБНОЙ МАШИНЫ

Студент _____ Викторов Денис Сергеевич

Руководитель _____ Мялкин А.О.

Нормоконтролёр _____ Фатеев А.Г.

Рецензент _____ Секаев Р.А.

Работа допущена к защите (протокол заседания кафедры от _____ №
_____)

Заведующий кафедрой _____ Зефилов С.Л.

Работа защищена с отметкой _____ (протокол заседания ГЭК
от _____ № _____)

Секретарь ГЭК _____ Иванов А.П.

Пенза 2017

СОДЕРЖАНИЕ

Обозначения и сокращения.....	4
Введение.....	5
1 Анализ предметной области	8
1.1 Структура сети и виды передаваемых данных	9
1.2 Общие сведения об операционной системе МСВС 3.0.....	11
1.3 Мандатная модель управления доступом в операционной системе МСВС 3.0.....	12
1.4 Функции создаваемой библиотеки в проектируемой системе защиты информации	14
1.4.1 Общие положения	14
1.4.2 Состав программного обеспечения системы защиты информации.....	14
1.4.3 Принцип работы системы защиты информации	15
2 Анализ угроз ИБ	20
2.1 Идентификация угроз ИБ	20
2.2 Определение актуальности выявленных угроз ИБ	21
3 Выбор мер защиты для противодействия реализации актуальных угроз.....	24
3.1 Меры защиты для противодействия реализации угрозы У1.....	24
3.2 Меры защиты для противодействия реализации угрозы У2.....	24
3.3 Меры защиты для противодействия реализации угрозы У3.....	25
3.4 Меры защиты для противодействия реализации угрозы У4.....	25
4 Разработка приложения	26
4.1 Реализация алгоритма на основе стандарта ГОСТ Р 34.12-201526	
4.2 Описание логической структуры	34
4.2.1 Функции, которые должна обеспечивать разрабатываемая библиотека.....	34
4.2.2 Режимы функционирования программного обеспечения, частью которого является разрабатываемая библиотека	35

5 Проверка работоспособности приложения	53
Заключение.....	62
Список использованных источников.....	63
Приложение А Структура типичной сети, в которой функционирует библиотека.....	64
Приложение Б Структура взаимодействия должностных лиц	66
Приложение В Структура авторизации должностного лица.....	68
Приложение Г Алгоритм обмена ключами.....	70
Приложение Д Алгоритм работы ГОСТ Р 34.12-2015	72
Приложение Е Исходный код разработанного приложения.....	74

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

АРМ – автоматизированное рабочее место;

КШМ – командно-штабная машина;
ОС – операционная система;
ДЛ – должностное лицо;
ОСЗИ – ответственный за систему защиты информации;
СЗИ – система защиты информации);
ТСД – таблица санкционированного доступа;
ЭВМ – электронно-вычислительная машина;
НСД – несанкционированный доступ
ВКР – выпускная квалификационная работа

ВВЕДЕНИЕ

С началом массового внедрения информационных технологий в разные сферы деятельности человека, объем данных, хранящихся в электронном виде, вырос во много раз, а с появлением компьютерных сетей отсутствие физического доступа к компьютеру перестало являться гарантией безопасности информационных ресурсов. Стали развиваться специализированные средства защиты информации, ориентированные, в основном, на решение задачи обеспечения безопасности системы (в редком случае, некоторого ограниченного набора задач). Так, организации, заинтересованные в сохранении в тайне некоторой информации или предотвращении подделок электронных документов, реализовывают определенный набор мер, ограждающий себя от «компьютерных» преступлений. Расширение сфер применения информационных технологий дает возможность возникновения и распространения различных правонарушений, связанных с использованием вычислительной техники. Для противодействия им и снижения ущерба появляется необходимость грамотного выбора мер и средств обеспечения защиты информации от кражи, несанкционированного доступа, порчи, умышленного разрушения, копирования и чтения. Также необходимо знание основных законодательных положений в области защиты информации, технических, организационных, экономических и других мер.

В области информационной безопасности наиболее актуальными являются три группы проблем[1]:

- нарушение конфиденциальности информации;
- нарушение доступности информации;
- нарушение целостности информации.

Приоритетными направлениями разработок и исследований в области информационной безопасности являются[1]:

- разграничение доступа и защита от несанкционированного доступа к информации в информационных системах коллективного пользования;
- идентификация и аутентификация пользователей;

- обеспечение защиты от появления дезинформации в системах передачи данных;
- создание программного обеспечения высокого уровня надежности с использованием международных, национальных и корпоративных стандартов по обеспечению безопасности информации;
- обеспечение защиты данных в телекоммуникационных сетях;
- совершенствование и разработка правовых норм информационной безопасности.

Отдельного внимания заслуживает обеспечение безопасности информации, передаваемой по компьютерным сетям. Существует условно три типа проблем информационной безопасности при работе в компьютерных сетях[2]:

- перехват информации (нарушение конфиденциальности);
- модификация информации (искажение информации или замена ее другой информацией);
- подмена авторства (кража информации и нарушение авторского права).

Развитие информационных технологий оказало серьезное влияние на производственную деятельность оборонных предприятий и Вооруженные Силы РФ.

В настоящее время в процессы производства и управления происходят с использованием информационных технологий. Большинство оборонных предприятий также используют специальные программные средства для решения задач определенного класса. Информационные технологии применяются в коммерческих целях, а также компьютерная техника внедрена в конечную продукцию оборонных предприятий[3].

Таким образом, проблема безопасности информации является актуальной для оборонных предприятий и Вооруженных Сил РФ. С учетом характера информации, имеющей ключевое значение для обеспечения обороноспособности государства, и, зачастую, составляющей государственную

тайну, проблема становится наиболее важной.

В дипломном проекте приводятся общие сведения об операционной системе MSVC 3.0, которая используется в Вооруженных Силах РФ, производится общий анализ угроз, выбор необходимых методов защиты и реализация приложения для обеспечения безопасности информации на автоматизированных рабочих местах командно-штабной машины.

1 Анализ предметной области

1.1 Структура сети и виды передаваемых данных

Узлами рассматриваемой в данной работе сети управления ВС РФ являются объекты, совокупность которых имеет строгую иерархию. Каждый из объектов имеет группу подчиненных объектов и группу руководящих объектов. Обмен информацией возможен только с объектами, входящими в эти группы, т.е. если один объект не входит в группу руководящий или подчиненных объектов второго объекта, то обмен информации между ними невозможен. Физически объектом может являться командный пункт, штаб, командно-штабная машина, воинская часть и др. В данной работе в качестве объектов рассматриваются командно-штабные машины.

Каждый объект сети имеет ограниченный набор абонентов. Совокупность абонентов также имеет строгую иерархию, и обмен информацией возможен только с подчиненными или руководящими абонентами. Физически абонентами являются АРМ должностных лиц, находящиеся на борту командно-штабных машин. Пример структуры взаимодействия должностных лиц представлен на рисунке 1 и в графической части работы в приложении А, рисунок А1 «Структура взаимодействия должностных лиц».

Связь между АРМ должностных лиц в пределах одного объекта может быть организована с использованием локальной вычислительной сети, которая в свою очередь подключается к аппаратуре передачи данных, которая обеспечивает дальнейшую передачу сигнала. Связь между объектами сети может быть реализована средствами проводной, радио-, радиорелейной, спутниковой связи. Логическая схема сети приведена на рисунке 2 и в графической части работы в приложении А, рисунок А2 «Схема типичной сети, в которой функционирует приложение». На АРМ должностных лиц хранится информация, которая содержит данные о местах дислокации, развертывания вооружений, количестве и видах боеприпасов, количестве топлива, продовольственном обеспечении и др.

1.2 Общие сведения об операционной системе MCBC 3.0

На АРМ должностных лиц используется ОС MCBC 3.0. Операционная система Мобильная система Вооруженных Сил (MCBC) – защищенная операционная система общего назначения, предназначенная для построения стационарных защищенных автоматизированных систем. Принята в ВС РФ в 2002 году.

Разработчик ОС MCBC 3.0 – Всероссийский научно-исследовательский институт автоматизации управления в непроизводственной сфере им. В. В. Соломатина (ВНИИНС)

Основные функциональные возможности ОС MCBC 3.0:

- многопользовательская многозадачная сетевая ОС;
- функционирование на аппаратных платформах Intel, SPARC, IBM, System/390 и MIPS;
- поддержка многопроцессорных конфигураций (SMP);
- встроенные средства защиты информации от несанкционированного доступа;
- встроенные средства мандатного управления доступом, списки контроля доступа, ролевая модель;
- поддержка широкого ряда современных встроенных и внешних устройств;
- широкий спектр поддерживаемых сетевых сервисов.

ОС MCBC 3.0 сертифицирована в системе сертификации средств защиты информации по требованиям безопасности информации Министерства обороны РФ по следующим критериям:

- по второму классу защищенности информации от НСД согласно руководящему документу «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации»;

- по первому уровню классификации контроля отсутствия недеklarированных возможностей согласно руководящему документу «Защита от несанкционированного доступа к информации Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недеklarированных возможностей».

1.3 Мандатная модель управления доступом в операционной системе MSVC 3.0

В основе мандатного управления доступом в ОС MSVC лежит мандатная модель безопасности. В отличие от дискреционного управления, в котором пользователям непосредственно прописываются права на чтение, запись и выполнение, в мандатной модели управление доступом происходит неявным образом. Всем пользователям (субъектам) и файлам (объектам) назначаются уровни доступа. Пример иерархии уровней доступа: не секретно -> секретно -> совершенно секретно -> особой важности. Уровни доступа упорядочиваются по доминированию одного уровня над другим. Затем доступ к защищённым файлам осуществляется по двум простым правилам:

- Пользователь имеет право читать только те документы, уровень безопасности которых не превышает его собственный уровень безопасности. Это правило обеспечивает защиту информации, обрабатываемой более высокоуровневыми пользователями, от доступа со стороны низкоуровневых пользователей.

- Пользователь имеет право заносить информацию только в те документы, уровень безопасности которых не ниже его собственного уровня безопасности. Это правило предотвращает нарушение режима доступа со стороны высокоуровневых участников процесса обработки информации к низкоуровневым пользователям. Функционирование каждого из правил

проиллюстрировано на рисунке 1.2. и в графической части работы на плакате «Отношения между субъектами в трехуровневой мандатной модели».

На рисунке изображены отношения пользователя с уровнем «секретно» с субъектами в трехуровневой мандатной модели.

Таким образом, назначив всем объектам и субъектам уровни доступа, можно определить все допустимые взаимодействия внутри системы. Из выше описанных правил видно, что мандатное управление доступом не ограничивает взаимодействие объектов и субъектов, находящихся на одном уровне доступа. Поэтому, для эффективного управления доступом, мандатная модель применяется совместно с дискреционной, которая описывает взаимодействие объектов и субъектов, находящихся на одном уровне[4].

Рисунок 3 – Отношения между субъектами в трехуровневой мандатной модели

1.4 Функции создаваемой библиотеки в проектируемой системе защиты информации

1.4.1 Общие положения

СЗИ представляет собой комплекс программных средств, именуемый «Система защиты от несанкционированного доступа абонентов к вычислительным ресурсам ЭВМ, информации и каналам связи», в котором программные средства взаимодействуют по определенным правилам с целью обеспечения безопасности информации.

1.4.2 Состав программного обеспечения системы защиты информации

В состав программного обеспечения системы защиты информации входят следующие модули:

- программное обеспечение рабочего места администратора обеспечения безопасности информации;
- агент системы защиты информации;
- сервис системы защиты информации;

- сервер блока интегрирующего задатчика;
- библиотека доступа к профилям системы защиты информации;
- библиотека операций системы защиты информации;
- программное обеспечение тестирования функций системы защиты информации;
- программное обеспечение восстановления функций системы защиты информации;
- программное обеспечение генерации сигнала «работа под контролем»;
- программное обеспечение контроля целостности ресурсов файловой системы;
- программное обеспечение экстренного стирания информации;
- программное обеспечение протоколирования запуска программ и задач;
- программное обеспечение формирования системного меню пользователя;
- программное обеспечение синхронизации таблиц санкционированного доступа;
- антивирусные средства;
- программное обеспечение блокировки консоли;
- программное обеспечение блокировки и разблокировки рабочих мест;
- программное обеспечение управления пользователями;
- средство регистрации и маркировки печатных документов;
- средство управления резервным копированием.

1.4.3 Принцип работы системы защиты информации

На рисунке 4 приведена структурная схема взаимодействия компонентов системы защиты информации. Зеленым цветом выделены компоненты, реализуемые ресурсами проектируемой в данной работе библиотеки.

Комплекс программ «Система защиты от несанкционированного доступа абонентов к вычислительным ресурсам ЭВМ, информации и каналам связи» предназначен для решения следующих задач:

- управления доступом;
- регистрации и учета;
- контроля целостности информации;
- обеспечения реакции системы на события блока интегрирующего задатчика.

Системообразующими компонентами системы защиты информации являются Агент системы защиты информации, сервер блока интегрирующего задатчика и программного обеспечения рабочего места администратора обеспечения безопасности информации. Администратор безопасности посредством программного обеспечения рабочего места администратора обеспечения безопасности информации формирует политику безопасности домена. Программное обеспечение рабочего места администратора обеспечения безопасности информации позволяет управлять списком подконтрольных должностных лиц.

Рисунок 4 – Структурная схема взаимодействия компонентов СЗИ

Администратор безопасности посредством программного обеспечения рабочего места администратора обеспечения безопасности информации формирует политику безопасности домена. Программное обеспечения рабочего места администратора обеспечения безопасности информации позволяет управлять списком подконтрольных должностных лиц. Программное обеспечения рабочего места администратора обеспечения безопасности информации позволяет управлять агентами системы защиты информации, формируя, таким образом, политику системы защиты информации. Агент системы защиты информации осуществляет реализацию политики безопасности домена на рабочих местах должностных лиц. Агенты системы

защиты информации обрабатывают управляющие воздействия администратора обеспечения безопасности информации.

Взаимодействие Агента системы защиты информации с программным обеспечением рабочего места администратора обеспечения безопасности информации осуществляется посредством обмена формализованными сообщениями подсистемы блока интегрирующего задатчика через функциональную подсистему коммуникаций, доступ к которой предоставляется транспортным модулем.

Сервер блока интегрирующего задатчика предназначен для приема и обработки запросов клиентов безопасности, поступающих в виде сообщений о событиях блока интегрирующего задатчика.

Сервис системы защиты информации предназначен для предоставления доступа к операциям, требующим привилегий суперпользователя.

Загружаемые модули системы защиты информации выполняют решение локальных задач системы защиты информации.

Библиотека операций системы защиты информации предназначена для доступа клиентского программного обеспечения к системным операциям в операционной системе, критичным с точки зрения безопасности.

Библиотека доступа к профилям пользователей предназначена для доступа клиентского программного обеспечения к профилям системы защиты информации.

Программное обеспечение контроля целостности ресурсов файловой системы позволяет контролировать целостность файловой системы в операционной системе.

Программное обеспечение управления пользователями предназначено для управления пользователями системы, их правами и полномочиями.

Программное обеспечение блокировки консоли используется в сигналах «свой-чужой» для должностного лица, генерируемых рабочим местом администратора обеспечения безопасности информации для проверки

легитимности должностного лица, работающего за подконтрольным рабочим местом.

Программное обеспечение экстренного стирания информации предназначено для экстренного стирания критичных с точки зрения безопасности информации данных на подконтрольном рабочем месте.

Программное обеспечение блокировки (разблокировки) рабочего места предназначено для блокировки или разблокировки подконтрольного рабочего места командами с рабочего места администратора обеспечения безопасности информации при реакции на события безопасности на подконтрольном рабочем месте.

Программное обеспечение протоколирования запуска программ предназначено для протоколирования запуска всего программного обеспечения системы защиты информации, а также ведения журнала выполнения этих программ.

Программное обеспечение формирования системного меню пользователя предназначено для разграничения доступа оператора к ресурсам (в частности к доступному программному обеспечению) посредством динамически формируемого меню оконного менеджера elk.

Программное обеспечение генерации сигнала «работа под контролем» предназначено для генерации сигнала «работа под контролем».

Программное обеспечение тестирования функций системы защиты информации предназначено для реализации программы тестирования функций системы защиты информации для получения уверенности в их правильной работе.

Программное обеспечение восстановления функций системы защиты информации предназначено для восстановления СЗИ после различных сбоев.

Программное обеспечение синхронизации таблиц санкционированного доступа предназначено для поддержки актуального состояния таблиц санкционированного доступа, хранящихся в центральной и локальных базах рабочих мест объекта управления[5].

Антивирусные средства защиты информации предназначены для предотвращения заражения системы компьютерными вирусами.

Средства регистрации и маркировки печатных документов предназначены для регистрации и маркировки печатных документов.

Средства управления резервным копированием предназначены для организации процесса резервного копирования информации.

2. Анализ угроз ИБ

2.1 Идентификация угроз ИБ

Была проведена идентификация угроз информационной безопасности. В качестве источника угроз рассматривались подразделения армии вероятного противника и должностные лица ВС РФ.

Библиотека функций устанавливается на АРМ должностных лиц внутри командно-штабных машин.

На основании приведенной, на рисунке 2 схемы типичной сети, в которой функционирует приложение, можно выделить следующие угрозы ИБ:

- доступ к файлу с пользователями и паролями;
- выдача себя за зарегистрированного пользователя;
- удаление или искажение ПО или файлов с важной информацией, в том числе системных;

- доступ к информации пользователем, не имеющим достаточный уровень доступа;
- физическое повреждение или уничтожение АРМ абонента.

Реализация угрозы «Доступ к файлу с пользователями и паролями» приведет к нарушению конфиденциальности паролей пользователей и даст вероятному противнику возможность войти в систему, используя идентификаторы любого пользователя, в том числе пользователя, имеющего доступ к секретным и совершенно секретным сведениям. Это может повлечь за собой компрометацию данных, в том числе секретных сведений, что повлечет за собой снижение эффективности и боеспособности подразделений ВС РФ. Также злоумышленник будет наделен полномочиями должностного лица, под чьим идентификатором он вошёл в систему, может получить возможность отдавать приказы и распоряжения, что может повлечь за собой несогласованность в действиях подразделений ВС РФ и снижение боеспособности.

Реализация угрозы «Выдача себя за зарегистрированного пользователя» наделяет злоумышленника полномочиями должностного лица, за которого он себя выдает, может получить возможность отдавать приказы и распоряжения, что может повлечь за собой несогласованность в действиях подразделений ВС РФ и снижение боеспособности. Это также может повлечь за собой компрометацию данных, в том числе секретных сведений, что повлечет за собой снижение эффективности и боеспособности подразделений ВС РФ.

Реализация угрозы «Удаление или искажение ПО или файлов с важной информацией, в том числе системных» может повлечь за собой полное или частичное нарушение работоспособности программного обеспечения АРМ, что также повлечет за собой значительное снижение эффективности и боеспособности подразделений ВС РФ.

Реализация угрозы «Доступ к информации пользователем, не имеющим достаточный уровень доступа» может повлечь за собой компрометацию данных, в том числе секретных сведений, что повлечет за собой снижение

эффективности и боеспособности подразделений ВС РФ. Более того, реализация данной угрозы может повлечь за собой ущерб, связанный с разглашением конфиденциальной информации представителю иностранного государства, что может повлечь за собой необходимость проведения дорогостоящих мероприятий по ликвидации последствий разглашения такой информации.

Реализация угрозы «Физическое повреждение или уничтожение АРМ абонента» вызовет нарушение доступности информации и приведет к замедлению или остановке деятельности пострадавшего объекта. Также вследствие реализации данной угрозы ВС РФ могут понести серьезный материальный ущерб.

2.2 Определение актуальности выявленных угроз ИБ

Угроза «Доступ к файлу с пользователями и паролями» представляется актуальной, так как нельзя исключать возможность получения доступа к ресурсам ЛВС авторизованным или неавторизованным человеком неавторизованным способом.

Угроза «Выдача себя за зарегистрированного пользователя» представляется актуальной, так как методы перехвата открытого трафика ЛВС, включая пароли, широко доступны сегодня. Также возможен перебор паролей.

Угроза «Удаление или искажение ПО или файлов с важной информацией, в том числе системных» представляется актуальной. Она может быть реализована только должностными лицами ВС РФ ввиду их возможной некомпетентности при работе на АРМ, т.к. АРМ абонентов находятся под постоянной охраной на территориях воинских частей и местах базирования подразделений ВС РФ.

Угроза «Доступ к информации пользователем, не имеющим достаточный уровень доступа» актуальна, так как нельзя исключать возможность получения доступа к ресурсам ЛВС авторизованным или неавторизованным человеком неавторизованным способом.

Угроза «физическое повреждение или уничтожение АРМ абонента» не является актуальной, т.к. средства передачи данных и АРМ абонентов находятся под постоянной охраной на территориях воинских частей и местах базирования подразделений ВС РФ.

Актуальность идентифицированных угроз представлена в таблице 1.

Таблица 1 – Актуальность идентифицированных угроз

Идентификатор угрозы	Описание угрозы	Актуальность
У1	Доступ к файлу с пользователями и паролями	Актуальна
У2	Выдача себя за зарегистрированного пользователя	Актуальна
У3	Удаление или искажение ПО или файлов с важной информацией, в том числе системных	Актуальна
У4	Доступ к информации пользователем, не имеющим достаточный уровень доступа	Актуальна
У5	Физическое повреждение или уничтожение АРМ абонента	Неактуальна

3 Выбор мер защиты для противодействия реализации актуальных угроз

3.1 Меры защиты для противодействия реализации угрозы У1

Для противодействия реализации угрозы «Доступ к файлу с пользователями и паролями» будет целесообразно применить методы криптографии. Такой подход позволяет сохранить конфиденциальность информации, содержащий имена пользователей и пароли, так как они хранятся в зашифрованном виде. Применение методов криптографии, а именно шифрование данных, не требует применение какого-либо специального оборудования и существенных затрат на разработку. Шифрование данных, содержащих имена пользователей и пароли позволит использовать всю имеющуюся инфраструктуру средств связи и АПД. Для реализации этого подхода необходимо каждое АРМ абонента снабдить ЭВМ с установленным программным обеспечением, реализующим криптографическое преобразование. В настоящее время все АРМ снабжены ЭВМ, поэтому применение шифрования данных представляется предпочтительным решением в финансовом смысле.

Техническое задание в качестве алгоритма криптографического преобразования и правил шифрования определяет действующий стандарт РФ в области шифрования ГОСТ Р 34.12-2015 Информационная технология. Криптографическая защита информации. Блочные шифры[4], т.к. этот стандарт обязателен для организаций, применяющих криптографическую защиту данных, хранимых и передаваемых в сетях ЭВМ, в отдельных вычислительных комплексах или в ЭВМ.

3.2 Меры защиты для противодействия реализации угрозы У2

Для реализации угрозы «Выдача себя за зарегистрированного пользователя» противнику необходимо осуществить неавторизованный доступ в ЛВС. С целью защиты от данной угрозы разрабатываемое приложение должно обеспечивать идентификацию и аутентификацию субъектов доступа путем использования системы паролей, подписей и профилей безопасности.

3.3 Меры защиты для противодействия реализации угрозы У3

С целью снижения ущерба, который может возникнуть вследствие реализации данной угрозы необходимо производить контроль целостности программных средств и обрабатываемой информации. Данная мера не осуществляет защиту от угрозы «Удаление или искажение ПО или файлов с важной информацией, в том числе системных», а лишь снижает ущерб от ее реализации. Решение о достаточности этой меры принято исходя из того, что реализовать её могут лишь должностные лица ВС РФ, и последствия от её реализации незначительны. Необходимо производить контроль целостности при каждой загрузке программного обеспечения, также необходимо иметь возможность производить контроль целостности по запросу ответственного за средства защиты информации и по запросу должностного лица. Контроль целостности целесообразно реализовать программными модулями из состава программного обеспечения.

3.4 Меры защиты для противодействия реализации угрозы У4

Доступ к информации пользователем, не имеющим достаточный уровень доступа, может происходить просто потому, что права доступа пользователей к ресурсу не назначены должным образом. Однако доступ может также происходить потому, что механизм управления доступом или механизм назначения привилегий обладают недостаточной степенью детализации. Мера защиты от данной угрозы – мандатная модель безопасности ОС МСВС 3.0, которая обеспечивает информации, обрабатываемой более высокоуровневыми пользователями, от доступа со стороны низкоуровневых пользователей.

4 Разработка приложения

4.1 Реализация алгоритма на основе стандарта ГОСТ Р 34.12-2015

Реализация алгоритма на основе стандарта ГОСТ Р 34.12-2015[4] определена как защитная мера от реализации угрозы «Доступ к файлу с пользователями и паролями». Алгоритм на основе стандарта ГОСТ Р 34.12-2015[4] предполагает использование циклически повторяемого раундового шифрования, где каждый раунд состоит из трех операций: преобразование в подстановочном блоке, линейное преобразование и сложение с соответствующим раундовым ключом[5].

На рисунке 4 представлена операция преобразования в подстановочном блоке.

Рисунок 4 – Преобразование в подстановочном блоке

На рисунке 5 представлена операция линейного преобразования.

Рисунок 5 – Линейное преобразование

Генерация раундовых ключей начинается с разбиения мастер-ключа пополам, в результате чего получается первая пара раундовых подключей. Для генерации последующей пары раундовых подключей применяется 8 итераций сети Фейстеля, в каждой итерации путем применения линейного преобразования алгоритма к значению номера итерации. Процедура представлена на рисунке 6.

Рисунок 6 – Процедура выработки раундовых ключей

На рисунке 7 представлена схема шифрования и расшифрования одного блока данных. Расшифрование реализуется обращением базовых преобразований и применением их в обратном порядке.

Для реализации шифрования и расшифрования было решено использовать режим ECB (англ. *Electronic Codebook*) – один из вариантов

использования симметричного блочного шифра, при котором каждый блок открытого текста заменяется блоком шифротекста[4].

Шифрование может быть описано следующим образом: $C_i = E_k(P_i)$, где i – номера блоков, C_i и P_i – блоки зашифрованного и открытого текстов соответственно, а E_k – функция блочного шифрования.

Рисунок 7 – Шифрование/расшифрование одного блока данных
Расшифровка аналогична. Схема шифрования представлена на рисунке 8.

Рисунок 8 – Шифрование в режиме ECB

Необходимые вспомогательные данные представлены ниже.

Нелинейное биективное преобразование:

```
static const unsigned char kPi[256] = {  
    252, 238, 221, 17, 207, 110, 49, 22, 251, 196, 250, 218, 35, 197, 4, 77,  
    233, 119, 240, 219, 147, 46, 153, 186, 23, 54, 241, 187, 20, 205, 95,  
193,  
    249, 24, 101, 90, 226, 92, 239, 33, 129, 28, 60, 66, 139, 1, 142, 79, 5,  
    132, 2, 174, 227, 106, 143, 160, 6, 11, 237, 152, 127, 212, 211, 31,  
235,  
    52, 44, 81, 234, 200, 72, 171, 242, 42, 104, 162, 253, 58, 206, 204,  
181,  
    112, 14, 86, 8, 12, 118, 18, 191, 114, 19, 71, 156, 183, 93, 135, 21,  
161,  
    150, 41, 16, 123, 154, 199, 243, 145, 120, 111, 157, 158, 178, 177, 50,  
117,  
    25, 61, 255, 53, 138, 126, 109, 84, 198, 128, 195, 189, 13, 87, 223,  
245,  
    36, 169, 62, 168, 67, 201, 215, 121, 214, 246, 124, 34, 185, 3, 224, 15,  
    236, 222, 122, 148, 176, 188, 220, 232, 40, 80, 78, 51, 10, 74, 167,  
151,  
    96, 115, 30, 0, 98, 68, 26, 184, 56, 130, 100, 159, 38, 65, 173, 69, 70,  
    146, 39, 94, 85, 47, 140, 163, 165, 125, 105, 213, 149, 59, 7, 88, 179,  
64,  
    134, 172, 29, 247, 48, 55, 107, 228, 136, 217, 231, 137, 225, 27, 131,  
73,  
    76, 63, 248, 254, 141, 83, 170, 144, 202, 216, 133, 97, 32, 113, 103,  
164,
```

```

45, 43, 9, 91, 203, 155, 37, 208, 190, 229, 108, 82, 89, 166, 116, 210,
230,
244, 180, 192, 209, 102, 175, 194, 57, 75, 99, 182};

```

Обратное нелинейное биективное преобразование множества двоичных векторов:

```

static const unsigned char kReversePi[256] = {
    0xa5, 0x2d, 0x32, 0x8f, 0x0e, 0x30, 0x38, 0xc0, 0x54, 0xe6, 0x9e, 0x39,
    0x55, 0x7e, 0x52, 0x91, 0x64, 0x03, 0x57, 0x5a, 0x1c, 0x60, 0x07, 0x18,
    0x21, 0x72, 0xa8, 0xd1, 0x29, 0xc6, 0xa4, 0x3f, 0xe0, 0x27, 0x8d, 0x0c,
    0x82, 0xea, 0xae, 0xb4, 0x9a, 0x63, 0x49, 0xe5, 0x42, 0xe4, 0x15, 0xb7,
    0xc8, 0x06, 0x70, 0x9d, 0x41, 0x75, 0x19, 0xc9, 0xaa, 0xfc, 0x4d, 0xbf,
    0x2a, 0x73, 0x84, 0xd5, 0xc3, 0xaf, 0x2b, 0x86, 0xa7, 0xb1, 0xb2, 0x5b,
    0x46, 0xd3, 0x9f, 0xfd, 0xd4, 0x0f, 0x9c, 0x2f, 0x9b, 0x43, 0xef, 0xd9,
    0x79, 0xb6, 0x53, 0x7f, 0xc1, 0xf0, 0x23, 0xe7, 0x25, 0x5e, 0xb5, 0x1e,
    0xa2, 0xdf, 0xa6, 0xfe, 0xac, 0x22, 0xf9, 0xe2, 0x4a, 0xbc, 0x35, 0xca,
    0xee, 0x78, 0x05, 0x6b, 0x51, 0xe1, 0x59, 0xa3, 0xf2, 0x71, 0x56, 0x11,
    0x6a, 0x89, 0x94, 0x65, 0x8c, 0xbb, 0x77, 0x3c, 0x7b, 0x28, 0xab, 0xd2,
    0x31, 0xde, 0xc4, 0x5f, 0xcc, 0xcf, 0x76, 0x2c, 0xb8, 0xd8, 0x2e, 0x36,
    0xdb, 0x69, 0xb3, 0x14, 0x95, 0xbe, 0x62, 0xa1, 0x3b, 0x16, 0x66, 0xe9,
    0x5c, 0x6c, 0x6d, 0xad, 0x37, 0x61, 0x4b, 0xb9, 0xe3, 0xba, 0xf1, 0xa0,
    0x85, 0x83, 0xda, 0x47, 0xc5, 0xb0, 0x33, 0xfa, 0x96, 0x6f, 0x6e, 0xc2,
    0xf6, 0x50, 0xff, 0x5d, 0xa9, 0x8e, 0x17, 0x1b, 0x97, 0x7d, 0xec, 0x58,
    0xf7, 0x1f, 0xfb, 0x7c, 0x09, 0x0d, 0x7a, 0x67, 0x45, 0x87, 0xdc, 0xe8,
    0x4f, 0x1d, 0x4e, 0x04, 0xeb, 0xf8, 0xf3, 0x3e, 0x3d, 0xbd, 0x8a, 0x88,
    0xdd, 0xcd, 0x0b, 0x13, 0x98, 0x02, 0x93, 0x80, 0x90, 0xd0, 0x24, 0x34,
    0xcb, 0xed, 0xf4, 0xce, 0x99, 0x10, 0x44, 0x40, 0x92, 0x3a, 0x01, 0x26,
    0x12, 0x1a, 0x48, 0x68, 0xf5, 0x81, 0x8b, 0xc7, 0xd6, 0x20, 0x0a, 0x08,
    0x00, 0x4c, 0xd7, 0x74};

```

Коэффициенты умножения в преобразовании L:

```

static const unsigned char kB[16] = {
    148, 32, 133, 16, 194, 192, 1, 251, 1, 192, 194, 16, 133, 32, 148, 1};

```

Далее поочередно представлены функции на языке C++, необходимые для реализации алгоритма.

