УТВЕРЖДЕН
07623615.00011-02 33 01-ЛУ

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СУПЕР-ЭВМ

Руководство программиста

07623615.00011-02 33 01

Листов 92

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

2020

АННОТАЦИЯ

Документ является руководством программиста СПО Супер-ЭВМ. В документе рассматриваются основные возможности, предоставляемые СПО-Супер-ЭВМ пользователям, компоненты СПО-Супер-ЭВМ и их интерфейсы.

СОДЕРЖАНИЕ

1. Функциональная схема СУПЕР-ЭВМ под управлением СПО Супер-ЭВМ 5
1.1. Система доступа 6
1.2. Вычислительное поле 6
1.3. Система хранения 6
1.4. Коммуникационная система 7
1.5. Сеть доступа 7
2. СПО Супер-ЭВМ 8
3. Предоставляемые пользователям ресурсы 12
4. Вход в систему 13
5. Описание операций 14
5.1. Управление пользователями 14
5.1.1. Работа с учетными записями 14
5.1.2. Работа с группами 15
5.2. Средства для обмена информацией 15
5.2.1. Доступ к файлам по протоколу FTP 15
5.2.2. Сетевые диски по SMB/CIFS 16
5.2.3. Электронная почта 17
5.2.4. Сервис передачи сообщений 18
5.3. Информационные команды 18
5.3.1. hostname – запрос имени узла 18
5.3.2. date – запрос (модификация) системных часов 18
5.3.3. man – справочная система Manual Pages 18
5.3.4. info – справочная системе GNU Info 18
5.4. Команды для работы с файловыми объектами 18
5.4.1. cd, pushd, pop – установка текущего каталога 18
5.4.2. pwd – вывод имени текущего каталога 19
5.4.3. mkdir – создание каталога 19
5.4.4. rmdir – удаление каталога 19
5.4.5. ls – просмотр содержимого каталога 19
5.4.6. rm – удаление файла 19
5.4.7. cp – копирование файлов 19
5.4.8. mv – перемещение файлов 19
5.4.9. chown – смена владельца файлов и каталогов 20
5.4.10. file – запрос информации о файле 20
5.4.11. mc – терминальный файловый менеджер 20
5.4.12. emacs – редактор и файловый менеджер GNU 20

5.5. Команды для работы с прикладным ПО 23
5.5.1. module – настройка окружения 23
5.5.2. Семейство компиляторов GNU 23
5.5.3. make – утилита для сборки проектов 24
5.5.4. gdb – отладчик GNU 24
5.6. Параллельное программирование с использованием MPI 25
5.6.1. Разработка параллельного приложения 25
5.6.2. Компиляция и сборка параллельного приложения 29
5.6.3. Запуск параллельного приложения 29
5.7. Система управления задачами 34
5.7.1. squeue – просмотр очереди задач 34
5.7.2. sinfo – просмотр состояния ВП 35
5.7.3. sbatch – постановка задачи в очередь 35
5.7.4. srun – запуск шага задачи (программы) 35
5.7.5. scancel – прекращение задачи 36
5.7.6. scontrol show job – получение информации о задаче 36
5.7.7. sview – графический монитор системы управления задачами 36
5.7.8. sacct – просмотр статистики по расчетам задач 36
5.7.9. sacctmgr – управление аккаунтами Slurm 37
5.8. WEB-сервис 38
5.9. СУБД 38
5.10. Системы хранения 38
5.10.1. Локальные системы хранения 38
5.10.2. Параллельная файловая система Lustre 38
5.10.3. Архивная система хранения АСХД 38
5.10.4. Облачная система хранения СТРИЖ 50
5.10.5. Subversion 88
Перечень сокращений 89
Перечень ссылочных документов 90

1. ФУНКЦИОНАЛЬНАЯ СХЕМА СУПЕР-ЭВМ ПОД УПРАВЛЕНИЕМ СПО СУПЕР-ЭВМ

Для понимания состава СПО Супер-ЭВМ, необходимо представлять типовую структуру Супер-ЭВМ. В данном документе мы приведем краткое описание Супер-ЭВМ, на примере одного из ее типов – Супер-ЭВМ, предназначенной для решения задач численного моделирования (далее по тексту Супер-ЭВМ).