Преобразование X:

```

int funcX(unsigned char* a, unsigned char* b, unsigned char* outdata)
{
    for(int i = 0; i < 16; ++i)
    {
        outdata[i] = a[i] ^ b[i];
    }
}

```

```

    return -1;
}

```

Преобразование S:

```

void funcS(unsigned char* indata, unsigned char* outdata){
    for(int i = 0; i < 16; ++i)
    {
        outdata[i] = kPi[indata[i]];
    }
}

```

Преобразование L:

```

int funcL(unsigned char* indata, unsigned char* outdata)
{
    unsigned char tmp[16];
    int i = 0;
    memcpy(tmp, indata, 16);
    for(i = 0; i < 16; ++i)
    {
        funcR(tmp, outdata);
        memcpy(tmp, outdata, 16);
    }
    return 0;
}

```

Преобразование R:

```

int funcR(unsigned char * indata , unsigned char *outdata ){
    unsigned long sum=0;
    for(int i = 0; i < 16; ++i)
    {
        sum ^= multTable[indata[i]*256 + kB[i]];
    }
    outdata[0] = sum;
    memcpy(outdata+1, indata, 15);
    return -1;
}

```

Обратное преобразование S:

```

int funcReverseS(unsigned char* indata, unsigned char* outdata)
{
    unsigned int i;
    for(i = 0; i < 16; ++i)
    {
        outdata[i] = kReversePi[indata[i]];
    }
}

```

```

    return 0;
}

```

Обратное преобразование L:

```

int funcReverseL(unsigned char* indata, unsigned char* outdata)
{
    unsigned char tmp[16];
    unsigned int i;
    memcpy(tmp, indata, 16);
    for(i = 0; i < 16; ++i)
    {
        funcReverseR(tmp, outdata);
        memcpy(tmp, outdata, 16);
    }
    return 0;
}

```

Обратное преобразование R:

```

int funcReverseR(unsigned char* indata, unsigned char* outdata)
{
    unsigned char tmp[16] = {0};
    unsigned char sum = 0;
    unsigned int i;
    memcpy(tmp, indata+1, 15);
    tmp[15] = indata[0];
    for(i = 0; i < 16; ++i)
    {
        sum ^= multTable[tmp[i]*256 + kB[i]];
    }
    memcpy(outdata, tmp, 15);
    outdata[15] = sum;
    return 0;
}

```

Функция выработки итерационных ключей:

```

int funcF(unsigned char* inputKey, unsigned char* inputKeySecond, unsigned
char* iterationConst, unsigned char* outputKey, unsigned char* outputKeySecond)
{
    unsigned char temp1[16] = {0};
    unsigned char temp2[16] = {0};
    funcLSX(inputKey, iterationConst, temp1);
    funcX(temp1, inputKeySecond, temp2);

    memcpy(outputKeySecond, inputKey, 16);
}

```

```

        memcpy(outputKey, temp2, 16);
        return 0;
    }

```

Функция выработки итерационных констант:

```

int funcC(unsigned char number, unsigned char* output)
{
    unsigned char tempI[16] = {0};
    tempI[15] = number;
    funcL(tempI, output);
    return 0;
}

```

Процедура получения раундовых ключей:

```

int ExpandKey(unsigned char* masterKey, unsigned char mass[8][16] )
{
    unsigned char C[16] = {0};
    unsigned char temp1[16] = {0};
    unsigned char temp2[16] = {0};
    unsigned char j, i;
    unsigned char keys[16];
    int g=0;
    memcpy(keys, masterKey, 16);
    memcpy(keys + 16, masterKey + 16, 16);
    memcpy(temp1, keys, 16);
    memcpy(temp1+16, keys+16, 16);
    for(j = 0; j < 4; ++j)
    {
        for( i = 1; i < 8; ++i )
        {
            funcC(j*8+i, C);
            funcF(temp1, temp2, C, temp1, temp2);
        }
        funcC(j*8+8, C);
        funcF(temp1, temp2, C, temp1, temp2); //два следующих ключа!
        memcpy(keys , temp1, 16);
        memcpy(keys + 16, temp2, 16);

        memcpy(mass[g], temp1, 16);
        g++;
        memcpy(mass[g], temp2, 16);
        g++;
    }
    return 0;
}

```

}

4.2 Описание логической структуры

4.2.1 Функции, которые должна обеспечивать разрабатываемая библиотека

Разрабатываемая библиотека предназначена для обеспечения безопасности информации, хранимой и обрабатываемой на автоматизированных рабочих местах, и должна обеспечивать:

- а) идентификацию и аутентификацию субъектов доступа (путем использования системы паролей, подписей и профилей безопасности);
- б) контроль целостности программных средств и обрабатываемой информации (при загрузке программного обеспечения, по запросу ответственного за систему защиты информации, по запросу должностного лица);
- в) управление пользователями (контроль доступа абонентов системы к вычислительным ресурсам, информации, каналам связи);
- г) экстренное стирание информации (удаление файлов и каталогов по маске, удаление файлов с затиранием);
- д) блокировку автоматизированного рабочего места (при попытках несанкционированного доступа, дистанционная блокировка автоматизированного рабочего места);
- е) протоколирование событий безопасности (регистрация входа-выхода субъектов доступа в (из) систему (системы), запуска (завершения) программ, доступа субъектов к защищаемым файлам, доступа субъектов к внешним устройствам ЭВМ);
- ж) просмотр событий безопасности ответственным за систему защиты информации;
- и) тестирование функций системы защиты информации;
- к) управление резервным копированием файлов и директорий (создание резервных копий по команде должностного лица, восстановление информации из резервных копий);

4.2.2 Режимы функционирования программного обеспечения, частью которого является разрабатываемая библиотека

Программное обеспечение, частью которого является разрабатываемая библиотека функций, функционирует в трёх режимах:

- а) инсталляция;
- б) первоначальная настройка;
- в) штатная работа.

4.2.2.1 Инсталляция программного обеспечения, частью которого является разрабатываемая библиотека.

На этапе инсталляции программного обеспечения, частью которого является разрабатываемая библиотека, производится установка конфигурационных файлов, необходимых для функционирования ОБЗИ:

а) **recovery.conf**, **recovery.tgz** – реализация функции восстановления работоспособности системы защиты информации после сбоев;

б) **backup.conf** - управление резервным копированием и восстановлением файлов и директорий;

в) **erase.conf** - экстренное стирание информации;

г) **Integrity.conf** - контроль целостности программных средств и обрабатываемой информации.

4.2.2.2 Первоначальная настройка программного обеспечения, частью которого является разрабатываемая библиотека.

Перед началом работы должностное лицо, ответственное за систему защиты информации должно сформировать таблицу санкционированного доступа в соответствии с оргштатной структурой. Для этого необходимо войти в ОС МСВС с учетной записью ответственного за систему защиты информации, запустить программное обеспечение, частью которого является разрабатываемая библиотека и зарегистрироваться в системе с технологической подписью «123456» (которая может быть изменена впоследствии). При успешной регистрации будет проведена проверка целостности программного обеспечения и откроется диалоговое окно «Управление пользователями». В

этом режиме ответственного за систему защиты информации предоставляются следующие возможности:

- а) вводить, корректировать и удалять информацию из таблицы санкционированного доступа;
- б) сохранять таблицу санкционированного доступа;
- в) сохранять таблицу санкционированного доступа под новым именем (резервная таблица санкционированного доступа);
- г) загружать таблицу санкционированного доступа;
- д) рассылать таблицу санкционированного доступа абонентам;
- е) производить смену системного пароля и подписи должностного лица;
- ж) вызывать помощника автоматизированного создания таблицы санкционированного доступа;
- к) изменять мандатный уровень доступа пользователя по обработке информации.

Первоначальная настройка реализована модулем `sp_user_control_form`. Алгоритм реализации модуля `sp_user_control_form` представлен на рисунке 9. Ниже представлены функции, реализующие первоначальную настройку.

Функции, реализующие первоначальную настройку:

```
void spUserControlForm::defaultTSDtoTable() {
    spTsdRecord* r;
    tlist.clear();
    Q3DictIterator<spTsdRecord> it( sp.usr()->tsd );
    int rows = sp.usr()->tsd.count();
    usersTable->setNumRows(rows);
    for(int row=0 ; (r=it.current()); ++it, row++ ){
        tlist += *r;
    }
}

void spUserControlForm::changedBL_Slot(bool f)
{
    changeBit(CP_BLK, f);
}

void spUserControlForm::changedKUF_Slot(bool f)
{
    changeBit(CP_KUF, f);
}
```

```

void spUserControlForm::changedNVO_Slot(bool f)
{
    changeBit(CP_NVO, f);
}

qSort(tlist.begin(),tlist.end(), qLessRecord<spTsdRecord>());
for(int row=0 ; row<rows; row++ ){
    spTsdRecord& rc = tlist[row];
        if(rc.post == sp.currentName() || rc.post == sp.usr()-
>otvSziName()){
                                usersTable->setItem(row, 0, new
Q3TableWidgetItem(usersTable, Q3TableWidgetItem::Never));
                                }
        usersTable->setText(row, 0, rc.post);
        usersTable->setText(row, 1, rc.sign);
    }
    tlistSetChanged(false);
    tableCurrentChangedSlot(0,0);
}

void spUserControlForm::changedBL_Slot(bool f)
{
    changeBit(CP_BLK, f);
}

void spUserControlForm::tsdToTable(QList<spTsdRecord>& externList){
    tlist.clear();
    QString osziName = sp.usr()->otvSziName();
    bool yesOszi = false;
    QList<spTsdRecord>::iterator it;
    for ( it = externList.begin(); it != externList.end(); ++it ){
        spTsdRecord& r0 = *it;
        if(r0.post == osziName){
            yesOszi = true;
        }
        r0.ord = sp.usr()->tsd.order(r0.post);
        if(r0.ord/100000 > 3){
            r0.prf = 0;
        }
        else{
            int dol = r0.ord%10;
            switch(dol){
                case 1:
                    r0.prf = CP_BLK | CP_VPO | CP_KUF | CP_NVO | CP_SVO | CP_PRN;
                    break;
                case 2:

```

```

        r0.prf = CP_BLK | CP_VPO | CP_KUF | CP_NVO | CP_SVO | CP_PRN;
        break;
        case 3:
        r0.prf = CP_BLK | CP_PLM | CP_PLO | CP_PLOG | CP_PLUS | CP_VPP |
        CP_OSZI | CP_OKUF | CP_NVO | CP_SVP | CP_PRN;
        break;
        case 4:
        r0.prf = CP_BLK | CP_PLM | CP_PLO | CP_PLOG | CP_PLUS | CP_VPP |
        CP_OKUF | CP_NVO | CP_SVP | CP_PRN;
        if(r0.post.contains(S_SOPR)){
            r0.prf |= CP_OSZI;
        }
        break;

int rows = tlist.count();
qSort(tlist.begin(), tlist.end(), qLessRecord<spTsdRecord>());

usersTable->setNumRows(rows);
for(int row=0 ; row<rows; row++ ){
    spTsdRecord& rc = tlist[row];
    if(rc.post == sp.currentName() || rc.post == sp.usr()-
>otvSziName()){
        usersTable->setItem(row, 0, new
Q3TableWidgetItem(usersTable, QTableWidgetItem::Never));
    }
    usersTable->setText(row, 0, rc.post);
    usersTable->setText(row, 1, rc.sign);
}

tlistSetChanged(true);
    usersTable->setCurrentCell(0,0);
    tableCurrentChangedSlot(0,0);
    tsdGenerateAction->setEnabled(false);
    fileSaveSlot();
}
.

```

Рисунок 9 - Алгоритм реализации модуля sp_user_control_form

4.2.2.3 Штатная работа программного обеспечения, частью которого является разрабатываемая библиотека.

Штатная работа включает в себя следующие основные задачи:

- а) регистрация;
- б) управление полномочиями пользователей;
- в) управление пользователями;
- г) АРМ обеспечения безопасности информации (ОБИ);
- д) блокировка и разблокировка подконтрольных АРМ;
- е) просмотр событий безопасности;
- ж) резервное копирование;
- и) восстановление из резервных копий;
- к) работа с внешними носителями информации.

Задача регистрации реализована модулем `sp_`, `sp_login_form`.

В рамках данной задачи реализована также реализована система авторизации, которая является мерой защиты для противодействия реализации угрозы «Выдача себя за зарегистрированного пользователя». Реализация данной задачи построена на основе принципов аутентификации пользователей. Алгоритм аутентификации пользователя представлен на рисунке 10.

При штатной работе должностному лицу необходимо войти в операционную систему МСВС со своей учетной записью (паролем). Запустить программное обеспечение, частью которого является разрабатываемая библиотека функций, после чего будет проведена проверка целостности программного обеспечения[6]. В случае нарушения целостности программного обеспечения произойдет немедленный выход из программного обеспечения. В случае успешной проверки будет выполнено согласование реальных пользователей операционной системы МСВС с указанными в таблице санкционированного доступа, лишние пользователи будут удалены из системы, новые – добавлены. Факты согласования будут зарегистрированы в журнале событий безопасности.

Далее должностное лицо должно зарегистрироваться в системе со своей подписью. В случае успешной регистрации должностное лицо будет допущено к программному обеспечению, частью которого является разрабатываемая библиотека функций, в соответствии со своими полномочиями.

Рисунок 10 – Алгоритм аутентификации пользователя

Ниже представлена функция, реализующая авторизацию пользователей.

```
bool Sp::login(QString msg/*=null*/, bool bl/*=false*/, bool visitor/*=false*/,
bool qu/*=false*/ ){
    QString curName;
    QString curSign;
    bool ok = false;
    if(visitor) bl = false;
    if(bl){
        curName = users->otvSziName();
        curSign = users->otvSziSign();
    }
    else{
        if(users->armIsBlocked()){
            ok = doReceipt(BlockWP, currentName());
        }
        curName = users->currentName();
        curSign = users->currentSign();
    }

    /*
        if(!visitor){
            QStringList ru = registeredUsers();
            QString curName16 = curName + " ";
            curName16.truncate(16);
            if(ru.contains(curName16)){
                //...
                QString err("Должностное лицо ");
                err += curName;
                err += " уже зарегистрировано в
системе.\nВыход.";
                QMessageBox::critical(0,
```

```

        "Регистрация...",
        err,
        QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
        ok = false;
        return ok;
    }
}
*/
repl2:;
    int trys = bl? 3000 : 3;

    for(int iTry=0; iTry<trys; iTry++){
        spLoginForm lf(curName, curSign, msg);
        connect(this, SIGNAL(exitWithTrueSignSignal()), &lf,
SLOT(exitWithTrueSignSlot()));
        if(lf.exec() == QDialog::Rejected){
            if(bl && !qu) continue;
            ok = false;
            return ok;
        }

        QString gS = lf.getSign();
        if(!latin()){
            gS = latRUSS(gS);
        }
        if(gS == curSign){ //Sign is right
            //Entry ok
            ok = true;
            usr()->armSetState(armStateWork);
            return ok;
        }
        else{ //Sign is not right
            msg = "Подпись не верна.";
            if(!bl && iTry==1)
                msg += "Последняя попытка.";
        }
    }
    if(!visitor){
        ok = doReceipt(BlockWP, currentName());
        msg.clear();
        goto repl2;
    }
    return ok;
}

```

```
}
```

Библиотечный модуль `sp_menu` динамически сформирует меню текущего пользователя с доступными ему в соответствии с полномочиями функциями системы защиты информации.

Программный код функции, реализующей формирование меню текущего пользователя с доступными ему, в соответствии с его полномочиями функциями системы защиты информации представлен ниже.

```
Q3PopupMenu* spMenu::menu(QWidget* w){

    aProfile = new QAction("Полномочия","Полномочия", 0, w);
    aProfile->setStatusTip("Полномочия действующего ДЛ");
    aProfile->setWhatsThis("Полномочия действующего ДЛ");
    connect( aProfile, SIGNAL(activated()), this, SLOT(slotProfile()) );

    aControlUsers = new QAction("Управление пользователями","Управление
пользователями", 0, w);
    aControlUsers->setStatusTip("Управление пользователями");
    aControlUsers->setWhatsThis("Управление пользователями");
    connect( aControlUsers, SIGNAL(activated()), this, SLOT(slotControlUsers()) );

    aAOBI = new QAction("АРМ ОБИ","АРМ ОБИ", 0, w);
    aAOBI->setStatusTip("АРМ ОБИ");
    aAOBI->setWhatsThis("АРМ ОБИ");
    connect( aAOBI, SIGNAL(activated()), this, SLOT(slotAOBI()) );

    aBlock = new QAction("Блокировка/Разблокировка РМ","Блокировка/Разблокировка
РМ", 0, w);
    aBlock->setStatusTip("Блокировка/Разблокировка РМ");
    aBlock->setWhatsThis("Блокировка/Разблокировка РМ");
    connect( aBlock, SIGNAL(activated()), this, SLOT(slotBlockUnblock()) );

    aEvents = new QAction("Просмотр событий безопасности","Просмотр событий
безопасности", 0, w);
    aEvents->setStatusTip("Просмотр событий безопасности");
    aEvents->setWhatsThis("Просмотр событий безопасности");
    connect( aEvents, SIGNAL(activated()), this, SLOT(slotEvents()) );

    aAudit = new QAction("Просмотр сщбытий аудита","Просмотр сщбытий аудита", 0,
w);
    aAudit->setStatusTip("Просмотр сщбытий аудита");
```

```

aAudit->setWhatsThis("Просмотр сщбытий аудита");
connect( aAudit, SIGNAL(activated()), this, SLOT(slotAudit()) );

pCntrMCBC = new Q3PopupMenu();
aTunningMandatLevels->addTo(pCntrMCBC);
aTunningAudit->addTo(pCntrMCBC);
aAudit->addTo(pCntrMCBC);

aIntegrity = new Q3Action("Контроль целостности", "Контроль целостности", 0,
w);
aIntegrity->setStatusTip("Контроль целостности");
aIntegrity->setWhatsThis("Контроль целостности");
connect( aIntegrity, SIGNAL(activated()), this, SLOT(slotIntegrity()) );

aTest = new Q3Action("Тестирование СЗИ", "Тестирование СЗИ", 0, w);
aTest->setStatusTip("Тестирование СЗИ");
aTest->setWhatsThis("Тестирование СЗИ");
connect( aTest, SIGNAL(activated()), this, SLOT(slotTest()) );

aRecovery = new Q3Action("Восстановление СЗИ", "Восстановление СЗИ", 0, w);
aRecovery->setStatusTip("Восстановление СЗИ");
aRecovery->setWhatsThis("Восстановление СЗИ");
connect( aRecovery, SIGNAL(activated()), this, SLOT(slotRecovery()) );

aArchive = new Q3Action("Резервное копионовление", "Резервное копирование", 0,
w);
aArchive->setStatusTip("Резервное копирование");
aArchive->setWhatsThis("Резервное копирование");
connect( aArchive, SIGNAL(activated()), this, SLOT(slotArchive()) );

aUnArchive = new Q3Action("Восстановление из резервных копий", "Восстановление
из резервных копий", 0, w);
aUnArchive->setStatusTip("Восстановление из резервных копий");
aUnArchive->setWhatsThis("Восстановление из резервных копий");
connect( aUnArchive, SIGNAL(activated()), this, SLOT(slotUnArchive()) );

```

Задача «Управление полномочиями пользователей» реализована модулем `sp_profile_form` и позволяет любому должностному лицу просмотреть доступные ему полномочия.

Задача «Управление пользователями» реализована модулем `sp_user_control_form`.

Этот функционал доступен для ДЛ с полномочиями «Ответственный за СЗИ». В этом режиме ДЛ предоставляются следующие возможности:

- а) вводить, корректировать и удалять информацию из ТСД;
- б) сохранять ТСД;
- в) сохранять ТСД под новым именем (резервная ТСД);
- г) загружать ТСД;
- д) рассылать ТСД абонентам;
- е) производить смену системного пароля и подписи ДЛ;
- ж) вызывать помощника автоматизированного создания ТСД;
- и) изменять мандатный уровень доступа пользователя по обработке информации.

Задача «АРМ обеспечения безопасности информации» реализована модулем `sp_obj_form`. Задача доступна для ДЛ с полномочиями «Ответственный за СЗИ» и выполняет следующие функции:

- а) опрос и индикация состояния подконтрольных ДЛ;
- б) проверка аутентичности ДЛ, работающего за АРМ оператора;
- в) блокировка и разблокировка подконтрольных АРМ;
- г) проверка целостности программного обеспечения.

Реализация проверки целостности программного обеспечения является мерой, значительно снижающий ущерб от реализации угрозы «удаление или искажение ПО или файлов с важной информацией, в том числе системных». Код реализации данной функции представлен ниже.

```
bool Sp::checkIntegrity(QString sender/*=QString::null*/, int autoHide){
    bool ok = false;
    QFileInfo fi(INTEGRITY_CONF);
    ok = fi.exists();
    QFile file,tmp;
    if(ok){
        file.setName( INTEGRITY_CONF );
        ok = file.open( IO_ReadOnly );
        if(ok){
            tmp.setName(INTEGRITY_CONF_TEMP);
            ok = tmp.open( IO_WriteOnly );
```

```

    }
}

QString lks, ksum;
    spLabel label("Проверка целостности ПО. Подождите...");
if(ok){
    QTextStream fs(&file);
    QTextStream stmp(&tmp);
    QRegExp sep( "\\s+" );
    QString l, lfn;
    bool Create = false;
    QStringList slist;
    while ( !fs.atEnd() ) {
        l = fs.readLine();
        if(l.isEmpty())
            continue;
        slist << l;
    }
    int n = slist.count();
    int sp = 2;

    for(int i=0; i<n; i++){
        l = slist[i];

        progress.setProgress( i +sp);
        QApplication->processEvents();

        lfn = l.section(sep, 0, 0);
        lfn = lfn.stripWhiteSpace();
        if(lfn[0]=='#')
            continue;

        QFileInfo lfni(lfn);
        if(!lfni.exists()){
            progress.setProgress( n );
            QString comp;
            if(!sender.isEmpty()){
                send(BadIntegrity, sender);
                usr()->armSetState(armStateBadIntegrity);
            }

            reg("СЗИ", "Проверка целостности ПО", "Нарушена целостность");
            comp = QString("Нарушена целостность ПО.\nОтсутствует компонент
('%1')").arg(lfn.ascii());

```

```

if(!sender.isEmpty()){
    if(ok) {
        send(OkIntegrity, sender);
        usr()->armSetState(armStateWork);
    }
    else {
        send(BadIntegrity, sender);
        usr()->armSetState(armStateBadIntegrity);
    }
}

if(ok){
    reg("СЗИ", "Проверка целостности ПО", "Целостность не нарушена");
    if(autoHide){
        QMessageBox mb( "Проверка целостности ...",
            "Целостность ПО не нарушена",
            QMessageBox::Information,
            QMessageBox::Ok | QMessageBox::Default
        );
    }
    label.hide();
    return ok;
}

```

Функция экстренного стирания информации с автоматизированного рабочего места необходима, в случае, если нарушителю удалось получить доступ к автоматизированному рабочему месту. Данная функция позволит не позволить ему получить искомую информацию в случае, если его доступ удалось вовремя заметить. Реализация данной функции представлена ниже.

```

if((fconf=fopen(FileName, "rt"))!=NULL)
{
    while(fgets(szBuff, sizeof(szBuff), fconf))
    {
        qApp->processEvents();
        int lb=strlen(szBuff)-1;
        char *szTmp=(char *)strchr(szBuff, '#');
        if(szTmp) szTmp[0]=0;
        else if (szBuff[lb]=='\n') szBuff[lb]=0;
        if(szBuff[0])
        {
            if(!strcmp("[erase]", szBuff))
                mode=0;
        }
    }
}

```

```

else
if(!strcmp("[overwrite", szBuff, 10))
{
mode=1;
szBuff[lb-1]=0;
if(szBuff[10]=='=')
dRepeat=atoi(&szBuff[11]); else dRepeat=dRep;

}
else
{
char *szMask=(char *)strrchr(szBuff, '/');
char *szMask_type=(char*)strrchr(szBuff, '*');

if(szMask_type)
{
szMask[0]=0;
szMask++;
szMask_type[0]=0;
szMask_type++;
if(szMask_type[0])
{ //удалени по маске *.rasshirenie
remrec(szMask_type, szBuff, mode, dRepeat, 0, dVerb);

}
else
{
//удаление по маске *
*szMask='*';
strcpy(szPath, szBuff);
remrec(szMask, szPath, mode, dRepeat, 0, dVerb);

}
}

label.hide();
fclose(fconf);
system ("sudo /sbin/poweroff");
return ok;
}

```

Задача «Блокировка и разблокировка подконтрольных АРМ» реализована модулем `sp_blocking_form`. Доступна для должностных лиц с полномочиями

«Блокировка/разблокировка». Реализация функции представлена ниже.

```
void spBlockingForm::blockSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Блокировка АРМ"),
                                     "Вы действительно хотите заблокировать "+
post,
                                     QMessageBox::Ok, QMessageBox::Cancel);

    if (exit==QMessageBox::Ok){
        sp.send(BlockWP, post);
    }
}

void spBlockingForm::unblockSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Разблокировка АРМ"),
                                     "Вы действительно хотите разблокировать "+
post,
                                     QMessageBox::Ok, QMessageBox::Cancel);

    if (exit==QMessageBox::Ok){
        sp.send(UnblockWP, post);
    }
}

void spBlockingForm::init(){
    const QList<spTsdRecord>& l = sp.usr()->inferiors();
    int n = l.count();
    int rw = 0;
    for(int i=0; i<n; i++){
        const QString& post = l[i].post;
        if(sp.findInTva(post)){
            tblUsers->setNumRows(rw+1);
            tblUsers->setText(rw, 0, post);
            tblUsers->setText(rw, 1, "Не известно");
            rw++;
        }
    }
    tblUsers->adjustColumn(1);
}
```

Задача «Просмотр событий безопасности» реализована модулем `sp_register_form`.

Все происходящие в системе безопасности события регистрируются в журнале. Просмотр журнала доступен должностным лицам с полномочиями «Ответственный за СЗИ».

При работе с журналом событий безопасности предоставляются следующие возможности:

- а) производить фильтрацию записей журнала по дате и времени;
- б) удалять записи из журнала по выбранной дате;
- в) удалять записи из журнала ранее выбранной даты.

Код реализации представлен ниже.

```
QFileInfo fi( conf );
if ( fi.exists() ) {
    if(!currentIsRoot())
        command << "sudo";
    command << "tar" << "-czf" << tgz << "-T" << conf <<"-P";

    spLabel label("Идет резервное копирование.\n Подождите..." );
    qApp->processEvents();
    ProcessManager pr(command);
    while(pr.isRunning())
        qApp->processEvents();
    label.hide();
    ok = true;
    QMessageBox::information(0, "Состояние", "Резервное копирование завершено");
    reg("СЗИ", "Резервное копирование", "Завершено");

bool Sp::restore(QString conf, QString tgz){
    bool ok = false;
    reg("СЗИ", "Резервное восстановление", "Вход");
    if(conf.isEmpty())
        conf = BACKUP_CONF;
    if(tgz.isEmpty())
        tgz = BACKUP_FILE;

    QFileInfo fi(tgz);
    if(fi.exists()){
```

```

QStringList command;
QFileInfo fc( conf );
if ( fc.exists() ) {
    if(!currentIsRoot())
        command << "sudo";
    command << "tar" << "-xzf" << tgz << "-T" << conf << "-P";

    spLabel label("Идет восстановление из резервных копий.\n Подождите..." );
    qApp->processEvents();

    ProcessManager pr(command);
    while(pr.isRunning())
        qApp->processEvents();
    label.hide();
    ok = true;
    QMessageBox::information(0, "Состояние", "Восстановление файлов
завершено");
    reg("СЗИ", "Резервное восстановление", "Завершено");
}
return ok;
}

```

Полный код реализации компонентного состава разрабатываемой библиотеки функций представлен в приложении Е.

5 Проверка работоспособности приложения

В соответствии с требованиями технического задания была проведена проверка работоспособности разработанного приложения. Проверка работоспособности проводилась в операционной системе MSVC 3.0. С целью получения возможности осуществить была разработана тестовая программа, имеющая возможности использования функций, содержащихся в разработанной библиотеке. Основное окно тестовой программы представлено на рисунке 11.

Рисунок 11 – Основное окно тестовой программы

Перед началом работы ответственный за систему защиты информации должен сформировать таблицу санкционированного доступа. Для этого ОСЗИ необходимо ввести в редактируемую текстовую строку «Подпись:» окна «СЗИ: Регистрация...» «123456», нажать кнопку **«Принять»**, при этом на экране АРМ отображается диалоговое окно «Управление пользователями». Окно представлено на рисунке 12.

Диалоговое окно «Управление пользователями» содержит главное меню, меню кнопок с пиктограммами, таблицу санкционированного доступа и зону «Полномочия».

При работе в режиме «Управление пользователями» ОСЗИ предоставляются следующие возможности:

- а) вводить, корректировать и удалять информацию из ТСД;
- б) сохранять ТСД;
- в) сохранять ТСД под новым именем (резервная ТСД);
- г) загружать ТСД;
- д) рассылать ТСД абонентам;
- е) производить смену системного пароля и подписи ДЛ;
- ж) вызывать помощника автоматизированного создания ТСД;
- и) изменять уровень доступа пользователя по обработке информации.

Рисунок 12 – Окно «Управление пользователями»

Для сохранения таблицы санкционированного доступа необходимо нажать в главном меню окна «СЗИ: Управление пользователями» кнопку «ТСД», в открывающемся подменю выбрать элемент «Сохранить как», при этом на экране автоматизированного рабочего места отображается окно «Сохранить как», представленное на рисунке 13.

Загрузить таблицу санкционированного доступа можно нажав в главном меню окна «СЗИ: Управление пользователями» кнопку «ТСД», в открывшемся подменю выбрать элемент «Загрузить», при этом на экране отображается окно «Открыть», представленное на рисунке 14.

Для смены пароля необходимо в таблице санкционированного доступа выбрать должностное лицо, чей пароль следует изменить, в главном меню окна «СЗИ: Управление пользователями» выбрать пункт «Пользователи», после чего в открывшемся подменю выбрать пункт «Изменить пароль», при этом отображается окно «СЗИ: Введите пароль», представленное на рисунке 15. Пользователь может ознакомиться с доступными ему режимами работы выбрав в меню «СЗИ» пункт полномочия, при этом отображается диалоговое окно «СЗИ: Полномочия», представленное на рисунке 16.

Рисунок 13 – Окно «Сохранить как»

Режим «АРМ ОБИ» доступен только для должностных лиц с полномочиями «Ответственный за СЗИ». Данный режим позволяет блокировать и разблокировать автоматизированные рабочие места должностных лиц, осуществлять проверку целостности программного обеспечения, а также тестирование и восстановление системы защиты информации, экстренное стирание информации с автоматизированного рабочего места. Для входа в него необходимо выбрать пункт «АРМ ОБИ» в меню «СЗИ». При этом отображается диалоговое окно, представленное на рисунке 17.

Рисунок 14 – Окно «Открыть»

Рисунок 15 – Окно «СЗИ: Введите пароль»

Рисунок 16 – Окно «СЗИ: Полномочия»

Проверка целостности программного обеспечения заключается в сравнении контрольных сумм основных файлов программного обеспечения с эталонными значениями.

В случае отсутствия нарушения целостности программного обеспечения на экране проверяемого автоматизированного рабочего места отображается окно «Проверка целостности», представленное на рисунке 18. В случае закрытия данного окна, на экране АРМ ОБИ отображается окно «Получено сообщение», представленное на рисунке 19, в случае нарушения целостности программного обеспечения на экране АРМ ОБИ отображается окно «Получено сообщение», представленное на рисунке 20.

Тестирование функций системы защиты информации на автоматизированном рабочем месте подконтрольного должностного лица можно осуществить путём выбора пункта меню «Тестирование СЗИ», предварительно выбрав в таблице требуемого абонента

Рисунок 17 – Окно «СЗИ: АРМ ОБИ»

Рисунок 18 – Окно «Проверка целостности»

Рисунок 19 – Окно «Получено сообщение»

Рисунок 20 – Окно «Получено сообщение»

В случае успешного тестирования функций системы защиты информации на экране АРМ ОБИ отображается окно «Получено сообщение», представленное на рисунке 21. В случае неудачного тестирования выдается окно «Получено сообщение, представленное на рисунке 22.

Рисунок 21 – Окно «Получено сообщение»

Рисунок 22 – Окно «Получено сообщение»

Восстановление работоспособности системы защиты информации на автоматизированном рабочем месте подконтрольного должностного лица можно осуществить путём выбора пункта меню «Восстановление СЗИ», после чего на экране отображается процесс восстановления. По окончании данного процесса на экране отображается окно «Состояние», представленное на рисунке 23.

Рисунок 23 – Окно «Состояние»

Режим «Блокировка/разблокировка» доступен для лиц с полномочиями «Блокировка/разблокировка». Для входа в данный режим необходимо в меню «СЗИ». При этом на экране автоматизированного рабочего места отображается окно «СЗИ: Блокировка/разблокировка АРМ», представленное на рисунке 24.

Рисунок 24 – Окно «СЗИ: Блокировка/разблокировка АРМ»

Осуществление резервного копирования доступно лишь для должностных лиц с полномочиями «Ответственный за СЗИ». Для его осуществления необходимо в меню «СЗИ» выбрать пункт «Резервное копирование». По окончании процесса на экране автоматизированного рабочего места отображается окно «Состояние» с сообщением «Резервное копирование завершено», представленное на рисунке 25.

Рисунок 25 – Окно «Резервное копирование завершено»,

Осуществление восстановления файлов из резервных копий доступно лишь для должностных лиц с полномочиями «Ответственный за СЗИ». Для его осуществления необходимо в меню «СЗИ» выбрать пункт «Восстановление из резервных копий». По окончании процесса на экране автоматизированного рабочего места отображается окно «Состояние» с сообщением «Восстановление файлов завершено», представленное на рисунке 26.

Рисунок 26 – Окно «Восстановление файлов завершено»,

ЗАКЛЮЧЕНИЕ

В результате выполнения ВКР было разработано приложение в виде библиотеки функций реализующее идентификацию и аутентификацию субъектов доступа, контроль целостности программных средств и обрабатываемой информации, управление пользователями, блокировку и разблокировку абонентов, экстренное стирание информации, протоколирование событий безопасности, просмотр событий безопасности, тестирование функций средств защиты информации от несанкционированного доступа, управление резервным копированием файлов и директорий, кодирование данных о субъектах доступа на основе ГОСТ Р 34.12-2015.

В процессе дипломного проектирования были рассмотрены общие сведения об операционной системе МСВС 3.0, выполнен обзор и анализ современных методов и средств криптографической защиты данных, был

проведен анализ угроз информационной безопасности в рассматриваемой предметной области, были выявлены актуальные угрозы и выбраны методы защиты от их реализации, выбранные методы были учтены и реализованы в созданном приложении, а также была проверена его работоспособность в операционной системе MSVC 3.0.

Результатом работы стало повышение моих профессиональных навыков, .

Таким образом, все пункты технического задания на выпускную квалификационную работу выполнены полностью.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Горбенко И.Д., Качко Е.Г., Потий, А.В. Решения и средства защиты информации. М.: «Форум-ИнфраМ», 2008. – 533 с.
- 2 Проблемы защиты информации [Электронный ресурс] // AB Solut. Инновации сегодня URL: http://ab-solut.net/ru/articles/problemi_zashiti (дата обращения: 19.03.2017).
- 3 Инновационный потенциал Военно-промышленного комплекса России [Электронный ресурс] // webeconomy.ru: Мировая экономика URL: <http://www.webeconomy.ru/index.php?page=cat&newsid=844&type=news> (дата обращения: 25.03.2017).
- 4 ГОСТ 34.12-2015. Информационная технология. Криптографическая защита информации. Блочные шифры. М.: «Стандартинформ», 2015. – 28с.
- 5 Атаки на криптографические системы [Электронный ресурс] URL: <http://algolist.manual.ru/defence/attack/> (дата обращения: 12.04.2017).
- 6 Qt 3.3.3 [Электронный ресурс] URL: <https://doc.qt.io/archives/3.3/classes.html> (дата обращения: 12.04.2017).
- 7 «Системы обработки информации. Защита криптографическая. Рекомендации по стандартизации. Задание узлов замены блока подстановки алгоритма шифрования ГОСТ 34.12-2015» [Электронный ресурс] // Технический комитет по стандартизации «Криптографическая защита информации» (ТК 26) URL: <https://tc26.ru/methods/recommendation/%D0%A2%D0%9A26%D0%A3%D0%97.pdf> (дата обращения: 16.05.2017).
- 8 Объектно-ориентированное программирование [Электронный ресурс] URL: https://ru.wikibooks.org/wiki/Объектно_ориентированное_программирование (дата обращения 22.05.2017).

ПРИЛОЖЕНИЕ А
(обязательное)

Структура типичной сети, в которой функционирует библиотека

Рисунок А.1

ПРИЛОЖЕНИЕ Б
(обязательное)

Структура взаимодействия должностных лиц

Рисунок Б.1

ПРИЛОЖЕНИЕ В
(обязательное)

Структура авторизации должностного лица на АРМ

Рисунок В.1

ПРИЛОЖЕНИЕ Г
(обязательное)

Алгоритм обмена ключами

Рисунок Г.1

ПРИЛОЖЕНИЕ Д
(обязательное)

Алгоритм работы ГОСТ Р 34.12-2015

Рисунок Д.1

ПРИЛОЖЕНИЕ Е

(обязательное)

Исходный код разработанного приложения

Компонентный состав библиотеки функций.

sp_ - основной класс, обеспечивающий весь функционал системы обеспечения безопасности, программный интерфейс с другими подсистемами, прием и отправку сообщений безопасности в каналы связи и т.п. Исходный код представлен ниже.

```
#include "sp_.h"
#include "sp_menu.h"
#include "sp_users.h"
#include <qmessagebox.h>
#include <qapplication.h>
#include <qfile.h>
#include <qtextstream.h>
#include <qdir.h>          //for QDir
#include <qlabel.h>
#include <qregexp.h>
#include <q3progressdialog.h>
#include <qdatetime.h>
#include <q3accel.h>
#include <qtimer.h>
#include <QDesktopWidget>

#include "sp_login_form.h"
#include "sp_usercontrol_form.h"
#include "sp_proc.h"
#include "sp_register.h"
#include "../sp_extern.h"
#include "sp_users.h"

#include <sys/types.h>
#include <sys/stat.h>
#include <fnmatch.h>
#include <unistd.h>
#include <dirent.h>
#include <q3ptrqueue.h>

#include <QtNetwork/qhostinfo.h>
#include <QtNetwork/qhostaddress.h>
#include <QtNetwork/qnetworkinterface.h>

#define INTEGRITY_CONF SP_PATH "/integrity.conf"
#define INTEGRITY_CONF_TEMP INTEGRITY_CONF ".temp"
#define BACKUP_CONF SP_PATH "/backup.conf"
#define BACKUP_FILE SP_PATH "/backup.tgz"
#define ERASE_CONF SP_PATH "/erase.conf"

#define RECOVERY_CONF "/var/preserve/recovery58.conf"
#define RECOVERY_FILE "/var/preserve/recovery58.tgz"

#define MSGTIMERINTERVAL 500
struct spMsg{
    spMsg(int type_, QString from_){
        type = type_;
        from = from_;
    }
    spMsg(QString path_, QString from_){
        type = File;
```

```

        from = from_;
        path = path_;
    }
    int type;
    QString from;
    QString path;
};

class spMsgQueue : public Q3PtrQueue<spMsg>
{
public:
    spMsgQueue():Q3PtrQueue<spMsg>(){
        setAutoDelete(true);
    }
};

static spMsgQueue msgQue;

spLabel::spLabel ( QString msg, QWidget* parent /*= 0*/, const char* name
/*= 0*/, Qt::WindowFlags f)
    : QLabel(parent, name, f)
{
    setCaption("C3И");
    setAlignment( Qt::AlignCenter );
    setText(msg);
    int w = 400;
    int h = 80;
    QDesktopWidget *d = QApplication::desktop();
    int W = d->width();      // returns desktop width
    int H = d->height();     // returns desktop height
    int x = (W-w)/2;
    int y = (H-h)/4;
    setGeometry(x, y, w, h);
    show();
}

void spLabel::closeEvent ( QCloseEvent* /*e*/){}

QList<QNetworkAddressEntry> getHostIP()
{
    QList<QNetworkAddressEntry> addressesEntry;
    addressesEntry.clear();
    for( int iCount = 0; iCount <
QNetworkInterface::allInterfaces().count(); iCount++ )
    {
        QNetworkInterface current =
QNetworkInterface::interfaceFromName(QNetworkInterface::allInterfaces().valu
e(iCount).name());
        if(current.isValid())
            addressesEntry += current.addressEntries();
    }
    for( int index = 0; index < addressesEntry.count(); index++ )
    {
        if(addressesEntry.value(index).ip().protocol() !=
QAbstractSocket::IPv4Protocol
        || addressesEntry.value(index).ip() == QHostAddress::LocalHost)
        {
            addressesEntry.removeAt(index);
            index--;
        }
    }
    return addressesEntry;
}

```

```

Sp::Sp() {
    menu = new spMenu();
    users = new spUsers();
    rg = new spRegister();
    localLoop = true;
    cf = 0;
    isStateChangedSignalConnected = false;
    //-----
    QDir tsdDir(TSD_PATH);
    if(tsdDir.exists()){
        commonAccess(BLOCKARMF);
        commonAccess(STATPOST);
        commonAccess(INTEGRITY_CONF);
        commonAccess(BACKUP_CONF);
        commonAccess(BACKUP_FILE);
        commonAccess(ERASE_CONF);
        commonAccess(RECOVERY_CONF);
        commonAccess(RECOVERY_FILE);
        commonAccess(TSD_FILE_NAME);
        commonAccess(DATABASE);

        QString command = "sudo chmac -fR 0:0:3f ";
        command += SP_PATH;
        system(command.ascii());
    }
}

Sp::~Sp() {
    delete menu;
    delete users;
    delete rg;
}

void Sp::addMenu(QWidget* w, Q3PopupMenu* parent){
    menu->add(w,parent);
}

void Sp::addMenu(QWidget* w, QMenuBar* parent){
    menu->add(w,parent);
}
//-----
Sp sp;

bool Sp::login(QString msg/*=null*/, bool bl/*=false*/, bool
visitor/*=false*/, bool qu/*=false*/ ){
    QString curName;
    QString curSign;
    bool ok = false;
    if(visitor) bl = false;
    if(bl){
        curName = users->otvSziName();
        curSign = users->otvSziSign();
    }
    else{
        if(users->armIsBlocked()){
            ok = doReceipt(BlockWP, currentName());
        }
        curName = users->currentName();
        curSign = users->currentSign();
    }
}

/*

```

```

        if(!visitor){
            QStringList ru = registeredUsers();
            QString curName16 = curName + " ";
            curName16.truncate(16);
            if(ru.contains(curName16)){
                //...
                QString err("Должностное лицо ");
                err += curName;
                err += " уже зарегистрировано в
системе.\nВыход.";
                QMessageBox::critical(0,
                    "Регистрация...",
                    err,
                    QMessageBox::Ok, QMessageBox::NoButton,
                    QMessageBox::NoButton );
                ok = false;
                return ok;
            }
        }
    */
    repl2::
        int trys = bl? 3000 : 3;

        for(int iTry=0; iTry<trys; iTry++){
            spLoginForm lf(curName, curSign, msg);
            connect(this, SIGNAL(exitWithTrueSignSignal()), &lf,
                SLOT(exitWithTrueSignSlot()));
            if(lf.exec() == QDialog::Rejected){
                if(bl && !qu) continue;
                ok = false;
                return ok;
            }

            QString gS = lf.getSign();
            if(!latin()){
                gS = latRUSS(gS);
            }
            if(gS == curSign){ //Sign is right
                //Entry ok
                ok = true;
                usr()->armSetState(armStateWork);
                return ok;
            }
            else{ //Sign is not right
                msg = "Подпись не верна.";
                if(!bl && iTry==1)
                    msg += "Последняя попытка.";
            }
        }
        if(!visitor){
            ok = doReceipt(BlockWP, currentName());
            msg.clear();
            goto repl2;
        }
        return ok;
    }

#define PRECHECK
extern QString developer;
bool Sp::init(){
    bool ok = false;
    QString msg;

```

```

//----- for workOnControl
connect(qApp, SIGNAL( workOnControlSignal() ), this,
SLOT( workOnControlSlot() ));
connect(qApp, SIGNAL( aboutToQuit() ), this, SLOT( logoutSlot() ));

QString currentUserName = users->currentName();
bool oszi = (QString(OTVSZINAME) == currentUserName);

#ifdef DEVELOPER
    developer = currentUserName;
    qDebug("*** developer=%s", developer.ascii());
#endif
users->tsd.load();

msgQue.clear();
idTimer = startTimer(MSGTIMERINTERVAL);

    if(users->tsd.exist()){        //TSD exist

#ifdef PRECHECK
        for(int ia=1; ia<qApp->argc(); ia++){
            QString arg = qApp->argv()[ia];
            if(arg.contains("precheck")){
                if(!checkIntegrity(QString::null, 2000)){
                    activateLogout();
                    logoutSlot();
                    _exit(0);
                }
            }
        }
#endif
label:
        spTsdRecord* r = users->tsd[users->currentName()];
        if( r && r->prf ){
            ;//Name exist in TSD
        }
        else{
            QMessageBox::critical(0,
                                "Регистрация...",
                                "Вы не допущены к работе с системой.\n",
                                QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
            msg = "Отказано.ДЛ отсутствует в ТСД.";
            reg("СЗИ", "Вход в СОД", msg);
            ok = false;
            return ok;
        }
        if(users->tsdIsActual()){ //TSD actual
            {
                ok = login();
                msg = ok? "Выполнен" : "Отказано.Ошибка регистрации.";
                reg("СЗИ", "Вход в СОД", msg);
                if(ok){
                    send(Login, usr()->parentOsziName());
                    /*msgQue.clear();
                    idTimer = startTimer(MSGTIMERINTERVAL);*/
                    if(oszi){
                        activateLogout();
                        menu->slotControlUsers();
                        logoutSlot();

```

```

        _exit(0);
    }
}
return ok;
}
}
else{ //TSD not actual
    //Make TSD actual
    int ans = QMessageBox::information(0,
        "Регистрация...",
        "ТСД не согласована с системой.",
        "Согласовать",
        "Выход");

    if(ans!=0){
        ok = false;
        return ok;
    }
    else{ //Make TSD actual
        ok = users->tsdMakeActual();
        if(ok)
            goto label;
    }
}
}
else{ //TSD not exist
    //Control sudoers and TSD directory
    if(!inited()){ //First call
        QMessageBox::critical(0,
            "Регистрация...",
            "СЗИ не инициализирована.\n"
            "Для первоначальной настройки
            СЗИ\n"
            "зарегистрируйтесь в системе
            как 'root'",
            QMessageBox::Ok,
            QMessageBox::NoButton, QMessageBox::NoButton );
        disconnect(qApp, SIGNAL( workOnControlSignal() ), this,
            SLOT( workOnControlSlot() ));
        disconnect(qApp, SIGNAL( aboutToQuit() ), this,
            SLOT( logoutSlot() ));
        ok = false;
        return ok;
    }
    else{ //sudoers ok
        int ans = QMessageBox::information(0,
            "Регистрация...",
            "ТСД не сформирована.",
            "Сформировать",
            "Выход");
        if(ans!=0){
            ok = false;
            return ok;
        }
        else{ //Creating TSD
            ok = users->creatingMinimalTSD();
            if(ok) goto label;
            else return ok;
        }
    }
}
return ok;

```

```

}

/** No descriptions */
bool Sp::inited(){
    bool ok = users->controlSudoers();
    if(ok){
        QDir tsdDir(TSD_PATH);
        ok = tsdDir.exists();
        if(!ok){
            system("sudo mkdir -m 0777 "TSD_PATH);
            system("echo > "TSD_FILE_NAME);
            ok = recovery();
        }
    }
    return ok;
}

bool Sp::checkIntegrity(QString sender/*=QString::null*/, int autoHide){
    bool ok = false;
    QFileInfo fi(INTEGRITY_CONF);
    ok = fi.exists();
    QFile file,tmp;
    if(ok){
        file.setName( INTEGRITY_CONF );
        ok = file.open( IO_ReadOnly );
        if(ok){
            tmp.setName(INTEGRITY_CONF_TEMP);
            ok = tmp.open( IO_WriteOnly );
        }
    }
}

QString lks, ksum;
    spLabel label("Проверка целостности ПО. Подождите...");
if(ok){
    QTextStream fs(&file);
    QTextStream stmp(&tmp);
    QRegExp sep( "\\s+" );
    QString l, lfn;
    bool Create = false;
    QStringList slist;
    while ( !fs.atEnd() ) {
        l = fs.readLine();
        if(l.isEmpty())
            continue;
        slist << l;
    }
    int n = slist.count();
    int sp = 2;

    Q3ProgressDialog progress( "Проверка целостности ПО ...",0, n+sp, 0,
"progress", TRUE );
    progress.setCaption("СЗИ:Проверка целостности ПО ...");

        reg("СЗИ", "Проверка целостности ПО", "Вход");

    for(int i=0; i<n; i++){
        l = slist[i];

        progress.setProgress( i +sp);
        QApplication->processEvents();
    }
}

```

```

lfn = l.section(sep, 0, 0);
lfn = lfn.stripWhiteSpace();
if(lfn[0]=='#')
    continue;

QFileInfo lfni(lfn);
if(!lfni.exists()){
    progress.setProgress( n );
    QString comp;
    if(!sender.isEmpty()){
        send(BadIntegrity, sender);
        usr()->armSetState(armStateBadIntegrity);
    }
    reg("СЗИ", "Проверка целостности ПО", "Нарушена целостность");
    comp = QString("Нарушена целостность ПО.\nОтсутствует компонент
('%1')").arg(lfn.ascii());
    QMessageBox::critical(0,
        "Проверка целостности ПО ...",
        comp,
        QMessageBox::Ok, QMessageBox::NoButton,
        QMessageBox::NoButton );
    label.hide();
    return false;
}

lks = l.section(sep, 1, 1);
lks = lks.stripWhiteSpace();
bool create = lks.isEmpty();
Create = Create || create;

QStringList command;
command << "md5sum" << lfn;
ProcessManager pr(command);
while(pr.isRunning())
    qApp->processEvents();
ok = !pr.answer.isEmpty();
if(!ok)
    break;
ksum = pr.answer[0].section(sep, 0, 0);
stmp << lfn.ascii() << " " << ksum.ascii() << "\n";

if(create){
    ok = !ksum.contains("md5");
    if(!ok)
        break;
}
else{
    //check
    ok = ksum == lks;
    if(!ok)
        break;
}
}

progress.setProgress( n +sp);
file.close();
tmp.close();

if(Create && ok){
    if(QFile::exists(INTEGRITY_CONF_TEMP))
    {

```

```

        QFile::remove(INTEGRITY_CONF);
        QFile::rename(INTEGRITY_CONF_TEMP, INTEGRITY_CONF);
    }
}
else{
    QFile::remove(INTEGRITY_CONF_TEMP);
}

if(!sender.isEmpty()){
    if(ok) {
        send(OkIntegrity, sender);
        usr()->armSetState(armStateWork);
    }
    else {
        send(BadIntegrity, sender);
        usr()->armSetState(armStateBadIntegrity);
    }
}

if(ok){
    reg("СЗИ", "Проверка целостности ПО", "Целостность не нарушена");
    if(autoHide){
        QMessageBox mb( "Проверка целостности ...",
            "Целостность ПО не нарушена",
            QMessageBox::Information,
            QMessageBox::Ok | QMessageBox::Default |
QMessageBox::Escape ,
            0, 0);
        QTimer tclose;
        tclose.start( autoHide, true );
        mb.connect( &tclose, SIGNAL(timeout()), SLOT(close()));
        mb.exec();
    }
    else{
        QMessageBox::information(0,
            "Проверка целостности ...",
            "Целостность ПО не нарушена",
            QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
    }
}
else{
    reg("СЗИ", "Проверка целостности ПО", "Нарушена целостность");
    QMessageBox::critical(0,
        "Проверка целостности ...",
        "Нарушена целостность ПО.",
        QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
}
}
else{
    if(!sender.isEmpty()){
        send(BadIntegrity, sender);
        usr()->armSetState(armStateBadIntegrity);
    }
    reg("СЗИ", "Проверка целостности ПО", "Ошибка открытия
конфигурационного файла");
    QMessageBox::critical(0,
        "Проверка целостности ...",
        "Ошибка открытия конфигурационного файла.",

```

```

        QMessageBox::Ok, QMessageBox::NoButton,
        QMessageBox::NoButton );
    }
    label.hide();
    return ok;
}

#define WAITMS 600
static void waitMS(int ms){
    QTime start = QTime::currentTime();
    while(start.msecsTo(QTime::currentTime())<ms){
        QApplication->processEvents();
    }
}

bool Sp::test(QString sender/*=QString::null*/){
    QString heap = "СЗИ:Тестирование ...";
    QString func = "";
    bool ok = false;
    reg("СЗИ", "Тестирование", "Вход");

    spLabel label(heap);

    while(1){
        int steps = 8;
        int step = 1;
        Q3ProgressDialog progress(heap, 0, steps, 0, "progress", TRUE );
        progress.setCaption(heap);

        //1
        func = "Проверка инициализации СЗИ...";
        ok = users->controlSudoers();
        if(!ok) break;
        progress.setLabelText(func);
        progress.setProgress(step++);
        waitMS(WAITMS);

        //2
        func = "Проверка создания групп...";
        ok = users->groupAdd("testgroup");
        if(!ok) break;
        ok = users->groupExist("testgroup");
        if(!ok) break;
        progress.setLabelText(func);
        progress.setProgress(step++);
        waitMS(WAITMS);

        //3
        func = "Проверка удаления групп...";
        ok = users->groupDel("testgroup");
        if(!ok) break;
        ok = !users->groupExist("testgroup");
        if(!ok) break;
        progress.setLabelText(func);
        progress.setProgress(step++);
        waitMS(WAITMS);

        //4
        func = "Проверка создания ДЛ...";
        spTsdRecord r; //cleared
        r.post = "testuser";
        r.sign = "testuser";
    }
}

```

```

    r.prfl = CP_OSZI;      //Пользователи с 0 профилем не должны создаваться
    r.id = 0;
    ok = users->osUserAdd(&r);
    if(!ok) break;
    ok = users->osUserExist("testuser");
    if(!ok) break;
    progress.setLabelText(func);
    progress.setProgress(step++);
    waitMS(WAITMS);

//5
    func = "Проверка удаления ДЛ...";
    ok = users->osUserDel("testuser");
    if(!ok) break;
    ok = !users->osUserExist("testuser");
    if(!ok) break;
    progress.setLabelText(func);
    progress.setProgress(step++);
    waitMS(WAITMS);

//6
    func = "Проверка блокировки АРМ...";
    ok = users->armSetBlocking(true);
    if(!ok) break;
    ok = users->armIsBlocked();
    if(!ok) break;
    progress.setLabelText(func);
    progress.setProgress(step++);
    waitMS(WAITMS);

//7
    func = "Проверка разблокировки АРМ...";
    ok = users->armSetBlocking(false);
    if(!ok) break;
    ok = !users->armIsBlocked();
    if(!ok) break;
    progress.setLabelText(func);
    progress.setProgress(step++);
    waitMS(WAITMS);

    func = "Тест прошел успешно";
    break;
}

    if(!sender.isEmpty()){
        if(ok) {
            usr()->armSetState(armStateWork);
            send(TestOk, sender);
        }
        else {
            usr()->armSetState(armStateTestError);
            send(TestError, sender);
        }
    }

    if(!ok){
        func = "Ошибка выполнения теста\n" + func;
        reg("СЗИ", "Тестирование", "Ошибка выполнения теста");
        QMessageBox::critical(0,
                                heap,
                                func,

```

```

        QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
    }
    else{
        reg("СЗИ", "Тестирование", "Тест прошел успешно");
        QMessageBox::information(0,
                                heap,
                                func,
                                QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
    }

    label.hide();

    return ok;
}

bool Sp::reg(const char* module, const char* func, const char* msg){
    return rg->put(usr()->currentName(), module, func, msg);
}

bool Sp::send(int type, QString to, QString buf){
    if(currentName() == usr()->otvSziName()) return false;
    if(to.isEmpty()) return false;
    if(to==currentName()){
        receipt(type, to);
        return true;
    }
    bool ok = false;

    QString msg;

    switch(type){
    case CheckIntegrity: //
    {
        msg = "Проверить целостность ПО (к " + to + ")";
        reg("СЗИ", "Отправка сообщения", msg);
        ok = ::send(CheckIntegrity, to, buf);
    }
    break;
    case OkIntegrity:
    {
        msg = "Целостность ПО не нарушена (к " + to + ")";
        reg("СЗИ", "Отправка сообщения", msg);
        ok = ::send(OkIntegrity, to, buf);
    }
    break;
    case BadIntegrity: //сообщение о нарушении целостности;
    {
        msg = "Нарушена целостность ПО (к " + to + ")";
        reg("СЗИ", "Отправка сообщения", msg);
        ok = ::send(BadIntegrity, to, buf);
    }
    break;
    case Test: //предписание на проведение тестирования функций СЗИ;
    {
        msg = "Протестировать СЗИ (к " + to + ")";
        reg("СЗИ", "Отправка сообщения", msg);
        ok = ::send(Test, to, buf);
    }
    break;

```

```

case TestOk:          //сообщение о работоспособности СЗИ;
{
    msg = "СЗИ работоспособна (к " + to + ")";
    reg("СЗИ", "Отправка сообщения", msg);
    ok = ::send(TestOk, to, buf);
}
break;
case TestError:      //сообщение о неработоспособности СЗИ.
{
    msg = "СЗИ не работоспособна (к " + to + ")";
    reg("СЗИ", "Отправка сообщения", msg);
    ok = ::send(TestError, to, buf);
}
break;
case Kill:           //экстренное стирание информации;
{
    msg = "Экстренное стирание информации (к " + to + ")";
    reg("СЗИ", "Отправка сообщения", msg);
    ok = ::send(Kill, to, buf);
}
break;
case Recovery:       //предписание на проведение восстановления функций
СЗИ;
{
    msg = "Восстановить работоспособность СЗИ (к " + to + ")";
    reg("СЗИ", "Отправка сообщения", msg);
    ok = ::send(Recovery, to, buf);
}
break;
case UnblockConsole: //команда на разблокировку консоли оператора;
{
    msg = "Разблокировать консоль (к " + to + ")";
    reg("СЗИ", "Отправка сообщения", msg);
    ok = ::send(UnblockConsole, to, buf);
}
break;
case BlockWP:        //команда на блокировку рабочего места;
{
    msg = "Заблокировать АРМ (к " + to + ")";
    reg("СЗИ", "Отправка сообщения", msg);
    ok = ::send(BlockWP, to, buf);
}
break;
case UnblockWP:      //команда на разблокировку рабочего места;
{
    msg = "Разблокировать АРМ (к " + to + ")";
    reg("СЗИ", "Отправка сообщения", msg);
    ok = ::send(UnblockWP, to, buf);
}
break;
case WorkOnControl:  //сообщение о работе под контролем;
{
    msg = "Работа под контролем (к " + to + ")";
    reg("СЗИ", "Отправка сообщения", msg);
    ok = ::send(WorkOnControl, to, buf);
}
break;
case Login:          //вход должностного лица в СОД;
{
    msg = "Вход ДЛ в СОД (к " + to + ")";
    reg("СЗИ", "Отправка сообщения", msg);

```

```

        ok = ::send(Login, to, buf);
    }
    break;
case Logout:          //выход должностного лица из СОД;
    {
        msg = "Выход ДЛ из СОД (к " + to + ")";
        reg("СЗИ", "Отправка сообщения", msg);
        ok = ::send(Logout, to, buf);
    }
    break;
case NoAccess:        //
    {
        ok = ::send(NoAccess, to);
    }
    break;
return true;
}
bool Sp::importFile(QString from, QString path){
    if(from==currentName()){
        receiptFile(path, from);
        return true;
    }
    bool ok = ::importFile(from, path);
    //QString msg = path + " (от " + from + ")";
    //reg("СЗИ", "Импорт файла", msg);
    return ok;
}
bool Sp::exportFile(QString to, QString path){
    if(to==currentName()){
        return true;
    }
    bool ok = ::exportFile(to, path);
    //QString msg = path + " (к " + to + ")";
    //reg("СЗИ", "Экспорт файла", msg);
    return ok;
}

//

bool Sp::doReceipt(int type, QString from){
    bool ok = false;
    QString msg;
    QString buf;
    switch(type){
    case CheckIntegrity: //
        {
            msg = "Проверить целостность ПО (от " + from + ")";
            reg("СЗИ", "Прием сообщения", msg);

            send(OkAccess, from, buf);

            ok = checkIntegrity(from);
            if(ok) {
                send(OkIntegrity, from);
                usr()->armSetState(armStateWork);
            }
            else {
                send(BadIntegrity, from);
                usr()->armSetState(armStateBadIntegrity);
            }
        }
    }
}
*/

```

```

    }
    break;
case OkIntegrity:
{
    qDebug("receipt OkIntegrity from %s", from.ascii());

    msg = "Целостность ПО не нарушена (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateWork);
    QMessageBox::information(0,
                            "Получено сообщение",
                            "Целостность ПО на " + from + " не нарушена",
                            QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
    ok = true;
}
break;
case BadIntegrity: //сообщение о нарушении целостности;
{
    qDebug("receipt BadIntegrity from %s", from.ascii());

    msg = "Нарушена целостность ПО (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateBadIntegrity);
    QMessageBox::information(0,
                            "Получено сообщение",
                            "Нарушена целостность ПО на " + from ,
                            QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
    ok = true;
}
break;
case Test: //предписание на проведение тестирования функций СЗИ;
{
    msg = "Протестировать СЗИ (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    ok = test(from); //Ответы находятся в функции
/*
    if(ok) {
        usr()->armSetState(armStateWork);
        send(TestOk,from);
    }
    else {
        usr()->armSetState(armStateTestError);
        send(TestError, from);
    }
*/
}
break;
case TestOk: //сообщение о работоспособности СЗИ;
{
    qDebug("receipt TestOk from %s", from.ascii());

    msg = "СЗИ работоспособна (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateWork);

```

```

        QMessageBox::information(0,
                                "Получено сообщение",
                                "Тестирование СЗИ на " + from + " прошло
успешно.",
                                QMessageBox::Ok, QMessageBox::NoButton,
                                QMessageBox::NoButton );
        ok = true;
    }
    break;
case TestError:      //сообщение о неработоспособности СЗИ.
{
    qDebug("receipt TestError from %s", from.ascii());

    msg = "СЗИ не работоспособна (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateTestError);
    QMessageBox::information(0,
                            "Получено сообщение",
                            "Тестирование СЗИ на " + from + " прошло с
ошибками.",
                            QMessageBox::Ok, QMessageBox::NoButton,
                            QMessageBox::NoButton );
    ok = true;
}
break;
case Kill:           //экстренное стирание информации;
{
    qDebug("receipt Kill from %s", from.ascii());

    msg = "Экстренное стирание информации (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    send(OkAccess, from, buf);

    usr()->armSetState(armStateNoAccess);
    exportFile(from, STATPOST);

    ok = erase();
}
break;
case Recovery:       //предписание на проведение восстановления функций
СЗИ;
{
    qDebug("receipt Recovery from %s", from.ascii());

    msg = "Восстановить функции СЗИ (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    send(OkAccess, from, buf);

    ok = recovery(QString::null, QString::null, from); //Ответы находятся в
функции
/*
    if(ok){
        usr()->armSetState(armStateWork);
    }
    else{
        usr()->armSetState(armStateTestError);
    }
    exportFile(from, STATPOST);

```

```

        */
    }
    break;
case WhoseWP:          //запрос «Свой/чужой» для оператора;
{
    qDebug("receipt WhoseWP from %s", from.ascii());

    msg = "Запрос 'СВОЙ-ЧУЖОЙ' (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    send(OkAccess, from, buf);

    spLabel label( "Получен запрос 'Свой-Чужой' от " +from+ ".\n"
                   "Подтвердите свою подпись...");
    ok = login("", false, true);
    label.hide();
    if(ok){
        usr()->armSetState(armStateWork);
        send(Member, from);
    }
    else{
        usr()->armSetState(armStateWorkVisitor);
        send(Visitor, from);
    }
}
break;
case WhoseWPRM:        //запрос «Свой/чужой» для рабочего места;
{
    qDebug("receipt WhoseWPRM from %s", from.ascii());

    msg = "Запрос 'СВОЙ-ЧУЖОЙ' (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    send(OkAccess, from, buf);

    spLabel label( "Получен запрос 'Свой-Чужой' от " +from+ ".\n"
                   "Подтвердите свою подпись...");
    ok = login("", false, true);
    label.hide();
    if(ok){
        usr()->armSetState(armStateWork);
        send(Member, from);
    }
    else{
        usr()->armSetState(armStateWorkVisitor);
        send(Visitor, from);
    }
}
break;
case Member:           //ответ на запрос «Свой-чужой» √ свой;
{
    qDebug("receipt Member from %s", from.ascii());

    msg = "Ответ 'СВОЙ' (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateWork);
    QMessageBox::information(0,
                             "Получено сообщение",
                             "Получен сигнал 'СВОЙ' от " + from ,

```

```

        QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
    ok = true;
}
break;
case Visitor:      //ответ на запрос «Свой-чужой» √ чужой;
{
    qDebug("receipt Visitor from %s", from.ascii());

    msg = "Ответ 'ЧУЖОЙ' (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateWorkVisitor);
rpl:;
    int ans = QMessageBox::critical(0,
        "Получено сообщение",
        "Получен сигнал 'ЧУЖОЙ' от " + from ,
        "Разрешить работу",
        "Заблокировать АРМ",
        "Уничтожить ПО на АРМ" );

    switch(ans){
        case 1: //block
            ans=QMessageBox::information(0, ("Блокировка АРМ"),
                "Вы действительно хотите заблокировать
"+ from,
                QMessageBox::Ok, QMessageBox::Cancel);
            if (ans==QMessageBox::Ok){
                sp.send(BlockWP, from);
            }
            else goto rpl;
            break;
        case 2: //erase
            ans=QMessageBox::information(0, ("Экстренное стирание"),
                "Вы действительно хотите уничтожить ПО
на "+ from,
                QMessageBox::Ok, QMessageBox::Cancel);
            if (ans==QMessageBox::Ok){
                sp.send(Kill, from);
            }
            else goto rpl;
            break;
        default:
            ans=QMessageBox::information(0, ("Разрешить работу"),
                "Вы действительно хотите разрешить
работы "+ from,
                QMessageBox::Ok, QMessageBox::Cancel);
            if (ans!=QMessageBox::Ok)
                goto rpl;
            break;
    }
    ok = true;
}
break;
case UnblockConsole: //команда на разблокировку консоли оператора;
{
    if(!from.isEmpty()){
        msg = "Разблокировать консоль (от " + from + ")";
        reg("СЗИ", "Прием сообщения", msg);
    }
    ok = true;
}
}