По своей структуре Супер-ЭВМ является многопроцессорным вычислительным комплексом (СУПЕР-ЭВМ) и представляет собой совокупность технических и программных средств системы обработки данных, способных функционировать самостоятельно или в составе других систем.

Общая схема Супер-ЭВМ представлена на рис. 1.

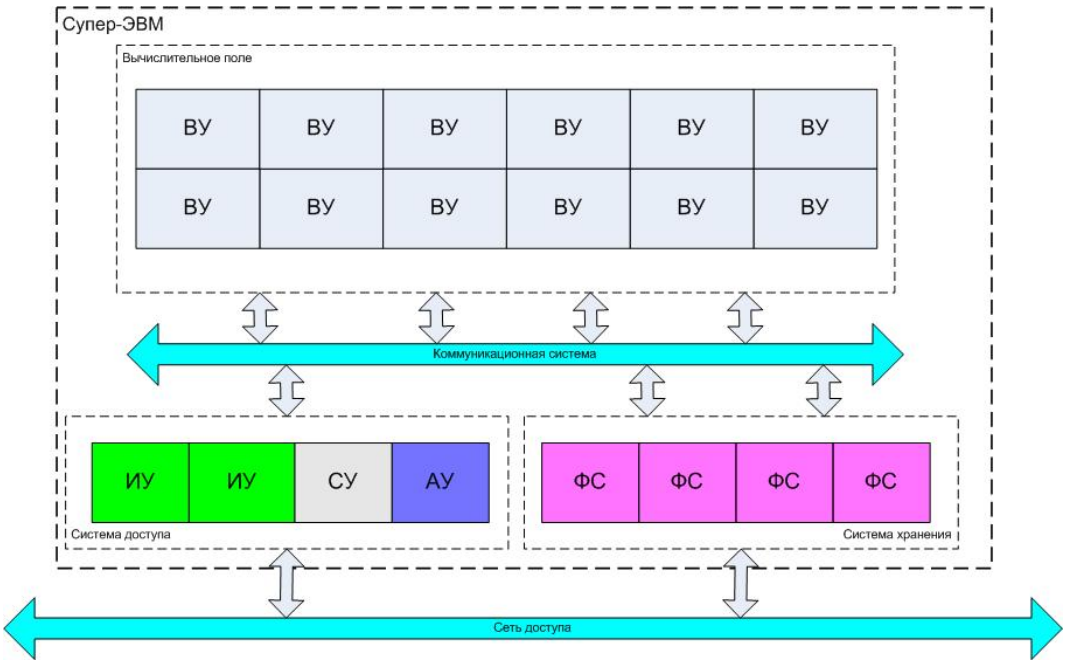

Рис. 1 – Общая схема Супер-ЭВМ

В состав Супер-ЭВМ входят следующие компоненты:

- система доступа;
- вычислительное поле;
- система хранения;
- коммуникационная система.

Супер-ЭВМ может быть подключена к другим компонентам ЛВС предприятия с помощью сети доступа.

Приведем описание основных компонент Супер-ЭВМ.

1.1. Система доступа

Система доступа – подсистема Супер-ЭВМ, предназначенная для организации доступа пользователей к ресурсам Супер-ЭВМ. Включает в себя множество узлов различного назначения, среди которых основными являются следующие:

– ИУ (инструментальные узлы) – выделенные в системе доступа технические и программные средства для разработки программ, подготовки начальных данных и обработки результатов расчетов;

– АУ (административные узлы) – выделенные в системе доступа технические и программные средства для организации системных сервисов управления задачами, мониторинга оборудования и пр.;

– СУ (сервисные узлы) – выделенные в системе доступа технические и программные средства для организации различных сервисов (информационных, загрузки бездисковых узлов, маршрутизации потоков данных между компонентами Супер-ЭВМ и транспортной подсистемой любой сопряженной автоматизированной системы.