```

```

        break;
    case BlockWP:          //команда на блокировку рабочего места;
    {
        if(!from.isEmpty()){
            msg = "Заблокировать АРМ (от " + from + ")";
            reg("СЗИ", "Прием сообщения", msg);

            send(OkAccess, from, buf);
        }
        if(users->armIsBlocked()){
            ok = true;
            spLabel label("АРМ заблокировано.\n"
                           "Введите подпись ОСЗИ...");
            ok = login("", true);
            if(ok)
            {
                label.hide();
                if(users->armIsBlocked())
                    ok = doReceipt(UnblockWP, from);
                QMessageBox::information(0,
                    "Разблокировка АРМ", "АРМ разблокировано",
                    QMessageBox::Ok, QMessageBox::NoButton,
                    QMessageBox::NoButton );
            }
        }
        else{
            ok = users->armSetBlocking(true);
            emit stateChangedSignal(from, armStateBlock);
            if(from == currentName())
                exportFile(usr()->parentOsziName(), STATPOST);
            else
                exportFile(from, STATPOST);

            if(ok){
                spLabel label("АРМ заблокировано.\n"
                               "Введите подпись ОСЗИ...");
                ok = login("", true);

                if(ok){
                    label.hide();
                    if(users->armIsBlocked())
                        ok = doReceipt(UnblockWP, from);
                    QMessageBox::information(0,
                        "Разблокировка АРМ",
                        "АРМ разблокировано",
                        QMessageBox::Ok, QMessageBox::NoButton,
                        QMessageBox::NoButton );
                }
            }
        }
        break;
    case UnblockWP:        //команда на разблокировку рабочего места;
    {
        qDebug("receipt UnblockWP from %s", from.ascii());
        if(!from.isEmpty()){
            msg = "Разблокировать АРМ (от " + from + ")";
            reg("СЗИ", "Прием сообщения", msg);

            send(OkAccess, from, buf);
        }
        if(!users->armIsBlocked()){
            ok = true;

```

```

        }
        else{
            ok = users->armSetBlocking(false);
            emit stateChangedSignal(from, armStateWork);
            emit exitWithTrueSignSignal();
            if(from == currentName())
                exportFile(usr()->parentOszName(), STATPOST);
            else
                exportFile(from, STATPOST);
        }
    }
    break;
case WorkOnControl: //сообщение о работе под контролем;
{
    msg = "Работа под контролем (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateWorkOnControl);
    //arr()->armSetState(armStateWorkOnControl);
    //exportFile(from, STATPOST);

rpl2::
    int ans = QMessageBox::critical(0,
                                    "Получено сообщение",
                                    "Получен сигнал 'Работа под контролем' от " +
from ,
                                    "Разрешить работу",
                                    "Заблокировать АРМ",
                                    "Уничтожить ПО на АРМ" );
    switch(ans){
        case 1: //block
            ans=QMessageBox::information(0, ("Блокировка АРМ"),
                                        "Вы действительно хотите заблокировать
"+ from,
                                        QMessageBox::Ok, QMessageBox::Cancel);
            if (ans==QMessageBox::Ok){
                sp.send(BlockWP, from);
            }
            else goto rpl2;
            break;
        case 2: //erase
            ans=QMessageBox::information(0, ("Экстренное стирание"),
                                        "Вы действительно хотите уничтожить ПО
на "+ from,
                                        QMessageBox::Ok, QMessageBox::Cancel);
            if (ans==QMessageBox::Ok){
                sp.send(Kill, from);
            }
            else goto rpl2;
            break;
        default:
            ans=QMessageBox::information(0, ("Разрешить работу"),
                                        "Вы действительно хотите разрешить
работы "+ from,
                                        QMessageBox::Ok, QMessageBox::Cancel);
            if (ans!=QMessageBox::Ok)
                goto rpl2;
            break;
    }
    ok = true;
}

```

```

    }
    break;
case Login:          //вход должностного лица в СОД;
{
    msg = "Вход ДЛ в СОД (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateWork);
    ok = true;
}
break;
case Logout:        //выход должностного лица из СОД;
{
    qDebug("receipt Logout from %s", from.ascii());

    msg = "Выход ДЛ из СОД (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateNoAccess);
    ok = true;
}
break;
case NoAccess:      //
{
    qDebug("receipt NoAccess from %s", from.ascii());

    msg = "Нет доступа (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateNoAccess);
    ok = true;
}
break;
case OkAccess:      //квитанция ✓ ответ на запрос;
{
    if(from==currentName()) { ok = true; break; }

    qDebug("receipt OkAccess from %s", from.ascii());

    msg = "Квитанция (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateWork);
    ok = true;
}
break;
case NotAuthDeliv:  //несанкционированная выдача команд
{
    msg = "Несанкционированная выдача команд (от " + from + ")";
    reg("СЗИ", "Прием сообщения", msg);

    emit stateChangedSignal(from, armStateWork);
    ok = true;
}
break;
case AttNotAuthDeliv://попытка несанкционированной выдачи команд
{
    msg = "Попытка несанкционированной выдачи команд (от " + from + ")";

```

```

        reg("СЗИ", "Прием сообщения", msg);

        emit stateChangedSignal(from, armStateWork);
        ok = true;
    }
    break;
}
return ok;
}
bool Sp::doReceiptFile(QString path, QString from){
    bool ok = false;
    QString msg;
    if(path == STATPOST){
        QString post;
        int state = armStateUnknown;
        bool st = usr()->armGetState(post, state);
        usr()->armSetState(); //Восстанавливаем свое состояние в
        файле
        if(st){
#ifdef TESTITSELF
            if(post==from)
#endif
            {
                QString st;
                switch(state){
                    case armStateWork:
                        st = "Работа";
                        break;
                    case armStateWorkOnControl:
                        st = "Работа под контролем";
                        break;
                    case armStateWorkVisitor:
                        st = "Работа (ЧУЖОЙ)";
                        break;
                    case armStateTestError:
                        st = "СЗИ неработоспособна";
                        break;
                    case armStateBadIntegrity:
                        st = "Нарушена целостность ПО";
                        break;
                    case armStateBlock:
                        st = "Заблокировано";
                        break;
                    case armStateNoAccess:
                        st = "Нет доступа";
                        break;
                    default:
                        st = "Не известно";
                        break;
                }

                msg = "Состояние " + from + ":" + st;
                reg("СЗИ", "Прием сообщения", msg);
                if(isStateChangedSignalConnected)
                    emit stateChangedSignal(post, state);
                else{
                    if(state == armStateBlock){
                        QMessageBox::information(0,
                            "СЗИ:Прием сообщения",
                            msg,

```

```

        QMessageBox::Ok, QMessageBox::NoButton,
        QMessageBox::NoButton );
    }
}
    }
    ok = true;
}
}
if(path == TSD_FILE_NAME){
    msg = "Получена новая ТСД от " + from + ".Перезагрузка.";
    reg("СЗИ", "Прием сообщения", msg);
    tsdChanged();
    QMessageBox::information(0,
        "Перезагрузка...",
        "Получена новая ТСД от " + from + ".\n"
        "АРМ будет перезагружено.",
        QMessageBox::Ok, QMessageBox::NoButton, QMessageBox::NoButton );
}

#if 1
    system("sudo killall elk-wm");
#else
    qApp->quit();
#endif

    ok = true;
}
return ok;
}

bool Sp::backup(QString conf, QString tgz){
    bool ok = false;
    reg("СЗИ", "Резервное копирование", "Вход");

    if(conf.isEmpty())
        conf = BACKUP_CONF;
    if(tgz.isEmpty())
        tgz = BACKUP_FILE;

    QStringList command;

    QFileInfo fi( conf );
    if ( fi.exists() ) {
        if(!currentIsRoot())
            command << "sudo";
        command << "tar" << "-czf" << tgz << "-T" << conf << "-P";

        spLabel label("Идет резервное копирование.\n Подождите..." );
        qApp->processEvents();

        ProcessManager pr(command);
        while(pr.isRunning())
            qApp->processEvents();
        label.hide();
        ok = true;
        QMessageBox::information(0, "Состояние", "Резервное копирование
завершено");
        reg("СЗИ", "Резервное копирование", "Завершено");
    }
}

```

```

    }
    else
    {
        QMessageBox::critical(0, "Резервное копирование/восстановление", "Ошибка
открытия файла конфигурации");
        reg("СЗИ", "Резервное копирование", "Ошибка открытия файла
конфигурации");
    }
    return ok;
}

bool Sp::restore(QString conf, QString tgz){
    bool ok = false;
    reg("СЗИ", "Резервное восстановление", "Вход");
    if(conf.isEmpty())
        conf = BACKUP_CONF;
    if(tgz.isEmpty())
        tgz = BACKUP_FILE;

    QFileInfo fi(tgz);
    if(fi.exists()){
        QStringList command;
        QFileInfo fc( conf );
        if ( fc.exists() ) {
            if(!currentIsRoot())
                command << "sudo";
            command << "tar" << "-xzf" << tgz << "-T" << conf << "-P";

            spLabel label("Идет восстановление из резервных копий.\n Подождите..."
);
            qApp->processEvents();

            ProcessManager pr(command);
            while(pr.isRunning())
                qApp->processEvents();
            label.hide();
            ok = true;
            QMessageBox::information(0, "Состояние", "Восстановление файлов
завершено");
            reg("СЗИ", "Резервное восстановление", "Завершено");
        }
        else {
            QMessageBox::critical(0, "Резервное копирование/восстановление",
"Ошибка открытия файла конфигурации");
            reg("СЗИ", "Резервное восстановление", "Ошибка открытия файла
конфигурации");
        }
    }
    else {
        QMessageBox::critical(0, "Резервное копирование/восстановление",
"Ошибка открытия файла архива");
        reg("СЗИ", "Резервное восстановление", "Ошибка открытия файла архива");
    }
    return ok;
}

bool Sp::recovery(QString conf, QString tgz, QString sender){
    bool ok = false;

```

```

reg("СЗИ", "Восстановление СЗИ", "Вход");
if(conf.isEmpty())
    conf = RECOVERY_CONF;
if(tgz.isEmpty())
    tgz = RECOVERY_FILE;

//-----
QFileInfo fi(tgz);
if(fi.exists()){
    QStringList command;
    QFileInfo fc( conf );
    if ( fc.exists() ) {
        if(!currentIsRoot())
            command << "sudo";
        command << "tar" << "-xzf" << tgz << "-T" << conf << "-P";

        spLabel label("Идет восстановление СЗИ.\n Подождите..." );
        qApp->processEvents();

        ProcessManager pr(command);
        while(pr.isRunning())
            qApp->processEvents();
        label.hide();
        ok = true;

        usr()->armSetState(armStateWork);
        if(!sender.isEmpty())
            exportFile(sender, STATPOST);
        QMessageBox::information(0, "Состояние", "Восстановление СЗИ
завершено");
        reg("СЗИ", "Восстановление СЗИ", "Завершено");
    }
    else {
        usr()->armSetState(armStateTestError);
        if(!sender.isEmpty())
            exportFile(sender, STATPOST);

        QMessageBox::critical(0, "Восстановление СЗИ", "Ошибка открытия файла
конфигурации");
        reg("СЗИ", "Восстановление СЗИ", "Ошибка открытия файла
конфигурации");
    }
}
else {
    usr()->armSetState(armStateTestError);
    if(!sender.isEmpty())
        exportFile(sender, STATPOST);
    QMessageBox::critical(0, "Восстановление СЗИ", "Ошибка открытия файла
архива");
    reg("СЗИ", "Восстановление СЗИ", "Ошибка открытия файла архива");
}
return ok;
}

void Sp::workOnControlSlot(){
    qDebug("workOnControlSlot");
    send(WorkOnControl, usr()->parentOszName());
}

```

```

void Sp::logoutSlot(){
    if(!localLoop){
        killTimer(idTimer);
        reg("СЗИ", "Выход из СОД", "Выполнен");
        send(Login, usr()->parentOsziName());
    }
}

bool Application::notify( QObject *o, QEvent *e ){
    if(e->type()==QEvent::KeyRelease){
        QKeyEvent* ke = (QKeyEvent*) e;
        if ( ( ke->state() & Qt::ControlButton ) && (ke->key() ==
Qt::Key_Period || ke->key() == Qt::Key_Slash) ){
            emit workOnControlSignal();
        }
    }
    return QApplication::notify(o, e);
}

static QString strTemp;
#define DEFAULT_REPEAT 1

/*
static void overwrite(const char *szFileName, int dRepCo, int dFileSize, int
dVerb)
{
#define BUFFSZI 256
#define BUFFSZ BUFFSZI*sizeof(int)
    int buff[BUFFSZI];
    FILE *ferase;
    int i, rep, dBlocks, dRest;

    for(rep=1; rep<=dRepCo; rep++)
    {
        for(i=0; i<BUFFSZI; i++)
            buff[i]=rand();
        if((ferase=fopen(szFileName, "r+b"))!=NULL)
        {
            dBlocks = dFileSize / BUFFSZ;
            dRest = dFileSize % BUFFSZ;
            for(i=0; i<dBlocks; i++)
                fwrite(buff, BUFFSZ, 1, ferase);
            if(dRest>0)
                fwrite(buff, dRest, 1, ferase);
            fclose(ferase);
        }
        sync();
    }
}
*/
void RMDIR(QString dir)
{
    QString cmd;
    if (!sp.currentIsRoot())
    {
        cmd= "sudo rmdir "+dir;
    }
    else
    {
        cmd= "rmdir "+dir;
    }
}

```

```

system (cmd);
}

void UNLINK (QString file_name)
{
    QString cmd;
    if (!sp.currentIsRoot())
    {
        cmd= "sudo unlink "+file_name;
    }
    else
    {
        cmd= "unlink "+file_name;
    }
    system (cmd);
}

void Sp::overwrite(const char *szFileName, int dRepCo, int dFileSize, int
/*dVerb*/)
{
#define BUFFSZ 256
#define BUFFSZ BUFFSZ*sizeof(int)
    int buff[BUFFSZ];
    FILE *ferase;
    int i, rep, dBlocks, dRest;
    strTemp=szFileName;
    qApp->processEvents();
    for(rep=1; rep<=dRepCo; rep++)
    {
        for(i=0; i<BUFFSZ; i++)
            buff[i]=rand();

        commonAccess(szFileName);
        if((ferase=fopen(szFileName, "r+b"))!=NULL)
        {
            dBlocks = dFileSize / BUFFSZ;
            dRest = dFileSize % BUFFSZ;
            for(i=0; i<dBlocks; i++)
                fwrite(buff, BUFFSZ, 1, ferase);
            if(dRest>0)
                fwrite(buff, dRest, 1, ferase);
            fclose(ferase);
        }
        else
        {
            {
                sync();
            }
        }
    }
}

static void remrec(char *szMask, char *szPath, int ow, int dRepeat, int
bSubdir, int dVerb)
{
    DIR *pdir;
    struct dirent *pdirent;

    sp.commonAccess(szPath);
    if((pdir=opendir(szPath))!=NULL)
    {

```

```

while((pdirent=readdir(pdir))!=NULL)
{
    struct stat st;
    char szFN[256];
    memset(szFN, 0, sizeof(szFN));
    strcpy(szFN, szPath);
    strcat(szFN, "/");
    strcat(szFN, pdirent->d_name);
    if(dVerb>1)
    printf("  %s\n", pdirent->d_name);
    QString Temp=(char*)strchr(szFN, '.');

    if(!stat(szFN, &st))
    {
        if(szMask==Temp)
        {
            strTemp+=szFN;
            sp.overwrite(szFN, dRepeat, st.st_size, dVerb);
            UNLINK(szFN);
        }
        else
        {
            if(bSubdir || !fnmatch(szMask, pdirent->d_name, 0))
            {
                if(st.st_mode & S_IFDIR)
                {
                    if(strcmp(pdirent->d_name, ".") && strcmp(pdirent->d_name,
".."))
                    {
                        remrec(szMask, szFN, ow, dRepeat, 1, dVerb);
                        RMDIR(szFN);
                    }
                }
                else
                {
                    if(ow)
                    sp.overwrite(szFN, dRepeat, st.st_size, dVerb);
                    UNLINK(szFN);
                }
            }
        }
        else {};//perr("can't stat!", szFN);
    }
    closedir(pdir);
}
else { strTemp=szPath;
    QString msg = "Ошибка открытия каталога"+strTemp ;
    QMessageBox::critical(0, "Экстренное удаление", msg );
    sp.reg("СЗИ", "Экстренное удаление", msg);
}
}

bool Sp::erase(QString FileName){
    bool ok = false;
    reg("СЗИ", "Экстренное удаление", "Вход");
    if(FileName.isEmpty())
    FileName = ERASE_CONF;
    spLabel label("Идёт экстренное удаление...");
    QApplication->processEvents();
    char szBuff[2048];

```

```

char szPath[2048];
int dRepeat=DEFAULT_REPEAT, dRep=dRepeat;
int mode=0;
int dVerb=0;
FILE *fconf;

memset(szBuff, 0, sizeof(szBuff));
memset(szPath, 0, sizeof(szPath));

if((fconf=fopen(FileName, "rt"))!=NULL)
{
    while(fgets(szBuff, sizeof(szBuff), fconf))
    {
        qApp->processEvents();
        int lb=strlen(szBuff)-1;
        char *szTmp=(char *)strchr(szBuff, '#');
        if(szTmp) szTmp[0]=0;
        else if (szBuff[lb] =='\n') szBuff[lb]=0;
        if(szBuff[0])
        {
            if(!strcmp("[erase]", szBuff))
                mode=0;
            else
                if(!strncmp("[overwrite", szBuff, 10))
                {
                    mode=1;
                    szBuff[lb-1]=0;
                    if(szBuff[10]=='=')
                        dRepeat=atoi(&szBuff[11]); else dRepeat=dRep;
                }
            else
            {
                char *szMask=(char *)strrchr(szBuff, '/');
                char *szMask_type=(char*)strrchr(szBuff, '*');

                if(szMask_type)
                {
                    szMask[0]=0;
                    szMask++;
                    szMask_type[0]=0;
                    szMask_type++;
                    if(szMask_type[0])
                    { //udaleni po maske *.rasshirenie
                        remrec(szMask_type, szBuff, mode, dRepeat, 0, dVerb);
                    }
                    else
                    {
                        //udalenie po maske *
                        *szMask='*';
                        strcpy(szPath, szBuff);
                        remrec(szMask, szPath, mode, dRepeat, 0, dVerb);
                    }
                }
            }
        }
        else
        {
            if(szMask)
            {
                szMask[0]=0;
            }
        }
    }
}

```

```

        szMask++;

        if(!szMask[0])
        {
            //если fail не указан удаляется все содержимое папки и
sama papka
            remrec(szMask, szBuff, mode, dRepeat, 1, dVerb);
            RMDIR(szBuff);

        }
        else
        { //удаление по маске

            strcpy(szPath, szBuff);
            remrec(szMask, szPath, mode, dRepeat, 0, dVerb);
        }
    }
    else
    {

        szMask=szBuff;
        remrec(szMask, szPath, mode, dRepeat, 0, dVerb);
    }
}
}
}
label.hide();
fclose(fconf);
system ("sudo /sbin/poweroff");
}
else
{
    QMessageBox::critical(0, "Экстренное удаление", "Ошибка открытия файла
конфигурации");
    reg("СЗИ", "Экстренное удаление", "Ошибка открытия файла конфигурации");
}
return ok;
}

QString Sp::sign(const QString& name){
    return usr()->sign(name);
}
QString Sp::post(const QString& sign){
    return usr()->post(sign);
}
uint Sp::prof(const QString& name){
    return usr()->prof(name);
}
int Sp::order(const QString& name){
    return usr()->order(name);
}
QString Sp::currentName(){
    return usr()->currentName();
}

QString Sp::currentSign(){
    return usr()->currentSign();
}

uint Sp::currentProf(){

```

```

    return usr()->currentProf();
}

bool Sp::currentIsRoot(){
    return usr()->currentIsRoot();
}

bool Sp::can(uint p){ //true, если все флаги разрешены текущему ДЛ
    uint cp = currentProf();
    return (cp & p) == p;
}

bool Sp::can2(uint p){ //true, если хотя бы один флаг разрешен текущему ДЛ
    uint cp = currentProf();
    return (cp & p) != 0;
}

bool Sp::can(QString name, uint p){//true, если все флаги разрешены
указанному ДЛ
    uint cp = prof(name);
    return (cp & p) == p;
}

bool Sp::can2(QString name, uint p){//true, если хотя бы один флаг разрешен
указанному ДЛ
    uint cp = prof(name);
    return (cp & p) != 0;
}

bool Sp::start(const char* module, const char* funcName, uint p){ //true,
если все флаги разрешены текущему ДЛ
    bool ans = can(p);
    if(ans){
        reg(module, funcName, "Вход");
    }
    else{
        QString msg = "Не достаточно полномочий для запуска";
        reg(module, funcName, msg);
        QMessageBox::critical(0,
                                "SZI",
                                msg + "\n" + module + ":" + funcName,
                                QMessageBox::Ok, QMessageBox::NoButton,
                                QMessageBox::NoButton );
    }
    return ans;
}

bool Sp::start2(const char* module, const char* funcName, uint p){//true,
если хотя бы один флаг разрешен текущему ДЛ
    bool ans = can2(p);
    if(ans){
        reg(module, funcName, "Вход");
    }
    else{
        QString msg = "Не достаточно полномочий для запуска";
        reg(module, funcName, msg);
        QMessageBox::critical(0,
                                "SZI",
                                msg + "\n" + module + ":" + funcName,
                                QMessageBox::Ok, QMessageBox::NoButton,
                                QMessageBox::NoButton );
    }
    return ans;
}

```

```

}

void Sp::finish(const char* module, const char* funcName, const char*
exitMsg){
    QString msg = exitMsg? exitMsg : "Нормальное завершение";
    reg(module, funcName, msg);
}

bool Sp::receipt(int type, const QString& from){
    QString fr = from.simplifyWhiteSpace();
    if(type == UnblockWP)
        doReceipt(type, fr);
    else
        msgQue.enqueue(new spMsg(type, fr));
    return true;
}

bool Sp::receiptFile(const QString& path, const QString& from){
    QString fr = from.simplifyWhiteSpace();
    msgQue.enqueue(new spMsg(path, fr));
    return true;
}

void Sp::timerEvent( QTimerEvent* /*e*/){
    while(!msgQue.isEmpty()){
        spMsg* msg = msgQue.dequeue();
        if(msg->type == File)
            doReceiptFile(msg->path, msg->from);
        else
            doReceipt(msg->type, msg->from);

        delete msg;
    }
}

bool Sp::armSetBlocking(bool bl){
    return usr()->armSetBlocking(bl);
}

bool Sp::armIsBlocked(){
    return usr()->armIsBlocked();
}

const Q3Dict<spTsdRecord>& Sp::tsd(){
    return usr()->tsd;
}

const QList<spTsdRecord>& Sp::usersList(){
    return usr()->users();
}

void Sp::usersControl(QList<spTsdRecord>& l){
    if(!cf){
        cf = new spUserControlForm();
    }
    cf->tsdToTable(l);
}

void Sp::raiseUsersControl(){
    if(!cf){
        cf = new spUserControlForm();
    }
    cf->hide();
    cf->show();
    while(!cf->isHidden())
        qApp->processEvents();
}

```

```

void Sp::blockControl() {
    menu->slotBlock();
}
void Sp::unblockControl() {
    menu->slotUnblock();
}
bool Sp::latin() {
#ifdef LATIN_DL
    return true;
#else
    return false;
#endif
}

bool Sp::findInTva(const QString& post) {
    char bf[17];
    strcpy(bf, post.ascii());
    for(int i=strlen(bf); i<16; i++)          bf[i] = ' ';
    bf[16] = 0;
    return ::findInTva(bf);
}
void Sp::tsdChanged() {
    ::tsdChanged();
}
bool Sp::generateTsd(bool erase) {
    return ::generateTsd(erase);
}
void Sp::activateLogout() {
    localLoop = false;
}
void Sp::sendLogout() {
    localLoop = false;
    logoutSlot();
}
bool Sp::commonAccess(const char* path) {
    if(!QFile::exists(path)) return -1;
    QString cmd = "sudo chmod a+rw ";
    cmd += path;
    return system(cmd.ascii());
}

bool receiptFile(QString path, QString from) {
    return sp.receiptFile(path, from);
}
bool receipt(int type, QString from) {
    return sp.receipt(type, from);
}

```

3.1.3.2 sp_blocking_form - форма, обеспечивающая блокировку и разблокировку АРМ. Исходный код представлен ниже.

```

#include "sp_blocking_form.h"

#include <qvariant.h>
#include <qbuttongroup.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include <q3table.h>

```

```

#include <qtoolbutton.h>
#include <qlayout.h>
#include <qtooltip.h>
#include <qwhatsthis.h>
#include <qimage.h>
#include <qpixmap.h>
#include <qmessagebox.h>
#include <q3accel.h>

#include "sp_.h"
#include "sp_tsd.h"
#include "sp_users.h"
#include "../sp_extern.h"

spBlockingForm::spBlockingForm( uint bl_unbl /*= 3*/, QWidget* parent,
const char* name, bool modal, Qt::WindowFlags fl )
    : QDialog( parent, name, modal, fl )
{
    QPixmap image0( ( const char** ) image0_data );
    if ( !name )
        setName( "spBlockingForm" );
    resize( 285, 320 );
    setCaption( ( "СЗИ:Блокировка/Разблокировка АРМ" ) );
    spBlockingFormLayout = new QGridLayout( this, 1, 1, 11, 6,
"spBlockingFormLayout");

    TextLabel1 = new QLabel( this, "TextLabel1" );
    TextLabel1->setText( ( "Контролируемые ДЛ" ) );

    spBlockingFormLayout->addWidget( TextLabel1, 0, 0 );

    tblUsers = new Q3Table( this, "tblUsers" );
    tblUsers->setNumCols( tblUsers->numCols() + 1 );    tblUsers-
>horizontalHeader()->setLabel( tblUsers->numCols() - 1, ( "Имя ДЛ" ) );
    tblUsers->setNumCols( tblUsers->numCols() + 1 );    tblUsers-
>horizontalHeader()->setLabel( tblUsers->numCols() - 1, ( "Состояние
АРМ" ) );
    tblUsers->setNumCols( 2 );
    tblUsers->setSelectionMode( Q3Table::SingleRow );
    tblUsers->setReadOnly( true );
    tblUsers->setFocusStyle( Q3Table::FollowStyle );
    tblUsers->setMinimumSize( QSize( 290, 0 ) );
    tblUsers->setColumnStretchable( 0, true );
    tblUsers->setColumnStretchable( 1, true );

    spBlockingFormLayout->addMultiCellWidget( tblUsers, 1, 1, 0, 2 );
    QSpacerItem* spacer = new QSpacerItem( 60, 0, QSizePolicy::Expanding,
QSizePolicy::Minimum );
    spBlockingFormLayout->addItem( spacer, 0, 1 );

    btnRefresh = new QToolButton( this, "btnRefresh" );
    btnRefresh->setText( ( "" ) );
    btnRefresh->setPixmap( image0 );
    QToolTip::add( btnRefresh, ( "Запрос состояния выбранного ДЛ" ) );

    spBlockingFormLayout->addWidget( btnRefresh, 0, 2 );

    ButtonGroup1 = new Q3ButtonGroup( this, "ButtonGroup1" );
    ButtonGroup1->setTitle( ( "" ) );
    ButtonGroup1->setColumnLayout( 0, Qt::Vertical );

```

```

ButtonGroup1->layout()->setSpacing( 6 );
ButtonGroup1->layout()->setMargin( 11 );
ButtonGroup1Layout = new QHBoxLayout( ButtonGroup1->layout() );
ButtonGroup1Layout->setAlignment( Qt::AlignTop );

if(bl_unbl & 2){
    btnBlock = new QPushButton( ButtonGroup1, "btnBlock" );
    btnBlock->setMinimumSize( QSize( 0, 0 ) );
    btnBlock->setText( ( "Заблокировать" ) );
    ButtonGroup1Layout->addWidget( btnBlock );
}
if(bl_unbl & 1){
    btnUnblock = new QPushButton( ButtonGroup1, "btnUnblock" );
    btnUnblock->setText( ( "Разблокировать" ) );
    ButtonGroup1Layout->addWidget( btnUnblock );
}
spBlockingFormLayout->addMultiCellWidget( ButtonGroup1, 2, 2, 0, 2 );

ButtonGroup2 = new Q3ButtonGroup( this, "ButtonGroup2" );
ButtonGroup2->setFrameShape( Q3ButtonGroup::NoFrame );
ButtonGroup2->setTitle( ( "" ) );
ButtonGroup2->setColumnLayout(0, Qt::Vertical );
ButtonGroup2->layout()->setSpacing( 6 );
ButtonGroup2->layout()->setMargin( 11 );
ButtonGroup2Layout = new QHBoxLayout( ButtonGroup2->layout() );
ButtonGroup2Layout->setAlignment( Qt::AlignTop );
QSpacerItem* spacer_2 = new QSpacerItem( 90, 0, QSizePolicy::Expanding,
QSizePolicy::Minimum );
ButtonGroup2Layout->addItem( spacer_2 );

btnExit = new QPushButton( ButtonGroup2, "btnExit" );
btnExit->setText( ( "Выход" ) );
ButtonGroup2Layout->addWidget( btnExit );

spBlockingFormLayout->addMultiCellWidget( ButtonGroup2, 3, 3, 0, 2 );

// signals and slots connections
connect( btnExit, SIGNAL( clicked() ), this, SLOT( reject() ) );
connect( btnRefresh, SIGNAL( clicked() ), this, SLOT( refreshSlot() ) );
if(bl_unbl & 2){
    connect( btnBlock, SIGNAL( clicked() ), this, SLOT( blockSlot() ) );
}
if(bl_unbl & 1){
    connect( btnUnblock, SIGNAL( clicked() ), this,
SLOT( unblockSlot() ) );
}

connect( &sp, SIGNAL(stateChangedSignal(const QString&, int) ), this,
SLOT( armRefreshStateSlot(const QString&, int) ) );
sp.isStateChangedSignalConnected = true;
init();

Q3Accel *a = new Q3Accel ( this );
a->connectItem ( a->insertItem ( Qt::Key_F1 ), this, SLOT
( slotHelp() ) );
}

spBlockingForm::~spBlockingForm()
{
    sp.isStateChangedSignalConnected = false;
}

```

```

void spBlockingForm::slotHelp ()
{
}

void spBlockingForm::blockSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Блокировка АРМ"),
                                     "Вы действительно хотите заблокировать
"+ post,
                                     QMessageBox::Ok, QMessageBox::Cancel);
    if (exit==QMessageBox::Ok){
        sp.send(BlockWP, post);
    }
}

void spBlockingForm::unblockSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Разблокировка АРМ"),
                                     "Вы действительно хотите разблокировать
"+ post,
                                     QMessageBox::Ok, QMessageBox::Cancel);
    if (exit==QMessageBox::Ok){
        sp.send(UnblockWP, post);
    }
}

void spBlockingForm::init(){
    const QList<spTsdRecord>& l = sp.usr()->inferiors();
    int n = l.count();
    int rw = 0;
    for(int i=0; i<n; i++){
        const QString& post = l[i].post;
        if(sp.findInTva(post)){
            tblUsers->setNumRows(rw+1);
            tblUsers->setText(rw, 0, post);
            tblUsers->setText(rw, 1, "Не известно");
            rw++;
        }
    }
    tblUsers->adjustColumn(1);
}

void spBlockingForm::refreshSlot()
{
    int rw = tblUsers->currentRow();
    tblUsers->setText(rw, 1, "Не известно");
    QString post = tblUsers->text(rw, 0);
    sp.importFile(post, STATPOST);
}

void spBlockingForm::armRefreshStateSlot(const QString& post, int state){
    int rows = tblUsers->numRows();
    for(int i=0; i<rows; i++){
        if(tblUsers->text(i, 0) == post){
            QString st;
            switch(state){
                case armStateWork:

```

```

        st = "Работа";
        break;
    case armStateWorkOnControl:
        st = "Работа под контролем";
        break;
    case armStateWorkVisitor:
        st = "Работа (ЧУЖОЙ)";
        break;
    case armStateTestError:
        st = "СЗИ неработоспособна";
        break;
    case armStateBadIntegrity:
        st = "Нарушена целостность ПО";
        break;
    case armStateBlock:
        st = "Заблокировано";
        break;
    case armStateNoAccess:
        st = "Нет доступа";
        break;
    //case armStateUnknown:
    default:
        st = "Не известно";
        break;
    }
    tblUsers->setText(i, 1, st);
    tblUsers->adjustColumn(1);
    break;
}
}
}

```

3.1.3.7 sp_menu - динамическое меню функций ОБЗИ в зависимости от полномочий ДЛ. Исходный код представлен ниже.

```

#include "sp_menu.h"
#include <q3popupmenu.h>
#include <qmenubar.h>
#include <qaction.h>
#include <q3accel.h>
#include <qmessagebox.h>
#include <qfileinfo.h>
#include <QTextStream>

#include "sp_usercontrol_form.h"
#include "sp_.h"
#include "sp_register_form.h"
#include "sp_blocking_form.h"
#include "sp_obi_form.h"
#include "sp_profile_form.h"

Q3PopupMenu* spMenu::menu(QWidget* w){

    aProfile = new Q3Action("Полномочия", "Полномочия", 0, w);
    aProfile->setStatusTip("Полномочия действующего ДЛ");
    aProfile->setWhatsThis("Полномочия действующего ДЛ");
    connect( aProfile, SIGNAL(activated()), this, SLOT(slotProfile()) );

```

```

aControlUsers = new QAction("Управление пользователями", "Управление
пользователями", 0, w);
aControlUsers->setStatusTip("Управление пользователями");
aControlUsers->setWhatsThis("Управление пользователями");
connect( aControlUsers, SIGNAL(activated()), this,
SLOT(slotControlUsers()) );

aAOBI = new QAction("APM ОБИ", "APM ОБИ", 0, w);
aAOBI->setStatusTip("APM ОБИ");
aAOBI->setWhatsThis("APM ОБИ");
connect( aAOBI, SIGNAL(activated()), this, SLOT(slotAOBI()) );

aBlock = new QAction("Блокировка/Разблокировка
PM", "Блокировка/Разблокировка PM", 0, w);
aBlock->setStatusTip("Блокировка/Разблокировка PM");
aBlock->setWhatsThis("Блокировка/Разблокировка PM");
connect( aBlock, SIGNAL(activated()), this, SLOT(slotBlockUnblock()) );

aEvents = new QAction("Просмотр событий безопасности", "Просмотр событий
безопасности", 0, w);
aEvents->setStatusTip("Просмотр событий безопасности");
aEvents->setWhatsThis("Просмотр событий безопасности");
connect( aEvents, SIGNAL(activated()), this, SLOT(slotEvents()) );

//-----
aTunningMandatLevels = new QAction("Настройка мандатных
уровней", "Настройка мандатных уровней", 0, w);
aTunningMandatLevels->setStatusTip("Настройка мандатных уровней");
aTunningMandatLevels->setWhatsThis("Настройка мандатных уровней");
connect( aTunningMandatLevels, SIGNAL(activated()), this,
SLOT(slotTunningMandatLevels()) );

aTunningAudit = new QAction("Настройка политики аудита", "Настройка
политики аудита", 0, w);
aTunningAudit->setStatusTip("Настройка политики аудита");
aTunningAudit->setWhatsThis("Настройка политики аудита");
connect( aTunningAudit, SIGNAL(activated()), this,
SLOT(slotTunningAudit()) );

aAudit = new QAction("Просмотр сщбытий аудита", "Просмотр сщбытий аудита",
0, w);
aAudit->setStatusTip("Просмотр сщбытий аудита");
aAudit->setWhatsThis("Просмотр сщбытий аудита");
connect( aAudit, SIGNAL(activated()), this, SLOT(slotAudit()) );

pCntrMCBC = new Q3PopupMenu();
aTunningMandatLevels->addTo(pCntrMCBC);
aTunningAudit->addTo(pCntrMCBC);
aAudit->addTo(pCntrMCBC);

//-----
aCdMount = new QAction("Подключить", "Подключить CD-ROM", 0, w);
aCdMount->setStatusTip("Подключить CD-ROM");
aCdMount->setWhatsThis("Подключить CD-ROM");
connect( aCdMount, SIGNAL(activated()), this, SLOT(slotCdMount()) );

aCdUnmount = new QAction("Отключить", "Отключить CD-ROM", 0, w);
aCdUnmount->setStatusTip("Отключить CD-ROM");
aCdUnmount->setWhatsThis("Отключить CD-ROM");
connect( aCdUnmount, SIGNAL(activated()), this, SLOT(slotCdUnmount()) );

```

```

pCd = new Q3PopupMenu();
aCdMount->addTo(pCd);
aCdUnmount->addTo(pCd);
connect( pCd, SIGNAL(aboutToShow()), this, SLOT(slotCd()) );

aFlashMount = new Q3Action("Подключить","Подключить УЗПИ", 0, w);
aFlashMount->setStatusTip("Подключить УЗПИ");
aFlashMount->setWhatsThis("Подключить УЗПИ");
connect( aFlashMount, SIGNAL(activated()), this, SLOT(slotFlashMount()) );

aFlashUnmount = new Q3Action("Отключить","Отключить УЗПИ", 0, w);
aFlashUnmount->setStatusTip("Отключить УЗПИ");
aFlashUnmount->setWhatsThis("Отключить УЗПИ");
connect( aFlashUnmount, SIGNAL(activated()), this,
SLOT(slotFlashUnmount()) );

pFlash = new Q3PopupMenu();
aFlashMount->addTo(pFlash);
aFlashUnmount->addTo(pFlash);
connect( pFlash, SIGNAL(aboutToShow()), this, SLOT(slotFlash()) );

//-----
aIntegrity = new Q3Action("Контроль целостности", "Контроль целостности",
0, w);
aIntegrity->setStatusTip("Контроль целостности");
aIntegrity->setWhatsThis("Контроль целостности");
connect( aIntegrity, SIGNAL(activated()), this, SLOT(slotIntegrity()) );

aTest = new Q3Action("Тестирование СЗИ","Тестирование СЗИ", 0, w);
aTest->setStatusTip("Тестирование СЗИ");
aTest->setWhatsThis("Тестирование СЗИ");
connect( aTest, SIGNAL(activated()), this, SLOT(slotTest()) );

aRecovery = new Q3Action("Восстановление СЗИ","Восстановление СЗИ", 0, w);
aRecovery->setStatusTip("Восстановление СЗИ");
aRecovery->setWhatsThis("Восстановление СЗИ");
connect( aRecovery, SIGNAL(activated()), this, SLOT(slotRecovery()) );

//-----

aArchive = new Q3Action("Резервное копирование","Резервное
копирование", 0, w);
aArchive->setStatusTip("Резервное копирование");
aArchive->setWhatsThis("Резервное копирование");
connect( aArchive, SIGNAL(activated()), this, SLOT(slotArchive()) );

aUnArchive = new Q3Action("Восстановление из резервных
копий","Восстановление из резервных копий", 0, w);
aUnArchive->setStatusTip("Восстановление из резервных копий");
aUnArchive->setWhatsThis("Восстановление из резервных копий");
connect( aUnArchive, SIGNAL(activated()), this, SLOT(slotUnArchive()) );

pCommon = new Q3PopupMenu();

aProfile->addTo(pCommon);

    if(sp.can2(CP_OSZI)){
        aControlUsers->addTo(pCommon);
        aAOBI->addTo(pCommon);
    }

```

```

        if(sp.can2(CP_OSZI | CP_BLK)){
            aBlock->addTo(pCommon);
        }
        if(sp.can2(CP_OSZI)){
            aEvents->addTo(pCommon);
        }
        pCommon->insertSeparator();

//-----
        aIntegrity->addTo(pCommon);

        if(sp.can2(CP_OSZI)){
            aTest ->addTo(pCommon);
            aRecovery ->addTo(pCommon);
        }
        pCommon->insertSeparator();
//-----
        if(sp.can2(CP_OSZI)){
            aArchive->addTo(pCommon);
            aUnArchive->addTo(pCommon);
        }

        if(sp.can(CP_VNI)){
            pCommon->insertSeparator();
            pCommon->insertItem("CD-ROM", (Q3PopupMenu*)pCd);
        }

        return pCommon;
    }
    void spMenu::add(QWidget* w, Q3PopupMenu* parent){
        parent->insertItem("C3И", menu(w));
    }

    void spMenu::add(QWidget* w, QMenuBar* parent){
        parent->insertItem("C3И", menu(w));
    }

    void spMenu::slotIntegrity()
    {
        sp.checkIntegrity();
    }

    void spMenu::slotTunningMandatLevels()
    {
        qDebug("slotTunningMandatLevels");
    }

    void spMenu::slotControlUsers()
    {
        if(!sp.cf){
            sp.cf = new spUserControlForm();
        }
        sp.cf->showModal();
    }

    void spMenu::slotArchive()
    {
        sp.backup();
    }

    void spMenu::slotUnArchive()
    {
        sp.restore();
    }

```

```

void spMenu::slotAOBI()
{
    spOBIForm of;
    of.exec();
}
void spMenu::slotBlockUnblock()
{
    spBlockingForm bf;
    bf.exec();
}
void spMenu::slotBlock()
{
    spBlockingForm bf(2);
    bf.exec();
}
void spMenu::slotUnblock()
{
    spBlockingForm bf(1);
    bf.exec();
}

void spMenu::slotEvents()
{
    spRegisterForm rf;
    rf.exec();
}

void spMenu::slotAudit()
{
    qDebug("slotAudit");
}
void spMenu::slotTunningAudit()
{
    qDebug("slotTunningAudit");
}
void spMenu::slotTest()
{
    sp.test();
}

void spMenu::slotRecovery()
{
    sp.recovery();
}

void spMenu::slotProfile(){
    spProfileForm pf;
    pf.exec();
}

static bool exists(const QString& point){
    QFile f("/etc/mtab"); bool ex = false;
    if ( f.open(IO_ReadOnly) ) {
        QTextStream t( &f );
        QString s = t.read();
        ex = s.contains(point) != 0;
        f.close();
    }
    return ex;
}

```

```

}

void spMenu::slotCdMount() {
    if(exists("/mnt/cdrom")) {
        QMessageBox::information(0, "СЗИ:Работа с ВНИ", "CD-ROM уже
подключен");
        return;
    }
    bool err;
    if (!sp.currentIsRoot()) {
        err = system("sudo cdrom.mount");
    }
    else{
        err = system("cdrom.mount");
    }
    QString msg = !err? "CD-ROM подключен" : "Ошибка подключения CD_ROM";
    QMessageBox::information(0, "СЗИ:Работа с ВНИ", msg);
}

void spMenu::slotCdUnmount() {
    if(!exists("/mnt/cdrom")) {
        QMessageBox::information(0, "СЗИ:Работа с ВНИ", "CD-ROM не
подключен");
        return;
    }
    bool err;
    if (!sp.currentIsRoot()) {
        err = system("sudo cdrom.umount");
    }
    else{
        err = system("cdrom.umount");
    }
    QString msg = !err? "CD-ROM отключен" : "Ошибка отключения CD_ROM";
    QMessageBox::information(0, "СЗИ:Работа с ВНИ", msg);
}

void spMenu::slotFlashMount() {
    if(exists("/mnt/uzpi")) {
        QMessageBox::information(0, "СЗИ:Работа с ВНИ", "УЗПИ уже
подключено");
        return;
    }
    bool err;
    if (!sp.currentIsRoot()) {
        err = system("sudo uzpi.mount");
    }
    else{
        err = system("uzpi.mount");
    }
    QString msg = !err? "УЗПИ подключено" : "Ошибка подключения УЗПИ";
    QMessageBox::information(0, "СЗИ:Работа с ВНИ", msg);
}

void spMenu::slotFlashUnmount() {
    if(!exists("/mnt/uzpi")) {
        QMessageBox::information(0, "СЗИ:Работа с ВНИ", "УЗПИ не
подключено");
        return;
    }
    bool err;
    if (!sp.currentIsRoot()) {
        err = system("sudo uzpi.umount");
    }
    else{

```

```

        err = system("uzpi.umount");
    }
    QString msg = !err? "УЗПИ отключено" : "Ошибка отключения УЗПИ";
    QMessageBox::information(0, "СЗИ:Работа с ВНИ", msg);
}

void spMenu::slotFlash(){
    bool flash =      exists("/mnt/uzpi");
    aFlashMount->setEnabled(!flash);
    aFlashUnmount->setEnabled(flash);
}

void spMenu::slotCd(){
    bool cd =      exists("/mnt/cdrom");
    aCdMount->setEnabled(!cd);
    aCdUnmount->setEnabled(cd);
}

```

3.1.3.8 sp_obi_form - реализация АРМ ОБИ (Блокировка/разблокировка пользователей, проверка целостности ПО, тестирование СЗИ, Восстановление СЗИ, Экстренное стирание). Исходный код представлен ниже.

```

#include "sp_obi_form.h"

#include <qvariant.h>
#include <qbuttongroup.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include <q3table.h>
#include <qtoolbutton.h>
#include <q3buttongroup.h>
#include <qlayout.h>
#include <qtooltip.h>
#include <qwhatsthis.h>
#include <qimage.h>
#include <qpixmap.h>
#include <qmessagebox.h>
#include "sp_.h"
#include "sp_tsd.h"
#include "sp_users.h"
#include "../sp_extern.h"

spOBIForm::spOBIForm( QWidget* parent,  const char* name, bool modal,
Qt::WindowFlags fl )
    : QDialog( parent, name, modal, fl )
{
    QPixmap image0( ( const char** ) image0_data );
    if ( !name )
        setName( "spOBIForm" );
    resize( 580, 350 );
    setCaption( ( "СЗИ:АРМ ОБИ" ) );
    spOBIFormLayout = new QGridLayout( this, 1, 1, 11, 6,
"spOBIFormLayout");

    TextLabel1 = new QLabel( this, "TextLabel1" );
    TextLabel1->setText( ( "Контролируемые ДЛ" ) );

    spOBIFormLayout->addWidget( TextLabel1, 0, 0 );
    QSpacerItem* spacer = new QSpacerItem( 68, 0, QSizePolicy::Expanding,
QSizePolicy::Minimum );

```

```

spOBIFormLayout->addItem( spacer, 0, 1 );

btnRefresh = new QPushButton( this, "btnRefresh" );
btnRefresh->setText( ( "" ) );
btnRefresh->setPixmap( image0 );
QToolTip::add( btnRefresh, ( "Запрос состояния выбранного ДЛ" ) );

spOBIFormLayout->addWidget( btnRefresh, 0, 2 );

tblUsers = new Q3Table( this, "tblUsers" );
tblUsers->setNumCols( tblUsers->numCols() + 1 );    tblUsers-
>horizontalHeader()->setLabel( tblUsers->numCols() - 1, ( "Имя ДЛ" ) );
tblUsers->setNumCols( tblUsers->numCols() + 1 );    tblUsers-
>horizontalHeader()->setLabel( tblUsers->numCols() - 1, ( "Состояние
APM" ) );
tblUsers->setNumCols( 2 );
tblUsers->setSelectionMode( Q3Table::SingleRow );
tblUsers->setReadOnly( true );
tblUsers->setFocusStyle( Q3Table::FollowStyle );
tblUsers->setMinimumSize( QSize( 290, 0 ) );
tblUsers->setColumnStretchable( 0, true );
tblUsers->setColumnStretchable( 1, true );

spOBIFormLayout->addMultiCellWidget( tblUsers, 1, 1, 0, 2 );

ButtonGroup1 = new Q3ButtonGroup( this, "ButtonGroup1" );
ButtonGroup1->setTitle( ( "" ) );
ButtonGroup1->setColumnLayout( 0, Qt::Vertical );
ButtonGroup1->layout()->setSpacing( 6 );
ButtonGroup1->layout()->setMargin( 11 );
ButtonGroup1Layout = new QVBoxLayout( ButtonGroup1->layout() );
ButtonGroup1Layout->setAlignment( Qt::AlignTop );

btnBlock = new QPushButton( ButtonGroup1, "btnBlock" );
btnBlock->setMinimumSize( QSize( 0, 0 ) );
btnBlock->setText( ( "Заблокировать" ) );
ButtonGroup1Layout->addWidget( btnBlock );

btnUnblock = new QPushButton( ButtonGroup1, "btnUnblock" );
btnUnblock->setText( ( "Разблокировать" ) );
ButtonGroup1Layout->addWidget( btnUnblock );

btnUnblockConsole = new QPushButton( ButtonGroup1,
"btnUnblockConsole" );
btnUnblockConsole->setText( ( "Разблокировать консоль" ) );
ButtonGroup1Layout->addWidget( btnUnblockConsole );

btnWhoseWP = new QPushButton( ButtonGroup1, "btnWhoseWP" );
btnWhoseWP->setText( ( "Послать запрос \"Свой-Чужой\"" ) );
ButtonGroup1Layout->addWidget( btnWhoseWP );

btnWhoseWPRM = new QPushButton( ButtonGroup1, "btnWhoseWPRM" );
btnWhoseWPRM->setText( ( "\"Свой-Чужой\" для рабочего места" ) );
ButtonGroup1Layout->addWidget( btnWhoseWPRM );

btnIntegrity = new QPushButton( ButtonGroup1, "btnIntegrity" );
btnIntegrity->setText( ( "Проверка целостности ПО" ) );
ButtonGroup1Layout->addWidget( btnIntegrity );

btnTest = new QPushButton( ButtonGroup1, "btnTest" );
btnTest->setText( ( "Тестирование СЗИ" ) );

```

```

ButtonGroup1Layout->addWidget( btnTest );

btnRecovery = new QPushButton( ButtonGroup1, "btnRecovery" );
btnRecovery->setText( ( "Восстановление СЗИ" ) );
ButtonGroup1Layout->addWidget( btnRecovery );

btnErase = new QPushButton( ButtonGroup1, "btnErase" );
btnErase->setText( ( "Экстренное стирание" ) );
ButtonGroup1Layout->addWidget( btnErase );

spOBIFormLayout->addWidget( ButtonGroup1, 1, 3 );

ButtonGroup2 = new Q3ButtonGroup( this, "ButtonGroup2" );
ButtonGroup2->setFrameShape( Q3ButtonGroup::NoFrame );
ButtonGroup2->setTitle( ( "" ) );
ButtonGroup2->setColumnLayout(0, Qt::Vertical );
ButtonGroup2->layout()->setSpacing( 6 );
ButtonGroup2->layout()->setMargin( 11 );
ButtonGroup2Layout = new QHBoxLayout( ButtonGroup2->layout() );
ButtonGroup2Layout->setAlignment( Qt::AlignTop );
QSpacerItem* spacer_2 = new QSpacerItem( 90, 0, QSizePolicy::Expanding,
QSizePolicy::Minimum );
ButtonGroup2Layout->addItem( spacer_2 );

btnExit = new QPushButton( ButtonGroup2, "btnExit" );
btnExit->setText( ( "Выход" ) );
ButtonGroup2Layout->addWidget( btnExit );

spOBIFormLayout->addMultiCellWidget( ButtonGroup2, 2, 2, 0, 3 );

connect( btnExit, SIGNAL( clicked() ), this, SLOT( reject() ) );
connect( btnRefresh, SIGNAL( clicked() ), this, SLOT( refreshSlot() ) );
connect( btnBlock, SIGNAL( clicked() ), this, SLOT( blockSlot() ) );
connect( btnUnblock, SIGNAL( clicked() ), this, SLOT( unblockSlot() ) );
connect( btnUnblockConsole, SIGNAL( clicked() ), this,
SLOT( unblockConsoleSlot() ) );
connect( btnWhoseWP, SIGNAL( clicked() ), this, SLOT( whoseWPslot() ) );
connect( btnWhoseWPRM, SIGNAL( clicked() ), this,
SLOT( whoseWPRMSlot() ) );
connect( btnIntegrity, SIGNAL( clicked() ), this,
SLOT( integritySlot() ) );
connect( btnTest, SIGNAL( clicked() ), this, SLOT( testSlot() ) );
connect( btnRecovery, SIGNAL( clicked() ), this,
SLOT( recoverySlot() ) );
connect( btnErase, SIGNAL( clicked() ), this, SLOT( eraseSlot() ) );

connect( &sp, SIGNAL(stateChangedSignal(const QString&, int) ), this,
SLOT( armRefreshStateSlot(const QString&, int) ) );
sp.isStateChangedSignalConnected = true;

init();

Q3Accel *a = new Q3Accel ( this );
a->connectItem ( a->insertItem ( Qt::Key_F1 ), this, SLOT
( slotHelp() ) );
}

/*
 * Destroys the object and frees any allocated resources
 */
spOBIForm::~spOBIForm()

```

```

{
    sp.isStateChangedSignalConnected = false;
}

void spOBIForm::slotHelp ()
{
}

void spOBIForm::init() {
    const QList<spTsdRecord>& l = sp.usr()->inferiors();
    int n = l.count();
    int rw = 0;
    for(int i=0; i<n; i++){
        const QString& post = l[i].post;
        if(sp.findInTva(post)){
            tblUsers->setNumRows(rw+1);
            tblUsers->setText(rw, 0, post);
            tblUsers->setText(rw, 1, "Не известно");
            rw++;
        }
    }
    tblUsers->adjustColumn(1);
}

void spOBIForm::refreshSlot()
{
    int rw = tblUsers->currentRow();
    tblUsers->setText(rw, 1, "Не известно");
    QString post = tblUsers->text(rw, 0);
    sp.importFile(post, STATPOST);
}

void spOBIForm::blockSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Блокировка APM"),
                                     "Вы действительно хотите заблокировать
"+ post,
                                     QMessageBox::Ok, QMessageBox::Cancel);
    if (exit==QMessageBox::Ok){
        sp.send(BlockWP, post);
    }
}

void spOBIForm::unblockSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Разблокировка APM"),
                                     "Вы действительно хотите разблокировать
"+ post,
                                     QMessageBox::Ok, QMessageBox::Cancel);
    if (exit==QMessageBox::Ok){
        sp.send(UnblockWP, post);
    }
}

void spOBIForm::unblockConsoleSlot()

```

```

{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Разблокировка консоль"),
                                     "Вы действительно хотите разблокировать
"+ post,
                                     QMessageBox::Ok, QMessageBox::Cancel);
    if (exit==QMessageBox::Ok){
        sp.send(UnblockConsole, post);
    }
}

void spOBIForm::armRefreshStateSlot(const QString& post, int state){
    int rows = tblUsers->numRows();
    for(int i=0; i<rows; i++){
        if(tblUsers->text(i, 0) == post){
            QString st;
            switch(state){
                case armStateWork:
                    st = "Работа";
                    break;
                case armStateWorkOnControl:
                    st = "Работа под контролем";
                    break;
                case armStateWorkVisitor:
                    st = "Работа (ЧУЖОЙ)";
                    break;
                case armStateTestError:
                    st = "СЗИ неработоспособна";
                    break;
                case armStateBadIntegrity:
                    st = "Нарушена целостность ПО";
                    break;
                case armStateBlock:
                    st = "Заблокировано";
                    break;
                case armStateNoAccess:
                    st = "Нет доступа";
                    break;
                //case armStateUnknown:
                default:
                    st = "Не известно";
                    break;
            }
            tblUsers->setText(i, 1, st);
            tblUsers->adjustColumn(1);
            break;
        }
    }
}

void spOBIForm::whoseWPSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Посылка сигнала 'Свой-
Чужой'"),
                                     "Вы действительно хотите послать\n"
                                     "запрос 'Свой-Чужой' к "+ post,
                                     QMessageBox::Ok, QMessageBox::Cancel);
}

```

```

        if (exit==QMessageBox::Ok){
            sp.send(WhoseWP, post);
        }
    }
void spOBIForm::whoseWPRMSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Посылка сигнала 'Свой-
Чужой'"),
                                    "Вы действительно хотите послать\n"
                                    "запрос 'Свой-Чужой' к "+ post,
                                    QMessageBox::Ok, QMessageBox::Cancel);

    if (exit==QMessageBox::Ok){
        sp.send(WhoseWPRM, post);
    }
}

void spOBIForm::testSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Тестирование СЗИ"),
                                    "Вы действительно хотите протестировать
СЗИ на "+ post,
                                    QMessageBox::Ok, QMessageBox::Cancel);

    if (exit==QMessageBox::Ok){
        sp.send(Test, post);
    }
}

void spOBIForm::integritySlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Проверка целостности"),
                                    "Вы действительно хотите проверить
целостность ПО на "+ post,
                                    QMessageBox::Ok, QMessageBox::Cancel);

    if (exit==QMessageBox::Ok){
        sp.send(CheckIntegrity, post);
    }
}

void spOBIForm::eraseSlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Экстренное стирание"),
                                    "Вы действительно хотите уничтожить ПО
на "+ post,
                                    QMessageBox::Ok, QMessageBox::Cancel);

    if (exit==QMessageBox::Ok){
        sp.send(Kill, post);
    }
}

void spOBIForm::recoverySlot()
{
    int row = tblUsers->currentRow();
    QString post = tblUsers->text(row, 0);
    int exit=QMessageBox::information(this, ("Восстановление СЗИ"),

```

```

        "Вы действительно хотите восстановить
СЗИ на "+ post,
        QMessageBox::Ok, QMessageBox::Cancel);
    if (exit==QMessageBox::Ok){
        sp.send(Recovery, post);
    }
}

```

sp_password_form - изменение пароля входа в ОС МСВС для ДЛ. Исходный код представлен ниже.

```

#include "sp_password_form.h"

#include <qvariant.h>
#include <qgroupbox.h>
#include <qlabel.h>
#include <qlineedit.h>
#include <qpushbutton.h>
#include <qlayout.h>
#include <qtooltip.h>
#include <qwhatsthis.h>
#include <qimage.h>
#include <qpixmap.h>

spPasswordForm::spPasswordForm( QString post, QWidget* parent, const char*
name, bool modal, Qt::WindowFlags fl )
    : QDialog( parent, name, modal, fl )
{
    QPixmap image0( ( const char** ) image0_data );
    if ( !name )
        setName( "spPasswordForm" );
    resize( 294, 192 );
    setCaption( "СЗИ:Введите пароль..." );
    setIcon( image0 );
    spPasswordFormLayout = new QVBoxLayout( this, 11, 6,
"spPasswordFormLayout");

    GroupBox2 = new Q3GroupBox( this, "GroupBox2" );
    GroupBox2->setTitle( ( "" ) );
    GroupBox2->setColumnLayout(0, Qt::Vertical );
    GroupBox2->layout()->setSpacing( 6 );
    GroupBox2->layout()->setMargin( 11 );
    GroupBox2Layout = new QGridLayout( GroupBox2->layout() );
    GroupBox2Layout->setAlignment( Qt::AlignTop );

    lblParol = new QLabel( GroupBox2, "lblParol" );
    lblParol->setText( "Пароль" );
    lblParol->setAlignment( Qt::AlignVCenter | Qt::AlignRight );

    GroupBox2Layout->addWidget( lblParol, 1, 0 );

    lblDl = new QLabel( GroupBox2, "lblDl" );
    lblDl->setText( "Должностное лицо" );
    lblDl->setAlignment( Qt::AlignVCenter | Qt::AlignRight );

    GroupBox2Layout->addWidget( lblDl, 0, 0 );

    lblParol2 = new QLabel( GroupBox2, "lblParol2" );

```

```

lblParol2->setText( "Повторите пароль" );
lblParol2->setAlignment( Qt::AlignVCenter | Qt::AlignRight );

GroupBox2Layout->addWidget( lblParol2, 2, 0 );

ledParol = new QLineEdit( GroupBox2, "ledParol" );

GroupBox2Layout->addWidget( ledParol, 1, 1 );

ledParol2 = new QLineEdit( GroupBox2, "ledParol2" );

GroupBox2Layout->addWidget( ledParol2, 2, 1 );

lblPost = new QLabel( GroupBox2, "lblPost" );
lblPost->setMinimumSize( QSize( 0, 22 ) );
lblPost->setFrameShape( QLabel::Panel );
lblPost->setFrameShadow( QLabel::Sunken );
lblPost->setLineWidth( 2 );
lblPost->setText( ( "" ) );

GroupBox2Layout->addWidget( lblPost, 0, 1 );

lblMsg = new QLabel( GroupBox2, "lblMsg" );
lblMsg->setText( ( "" ) );

GroupBox2Layout->addMultiCellWidget( lblMsg, 3, 3, 0, 1 );
spPasswordFormLayout->addWidget( GroupBox2 );

GroupBox3 = new Q3GroupBox( this, "GroupBox3" );
GroupBox3->setTitle( ( "" ) );
GroupBox3->setColumnLayout(0, Qt::Vertical );
GroupBox3->layout()->setSpacing( 6 );
GroupBox3->layout()->setMargin( 11 );
GroupBox3Layout = new QHBoxLayout( GroupBox3->layout() );
GroupBox3Layout->setAlignment( Qt::AlignTop );

PushButtonOk = new QPushButton( GroupBox3, "PushButtonOk" );
PushButtonOk->setText( "Принять" );
GroupBox3Layout->addWidget( PushButtonOk );
QSpacerItem* spacer = new QSpacerItem( 30, 0, QSizePolicy::Expanding,
QSizePolicy::Minimum );
GroupBox3Layout->addItem( spacer );

PushButtonCancel = new QPushButton( GroupBox3, "PushButtonCancel" );
PushButtonCancel->setText( "Выход" );
GroupBox3Layout->addWidget( PushButtonCancel );
spPasswordFormLayout->addWidget( GroupBox3 );

connect( ledParol, SIGNAL( textChanged(const QString&) ), this,
SLOT( psw1ChangedSlot(const QString&) ) );
connect( ledParol2, SIGNAL( textChanged(const QString&) ), this,
SLOT( psw2ChangedSlot(const QString&) ) );
connect( PushButtonOk, SIGNAL( clicked() ), this, SLOT( okSlot() ) );
connect( PushButtonCancel, SIGNAL( clicked() ), this, SLOT( cancelSlot()
) );

PushButtonOk->setEnabled(false);
lblPost->setText(post);
}

spPasswordForm::~spPasswordForm()

```

```

{
}

void spPasswordForm::cancelSlot()
{
    reject();
}

void spPasswordForm::okSlot()
{
    accept();
}

void spPasswordForm::psw1ChangedSlot(const QString& pw1)
{
    bool ok = pw1 == ledParol2->text();
    PushButtonOk->setEnabled(ok);
    if(ok) lblMsg->setText("");
    else lblMsg->setText("Подписи не совпадают.");
}

void spPasswordForm::psw2ChangedSlot(const QString& pw2)
{
    bool ok = pw2 == ledParol->text();
    PushButtonOk->setEnabled(ok);
    if(ok) lblMsg->setText("");
    else lblMsg->setText("Подписи не совпадают.");
}

QString spPasswordForm::getSign() {
    return ledParol->text();
}

```

3.1.3.11 sp_profile_form - просмотр полномочий текущего ДП. Исходный код представлен ниже.

```

#include "sp_profile_form.h"
#include "sp_.h"

#include <qvariant.h>
#include <qbuttongroup.h>
#include <qcheckbox.h>
#include <qgroupbox.h>
#include <qpushbutton.h>
#include <qlayout.h>
#include <qtooltip.h>
#include <qwhatsthis.h>

spProfileForm::spProfileForm( QWidget* parent,  const char* name, bool
modal, Qt::WindowFlags fl )
    : QDialog( parent, name, modal, fl )
{
    if ( !name )
        setName( "spProfileForm" );
    resize( 493, 301 );
    setCaption( "СЗИ: Полномочия" );
    spProfileFormLayout = new QVBoxLayout( this, 11, 6,
"spProfileFormLayout");
    profileButtonGroup = new Q3ButtonGroup( this, "profileButtonGroup" );
    profileButtonGroup->setTitle( "Ваши полномочия (только просмотр)" );