1.2. Вычислительное поле

Вычислительное поле Супер-ЭВМ, состоящее из множества вычислительных узлов (ВУ), является основным вычислительным ресурсом Супер-ЭВМ и предназначено для проведения ресурсоемких расчетов (в частности, численного моделирования физических процессов).

1.3. Система хранения

Система хранения (СХ) обеспечивает хранение системного и прикладного ПО, а также исходных данных и результатов расчетов и может состоять из нескольких типов систем хранения, например:

– POSIX-совместимой высокопроизводительной файловой системой с параллельным доступом;

– POSIX-совместимой файловой системой общего назначения;

– СУБД;

– подсистемы ведения проектов (ПсВП);

– архивных систем хранения.

Система хранения состоит из файловых серверов разного назначения (сетевые, параллельные, архивные и пр.):

– ФС (файловые сервера) – выделенные в системе хранения узлы, предназначенные для хранения информации. Можно выделить множество ФС, составляющих сетевую

файловую систему общего назначения (типа NFS), параллельную (оперативную) файловую систему (например, типа Lustre), архивную подсистему (систему хранения второго уровня);

– СУБД – сервера различных СУБД, входящих в СПО Супер-ЭВМ;

– медиа-сервера – сервера, предназначенные для переноса информации на различные устройства хранения.

1.4. Коммуникационная система

Коммуникационная система Супер-ЭВМ – совокупность коммуникационного оборудования и линий связи, обеспечивающая передачу данных между компонентами Супер-ЭВМ. Реализует несколько типов сетей: сеть передачи сообщений, сеть данных, сеть управления, сеть мониторинга и т.д. Может включать различные типы сетевого оборудования для выполнения разных целей (Ethernet, Infiniband и пр.).

1.5. Сеть доступа

Транспортная подсистема Супер-ЭВМ – совокупность коммуникационного оборудования и линий связи, обеспечивающая передачу данных между Супер-ЭВМ, другими компонентами ЛВС предприятия, в состав которой Супер-ЭВМ включен.

2. СПО СУПЕР-ЭВМ

Дистрибутив СПО Супер-ЭВМ основан на свободно распространяемом дистрибутиве Scientific Linux [1], дополненном компонентами, необходимыми для функционирования Супер-ЭВМ, в том числе собственной разработки, и включает в себя следующие программные средства:

– ОС узла – совокупность системного программного обеспечения нижнего уровня, обеспечивающая работу аппаратных составляющих узла (сервера);

– программное обеспечение (ПО), используемое для разработки прикладных программ – компиляторы с языков программирования C, C++, Fortran и др., системы разработки, прикладные, научные и графические библиотеки и пр.;

– ПО транспортных подсистем – системное ПО, обеспечивающее передачу различных категорий данных по каналам связи, по специфичным для конкретного вида сетевого оборудования протоколам;

– коммуникационное ПО – системное ПО, используемое для передачи данных между процессами прикладных задач и использующее ПО транспортных подсистем;

– ПО системы хранения – системное ПО, используемое для организации доступа и хранения данных в системе хранения. Специфично для каждого типа систем хранения;

– подсистема учета пользователей – системное ПО, предназначенное для хранения учетных записей пользователей и содержащее механизмы, используемые для аутентификации;

– подсистема конфигурации – совокупность системного ПО и таблиц, содержащих информацию об оборудовании Супер-ЭВМ, описывающих состав Супер-ЭВМ и его компонентов и позволяющих использовать Супер-ЭВМ как единое целое;

– подсистема управления задачами и ресурсами – системное ПО, предназначенное для организации многозадачного и многопользовательского режима выполнения задач на вычислительном поле;

– средства управления и мониторинга узлов Супер-ЭВМ – системное ПО, используемое в процессе эксплуатации Супер-ЭВМ для управления узлами, диагностики работы оборудования;

– сервисы удаленного доступа к узлам Супер-ЭВМ – системные компоненты, установленные на узлах Супер-ЭВМ и обеспечивающие подключение пользователей со своих АРМ к узлам Супер-ЭВМ;

– web–сервис, предоставляющий доступ к общедоступным информационным ресурсам по протоколам HTTP и HTTPS, а также к компонентам систем управления задачами и узлами;

– сервисы передачи сообщений между компонентами системного ПО, оборудованием Супер-ЭВМ и обслуживающим персоналом на основе электронной почты и ПО сервиса мгновенных сообщений.