```

```

profileButtonGroup->setAlignment( Qt::AlignHCenter );
profileButtonGroup->setColumnLayout(0, Qt::Vertical );
profileButtonGroup->layout()->setSpacing( 6 );
profileButtonGroup->layout()->setMargin( 11 );
profileButtonGroupLayout = new QGridLayout( profileButtonGroup->layout()
);
profileButtonGroupLayout->setAlignment( Qt::AlignTop );

CheckBoxBL = new QCheckBox( profileButtonGroup, "CheckBoxBL" );
CheckBoxBL->setText( "Блокировка/разблокировка" );

profileButtonGroupLayout->addWidget( CheckBoxBL, 0, 0 );

CheckBoxPLO = new QCheckBox( profileButtonGroup, "CheckBoxPLO" );
CheckBoxPLO->setText( "Планирование обеспечения" );

profileButtonGroupLayout->addWidget( CheckBoxPLO, 0, 1 );

CheckBoxPLOG = new QCheckBox( profileButtonGroup, "CheckBoxPLOG" );
CheckBoxPLOG->setText( "Планирование огня" );

profileButtonGroupLayout->addWidget( CheckBoxPLOG, 1, 1 );

CheckBoxPUS = new QCheckBox( profileButtonGroup, "CheckBoxPUS" );
CheckBoxPUS->setText( "Планирование управления связью" );

profileButtonGroupLayout->addWidget( CheckBoxPUS, 2, 1 );

CheckBoxOtvSZI = new QCheckBox( profileButtonGroup, "CheckBoxOtvSZI" );
CheckBoxOtvSZI->setText( "Ответственный за СЗИ" );

profileButtonGroupLayout->addWidget( CheckBoxOtvSZI, 1, 0 );

CheckBoxVni = new QCheckBox( profileButtonGroup, "CheckBoxOBI" );
CheckBoxVni->setText( "Работа с ВНИ" );

profileButtonGroupLayout->addWidget( CheckBoxVni, 6, 0 );

CheckBoxPLM = new QCheckBox( profileButtonGroup, "CheckBoxPLM" );
CheckBoxPLM->setText( "Планирование манёвра" );

profileButtonGroupLayout->addWidget( CheckBoxPLM, 3, 1 );

CheckBoxSVO = new QCheckBox( profileButtonGroup, "CheckBoxSVO" );
CheckBoxSVO->setText( "Свои войска (обработка)" );

profileButtonGroupLayout->addWidget( CheckBoxSVO, 4, 1 );

CheckBoxSVP = new QCheckBox( profileButtonGroup, "CheckBoxSVP" );
CheckBoxSVP->setText( "Свои войска (просмотр)" );

profileButtonGroupLayout->addWidget( CheckBoxSVP, 5, 1 );

CheckBoxVPO = new QCheckBox( profileButtonGroup, "CheckBoxVPO" );
CheckBoxVPO->setText( "Войска противника (обработка)" );

profileButtonGroupLayout->addWidget( CheckBoxVPO, 6, 1 );

CheckBoxVPP = new QCheckBox( profileButtonGroup, "CheckBoxVPP" );
CheckBoxVPP->setText( "Войска противника (просмотр)" );

```

```

profileButtonGroupLayout->addWidget( CheckBoxVPP, 7, 1 );

CheckBoxUO = new QCheckBox( profileButtonGroup, "CheckBoxUO" );
CheckBoxUO->setText( "Управление огнём" );

profileButtonGroupLayout->addWidget( CheckBoxUO, 7, 0 );

CheckBoxUP = new QCheckBox( profileButtonGroup, "CheckBoxUP" );
CheckBoxUP->setText( "Работа с принтером" );

profileButtonGroupLayout->addWidget( CheckBoxUP, 5, 0 );

CheckBoxKUF = new QCheckBox( profileButtonGroup, "CheckBoxKUF" );
CheckBoxKUF->setText( "Работа с КУФ" );

profileButtonGroupLayout->addWidget( CheckBoxKUF, 3, 0 );

CheckBoxOtvKUF = new QCheckBox( profileButtonGroup, "CheckBoxOtvKUF" );
CheckBoxOtvKUF->setText( "Ответственный за КУФ" );

profileButtonGroupLayout->addWidget( CheckBoxOtvKUF, 2, 0 );

CheckBoxNVO = new QCheckBox( profileButtonGroup, "CheckBoxNVO" );
CheckBoxNVO->setText( "Работа с НВО и ТГО" );

profileButtonGroupLayout->addWidget( CheckBoxNVO, 4, 0 );
spProfileFormLayout->addWidget( profileButtonGroup );
//-----
GroupBox2 = new Q3GroupBox( this, "GroupBox2" );
GroupBox2->setMaximumSize( QSize( 32767, 40 ) );
GroupBox2->setFrameShape( Q3GroupBox::NoFrame );
GroupBox2->setTitle( trUtf8( "" ) );
GroupBox2->setColumnLayout(0, Qt::Vertical );
GroupBox2->layout()->setSpacing( 6 );
GroupBox2->layout()->setMargin( 11 );
GroupBox2Layout = new QHBoxLayout( GroupBox2->layout() );
GroupBox2Layout->setAlignment( Qt::AlignTop );
QSpacerItem* spacer = new QSpacerItem( 90, 0, QSizePolicy::Expanding,
QSizePolicy::Minimum );
GroupBox2Layout->addItem( spacer );

btnClose = new QPushButton( GroupBox2, "btnClose" );
btnClose->setText( ( "Закрыть" ) );
GroupBox2Layout->addWidget( btnClose );
QSpacerItem* spacer_2 = new QSpacerItem( 110, 0, QSizePolicy::Expanding,
QSizePolicy::Minimum );
GroupBox2Layout->addItem( spacer_2 );
spProfileFormLayout->addWidget( GroupBox2 );

connect( btnClose, SIGNAL( clicked() ), this, SLOT( reject() ) );
connect( CheckBoxBL, SIGNAL( toggled(bool) ), this,
SLOT( changedBL_Slot(bool) ) );
connect( CheckBoxOtvSZI, SIGNAL( toggled(bool) ), this,
SLOT( changedOtvSZI_Slot(bool) ) );
connect( CheckBoxVni, SIGNAL( toggled(bool) ), this,
SLOT( changedVni_Slot(bool) ) );
connect( CheckBoxOtvKUF, SIGNAL( toggled(bool) ), this,
SLOT( changedOtvKUF_Slot(bool) ) );
connect( CheckBoxKUF, SIGNAL( toggled(bool) ), this,
SLOT( changedKUF_Slot(bool) ) );

```

```

        connect( CheckBoxNVO, SIGNAL( toggled(bool) ), this,
        SLOT( changedNVO_Slot(bool) ) );
        connect( CheckBoxUP, SIGNAL( toggled(bool) ), this,
        SLOT( changedUP_Slot(bool) ) );
        connect( CheckBoxUO, SIGNAL( toggled(bool) ), this,
        SLOT( changedUO_Slot(bool) ) );
        connect( CheckBoxPLO, SIGNAL( toggled(bool) ), this,
        SLOT( changedPLO_Slot(bool) ) );
        connect( CheckBoxPLOG, SIGNAL( toggled(bool) ), this,
        SLOT( changedPLOG_Slot(bool) ) );
        connect( CheckBoxPUS, SIGNAL( toggled(bool) ), this,
        SLOT( changedPUS_Slot(bool) ) );
        connect( CheckBoxPLM, SIGNAL( toggled(bool) ), this,
        SLOT( changedPLM_Slot(bool) ) );
        connect( CheckBoxSVO, SIGNAL( toggled(bool) ), this,
        SLOT( changedSVO_Slot(bool) ) );
        connect( CheckBoxSVP, SIGNAL( toggled(bool) ), this,
        SLOT( changedSVP_Slot(bool) ) );
        connect( CheckBoxVPO, SIGNAL( toggled(bool) ), this,
        SLOT( changedVPO_Slot(bool) ) );
        connect( CheckBoxVPP, SIGNAL( toggled(bool) ), this,
        SLOT( changedVPP_Slot(bool) ) );

        prof = sp.currentProf();
        toCheckBox( prof );

        Q3Accel *a = new Q3Accel ( this );
        a->connectItem ( a->insertItem ( Qt::Key_F1 ), this, SLOT
( slotHelp() ) );
    }

spProfileForm::~spProfileForm()
{

}

void spProfileForm::slotHelp ()
{

}

void spProfileForm::toCheckBox(uint prf){
    CheckBoxBL->setChecked    (prf & CP_BLK);
    CheckBoxPLO->setChecked    (prf & CP_PLO);
    CheckBoxPLOG->setChecked    (prf & CP_PLOG);
    CheckBoxPUS->setChecked    (prf & CP_PLUS);
    CheckBoxOtvSZI->setChecked (prf & CP_OSZI);
    CheckBoxVni->setChecked    (prf & CP_VNI);
    CheckBoxPLM->setChecked    (prf & CP_PLM);
    CheckBoxSVO->setChecked    (prf & CP_SVO);
    CheckBoxSVP->setChecked    (prf & CP_SVP);
    CheckBoxVPO->setChecked    (prf & CP_VPO);
    CheckBoxVPP->setChecked    (prf & CP_VPP);
    CheckBoxUO->setChecked    (prf & CP_UO);
    CheckBoxUP->setChecked    (prf & CP_PRN);
    CheckBoxKUF->setChecked    (prf & CP_KUF);
    CheckBoxOtvKUF->setChecked (prf & CP_OKUF);
    CheckBoxNVO->setChecked    (prf & CP_NVO);
}

void spProfileForm::changedBL_Slot(bool){
    toCheckBox( prof );
}

```

```

}
void spProfileForm::changedKUF_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedNVO_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedVni_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedOtvKUF_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedOtvSZI_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedPLM_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedPLOG_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedPLO_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedPUS_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedSVO_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedSVP_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedUO_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedUP_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedVPO_Slot (bool) {
    toCheckBox (prof);
}
void spProfileForm::changedVPP_Slot (bool) {
    toCheckBox (prof);
}
}

```

sp_register - база данных регистрационного журнала событий безопасности.
Исходный код представлен ниже.

```

#include "sp_register.h"
#include <qmessagebox.h>
#include <qdatetime.h>
#include <q3datatable.h>
#include <q3sqlcursor.h>

spRegister::spRegister() {
}
spRegister::~spRegister() {
}

```

```

bool spRegister::put(const char* post, const char* module, const char* func,
const char* msg){

    if(init()){
        Q3SqlCursor cur(REGISTER_TBL);
        QSqlRecord *buffer = cur.primeInsert();
        QString t;

        buffer->setValue( "date", QDate::currentDate() );

        buffer->setValue( "time", QTime::currentTime() );

        t = post;
        buffer->setValue( "post", t );

        t = module;
        buffer->setValue( "module", t );

        t = func;
        buffer->setValue( "func", t );

        t = msg;
        buffer->setValue( "msg", t );

        cur.insert();
        return true;
    }
    return false;
}

bool spRegister::init(){
    if(!db.isValid()){
        db = QSqlDatabase::addDatabase( DRIVER );
        if(!db.isValid()){
            QMessageBox::critical(0,
                                   "СЗИ",
                                   "Ошибка доступа к регистрационному журналу.",
                                   QMessageBox::Ok, QMessageBox::NoButton,
                                   QMessageBox::NoButton );
            return false;
        }
    }
    else{
        db.setDatabaseName( DATABASE );
        db.setUserName( USER );
        db.setPassword( PASSWORD );
        db.setHostName( HOST );

        if( !db.open() ){
            QString errt = db.lastError().text();
            qWarning (errt);
            return false;
        }

        QStringList existTables = db.tables(QSql::Tables);
        if (!existTables.contains(REGISTER_TBL)) {
            QSqlQuery q("create table " REGISTER_TBL
                        "(date date, time time, post varchar(20),
module varchar(20), func varchar(32), msg varchar(64))", db);
        }
    }
}

```

```

    }
}
return true;
}

```

sp_register_form - форма просмотра регистрационного журнала безопасности.
Исходный код представлен ниже.

```

#include "sp_register.h"
#include "sp_register_form.h"
#include "sp_.h"
#include "sp_tsd.h"

#include <qvariant.h>
#include <qcombobox.h>
#include <qdatetimeedit.h>
#include <qgroupbox.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include <q3datatable.h>
#include <qtoolbutton.h>
#include <qlayout.h>
#include <qtooltip.h>
#include <qwhatsthis.h>
#include <qimage.h>
#include <qpixmap.h>
#include <q3sqlcursor.h>
#include <q3dict.h>
#include <qmessagebox.h>
#include <qstringlist.h>
#include <qtextstream.h>
#include <qfile.h>

spRegisterForm::spRegisterForm( QWidget* parent,  const char* name, bool
modal, Qt::WindowFlags fl )
    : QDialog( parent, name, modal, fl )
{
    QPixmap image7( ( const char** ) image1_data );
    QPixmap image8( ( const char** ) image0_data );
    QPixmap image9( ( const char** ) image9_data );

    if ( !name )
        setName( "spRegisterForm" );
    resize( 735, -1 );
    setCaption( ( "СЗИ:Журнал событий безопасности" ) );
    spRegisterFormLayout = new QGridLayout( this, 1, 1, 11, 6,
"spRegisterFormLayout");

    GroupBox1 = new Q3GroupBox( this, "GroupBox1" );
    GroupBox1->setTitle( ( "Фильтрация" ) );
    GroupBox1->setAlignment( Qt::AlignHCenter );
    GroupBox1->setColumnLayout(0, Qt::Vertical );
    GroupBox1->layout()->setSpacing( 6 );
    GroupBox1->layout()->setMargin( 11 );
    GroupBox1Layout = new QHBoxLayout( GroupBox1->layout() );
    GroupBox1Layout->setAlignment( Qt::AlignTop );

    lblDate = new QLabel( GroupBox1, "lblDate" );
    lblDate->setText( ( "Дата:" ) );
    GroupBox1Layout->addWidget( lblDate );

```

```

cmbDate = new QComboBox( FALSE, GroupBox1, "cmbDate" );
cmbDate->insertItem( ( "Все" ) );
cmbDate->insertItem( ( "Вчера" ) );
cmbDate->insertItem( ( "Сегодня" ) );
cmbDate->insertItem( ( "Конкретный день" ) );
QToolTip::add( cmbDate, ( "Режим фильтрации по дате" ) );
GroupBox1Layout->addWidget( cmbDate );

spinDate = new Q3DateEdit( GroupBox1, "spinDate" );
spinDate->setOrder( Q3DateEdit::DMY );
spinDate->setAutoAdvance( TRUE );
spinDate->setMaxValue( QDate::currentDate() );

GroupBox1Layout->addWidget( spinDate );
QSpacerItem* spacer = new QSpacerItem( 16, 0, QSizePolicy::Expanding,
QSizePolicy::Minimum );
GroupBox1Layout->addItem( spacer );

    lblTime0 = new QLabel( GroupBox1, "lbkTime0" );
    lblTime0->setText( ( "Время:" ) );
    GroupBox1Layout->addWidget( lblTime0 );

    spinTime0 = new Q3TimeEdit( GroupBox1, "spinTime0" );
    spinTime0->setAutoAdvance( TRUE );
    spinTime0->setMinValue( QTime(0,0,0) );
    spinTime0->setMaxValue( QTime(23,59,59) );
    spinTime0->setTime( QTime(0,0,0) );
    GroupBox1Layout->addWidget( spinTime0 );

    spinTime1 = new Q3TimeEdit( GroupBox1, "spinTime1" );
    spinTime1->setAutoAdvance( TRUE );
    spinTime1->setMinValue( QTime(0,0,0) );
    spinTime1->setMaxValue( QTime(23,59,59) );
    spinTime1->setTime( QTime(23,59,59) );
    GroupBox1Layout->addWidget( spinTime1 );

    btnClearTime = new QToolButton( GroupBox1, "btnClearTime" );
    btnClearTime->setText( "0:0" );
    QToolTip::add( btnClearTime, ( "Сбросить время" ) );
    GroupBox1Layout->addWidget( btnClearTime );

    spRegisterFormLayout->addWidget( GroupBox1, 0, 0 );

    GroupBox2 = new Q3GroupBox( this, "GroupBox2" );
    GroupBox2->setTitle( ( "Удаление" ) );
    GroupBox2->setAlignment( Qt::AlignHCenter );
    GroupBox2->setColumnLayout(0, Qt::Vertical );
    GroupBox2->layout()->setSpacing( 6 );
    GroupBox2->layout()->setMargin( 11 );
    GroupBox2Layout = new QHBoxLayout( GroupBox2->layout() );
    GroupBox2Layout->setAlignment( Qt::AlignHCenter );

    btnDeleteDefDate = new QToolButton( GroupBox2, "btnDeleteDefDate" );
    btnDeleteDefDate->setPixmap( image7 );
    btnDeleteDefDate->setMaximumSize( QSize( 32, 32 ) );
    QToolTip::add( btnDeleteDefDate, ( "Удалить записи по выбранной
дате" ) );
    GroupBox2Layout->addWidget( btnDeleteDefDate );

```

```

btnDeletePrevDate = new QToolButton( GroupBox2, "btnDeletePrevDate" );
btnDeletePrevDate->setPixmap( image8 );
btnDeletePrevDate->setMaximumSize( QSize( 32, 32 ) );
QToolTip::add( btnDeletePrevDate, ( "Удалить записи ранее выбранной
даты" ) );
GroupBox2Layout->addWidget( btnDeletePrevDate );

spRegisterFormLayout->addWidget( GroupBox2, 0, 1 );

GroupBox3 = new Q3GroupBox( this, "GroupBox3" );
GroupBox3->setTitle( ( "Печать" ) );
GroupBox3->setAlignment( Qt::AlignHCenter );
GroupBox3->setColumnLayout(0, Qt::Vertical );
GroupBox3->layout()->setSpacing( 6 );
GroupBox3->layout()->setMargin( 11 );
GroupBox3Layout = new QHBoxLayout( GroupBox3->layout() );
GroupBox3Layout->setAlignment( Qt::AlignHCenter );

btnPrint = new QToolButton( GroupBox3, "btnPrint" );
btnPrint->setMaximumSize( QSize( 32, 32 ) );
btnPrint->setPixmap( image9 );
QToolTip::add( btnPrint, ( "Распечатать отобранные записи" ) );
GroupBox3Layout->addWidget( btnPrint );

spRegisterFormLayout->addWidget( GroupBox3, 0, 3 );
QSpacerItem* spacer_3 = new QSpacerItem( 16, 0, QSizePolicy::Expanding,
QSizePolicy::Minimum );
spRegisterFormLayout->addItem( spacer_3, 0, 2 );

cursor = new QSqlCursor("rgtbl");
table = new Q3DataTable( this, "table" );

table->setSqlCursor(cursor);
table->addColumn( "date", "Дата" );
table->addColumn( "time", "Время" );
table->addColumn( "post", "ДЛ" );
table->addColumn( "module", "Подсистема" );
table->addColumn( "func", "Функция" );
table->addColumn( "msg", "Событие" );

table->setColumnStretchable( 3, true );
table->setColumnStretchable( 4, true );
table->setColumnStretchable( 5, true );

table->setDateFormat( Qt::LocalDate );
table->setSorting(true);
table->setReadOnly(true);
table->refresh();
table->show();

spRegisterFormLayout->addMultiCellWidget( table, 1, 1, 0, 3 );

connect( cmbDate, SIGNAL( activated(int) ), this,
SLOT( dateFilterSlot(int) ) );
connect( spinDate, SIGNAL( valueChanged(const QDate&) ), this,
SLOT( dateChangedSlot(const QDate&) ) );
connect( spinTime0, SIGNAL( valueChanged(const QTime&) ), this,
SLOT( time0ChangedSlot(const QTime&) ) );
connect( spinTime1, SIGNAL( valueChanged(const QTime&) ), this,
SLOT( time1ChangedSlot(const QTime&) ) );

```

```

        connect( btnDeleteDefDate, SIGNAL( clicked() ), this,
SLOT( deleteDefDateSlot() ) );
        connect( btnDeletePrevDate, SIGNAL( clicked() ), this,
SLOT( deletePrevDateSlot() ) );
        connect( btnPrint, SIGNAL( clicked() ), this, SLOT( printSlot() ) );
        connect( btnClearTime, SIGNAL( clicked() ), this,
SLOT( clearTimeSlot() ) );

        dateFilterSlot(0);

        Q3Accel *a = new Q3Accel ( this );
        a->connectItem ( a->insertItem ( Qt::Key_F1 ), this, SLOT
( slotHelp() ) );
    }

/*
 * Destroys the object and frees any allocated resources
 */
spRegisterForm::~spRegisterForm()
{
    delete cursor;
}

void spRegisterForm::slotHelp ()
{
}

void spRegisterForm::dateChangedSlot(const QDate& d)
{
    if(d == QDate::currentDate())
        btnDeleteDefDate->setEnabled(false);
    else
        btnDeleteDefDate->setEnabled(true);

    selectDT();
}

void spRegisterForm::dateFilterSlot(int index)
{
    //0 Все
    //1 Вчера
    //2 Сегодня
    //3 Конкр день
    switch(index){
        case 0:
            spinDate->setEnabled(false);
            spinTime0->setEnabled(false);
            spinTime1->setEnabled(false);
            btnClearTime->setEnabled(false);
            clearTimeSlot();
            GroupBox3->setEnabled(false);
            GroupBox2->setEnabled(false);

            table->setFilter("");
            break;
        case 1:
            spinDate->setDate(QDate::currentDate().addDays(-1));
            spinDate->setEnabled(false);
            spinTime0->setEnabled(true);
            spinTime1->setEnabled(true);
            btnClearTime->setEnabled(true);
    }
}

```

```

        GroupBox3->setEnabled(true);
        GroupBox2->setEnabled(true);
        btnDeleteDefDate->setEnabled(true);
        btnDeletePrevDate->setEnabled(true);

    break;
case 2:
    spinDate->setDate(QDate::currentDate());
    spinDate->setEnabled(false);
    spinTime0->setEnabled(true);
    spinTime1->setEnabled(true);
    btnClearTime->setEnabled(true);
    GroupBox3->setEnabled(true);
    GroupBox2->setEnabled(true);
    btnDeleteDefDate->setEnabled(false);
    btnDeletePrevDate->setEnabled(true);

    break;
case 3:
    spinDate->setEnabled(true);
    spinTime0->setEnabled(true);
    spinTime1->setEnabled(true);
    btnClearTime->setEnabled(true);
    GroupBox3->setEnabled(true);
    GroupBox2->setEnabled(true);

    if(spinDate->date() == QDate(0,0,0))
        spinDate->setDate(QDate::currentDate());

    if(spinDate->date() == QDate::currentDate())
        btnDeleteDefDate->setEnabled(false);
    else
        btnDeleteDefDate->setEnabled(true);
    btnDeletePrevDate->setEnabled(true);

    break;
}
table->refresh();
adjustForm();
}

void spRegisterForm::deleteDefDateSlot()
{
    QDate d = spinDate->date();
    QString sd = "'" + d.toString(Qt::ISODate) + "'";
    int ans = QMessageBox::information(0, ("Корректировка регистрационного
журнала"),
                                     "Вы действительно хотите удалить все
записи\n"
                                     "зарегистрированные "+ sd,
                                     QMessageBox::Ok, QMessageBox::Cancel);
    if (ans!=QMessageBox::Ok) return;
    spLabel label("Корректировка регистрационного журнала. Подождите...");

    QString q = "DELETE FROM rgtbl WHERE date=" + sd;
    QSqlQuery qu(q);
    if(qu.isActive()){
        qDebug("%d rows deleted", qu.numRowsAffected());
    }
    table->refresh();
    adjustForm();
}

```

```

void spRegisterForm::deletePrevDateSlot()
{
    QDate d = spinDate->date();
    QString sd = "'" + d.toString(Qt::ISODate) + "'";
    int ans = QMessageBox::information(0, ("Корректировка регистрационного
журнала"),
                                     "Вы действительно хотите удалить все
записи\n"
                                     "зарегистрированные ранее "+ sd,
                                     QMessageBox::Ok, QMessageBox::Cancel);
    if (ans!=QMessageBox::Ok) return;

    spLabel label("Корректировка регистрационного журнала. Подождите...");

    QString q = "DELETE FROM rgtbl WHERE date<" + sd;
    QSqlQuery qu(q);
    if(qu.isActive()){
        qDebug("%d rows deleted", qu.numRowsAffected()) ;
    }
    table->refresh();
    adjustForm();
}

#define _lenL_ 76
#define _len0_ 8
#define _len1_ 16

#define _tab0_ 0
#define _tab1_ (_len0_+1)
#define _tab2_ (_tab1_+_len1_+1)
#define _tab3_ _tab2_
#define _len3_ (_lenL_ + 1 - _tab3_)
#define _len2_ (_lenL_ + 1 - _tab2_)

void spRegisterForm::printSlot()
{
    //Имя временного файла для печати
    QString tmpFile = QTime::currentTime().toString(Qt::ISODate);
    tmpFile.replace(":", "");
    tmpFile.prepend(SP_PATH"/rg");
    tmpFile+="".prn";
    QFile f(tmpFile);
    QString field[4];
    if(f.open(IO_WriteOnly)){
        QTextStream stream(&f);
        //-----
        QString header = "*** События безопасности, зарегистрированные ";
        header += spinDate->date().toString(Qt::ISODate) + " ("

+       spinTime0->time().toString(Qt::ISODate)

+       "-"

+       spinTime1->time().toString(Qt::ISODate)

```

```

+ " )";
    stream << header << "\n";
    header.fill('-', _lenL_);
    stream << header << "\n";

    field[0] = "Время";
    field[1] = "Должн.лицо";
    field[2] = "Система:функция / Событие";

    QString line;
    line.fill(' ', _lenL_);
    line.replace(_tab0_, _len0_, field[0]);
    line.replace(_tab1_, _len1_, field[1]);
    line.replace(_tab2_, _len2_, field[2]);
    stream << line << "\n";

    stream << header << "\n";

    QSqlCursor cur("rgtbl");
    cur.select(table->filter());
    while(cur.next()){
        field[0] = cur.value(1).toString();
        field[1] = cur.value(2).toString();
        field[2] = cur.value(3).toString() + ":"
+cur.value(4).toString();
        field[3] = cur.value(5).toString();

        line.fill(' ', _lenL_);
        line.replace(_tab0_, _len0_, field[0]);
        line.replace(_tab1_, _len1_, field[1]);
        line.replace(_tab2_, _len2_, field[2]);
        stream << line << "\n";

        while(!field[3].isEmpty()){
            line.fill(' ', _lenL_);
            line.replace(_tab3_, _len3_, field[3].left(_len3_));
            field[3].remove(0, _len3_);
            stream << line << "\n";
        }
    }

    //-----
    stream << header << "\n";
    f.close();
    //-----
}
else{
    QMessageBox::critical(0,
        "Печать ...",
        QString("Ошибка открытия файла ") + tmpFile,
        QMessageBox::Ok, QMessageBox::NoButton, QMessageBox::NoButton );
}
}

void spRegisterForm::time0ChangedSlot(const QTime& t){
    QTime t1 = spinTime1->time().addSecs(-1);
    if(t > t1)
        spinTime0->setTime(t1);
    selectDT();
}

```

```

}

void spRegisterForm::time1ChangedSlot(const QTime& t){
    QTime t0 = spinTime0->time().addSecs(1);
    if(t < t0)
        spinTime1->setTime(t0);
    selectDT();
}

void spRegisterForm::selectDT(){
    QDate d = spinDate->date();
    QString sd = "'" + d.toString(Qt::ISODate) + "'";

    QTime t0 = spinTime0->time();
    QString st0 = "'" + t0.toString(Qt::ISODate) + "'";

    QTime t1 = spinTime1->time();
    QString st1 = "'" + t1.toString(Qt::ISODate) + "'";

    QString q = "date=" + sd + " and time>=" + st0 + "and time<=" + st1;
    table->setFilter(q);
    table->refresh();
    adjustForm();
}

void spRegisterForm::clearTimeSlot(){
    spinTime0->setTime(spinTime0->minValue());
    spinTime1->setTime(spinTime1->maxValue());
}

void spRegisterForm::adjustForm(){
    table->adjustColumn(2);
    table->adjustColumn(3);
    table->adjustColumn(4);
    table->adjustColumn(5);
    adjustSize();
}

```

sp_tsd - таблица санкционированного доступа. Исходный код представлен ниже.

```

#include "sp_tsd.h"
#include <qfile.h>
#include <qdatastream.h>
#include <qtextstream.h>
#include <qdir.h>
#include <qmessagebox.h>
#include <stdlib.h>
#include "sp_users.h"
#include "sp_.h"

extern QString developer;

spTsdRecord::spTsdRecord(){
    clear();
}

spTsdRecord::spTsdRecord(const spTsdRecord& r){
    post = r.post;
    sign = r.sign;
    prf   = r.prf;
    ord   = r.ord;
    id    = r.id;
    state = r.state;
}

```

```

QTextStream& operator>>(QTextStream& s, spTsdRecord& r){
    codeRead(r.post, s);
    codeRead(r.sign, s);
    s >> r.prf;
    s >> r.ord;
    s >> r.id;
    r.state = USER_STATE_UNKNOWN;
    return s;
}

QTextStream& operator<<(QTextStream& s, spTsdRecord& r){
    codeWrite(r.post, s);
    codeWrite(r.sign, s);
    s << r.prf << ' ';
    s << r.ord << ' ';
    s << r.id << ' ';
    return s;
}

bool operator<= ( const spTsdRecord& r1, const spTsdRecord& r2){
    return r1.ord <= r2.ord;
}

bool operator< ( const spTsdRecord& r1, const spTsdRecord& r2){
    return r1.ord < r2.ord;
}

bool operator> ( const spTsdRecord& r1, const spTsdRecord& r2){
    return r1.ord > r2.ord;
}

bool operator== ( const spTsdRecord& r1, const spTsdRecord& r2){
    return r1.ord == r2.ord;
}

}

void spTsdRecord::clear(){
    post = "";
    sign = "";
    prf = 0;
    ord = 0;
    id = 0;
    state = USER_STATE_UNKNOWN;
}

spTsd::spTsd(QString tsdFile){
    setAutoDelete(true);
    tsdFileName = tsdFile;
}

spTsd::~spTsd(){
}

bool spTsd::load(QString fname){
    QFile file( fname );
    bool ok = file.open( IO_ReadOnly );
    if(ok){
        QTextStream s( &file );
        spTsdRecord r;
        while(!s.atEnd()){
            s >> r;
            if(!r.post.isEmpty() && !r.sign.isEmpty())
                insert(r.post, new spTsdRecord(r));
        }
        emit changed_signal();
    }
    return ok? control() : ok;
}

```

```

bool spTsd::save(QString fname){
    QFile file( fname );
    bool existing = file.exists();
    bool ok = file.open( IO_WriteOnly );
    if(ok){
        QTextStream s( &file );
        Q3DictIterator<spTsdRecord> it( *this );
        spTsdRecord* r;
        for( ; (r=it.current()); ++it ){
            s << *r;
        }
    }
    file.close();
    if(existing){
        QString cmd = "chmod 0666 " + fname;
        system(cmd.ascii());
    }
    emit changed_signal();

    return ok? control() : ok;
}

bool spTsd::save(){
    return save(tsdFileName);
}

bool spTsd::load(){
    return load(tsdFileName);
}

bool spTsd::exist(){
    return !isEmpty();
}

bool spTsd::recreate(QString fname){
    clear();
    return load(fname);
}

bool spTsd::addRecordAndSave(spTsdRecord& r){
    r.ord = order(r.post);
    insert(r.post, new spTsdRecord(r));
    return save();
}

bool spTsd::delRecordAndSave(QString name){

    bool ok = remove(name);
    if(ok)
        save();
    return true;
}

int spTsd::order(QString post){
    int lv = 0;
    int nm = 0;
    int d = 0;
    if(post == "root")
        return 2001;
    QString dl = post.section(S_, 0, 0);
    QString dv = post.section(S_, 1,1);
    if(!dv.isEmpty()){

```

```

    nm = atoi(dv);
    if(dv.contains(_BR_)) {
        lv = 1;
    }
    else if(dv.contains(_DN_)){
        lv = 2;
    }
    else if(dv.contains(_BATR_)){
        lv = 3;
    }
    else if(dv.contains(_BM_)){
        lv = 4;
    }
    else if(dv.contains(_PU_)){
        lv = 4;
    }
    else {
#ifdef DEVELOPER
        if(post == developer)
            return 2001;
#endif
        lv = 5; //Unknown level
    }
}

if(!dl.isEmpty()){
    if(dl.contains(S_OFS)) {
        d = 2;
    }
    else if(dl.contains(S_KR)){
        d = 1;
    }
    else if(dl.contains(S_NS) || dl.contains(S_NOO)){
        d = 3;
    }
    else if(dl.contains(S_SOPR)){
        d = 4;
    }
    else {
        d = 5; //Unknown level
    }
}

return lv*100000 + nm*10 + d;
}

bool spTsd::control(){
    bool ok = true;
    Q3DictIterator<spTsdRecord> it1( *this );
    spTsdRecord* r1;
    for(int i1=0 ; (r1=it1.current()); ++it1, i1++ ){
        if(r1->sign.length()<6){
            QMessageBox::critical(0,
                                   "Проверка ТСД...",

                                   QString("Требуется корректировка ТСД\n"
                                           "Подпись ДЛ должна содержать 6 символов ") + ""
                                   + r1->sign + "",
                                   QMessageBox::Ok, QMessageBox::NoButton,
                                   QMessageBox::NoButton );
            ok = false;
        }
    }
}

```

```

    }
    if(r1->ord/100000<4 && r1->prf==0){
        QMessageBox::critical(0,
                               "Проверка ТСД...",

                               QString("Требуется корректировка ТСД\n"
                                       "Не заданы полномочия ДЛ ") + "'" + r1->post +
                               "'",
                               QMessageBox::Ok, QMessageBox::NoButton,
                               QMessageBox::NoButton );
        ok = false;
    }

    Q3DictIterator<spTsdRecord> it2( *this );
    spTsdRecord* r2;

    for(int i2=0 ; (r2=it2.current()); ++it2, i2++ ){
        if(i2<=i1) continue;

        if(r1->post == r2->post){
            QMessageBox::critical(0,
                                   "Проверка ТСД...",

                                   QString("Требуется корректировка ТСД\n"
                                           "Дубликатное имя ДЛ ") + "'" + r1->post + "'",
                                   QMessageBox::Ok, QMessageBox::NoButton,
                                   QMessageBox::NoButton );
            ok = false;
        }
        if(r1->sign == r2->sign){
            QMessageBox::critical(0,
                                   "Проверка ТСД...",

                                   QString("Требуется корректировка ТСД\n"
                                           "Дубликатная подпись ДЛ ") + "'" + r1->sign +
                                   "'",
                                   QMessageBox::Ok, QMessageBox::NoButton,
                                   QMessageBox::NoButton );
            ok = false;
        }
    }
    return ok;
}