В состав СПО Супер-ЭВМ входят компоненты, перечисленные в таблице 1. В столбце «Размещение» для каждой компоненты указывается тип серверов Супер-ЭВМ, на которых эта компонента устанавливается, а также необходимость установки этой компоненты при использовании СПО Супер-ЭВМ на автономных АРМ.

Таблица 1 – Функциональный состав СПО Супер-ЭВМ

Наименование системной компоненты	Описание	Размещение (при применении на Супер-ЭВМ)
1. Средства установки дистрибутива СПО Супер-ЭВИ		
УДист	Совокупность СПО, обеспечивающая подготовку СВТ к установку ОС, установку ядра и необходимых пакетов, первичную настройку параметров работы ОС	Все узлы, АРМ
2. Подсистема Ядро ОС		
Ядро ОС	Совокупность СПО нижнего уровня, обеспечивающая работу аппаратных составляющих узла (сервера). Включает в себя подсистему идентификации, подсистему виртуальной памяти, виртуальную файловую систему*) и подсистему организации процессов	Все узлы, АРМ
3. Подсистема инициализации (ПсИни)		
INIT, Systemd	Системный процесс, обеспечивающий загрузку системных сервисов	Все узлы, АРМ
4. Подсистема идентификации пользователей (ПсИП)		
PAM	Модуль аутентификации пользователей	Все узлы, АРМ
OpenLDAP	Служба каталогов, хранящая учетные записи пользователей и используемая для организации ЕПП	АУ
su, sudo, sg, newgrp	Утилиты модификации учетных данных пользователя для процессов	Все узлы, АРМ
PolKit	подсистема аутентификации	Все узлы, АРМ
5. Подсистема идентификации оборудования (ПсИО)		
OpenSM	Менеджер сетей Infiniband	Все узлы
DNS	Подсистема доменных имен	АУ

Продолжение таблицы 1

Наименование системной компоненты	Описание	Размещение (при применении на Супер-ЭВМ)
DHCP	Служба динамического конфигурирования узлов в сетях Ethernet	АУ
6. Подсистема виртуализации (ПсВирт)		
KVM	модуль ядра и утилиты для работы с ВМ	ИУ, АРМ
7. Сервисы доступа (СД)		
OpenSSH	Сервис удаленного доступа	Все узлы, АРМ
Getty, Login	Сервис локального доступа	Все узлы, АРМ
XDM, LightDM	Сервисы удаленного доступа в графическом режиме	ИУ, АУ, АРМ
VNC	Сервис удаленного доступа в графическом режиме	ИУ
FTP	Сервис передачи файлов	ИУ, АУ, АРМ
Samba	Сетевая файловая система	ИУ
Apache	Web-сервис	ИУ
8. Утилиты управления доступом (УтУД)		
chown, chgrp, umask, getfacl, setfacl	Утилиты управления доступом	ИУ, АУ, АРМ
9. Подсистема управления задачами (СУЗ)		
Slurm	Система управления заданиями	ИУ, АУ, ВУ
Jam	Система управления заданиями	ИУ, АУ, ВУ
10. Коммуникационное СПО (КСПО)		
OFED	Программное обеспечение низкого уровня для сетей InfiniBand	Все узлы
11. Системы хранения оперативного уровня (СХ)		
NFS	Сетевая файловая система	ФС[1], АРМ
Lustre	Сетевая файловая система	ФС[1]