```

sp_user_control_form - форма управление пользователями. Исходный код представлен ниже.

```

#include "sp_usercontrol_form.h"

#include <qvariant.h>
#include <qbuttongroup.h>
#include <qcheckbox.h>
#include <q3groupbox.h>
#include <qpushbutton.h>
#include <q3table.h>
#include <qlayout.h>
#include <qtooltip.h>
#include <qwhatsthis.h>
#include <qaction.h>

```

```

#include <qmenubar.h>
#include <q3popupmenu.h>
#include <qtoolbar.h>
#include <qimage.h>
#include <qpixmap.h>
#include <qapplication.h>
#include <qmessagebox.h>
#include <qfiledialog.h>
#include <qlineedit.h>

#include "sp_.h"
#include "sp_users.h"
#include "sp_password_form.h"

#include <mac.h>

template <typename spTsdRecord>
class qLessRecord
{
public:
    inline bool operator()(const spTsdRecord &t1, const spTsdRecord &t2)
const
    {
        return (t1.id < t2.id);
    }
};

usTable::usTable(QWidget * parent, const char * name):Q3Table ( parent,
name){}
QWidget* usTable::beginEdit ( int row, int col, bool replace ){
    QWidget* w = Q3Table::beginEdit ( row, col, replace );
    QLineEdit* e = (QLineEdit*)cellWidget(row, col);
    if(e){
        connect( e, SIGNAL( textChanged ( const QString & ) ), this,
SLOT( textChanged ( const QString & ) ) );
    }
    return w;
}

void usTable::textChanged ( const QString & txt){
    emit keyPressCell(txt);
}

spUserControlForm::spUserControlForm( QWidget* parent, const char* name,
Qt::WindowFlags fl )
    : QMainWindow( parent, name, fl )
{
    (void)statusBar();
    QPixmap image0( ( const char** ) image0_data );
    QPixmap image1( ( const char** ) image1_data );
    QPixmap image2( ( const char** ) image2_data );
    QPixmap image3( ( const char** ) image3_data );
    QPixmap image4( ( const char** ) image4_data );
    QPixmap image5( ( const char** ) image5_data );
    QPixmap image6( ( const char** ) image6_data );

    QPixmap imageUserDelete( ( const char** ) imageUserDelete_data );
    QPixmap imageTsdDelete( ( const char** ) imageTsdDelete_data );
    QPixmap imageTsdGenerate( ( const char** ) imageTsdGenerate_data );

    QPixmap image8( ( const char** ) image8_data );
    QPixmap image9( ( const char** ) image9_data );

```

```

QPixmap image10( ( const char** ) image10_data );
if ( !name )
    setName( "spUserControlForm" );
resize( 800, 330 );
setCaption( "СЗИ:Управление пользователями" );
setIcon( image0 );
setCentralWidget( new QWidget( this, "qt_central_widget" ) );
spUserControlFormLayout = new QHBoxLayout( centralWidget(), 11, 6,
"spUserControlFormLayout");

GroupBox1 = new Q3GroupBox( centralWidget(), "GroupBox1" );
GroupBox1->setTitle( ( "" ) );
GroupBox1->setColumnLayout(0, Qt::Vertical );
GroupBox1->layout()->setSpacing( 6 );
GroupBox1->layout()->setMargin( 11 );
GroupBox1Layout = new QHBoxLayout( GroupBox1->layout() );
GroupBox1Layout->setAlignment( Qt::AlignTop );

usersTable = new usTable( GroupBox1, "usersTable" );
usersTable->setNumCols( usersTable->numCols() + 1 );
usersTable->horizontalHeader()->setLabel( usersTable->numCols() - 1,
( "Долж.лицо" ) );
usersTable->setNumCols( usersTable->numCols() + 1 );
usersTable->horizontalHeader()->setLabel( usersTable->numCols() - 1,
( "Подпись" ) );
usersTable->setMinimumSize( QSize( 280, 300 ) );
usersTable->setNumRows( 2 );
usersTable->setNumCols( 2 );
usersTable->setColumnStretchable( 0, true );
usersTable->setColumnStretchable( 1, true );
usersTable->setSelectionMode(Q3Table::NoSelection);
GroupBox1Layout->addWidget( usersTable );

QFrame *frame = new QFrame(GroupBox1);
QVBoxLayout *vbox = new QVBoxLayout(frame);
vbox->setMargin(0);

profileButtonGroup = new Q3ButtonGroup( /*GroupBox1*/frame,
"profileButtonGroup" );
profileButtonGroup->setTitle( "Полномочия" );
profileButtonGroup->setAlignment( Qt::AlignHCenter );
profileButtonGroup->setColumnLayout(0, Qt::Vertical );
profileButtonGroup->layout()->setSpacing( 6 );
profileButtonGroup->layout()->setMargin( 11 );
profileButtonGroupLayout = new QGridLayout( profileButtonGroup->layout()
);
profileButtonGroupLayout->setAlignment( Qt::AlignTop );

CheckBoxBL = new QCheckBox( profileButtonGroup, "CheckBoxBL" );
CheckBoxBL->setText( "Блокировка/разблокировка" );

profileButtonGroupLayout->addWidget( CheckBoxBL, 0, 0 );

CheckBoxPLO = new QCheckBox( profileButtonGroup, "CheckBoxPLO" );
CheckBoxPLO->setText( "Планирование обеспечения" );

profileButtonGroupLayout->addWidget( CheckBoxPLO, 0, 1 );

CheckBoxPLOG = new QCheckBox( profileButtonGroup, "CheckBoxPLOG" );
CheckBoxPLOG->setText( "Планирование огня" );

```

```

profileButtonGroupLayout->addWidget( CheckBoxPLOG, 1, 1 );

CheckBoxPUS = new QCheckBox( profileButtonGroup, "CheckBoxPUS" );
CheckBoxPUS->setText( "Планирование управления связью" );

profileButtonGroupLayout->addWidget( CheckBoxPUS, 2, 1 );

CheckBoxOtvSZI = new QCheckBox( profileButtonGroup, "CheckBoxOtvSZI" );
CheckBoxOtvSZI->setText( "Ответственный за СЗИ" );

profileButtonGroupLayout->addWidget( CheckBoxOtvSZI, 1, 0 );

CheckBoxVni = new QCheckBox( profileButtonGroup, "CheckBoxOBI" );
CheckBoxVni->setText( "Работа с ВНИ" );

profileButtonGroupLayout->addWidget( CheckBoxVni, 6, 0 );

CheckBoxPLM = new QCheckBox( profileButtonGroup, "CheckBoxPLM" );
CheckBoxPLM->setText( "Планирование манёвра" );

profileButtonGroupLayout->addWidget( CheckBoxPLM, 3, 1 );

CheckBoxSVO = new QCheckBox( profileButtonGroup, "CheckBoxSVO" );
CheckBoxSVO->setText( "Свои войска (обработка)" );

profileButtonGroupLayout->addWidget( CheckBoxSVO, 4, 1 );

CheckBoxSVP = new QCheckBox( profileButtonGroup, "CheckBoxSVP" );
CheckBoxSVP->setText( "Свои войска (просмотр)" );

profileButtonGroupLayout->addWidget( CheckBoxSVP, 5, 1 );

CheckBoxVPO = new QCheckBox( profileButtonGroup, "CheckBoxVPO" );
CheckBoxVPO->setText( "Войска противника (обработка)" );

profileButtonGroupLayout->addWidget( CheckBoxVPO, 6, 1 );

CheckBoxVPP = new QCheckBox( profileButtonGroup, "CheckBoxVPP" );
CheckBoxVPP->setText( "Войска противника (просмотр)" );

profileButtonGroupLayout->addWidget( CheckBoxVPP, 7, 1 );

CheckBoxUO = new QCheckBox( profileButtonGroup, "CheckBoxUO" );
CheckBoxUO->setText( "Управление огнём" );

profileButtonGroupLayout->addWidget( CheckBoxUO, 7, 0 );

CheckBoxUP = new QCheckBox( profileButtonGroup, "CheckBoxUP" );
CheckBoxUP->setText( "Работа с принтером" );

profileButtonGroupLayout->addWidget( CheckBoxUP, 5, 0 );

CheckBoxKUF = new QCheckBox( profileButtonGroup, "CheckBoxKUF" );
CheckBoxKUF->setText( "Работа с КУФ" );

profileButtonGroupLayout->addWidget( CheckBoxKUF, 3, 0 );

CheckBoxOtvKUF = new QCheckBox( profileButtonGroup, "CheckBoxOtvKUF" );
CheckBoxOtvKUF->setText( "Ответственный за КУФ" );

profileButtonGroupLayout->addWidget( CheckBoxOtvKUF, 2, 0 );

```

```

CheckBoxNVO = new QCheckBox( profileButtonGroup, "CheckBoxNVO" );
CheckBoxNVO->setText( "Работа с HBO и ТГО" );

profileButtonGroupLayout->addWidget( CheckBoxNVO, 4, 0 );
vbox->addWidget( profileButtonGroup );

macUser = NULL;
macUser = new mac_user;
memset(macUser, 0, sizeof(mac_user));

macButtonGroup = new Q3ButtonGroup( frame, "profileButtonGroup" );
macButtonGroup->setTitle( "Мандатный уровень безопасности" );
macButtonGroup->setAlignment( Qt::AlignHCenter );
macButtonGroup->setColumnLayout(0, Qt::Vertical );
macButtonGroup->layout()->setSpacing( 6 );
macButtonGroup->layout()->setMargin( 11 );
macButtonGroupLayout = new QGridLayout( macButtonGroup->layout() );
macButtonGroupLayout->setAlignment( Qt::AlignTop );

QLabel *LabelMAC = new QLabel( macButtonGroup, "LabelMAC" );
LabelMAC->setText("Максимальный уровень:");
macButtonGroupLayout->addWidget( LabelMAC, 0, 0 );

ComboBoxMAC = new QComboBox( macButtonGroup, "ComboBoxMAC" );
QStringList sl;
sl << "Уровень=0(Несекретно)" << "Уровень=1(Секретно)" <<
"Уровень=2(Сов.Секретно)";
ComboBoxMAC->addItem(sl);
ComboBoxMAC->setCurrentIndex(-1);
macButtonGroupLayout->addWidget( ComboBoxMAC, 0, 1 );

QPushButton *PushButtonMAC = new QPushButton( macButtonGroup,
"PushButtonMAC" );
PushButtonMAC->setText("Установить");
connect( PushButtonMAC, SIGNAL( clicked() ), this,
SLOT( fromComboBoxMAC() ) );
macButtonGroupLayout->addWidget( PushButtonMAC, 0, 2 );

vbox->addWidget( macButtonGroup );
GroupBox1Layout->addWidget( frame );

spUserControlFormLayout->addWidget( GroupBox1 );

fileLoadAction = new QAction( this, "fileLoadAction" );
fileLoadAction->setIconSet( QIconSet( image1 ) );
fileLoadAction->setText( "Загрузить ТСД" );
fileLoadAction->setMenuText( "Загрузить" );
fileLoadAction->setAccel( 0 );
fileSaveAction = new QAction( this, "fileSaveAction" );
fileSaveAction->setIconSet( QIconSet( image2 ) );
fileSaveAction->setText( "Сохранить ТСД" );
fileSaveAction->setMenuText( ( "Сохранить" ) );
fileSaveAction->setAccel( 0 );
fileSaveAsAction = new QAction( this, "fileSaveAsAction" );
fileSaveAsAction->setIconSet( QIconSet( image3 ) );
fileSaveAsAction->setText( "Сохранить ТСД как" );
fileSaveAsAction->setMenuText( ( "Сохранить как" ) );
fileSaveAsAction->setAccel( 0 );
fileSendAction = new QAction( this, "fileSendAction" );

```

```

fileSendAction->setIconSet( QIconSet( image4 ) );
fileSendAction->setText( "Отослать ТСД выбранному ДЛ" );
fileSendAction->setMenuText( "Отослать" );
fileSendAction->setAccel( 0 );
fileExitAction = new QAction( this, "fileExitAction" );
fileExitAction->setIconSet( QIconSet( image5 ) );
fileExitAction->setText( "Выход" );
fileExitAction->setMenuText( "Выход" );
fileExitAction->setAccel( 0 );
usersNewAction = new QAction( this, "usersNewAction" );
usersNewAction->setIconSet( QIconSet( image6 ) );
usersNewAction->setText( "Добавить пользователя" );
usersNewAction->setMenuText( "Добавить" );
usersNewAction->setAccel( 0 );
usersDeleteAction = new QAction( this, "usersDeleteAction" );
usersDeleteAction->setIconSet( QIconSet( imageUserDelete ) );
usersDeleteAction->setText( "Удалить пользователя" );
usersDeleteAction->setMenuText( "Удалить" );
usersDeleteAction->setAccel( 0 );

tsdDeleteAction = new QAction( this, "tsdDeleteAction" );
tsdDeleteAction->setIconSet( QIconSet( imageTsdDelete ) );
tsdDeleteAction->setText( "Удалить всех пользователей" );
tsdDeleteAction->setMenuText( "Удалить всех" );
tsdDeleteAction->setAccel( 0 );

tsdGenerateAction = new QAction( this, "tsdGenerateAction" );
tsdGenerateAction->setIconSet( QIconSet( imageTsdGenerate ) );
tsdGenerateAction->setText( "Вызвать помощника создания ТСД" );
tsdGenerateAction->setMenuText( "Мастер ТСД" );
tsdGenerateAction->setAccel( 0 );

helpContentsAction = new QAction( this, "helpContentsAction" );
helpContentsAction->setIconSet( QIconSet( image8 ) );
helpContentsAction->setText( "Информация" );
helpContentsAction->setMenuText( "Информация" );
helpContentsAction->setAccel( Qt::Key_F1 );
helpAboutAction = new QAction( this, "helpAboutAction" );
helpAboutAction->setIconSet( QIconSet( image9 ) );
helpAboutAction->setText( "О программе" );
helpAboutAction->setMenuText( ( "О программе..." ) );
helpAboutAction->setAccel( 0 );
usersPasswordAction = new QAction( this, "usersPasswordAction" );
usersPasswordAction->setIconSet( QIconSet( image10 ) );
usersPasswordAction->setText( "Изменить пароль" );

toolBar = addToolBar( "toolbar" );

toolBar->setLabel( ( "Опции" ) );
fileLoadAction->addTo( toolBar );
fileSaveAction->addTo( toolBar );
fileSaveAsAction->addTo( toolBar );
fileSendAction->addTo( toolBar );
toolBar->addSeparator();
fileExitAction->addTo( toolBar );
toolBar->addSeparator();
usersNewAction->addTo( toolBar );
usersDeleteAction->addTo( toolBar );

```

```

tsdDeleteAction->addTo( toolBar );
tsdGenerateAction->addTo( toolBar );
usersPasswordAction->addTo( toolBar );
toolBar->addSeparator();
helpContentsAction->addTo( toolBar );

menubar = new QMenuBar( this, "menubar" );
setMenuBar(menubar);

fileMenu = new QMenu( "ТЦД", this );
fileLoadAction->addTo( fileMenu );
fileSaveAction->addTo( fileMenu );
fileSaveAsAction->addTo( fileMenu );
fileMenu->insertSeparator();
fileSendAction->addTo( fileMenu );
fileMenu->insertSeparator();
fileExitAction->addTo( fileMenu );
menubar->addMenu( fileMenu );

usersMenu = new QMenu( "Пользователи" , this );
usersNewAction->addTo( usersMenu );
usersDeleteAction->addTo( usersMenu );
tsdDeleteAction->addTo( usersMenu );
tsdGenerateAction->addTo( usersMenu );
usersMenu->insertSeparator();
usersPasswordAction->addTo( usersMenu );
menubar->addMenu( usersMenu );

helpMenu = new QMenu( "Помощь" , this );
helpContentsAction->addTo( helpMenu );
helpMenu->insertSeparator();
helpAboutAction->addTo( helpMenu );
menubar->addMenu( helpMenu );

connect( helpAboutAction, SIGNAL( activated() ), this, SLOT( helpAbout()
) );
connect( helpContentsAction, SIGNAL( activated() ), this,
SLOT( helpContentsSlot() ) );
connect( usersDeleteAction, SIGNAL( activated() ), this,
SLOT( usersDeleteSlot() ) );
connect( tsdDeleteAction, SIGNAL( activated() ), this,
SLOT( tsdDeleteSlot() ) );
connect( tsdGenerateAction, SIGNAL( activated() ), this,
SLOT( tsdGenerateSlot() ) );
connect( usersNewAction, SIGNAL( activated() ), this,
SLOT( usersNewSlot() ) );
connect( fileExitAction, SIGNAL( activated() ), this,
SLOT( fileExitSlot() ) );
connect( fileSendAction, SIGNAL( activated() ), this,
SLOT( fileSendSlot() ) );
connect( fileSaveAsAction, SIGNAL( activated() ), this,
SLOT( fileSaveAsSlot() ) );
connect( fileSaveAction, SIGNAL( activated() ), this,
SLOT( fileSaveSlot() ) );
connect( fileLoadAction, SIGNAL( activated() ), this,
SLOT( fileLoadSlot() ) );
connect( CheckBoxBL, SIGNAL( toggled(bool) ), this,
SLOT( changedBL_Slot(bool) ) );
connect( CheckBoxOtvSZI, SIGNAL( toggled(bool) ), this,
SLOT( changedOtvSZI_Slot(bool) ) );

```

```

        connect( CheckBoxVni, SIGNAL( toggled(bool) ), this,
        SLOT( changedVni_Slot(bool) ) );
        connect( CheckBoxOtvKUF, SIGNAL( toggled(bool) ), this,
        SLOT( changedOtvKUF_Slot(bool) ) );
        connect( CheckBoxKUF, SIGNAL( toggled(bool) ), this,
        SLOT( changedKUF_Slot(bool) ) );
        connect( CheckBoxNVO, SIGNAL( toggled(bool) ), this,
        SLOT( changedNVO_Slot(bool) ) );
        connect( CheckBoxUP, SIGNAL( toggled(bool) ), this,
        SLOT( changedUP_Slot(bool) ) );
        connect( CheckBoxUO, SIGNAL( toggled(bool) ), this,
        SLOT( changedUO_Slot(bool) ) );
        connect( CheckBoxPLO, SIGNAL( toggled(bool) ), this,
        SLOT( changedPLO_Slot(bool) ) );
        connect( CheckBoxPLOG, SIGNAL( toggled(bool) ), this,
        SLOT( changedPLOG_Slot(bool) ) );
        connect( CheckBoxPUS, SIGNAL( toggled(bool) ), this,
        SLOT( changedPUS_Slot(bool) ) );
        connect( CheckBoxPLM, SIGNAL( toggled(bool) ), this,
        SLOT( changedPLM_Slot(bool) ) );
        connect( CheckBoxSVO, SIGNAL( toggled(bool) ), this,
        SLOT( changedSVO_Slot(bool) ) );
        connect( CheckBoxSVP, SIGNAL( toggled(bool) ), this,
        SLOT( changedSVP_Slot(bool) ) );
        connect( CheckBoxVPO, SIGNAL( toggled(bool) ), this,
        SLOT( changedVPO_Slot(bool) ) );
        connect( CheckBoxVPP, SIGNAL( toggled(bool) ), this,
        SLOT( changedVPP_Slot(bool) ) );
        connect( usersTable, SIGNAL( valueChanged(int,int) ), this,
        SLOT( tableViewValueChangedSlot(int,int) ) );
        connect( usersTable, SIGNAL( keyPressEvent(const QString&) ), this, SLOT(
        tableViewKeyPressCellSlot(const QString&) ) );
        connect( usersTable, SIGNAL( currentChanged(int,int) ), this,
        SLOT( tableViewCurrentChangedSlot(int,int) ) );
        connect( usersPasswordAction, SIGNAL( activated() ), this,
        SLOT( usersPasswordSlot() ) );

        tlistSetChanged(false);
        prevRow = 99999;
        inCheck = false;
        defaultTSDtoTable();

        itIsSzi = sp.usr()->currentName() == sp.usr()->otvSziName();
        tsdGenerateAction->setEnabled(itIsSzi);
        wizardExist = false;
    }

    spUserControlForm::~spUserControlForm()
    {
        delete macUser;
        macUser = NULL;
    }

    void spUserControlForm::changeBit(uint cap, bool f){
        uint& prf = tlist[usersTable->currentRow()].prf;
        bool ff = prf & cap;
        if(ff != f){
            if(f) prf |= cap;
            else prf &= ~cap;
            tlistSetChanged(true);
        }
    }

```

```

}
void spUserControlForm::changedBL_Slot(bool f)
{
    changeBit(CP_BLK, f);
}

void spUserControlForm::changedKUF_Slot(bool f)
{
    changeBit(CP_KUF, f);
}

void spUserControlForm::changedNVO_Slot(bool f)
{
    changeBit(CP_NVO, f);
}

void spUserControlForm::changedVni_Slot(bool f)
{
    changeBit(CP_VNI, f);
}

void spUserControlForm::changedOtvKUF_Slot(bool f)
{
    changeBit(CP_OKUF, f);
}

void spUserControlForm::changedOtvSZI_Slot(bool f)
{
    changeBit(CP_OSZI, f);
}

void spUserControlForm::changedPLM_Slot(bool f)
{
    changeBit(CP_PLM, f);
}

void spUserControlForm::changedPLOG_Slot(bool f)
{
    changeBit(CP_PLOG, f);
}

void spUserControlForm::changedPLO_Slot(bool f)
{
    changeBit(CP_PLO, f);
}

void spUserControlForm::changedPUS_Slot(bool f)
{
    changeBit(CP_PLUS, f);
}

void spUserControlForm::changedSVO_Slot(bool f)
{
    changeBit(CP_SVO, f);
}

void spUserControlForm::changedSVP_Slot(bool f)
{
    changeBit(CP_SVP, f);
}

```

```

void spUserControlForm::changedUO_Slot(bool f)
{
    changeBit(CP_UO, f);
}

void spUserControlForm::changedUP_Slot(bool f)
{
    changeBit(CP_PRN, f);
}

void spUserControlForm::changedVPO_Slot(bool f)
{
    changeBit(CP_VPO, f);
}

void spUserControlForm::changedVPP_Slot(bool f)
{
    changeBit(CP_VPP, f);
}

void spUserControlForm::fileExitSlot()
{
    if(tlistChanged){
        int exit=QMessageBox::information(this, "Выход...",
                                           "Сохранить_изменения?",
                                           QMessageBox::Ok, QMessageBox::Cancel);

        if (exit==QMessageBox::Ok){
            if(!fileSaveSlot())
                return;
        }
    }
    hide();
}

void spUserControlForm::fileLoadSlot()
{
    QString s = QFileDialog::getOpenFileName(
        TSD_PATH,
        "Тсд файлы (*.sp)",
        this,
        "Загрузка ТсД файла..."
        "Выберите имя файла для загрузки" );
    if(!s.isEmpty()){
        sp.usr()->tsd.recreate(s);
        defaultTSDtoTable();
    }
}

bool spUserControlForm::fileSaveAsSlot()
{
    QString s = QFileDialog::getSaveFileName(
        TSD_PATH,
        "Тсд файлы (*.sp)",
        this,
        "Сохранить ТсД файл как..."
        "Выберите имя под которым хотите сохранить" );
    if(!s.isEmpty()){
        QFile fi(s);
        if(fi.extension().isEmpty())
            s += ".sp";
        return sp.usr()->tsd.save(s);
    }
}

```

```

    }
    return false;
}

bool spUserControlForm::fileSaveSlot()
{
    bool ok = true;
    if(tlistChanged){
        if(ok = tableCheck()){
            spTsd& tsd = sp.usr()->tsd;
            tsd.clear();
            QList<spTsdRecord>::iterator it;
            for ( it = tlist.begin(); it != tlist.end(); ++it ){
                spTsdRecord& r = *it;
                r.ord = tsd.order(r.post);
                tsd.insert(r.post, new spTsdRecord(r));
            }
            ok = tsd.save();
            sp.reg("СЗИ", "Управление пользователями", "Изменена ТСД");
            if(ok){
                sp.usr()->tsdMakeActual();
                tlistSetChanged(false);
            }
        }
    }
    return ok;
}

bool spUserControlForm::fileSendSlot()
{
    QString post = tlist[usersTable->currentRow()].post;
#ifdef TESTITSELF
    if(post == sp.usr()->currentName())
        return true;
#endif

    if(QMessageBox::Ok != QMessageBox::information(0, ("Рассылка
ТСД"),
                                                "Вы действительно хотите отослать ТСД к
"+ post,
                                                QMessageBox::Ok, QMessageBox::Cancel)
    ) return false;

    QString file = TSD_FILE_NAME;
    QString msg;
    if(!exportFile(post, file)){
        msg = QString("Ошибка доступа к ДЛ ") + post;
        QMessageBox::critical(0,
                              "Рассылка ТСД...",
                              msg,
                              QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );

        return false;
    }
    msg = QString("ТСД отослана к ДЛ ") + post;
    QMessageBox::information(0,
                              "Рассылка ТСД...",
                              msg,
                              QMessageBox::Ok, QMessageBox::NoButton,
QMessageBox::NoButton );
}

```

```

        return true;
    }

void spUserControlForm::helpAbout()
{
    qWarning( "spUserControlForm::helpAbout(): Not implemented yet!" );
}

void spUserControlForm::helpContentsSlot()
{
}

void spUserControlForm::usersPasswordSlot()
{
    if(!fileSaveSlot())
        return;
    QString post = tlist[usersTable->currentRow()].post;
    spPasswordForm pwf(post);
    if(pwf.exec() == QDialog::Accepted){
        QString sign = pwf.getSign();
        if(!sp.usr()->osUserChangePassword(post, sign))
            QMessageBox::critical(0,
                                   "Смена пароля...",
                                   "Ошибка смены пароля ДЛ",
                                   QMessageBox::Ok, QMessageBox::NoButton,
                                   QMessageBox::NoButton );
    }
}

void spUserControlForm::tableCurrentChangedSlot(int row,int)
{
    spTsdRecord r = tlist[row];
    toCheckBox(r.prf);
    toComboBoxMAC(r.post);
    QString oszi(OTVSZINAME);
    if(r.post == oszi || r.ord/100000 > 3 || r.post == sp.currentName())
    {
        profileButtonGroup->setEnabled(false);
        macButtonGroup->setEnabled(false);
    }
    else
    {
        profileButtonGroup->setEnabled(true);
        macButtonGroup->setEnabled(true);
    }

    if(r.post == oszi || r.post == sp.currentName() || !
    sp.findInTva(r.post) || r.ord/100000 > 3){
        fileSendAction->setEnabled(false);
    }
    else
        fileSendAction->setEnabled(true);

    if(r.post == oszi || r.post == sp.currentName()){
        usersDeleteAction->setEnabled(false);
    }
    else
        usersDeleteAction->setEnabled(true);

    if(r.ord/100000 > 3 || !r.prf){

```

```

        usersPasswordAction->setEnabled(false);
    }
    else{
        usersPasswordAction->setEnabled(true);
    }

    if(itIsSzi){
        fileSendAction->setEnabled(false);
    }

    if(prevRow != row){
prevRow = row;
    if(tlistChanged && !inCheck){
        tableCheck();
    }
}
}

void spUserControlForm::tableKeyPressCellSlot(const QString& txt){
    int row = usersTable->currentRow();
    int col = usersTable->currentColumn();
    QLineEdit* e = (QLineEdit*)usersTable->cellWidget(row, col);
    spTsdRecord& r = tlist[row];
    if(e){
        if(col==0){//Post
            r.post = txt;
            if(!sp.latin()){        //Принудительно в РУССКИЙ
                r.post = latRUSS(r.post);
            }
            r.post.truncate(16);//Принудительно <=16
            e->setText(r.post);
            tlistSetChanged(true);
        }
        if(col==1){//Sign
            r.sign = txt;
            if(!sp.latin()){        //Принудительно в РУССКИЙ
                r.sign = latRUSS(r.sign);
            }
            r.sign.truncate(6);//Принудительно <=16
            e->setText(r.sign);
            tlistSetChanged(true);
        }
    }
}

void spUserControlForm::tableValueChangedSlot(int row, int col)
{
    qDebug("tableValueChangedSlot %d %d", row, col);
    spTsdRecord& r = tlist[row];
    switch(col){
    case 0 : //Post
        if(r.post == sp.currentName()){
            QMessageBox::critical(0,
                "Ошибка...",
                "Вы не можете редактировать имя текущего ДЛ",
                QMessageBox::Ok, QMessageBox::NoButton,
                QMessageBox::NoButton );

            usersTable->setText(row, col, r.post);
            break;
        }
    }
}

```

```

        if(r.post == OTVSZINAME){
            QMessageBox::critical(0,
                                   "Ошибка...",
                                   "Вы не можете редактировать имя ДЛ ОСЗИ",
                                   QMessageBox::Ok, QMessageBox::NoButton,
                                   QMessageBox::NoButton );

            usersTable->setText(row, col, r.post);
            break;
        }

        if(!sp.latin()){ //Принудительно в
РУССКИЙ
            r.post = latRUSS(r.post);
        }
        r.post.truncate(16); //Принудительно <=16

        usersTable->setText(row, col, r.post);
        r.ord = sp.usr()->tsd.order(r.post);
        if(r.ord/100000 > 3){
            usersPasswordAction-
>setEnabled(false);
        }
        else{
            usersPasswordAction-
>setEnabled(true);
        }

        tlistSetChanged(true);

        break;
    case 1 : //Sign
        r.sign = usersTable->text(row, col);
        if(!sp.latin()){ //Принудительно в
РУССКИЙ
            r.sign = latRUSS(r.sign);
        }
        r.sign.truncate(6); //Принудительно <=16

        usersTable->setText(row, col, r.sign);
        tlistSetChanged(true);
        break;
    }
}

void spUserControlForm::usersDeleteSlot()
{
    int cr = usersTable->currentRow();
    spTsdRecord& r = tlist[cr];
    if(r.post == sp.currentName()){
        QMessageBox::critical(0,
                               "Ошибка...",
                               "Вы не можете удалить текущее ДЛ",
                               QMessageBox::Ok, QMessageBox::NoButton,
                               QMessageBox::NoButton );

        return;
    }
    if(r.post == OTVSZINAME){
        QMessageBox::critical(0,
                               "Ошибка...",
                               "Вы не можете удалить ДЛ ОСЗИ",

```

```

        QMessageBox::Ok, QMessageBox::NoButton,
        QMessageBox::NoButton );

        return;
    }
    usersTable->removeRow(cr);
    tlist.remove(tlist.at(cr));
    tlistSetChanged(true);
}

void spUserControlForm::usersNewSlot()
{
    if(tableCheck()){
        tlistSetChanged(false);
        int cr = usersTable->numRows();
        usersTable->insertRows(cr);
        usersTable->ensureCellVisible ( cr, 0 );
        spTsdRecord r;
        r.post = "?";
        r.sign = "??";
        r.id = tlist.count();
        tlist.append(r);
        usersTable->setText(cr, 0, r.post);
        usersTable->setText(cr, 1, r.sign);
        usersTable->setCurrentCell(cr, 0);
        tlistSetChanged(true);
    }
}

void spUserControlForm::showModal(){
    show();
    while(!isHidden() || wizardExist)
        qApp->processEvents();
}

void spUserControlForm::defaultTSDtoTable(){
    spTsdRecord* r;
    tlist.clear();
    Q3DictIterator<spTsdRecord> it( sp.usr()->tsd );
    int rows = sp.usr()->tsd.count();
    usersTable->setNumRows(rows);
    for(int row=0 ; (r=it.current()); ++it, row++ ){
        tlist += *r;
    }
    qSort(tlist.begin(),tlist.end(), qLessRecord<spTsdRecord>());
    for(int row=0 ; row<rows; row++){
        spTsdRecord& rc = tlist[row];
        if(rc.post == sp.currentName() || rc.post == sp.usr()-
>otvSziName()){
            usersTable->setItem(row, 0, new
Q3TableWidgetItem(usersTable, QTableWidgetItem::Never));
        }
        usersTable->setText(row, 0, rc.post);
        usersTable->setText(row, 1, rc.sign);
    }
    tlistSetChanged(false);
    tableCurrentChangedSlot(0,0);
}

void spUserControlForm::tsdToTable(QList<spTsdRecord>& externList){
    tlist.clear();

```

```

QString osziName = sp.usr()->otvSziName();
bool yesOszi = false;
    QList<spTsdRecord>::iterator it;
    for ( it = externList.begin(); it != externList.end(); ++it ){
        spTsdRecord& r0 = *it;
if(r0.post == osziName){
        yesOszi = true;
        }
        r0.ord = sp.usr()->tsd.order(r0.post);
        if(r0.ord/100000 > 3){
            r0.prf = 0;
        }
        else{
            int dol = r0.ord%10;
            switch(dol){
                case 1:
r0.prf = CP_BLK |
            CP_VPO |
            CP_KUF |
            CP_NVO |
            CP_SVO |
            CP_UO |
            CP_PRN;
                    break;
                case 2:
r0.prf = CP_BLK |
            CP_VPP |
            CP_KUF |
            CP_NVO |
            CP_SVP |
            CP_PRN;
                    break;
                case3:
r0.prf = CP_BLK |
            CP_PLM |
            CP_PLO |
            CP_PLOG |
            CP_PLUS |
            CP_VPP |
            CP_OSZI |
            CP_OKUF |
            CP_NVO |
            CP_SVP |
            CP_PRN;
                    break;
                case4:
r0.prf = CP_BLK |
            CP_PLM |
            CP_PLO |
            CP_PLOG |
            CP_PLUS |
            CP_VPP |
            CP_OKUF |
            CP_NVO |
            CP_SVP |
            CP_PRN;
                    if(r0.post.contains(S_SOPR)){
                        r0.prf |= CP_OSZI;
                    }
                    break;
                default:

```

```

        if(r0.prf & CP_UO){
            r0.prf = CP_BLK |
                CP_VPO |
                CP_KUF |
                CP_NVO |
                CP_SVO;
        }
        break;
    }

    }

    tlist << r0;
}

    if(!yesOszi){
        spTsdRecord r;
        r.post = OTVSZINAME;
        r.prf = CP_OSZI;
        r.sign = DEFSIGNSZI;
        r.ord = 0;
        r.id = 0;
        tlist << r;
    }
int rows = tlist.count();
qSort(tlist.begin(), tlist.end(), qLessRecord<spTsdRecord>());

usersTable->setNumRows(rows);
for(int row=0 ; row<rows; row++){
    spTsdRecord& rc = tlist[row];
    if(rc.post == sp.currentName() || rc.post == sp.usr()-
>otvSziName()){
        usersTable->setItem(row, 0, new
Q3TableWidgetItem(usersTable, QTableWidgetItem::Never));
    }
    usersTable->setText(row, 0, rc.post);
    usersTable->setText(row, 1, rc.sign);
}

tlistSetChanged(true);
usersTable->setCurrentCell(0,0);
tableCurrentChangedSlot(0,0);
tsdGenerateAction->setEnabled(false);
fileSaveSlot();
}

void spUserControlForm::toCheckBox(uint prf){
    CheckBoxBL->setChecked    (prf & CP_BLK);
    CheckBoxPLO->setChecked    (prf & CP_PLO);
    CheckBoxPLOG->setChecked    (prf & CP_PLOG);
    CheckBoxPUS->setChecked    (prf & CP_PLUS);
    CheckBoxOtvSZI->setChecked (prf & CP_OSZI);
    CheckBoxVni->setChecked    (prf & CP_VNI);
    CheckBoxPLM->setChecked    (prf & CP_PLM);
    CheckBoxSVO->setChecked    (prf & CP_SVO);
    CheckBoxSVP->setChecked    (prf & CP_SVP);
    CheckBoxVPO->setChecked    (prf & CP_VPO);
    CheckBoxVPP->setChecked    (prf & CP_VPP);
    CheckBoxUO->setChecked     (prf & CP_UO);
    CheckBoxUP->setChecked     (prf & CP_PRN);
    CheckBoxKUF->setChecked     (prf & CP_KUF);
    CheckBoxOtvKUF->setChecked (prf & CP_OKUF);
}

```

```

        CheckBoxNVO->setChecked    (prf & CP_NVO);
    }

uint spUserControlForm::fromCheckBox() {
    uint prf = 0;
    if (CheckBoxBL->isChecked())    prf |= CP_BLK;
    if (CheckBoxPLO->isChecked())    prf |= CP_PLO;
    if (CheckBoxPLOG->isChecked())   prf |= CP_PLOG;
    if (CheckBoxPUS->isChecked())    prf |= CP_PLUS;
    if (CheckBoxOtvSZI->isChecked()) prf |= CP_OSZI;
    if (CheckBoxVni->isChecked())    prf |= CP_VNI;
    if (CheckBoxPLM->isChecked())    prf |= CP_PLM;
    if (CheckBoxSVO->isChecked())    prf |= CP_SVO;
    if (CheckBoxSVP->isChecked())    prf |= CP_SVP;
    if (CheckBoxVPO->isChecked())    prf |= CP_VPO;
    if (CheckBoxVPP->isChecked())    prf |= CP_VPP;
    if (CheckBoxUO->isChecked())     prf |= CP_UO;
    if (CheckBoxUP->isChecked())     prf |= CP_PRN;
    if (CheckBoxKUF->isChecked())    prf |= CP_KUF;
    if (CheckBoxOtvKUF->isChecked()) prf |= CP_OKUF;
    if (CheckBoxNVO->isChecked())    prf |= CP_NVO;
    return prf;
}

void spUserControlForm::toComboBoxMAC(QString name)
{
    struct mac_user* mu = NULL;
    memset(macUser, 0, sizeof(mac_user));
    name = name.simplified().replace(" ", "_");
    mu = getmacnam(qPrintable(name));
    if (mu)
    {
        memcpy(macUser, mu, sizeof(mac_user));
        ComboBoxMAC->setCurrentIndex(macUser->max.lev);
    }
    else
    {
        ComboBoxMAC->setCurrentIndex(-1);
    }
}

void spUserControlForm::fromComboBoxMAC()
{
    QString name = tlist[usersTable->currentRow()].post;
    name = name.simplified().replace(" ", "_");

    if (!name.isEmpty() && ComboBoxMAC->currentIndex() != -1)
    {
        macUser->name = (char *)qPrintable(name);
        macUser->max.lev = ComboBoxMAC->currentIndex();
        putmacent(macUser);
    }
}

void spUserControlForm::tlistSetChanged(bool f) {
    tlistChanged = f;

    fileSaveAction->setEnabled(tlistChanged);
    fileSaveAsAction->setEnabled(!tlistChanged);
    fileSendAction->setEnabled(!tlistChanged);
}

```

```

void spUserControlForm::closeEvent(QCloseEvent* /*e*/){
    fileExitSlot();
}

bool spUserControlForm::tableCheck(){
    inCheck = true;
    bool ok = false;
    int n = tlist.count();
    QString err;
    for(int i=0; i<n; i++){
        spTsdRecord& r1 = tlist[i];
        if(r1.post.contains("?")) {
            err = "Не корректное имя ДЛ ";
            err += "'" + r1.post + "'";
            QMessageBox::critical(0,
                                "Проверка ТСД...",
                                err,
                                QMessageBox::Ok, QMessageBox::NoButton,
                                QMessageBox::NoButton );
            usersTable->setCurrentCell(i, 0);
            inCheck = false;
            ok = false;
            return ok;
        }
        if(r1.sign.length()<6){
            err = "Подпись ДЛ должна содержать 6 символов ";
            err += "'" + r1.sign + "'";
            QMessageBox::critical(0,
                                "Проверка ТСД...",
                                err,
                                QMessageBox::Ok, QMessageBox::NoButton,
                                QMessageBox::NoButton );
            usersTable->setCurrentCell(i, 1);
            inCheck = false;
            ok = false;
            return ok;
        }
        if(r1.ord/100000<4 && r1.prf==0){
            err = "Не заданы полномочия ДЛ ";
            err += "'" + r1.post + "'";
            QMessageBox::critical(0,
                                "Проверка ТСД...",
                                err,
                                QMessageBox::Ok, QMessageBox::NoButton,
                                QMessageBox::NoButton );
            usersTable->setCurrentCell(i, 0);
            inCheck = false;
            ok = false;
            return ok;
        }
        for(int j=i+1; j<n; j++){
            spTsdRecord& r2 = tlist[j];
            if(r1.post == r2.post){
                err = "Дубликатное имя ДЛ ";
                err += "'" + r1.post + "'";
                QMessageBox::critical(0,
                                    "Проверка ТСД...",
                                    err,
                                    QMessageBox::Ok, QMessageBox::NoButton,
                                    QMessageBox::NoButton );
                usersTable->setCurrentCell(j, 0);
            }
        }
    }
}

```

```

        inCheck = false;
        ok = false;
        return ok;
    }
    if(r1.sign == r2.sign){
        err = "Дубликатная подпись ДЛ ";
        err += "'" + r1.sign + "'";
        QMessageBox::critical(0,
            "Проверка ТСД...",
            err,
            QMessageBox::Ok, QMessageBox::NoButton,
            QMessageBox::NoButton );
        usersTable->setCurrentCell(j, 1);
        inCheck = false;
        ok = false;
        return ok;
    }
}
ok = true;
}
inCheck = false;
return ok;
}
void spUserControlForm::tsdDeleteSlot()
{
    QString oszi = sp.usr()->otvSziName();
    QString curr = sp.currentName();

    QList<spTsdRecord> tmp;

    QList<spTsdRecord>::iterator it;
    for ( it = tlist.begin(); it != tlist.end(); ++it ){
        spTsdRecord& r = *it;
        if(r.post == oszi) tmp << r;
        else
            if(r.post == curr) tmp << r;
    }
    tlist = tmp;
    tlistSetChanged(true);

    int rows = tlist.count();
    usersTable->setNumRows(rows);
    for(int row=0 ; row<rows; row++ ){
        spTsdRecord& rc = tlist[row];
        if(rc.post == curr || rc.post == oszi){
            usersTable->setItem(row, 0, new
Q3TableWidgetItem(usersTable, QTableWidgetItem::Never));
        }
        usersTable->setText(row, 0, rc.post);
        usersTable->setText(row, 1, rc.sign);
    }
}
void spUserControlForm::tsdGenerateSlot()
{
    qDebug("tsdGenerateSlot");
    tsdGenerateAction->setEnabled(false);
    wizardExist = true;
    hide();
    sp.generateTsd(true);
    wizardExist = false;
    tsdGenerateAction->setEnabled(true);
}

```

```
}
```

sp_users – управление пользователями ОС МСВС

```
#include "sp_users.h"
#include <pwd.h>
#include <grp.h>
#include <sys/types.h>
#include "sp_.h"
#include <unistd.h>
#include <crypt.h>
#include "sp_proc.h"
#include <qapplication.h>
#include <qregexp.h>
#include <qprogressdialog.h>
#include <qfile.h>
#include <qfileinfo.h>

template <typename spTsdRecord>
class qLessRecord
{
public:
    inline bool operator()(const spTsdRecord &t1, const spTsdRecord &t2)
const
    {
        return (t1.id < t2.id);
    }
};

QString developer;
QString nameU(QString name){
    QString ret = name;
    ret.replace(" ", "_");
    return ret;
}
QString nameS(QString name){
    QString ret = name;
    ret.replace("_", " ");
    return ret;
}

spUsers::spUsers(){
    connect(&tsd, SIGNAL(changed_signal()), this, SLOT(tsdChanged_slot()));
    tsdChanged_ = true;
    state = armStateUnknown;
}
spUsers::~spUsers(){
}

QString spUsers::currentName(){
    if(curName.isEmpty()){
#ifdef DEVELOPER
        QFile file( "login" );
        bool ok = file.open( IO_ReadOnly );
        if(ok){
            QString tmp;
            QTextStream s( &file );
            tmp = s.readLine();
            file.close();
            curName = tmp.simplifyWhiteSpace();

```

```

        if(!curName.isEmpty()){
            return nameS(curName);
        }
    }
#endif

    QStringList command;
    command<<"whoami";
    ProcessManager pr(command);
    while(pr.isRunning())
        qApp->processEvents();
    if(pr.answer.isEmpty()){
        qDebug("Error executing: whoami");
        curName = "";
    }
    else
        curName = pr.answer[0];
}
return nameS(curName);
}
QString spUsers::otvSziName(){
    return OTVSZINAME;
}
QString spUsers::otvSziSign(){
    QString sign;
    QString cn = OTVSZINAME;
    spTsdRecord* r = tsd[cn];
    if(r){
        sign = r->sign;
    }
    return sign;
}

uint spUsers::groupId(QString gn){
    struct group* gr = getgrnam(gn);
    uint grId = 0;
    if(gr){
        grId = gr->gr_gid;
    }
    return grId;
}

QStringList spUsers::osUsersList(uint gId){
    QStringList l;
    struct passwd* pw = 0;
    if(gId){
        setpwent();
        while((pw=getpwent())){
            if(pw->pw_gid==gId
#ifdef DEVELOPER
            || (developer!="root" && QString(pw->pw_name) == developer)
#endif
            )
            {
                l<<nameS(pw->pw_name);
            }
        }
        endpwent();
    }
    return l;
}
QStringList spUsers::osUsersList(QString gn){

```

```

    return osUsersList( groupId(gn) );
}

bool spUsers::tsdIsActual(){
    bool ok = false;
    QStringList osUsers = osUsersList();
    if(!osUsers.isEmpty()){
        for ( QStringList::Iterator it = osUsers.begin(); it != osUsers.end(); +
+it ) {
            const QString& its = *it;
            if(tsd[its] == 0){
                return ok;
            }
        }
        Q3DictIterator<spTsdRecord> it( tsd );
        spTsdRecord* r;
        for( ; (r=it.current()); ++it ){
            if(r->prf)
                if(osUsers.contains(r->post)==0){
                    return ok;
                }
        }
        ok = true;
    }
    return ok;
}

bool spUsers::tsdMakeActual(){
    bool ok = false;
    QStringList osUsers = osUsersList();
    int steps = osUsers.count() + tsd.count();
    int step = 0;
    Q3ProgressDialog progress( "Корректировка системных пользователей...", 0,
steps,
                                0, "progress", true,
Qt::WindowStaysOnTopHint );

    progress.setCaption("СЗИ");
    progress.setAutoClose(true);

    if(!osUsers.isEmpty()){
        for ( QStringList::Iterator it = osUsers.begin(); it != osUsers.end(); +
+it ) {
            progress.setProgress( step++ );
            QApplication->processEvents();

            if(tsd[*it] == 0){
                if(!osUserDel(*it))
                    return ok;
            }
        }
    }
    Q3DictIterator<spTsdRecord> it( tsd );
    spTsdRecord* r;
    for( ; (r=it.current()); ++it ){
        progress.setProgress( step++ );
        QApplication->processEvents();
        if(r->prf)
            if(osUsers.contains(r->post)==0){
                if(!osUserAdd(r))
                    return ok;
            }
    }
}

```

```

        }
    }
    progress.setProgress( steps );
    ok = true;
    return ok;
}

bool spUsers::osUserAdd(QString name, QString bGroup, QStringList
iGroups, QString password) {
#ifdef DEVELOPER
    if(name == developer)
        return true;
#endif
    if(name == "root")
        return true;
    if(currentName() == name)
        return true;
    bool ok = groupAdd(bGroup);
    if(ok) {

        ok = osUserDel(name);
        if(ok) {
            QStringList command;
            if(!currentIsRoot())
                command << "sudo";
            command << "/usr/sbin/useradd" << "-g" << bGroup;
            if(!iGroups.isEmpty()) {
                command << "-G" ;
                command << iGroups.join(",");
            }

            char* hash = crypt(password.ascii(), "ur");
            command << "-p" << hash << nameU(name);
            ProcessManager pr(command);
            while(pr.isRunning())
                qApp->processEvents();
            if(!pr.answer.isEmpty() && !pr.answer[0].isEmpty()) {
                qDebug("Error executing: useradd %s", pr.answer[0].ascii());
                ok = false;
            }
        }
        else {
            command.clear();
            if(!currentIsRoot())
                command << "sudo";
            command << "usermac" << "-m" << "0:2" << nameU(name);
            ProcessManager pr(command);
            while(pr.isRunning())
                qApp->processEvents();

            QString msg = "Добавлен пользователь " + name;
            sp.reg("СЗИ", "Управление пользователями", msg);
        }
    }
}

return ok;
}

bool spUsers::osUserAdd(spTsdRecord* r, QString bGroup, QString password) {
    bool ok = false;
    if(r && r->prf) {
        QStringList iGroups;
        iGroups << "wheel";

```

```

        if(r->prf & (CP_OSZI | CP_NVO | CP_PRN) )
            iGroups << "root";
        ok = osUserAdd(r->post, bGroup, iGroups, password);
    }
    return ok;
}

bool spUsers::osUserDel(QString name){

#ifdef DEVELOPER
    if(name == developer)
        return true;
#endif
    if(name == "root")
        return true;

    bool ok = true;
    if(currentName() != name)
    if(osUserExist(name)){
        QStringList command;
        if(!currentIsRoot())
            command << "sudo";
        command<<"/usr/sbin/userdel" << "-r" << nameU(name);
        ProcessManager pr(command);
        while(pr.isRunning())
            qApp->processEvents();
        if(!pr.answer.isEmpty() && !pr.answer[0].isEmpty()){
            qDebug("Error executing: userdel");
            ok = false;
        }
        else{
            QString msg = "Удален пользователь " + name;
            sp.reg("СЗИ", "Управление пользователями", msg);
        }
    }
    return ok;
}

bool spUsers::groupExist(QString group){
    return (getgrnam (group) != 0);
}

bool spUsers::groupAdd(QString name){
    bool ok = true;
    if(!groupExist(name)){
        QStringList command;
        if(!currentIsRoot())
            command << "sudo";
        command<<"/usr/sbin/groupadd" << name ;
        ProcessManager pr(command);
        while(pr.isRunning())
            qApp->processEvents();
        if(!pr.answer.isEmpty() && !pr.answer[0].isEmpty()){
            qDebug("Error executing: groupadd");
            ok = false;
        }
        else{
            QString msg = "Добавлена группа " + name;
            sp.reg("СЗИ", "Управление пользователями", msg);
        }
    }
}

```

```

        return ok;
    }

bool spUsers::groupDel(QString name){
    bool ok = true;
    if(groupExist(name)){
        QStringList command;
        if(!currentIsRoot()){
            command << "sudo";
            command<<"/usr/sbin/groupdel" << name ;
            ProcessManager pr(command);
            while(pr.isRunning()){
                qApp->processEvents();
                if(!pr.answer.isEmpty() && !pr.answer[0].isEmpty()){
                    qDebug("Error executing: groupdel");
                    ok = false;
                }
            }
        }
        else{
            QString msg = "Удалена группа " + name;
            sp.reg("СЗИ", "Управление пользователями", msg);
        }
    }
    return ok;
}

bool spUsers::osUserExist(QString name){
    return (getpwnam (nameU(name)) != 0);
}

bool spUsers::currentUserBelongTo(QString grName){

    struct group* gr = getgrnam (grName);
    bool ok = false;
    if(gr && gr->gr_mem){
        QString nm;
        int i = 0;
        while (gr->gr_mem[i]) {
            nm = (const char *)gr->gr_mem[i];
            if(nameU(currentName()) == nm) { ok = true; break;}
            i++;
        }
    }
    return ok;
}

uint spUsers::prof(const QString& name){
    uint prf = 0;
    if(name == "root")
        prf = 0xffffffff;
    else{
        spTsdRecord* r = tsd[name];
        if(r){
            prf = r->prf;
        }
    }
    return prf;
}

uint spUsers::currentProf(){
    return prof(currentName());
}

QString spUsers::post(const QString& sign){
    QString pst;

```

```

    spTsdRecord* r = rttd()[sign];
    if(r){
        pst = r->post;
    }
    if(pst == "root"){
        pst = OTVSZINAME;
    }
    return pst;
}
QString spUsers::sign(const QString& name){
    QString sgn;
    spTsdRecord* r = 0;
    if(name == "root"){
        r = tsd[ OTVSZINAME ];
    }
    else r = tsd[name];
    if(r){
        sgn = r->sign;
    }
    return sgn;
}
QString spUsers::currentSign(){
    return sign(currentName());
}

bool spUsers::creatingMinimalTSD(){
    tsd.clear();
    spTsdRecord r;
    r.post = OTVSZINAME;

    r.prf = CP_OSZI;

    r.sign = DEFSIGNSZI;
    r.ord = tsd.order(r.post);
    r.id = 0;
    bool ok = tsd.addRecordAndSave(r);

    sp.reg("СЗИ", "Управление пользователями", "Создана ТСД");
    if(ok)
        ok = tsdMakeActual();
    return ok;
}

bool spUsers::controlSudoers(){
    bool ok = false;
    QFile file( "/etc/sudoers" );
    if ( !file.open( IO_ReadOnly ) ) {
        qDebug("Error: open sudoers");
        return ok;
    }
    QTextStream fs(&file);
    while ( !fs.atEnd() ) {
        QString l = fs.readLine();
        if(!l.contains("#")){
            if(l.contains("NOPASSWD:") && l.contains("%wheel")){
                ok = true;
                break;
            }
        }
    }
}

```

```

file.close();

if(!ok){
    if(currentIsRoot()){
        if ( !file.open( IO_ReadWrite ) ) {
            qDebug("Error: open sudoers");
            return false;
        }
        QTextStream s(&file);
        uint sz = file.size();
        file.at(sz);
        s << "%wheel ALL=(ALL) NOPASSWD: ALL\n";
        file.close();
        sp.reg("СЗИ", "Инициализация", "Выполнена");
        ok = true;
    }
    else{
        qDebug("Error: access to sudoers (not root)");
        sp.reg("СЗИ", "Инициализация", "Ошибка: пользователь не 'root'");
    }
}
return ok;
}

bool spUsers::currentIsRoot(){
    return currentName()=="root";
}

bool spUsers::osUserChangePassword(QString name, QString password){
#ifdef DEVELOPER
    if(name == developer)
        return true;
#endif
    if(name == "root")
        return true;

    char* hash = crypt(password.ascii(), "ur");

    QStringList command;
    if(!currentIsRoot())
        command << "sudo";
    command << "/usr/sbin/usermod" << "-p" << hash << nameU(name).ascii() ;
    ProcessManager pr(command);
    while(pr.isRunning())
        QApplication->processEvents();
    if(!pr.answer.isEmpty() && !pr.answer[0].isEmpty()){
        qDebug("Error executing: useradd %s", pr.answer[0].ascii());
        return false;
    }
    else{
        QString msg = "Изменен пароль у "+name;
        sp.reg("СЗИ", "Управление пользователями", msg);
    }
    return true;
}

bool spUsers::armIsBlocked(){
    QFile fi(BLOCKARMF);
    return fi.exists();
}

```

```

bool spUsers::armSetBlocking(bool bl){

    if(bl != armIsBlocked()){
        QStringList command;
        if(!currentIsRoot())
            command << "sudo";
        if(bl){
            command << "touch" << BLOCKARMF;
        }
        else{
            command << "rm" << "-f" << BLOCKARMF;
        }
        ProcessManager pr(command);
        while(pr.isRunning())
            qApp->processEvents();
        if(!pr.answer.isEmpty() && !pr.answer[0].isEmpty()){
            qDebug("Error executing: armSetBlocking %s", pr.answer[0].ascii());
            return false;
        }
    }
    armSetState(bl? armStateBlock : armStateWork);

    QString msg = bl? "Заблокировано" : "Разблокировано";
    sp.reg("СЗИ", "Блокировка АРМ", msg);

    return true;
}

const QList<spTsdRecord>& spUsers::superiors(){
    getLists();
    return superiors_;
}

const QList<spTsdRecord>& spUsers::inferiors(){
    getLists();
    return inferiors_;
}

const Q3Dict<spTsdRecord>& spUsers::rtsd(){
    getLists();
    return rtsd_;
}

int spUsers::order(const QString& name){
    int ord = 0;
    spTsdRecord* r = tsd[name];
    if(r){
        ord = r->ord;
    }
    else{
        if(name=="root")
            return 201;
#ifdef DEVELOPER
        if(name==developer)
            return 201;
#endif
    }
    return ord;
}

int spUsers::currentOrder(){
    return order(currentName());
}

void spUsers::getLists(){

```

```

        if(tsdChanged_){
            inferiors_.clear();
            superiors_.clear();
            rtsd_.clear();
            sortedUsers_.clear();
            int curOrd = currentOrder();
            QString oszi(OTVSZINAME);
            if(curOrd>0){
                Q3DictIterator<spTsdRecord> it( tsd );
                spTsdRecord* r;
                for( ; (r=it.current()); ++it ){
                    if(r->post==oszi) continue;
                    sortedUsers_.append(*r);
#ifdef TESTITSELF
                    if(r->post==currentName()) continue;
#endif
                    if(r->ord == curOrd){
                        superiors_.append(*r);
                        inferiors_.append(*r);
                    }
                    if(r->ord/100000 < curOrd/100000){
                        superiors_.append(*r);
                    }
                    if(r->ord/100000 > curOrd/100000){
                        inferiors_.append(*r);
                    }
                    rtsd_.insert(r->sign, new spTsdRecord(*r));
                }
                qSort(superiors_.begin(), superiors_.end(),
qLessRecord<spTsdRecord>());
                qSort(inferiors_.begin(),
inferiors_.end(),qLessRecord<spTsdRecord>());
                qSort(sortedUsers_.begin(), sortedUsers_.end(),
qLessRecord<spTsdRecord>());
            }

            tsdChanged_ = false;
        }
    }

void spUsers::tsdChanged_slot(){
    tsdChanged_ = true;
    sp.tsdChanged();
}

bool spUsers::armSetState(){
    return armSetState(state);
}

bool spUsers::armSetState(int st){
    state = st;
    bool ok = false;
    QFile file(STATPOST);
    ok = file.open(IO_WriteOnly );
    if(ok){
        QTextStream s(&file);

        codeWrite(currentName(), s);
        s << " " << st;

        file.close();
        QString cmd = "chmod 0666 " STATPOST;

```

```

        system(cmd.ascii());
    }
    return ok;
}
bool spUsers::armGetState(QString& post, int& state){
    bool ok = false;
    QFile file(STATPOST);
    ok = file.open(IO_ReadOnly );
    if(ok){
        QTextStream s(&file);
        codeRead(post, s);
        s >> state;
        file.close();
    }
    return ok;
}
QString spUsers::parentOszName(){
    getLists();
    parentOszName_ = QString::null;
    uint curOrd = currentOrder();
    if(curOrd){
        int ord = curOrd/100000 - 1;
        int n = superiors_.count();
        for(int i=n-1; i>=0; i--){
            const spTsdRecord& r = superiors_[i];
            if(r.ord/100000<=ord){
                if( (r.prf & CP_OSZI) && (sp.findInTva(r.post))){
                    parentOszName_ = r.post;
                    break;
                }
            }
        }
    }
    return parentOszName_;
}
const QList<spTsdRecord>& spUsers::users(){
    getLists();
    return sortedUsers_;
}

static unsigned char a_lin[] =
    "АВВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдежзийклмноп";
static unsigned char b_lin[] = "рстуфхцчшщъыьэюя"
static unsigned char dos_lin(unsigned char c){
    if(c>=0x80 && c<=0xAF) return (unsigned char)a_lin[c-0x80];
    else if(c>=0xE0 && c<=0xEF) return (unsigned char)b_lin[c-0xE0];
    else return c;
}
static unsigned char lin_dos(unsigned char c){
    if(c>=0xC0) return (unsigned char)l_dos[c-0xC0];
    else return c;
}

void koi8_koi7(char *str){
    while (*str) { *str = lin_dos(*str); str++;}
}
void koi7_koi8(char *str){
    while (*str) { *str = dos_lin(*str); str++;}
}
static const char spGamma[256] =

```

```

{
    7,    6,    9,   19,   17,   31,   10,   12,    9,   13,   26,   11,   18,   27,    3,
6,
    28,    2,   20,   24,   27,    8,    7,   13,   22,   26,   14,    3,   19,   31,    9,
26,
    6,   18,   13,   23,   17,   24,    3,   26,    5,   29,    5,   23,   24,    9,   30,
20,
    11,   18,   13,    6,   27,   20,   20,   17,   14,    2,   20,    1,    1,   29,   28,
7,
    16,    9,   30,    1,    1,    1,   28,    7,   30,    1,   30,   23,   10,   28,   11,
22,
    15,   24,   28,   10,   12,   16,   27,   27,   18,   15,   28,   20,   12,   24,   27,
28,
    2,   26,   30,    3,   27,   26,   10,   26,   27,    9,   17,    6,    5,   28,   28,
20,
    21,   24,   30,    1,    9,   25,   28,   27,    8,   25,   15,   21,   17,   11,   17,
19,
    5,   15,   23,    0,    9,    1,   26,    5,   10,   11,   11,   16,    8,    7,    4,
29,
    31,    3,   30,    8,   28,   27,    4,    5,   20,   19,   26,    5,   30,   11,   25,
3,
    27,   16,    4,    4,   17,   30,    9,   28,   10,   20,   12,   18,   27,   16,   15,
27,
    19,   13,    3,   16,    8,    7,   21,   28,   27,   15,    2,   25,   26,   27,   29,
21,
    11,    1,   26,   28,   31,    3,   24,    9,   24,    4,   27,   19,   21,   10,   14,
8,
    24,   18,   24,    0,   25,   13,   29,   20,   28,   31,   14,   23,   26,   11,   12,
5,
    12,    6,    1,   11,   10,   26,   21,    2,   30,   16,   21,   19,   27,    4,   28,
19,
    22,   20,   19,   15,    2,   16,    4,   30,   15,   18,   21,    9,   29,    2,   14,
9
};
/*
//Нормально шифруются ТОЛЬКО ЗАГЛАВНЫЕ РУС-ЛАТ
void codeWrite(const QString& str, QTextStream& s){
    const char* src = str.ascii();
    int l = strlen(src);
    char* d = (char*)calloc(1, l+1);
    strcpy(d, src);
    k8k7(d);
    int i0 = d[0];
    for(int i=1; i<l; i++){
        d[i] ^= spGamma[(i0+i)%256];
    }
    k7k8(d);
    s << l << ' ';
    s.writeRawBytes (d, l);
    free(d);
}

void codeRead(QString& str, QTextStream& s){
    int l;
    s >> l;
    char* d = (char*)calloc(1, l+1);
    s.readRawBytes (d, l);
    k8k7(d);
    int i0 = d[0];
    for(int i=1; i<l; i++){
        d[i] ^= spGamma[(i0+i)%256];
    }
}

```

```

    }
    k7k8(d);
    str = d;
    free(d);
}
*/

QString code(const QString& str){
    const char* src = str.ascii();
    unsigned int l = strlen(src);
    unsigned char* d = (unsigned char*)strdup(src);
    QString out;
    QString ch;
    unsigned int i0 = d[0];
    for(unsigned int i=1; i<l; i++){
        d[i] ^= spGamma[(i0+i)%256];
    }
    for(unsigned int i=0; i<l; i++){
        ch.sprintf("%02X", (unsigned char)d[i]);
        out += ch;
    }
    free(d);
    return out;
}

QString decode(const QString& str){
    int l = str.length()/2;
    QString out;
    QString ch;
    bool ok;
    unsigned char* d = (unsigned char*)calloc(1, l+1);
    for(int i=0; i<l; i++){
        ch = str.mid(i<<1, 2);
        d[i] = ch.toInt(&ok, 16);
    }
    int i0 = d[0];
    for(int i=1; i<l; i++){
        d[i] ^= spGamma[(i0+i)%256];
    }
    out = (char*)d;
    free(d);
    return out;
}

void codeWrite(const QString& str, QTextStream& s){
    s << code(str) << " ";
}

void codeRead(QString& str, QTextStream& s){
    QString tmp;
    s >> tmp;
    str = decode(tmp);
}

static unsigned char e_win[] = "`abcdefghijklmnopqrstuvwxyz{|}~'";
static unsigned char c_win[] = "ЮАБЦДЕФГХИЙКЛМНОПЯРСТУЖВЬЫЗШЭЩЧЪ";
static unsigned char r_win[] = ">F<WLTAU{BQRKVYJGZHCNE;DMSPI\"OX}";
static unsigned char rm_win[] = ".f,wltau[bqrkvyjgzhcne;dmspi'ox]";
static unsigned char lat_russ(unsigned char c){
    short i;
    for(i=0; i<32; i++) if(c==r_win[i]||c==rm_win[i]) return c_win[i];
    return c;
}

```

```

}
static void lat_russ(char *str){
    while (*str) { *str = lat_russ(*str); str++;}
}
QString latRUSS(const QString& lat){
    QString upperLat = lat.upper();
    char* bf = strdup(upperLat.ascii());
    lat_russ(bf);
    QString out(bf);
    free(bf);
    return out;
}

static unsigned char koi7_win(unsigned char c){
    if(c>=96 && c<=126) return (unsigned char)c_win[c-96];
    else if(c==39) return (unsigned char)c_win[31];
    else return c;
}
static unsigned char win_koi7(unsigned char c){
    short i;
    if(c>=0xC0 && c<=0xDF){ for(i=0; i<32; i++) if(c==c_win[i]) return
e_win[i]; }
    return c;
}
void k8k7(char* str, int len){
    for (int i=0; i<len; i++)
        str[i]=win_koi7(str[i]);
}
void k7k8(char* str, int len){
    for (int i=0; i<len; i++)
        str[i]=koi7_win(str[i]);
}
void k8k7(char *str){
    while (*str) { *str = win_koi7(*str); str++;}
}
void k7k8(char *str){
    while (*str) { *str = koi7_win(*str); str++;}
}

```