[1] Серверная часть. Клиентская часть размещена на ИУ, ВУ. АУ

Продолжение таблицы 1

Наименование системной компоненты	Описание	Размещение (при применении на Супер-ЭВМ)
12. Системы хранения второго уровня (СХ2)		
RFNC-dss	Пакет программных средств, реализующих распределенную систему хранения данных	ФС[2]
АСХД	Пакет программных средств, реализующих распределенную систему хранения данных	ФС[2]
ОСХД Стриж	Пакет программных средств, реализующих распределенную систему хранения данных	ФС[2]
13. Сервисы баз данных (СУБД)		
MariaDB	Сервисы баз данных	ФС[1], АРМ
PostgreSQL	Сервисы баз данных	ФС[1], АРМ
14. Подсистема ведения проектов (ПсВП)		
SVN	Система ведения версий проектов	ФС[2]
15. Подсистема точного времени (ПсТВ)		
Ntpd	Служба времени	ИУ, АУ, ВУ, ФС
16. Подсистема контроля целостности (ПсКЦ)		
AIDE	Система контроля целостности	Все узлы, АРМ
17. Подсистема регистрации событий безопасности (ПсРСБ)		
Auditd	Демон регистрации событий	Все узлы, АРМ
18. Сервисы обмена информацией (ИфноСерв)		
PostFIX	Почтовый сервер	АУ[2]
Jabber	Сервис передачи мгновенных сообщений	АУ[2]

[2] Серверная часть. Клиентская часть размещена на ИУ и АРМ пользователей

3. ПРЕДОСТАВЛЯЕМЫЕ ПОЛЬЗОВАТЕЛЯМ РЕСУРСЫ

При регистрации на Супер-ЭВМ пользователь получает доступ к следующим видам ресурсов:

– инструментальный узел – узел системы доступа Супер-ЭВМ, на котором пользователь может проводить разработку прикладного ПО, подготовку начальных данных и обработку результатов расчетов, постановку задач в очередь на исполнение и управление запущенными задачами;

– вычислительное поле – совокупность вычислительных узлов Супер-ЭВМ, на которых можно запускать последовательные и параллельные задачи в пакетном режиме;

– домашняя файловая система (/home) – сетевая файловая система, предназначенная для хранения текстов и кодов прикладного ПО. Не рекомендуется использовать данную систему для расчета задач с массовым выводом;

– файловая система для оперативного хранения (например, /lustre) – сетевая файловая система, предназначенная для хранения результатов счета, отличающаяся большим объемом и скоростью доступа со стороны параллельных программ, а также обеспечивающая возможность записи при расчете задач с массовым выводом;

– каталоги с дополнительным ПО (/opt), содержащие различное системное и прикладное программное обеспечение, включая библиотеки MPI, математические библиотеки и пр.

Доступ ко всем этим каталогам возможен как с ИУ, так и со всех ВУ Супер-ЭВМ.

4. ВХОД В СИСТЕМУ

СПО Супер-ЭВМ предоставляет следующие способы входа в систему:

– прямой вход на узел Супер-ЭВМ через подключенный к узлу символьный терминал;

– прямой вход на узел Супер-ЭВМ через подключенный к узлу графический терминал;

– удаленный вход на узел Супер-ЭВМ в режиме символьного терминала с АРМ пользователя или другого узла по SSH [2];

– удаленный вход на узел Супер-ЭВМ в режиме графического терминала с АРМ пользователя или другого узла, используя X11 [3].

Прямой вход на узел осуществляется после включения терминала, подключенного непосредственно к узлу. При этом запрашивается учетное имя пользователя и его пароль. Далее, в зависимости от вида подключения (символьный или графический) запускается оболочка shell для выполнения команд пользователя или оконный менеджер для запуска графических приложений (в том числе графического терминала, в котором также запускается оболочка shell).

Удаленный вход в режиме символьного терминала производится по протоколу SSH, для этого можно воспользоваться любым ssh-клиентом на своей машине. При вызове ssh-клиента ему указывается имя пользователя на узле Супер-ЭВМ и имя узла Супер-ЭВМ (обычно в виде <имя_пользователя>@<имя_узла>), после установки соединения с узлом вводится пароль пользователя, после этого, если все данные введены правильно, клиент отображает символьный терминал, в котором запускается оболочка shell.

Для удаленного входа в режиме графического терминала на АРМ пользователя должен быть запущен X-сервер. При этом возможны два режима работы:

– X-сервер запущен в пассивном режиме. В этом случае вход на узел производится по SSH, в появившемся терминале устанавливается переменная окружения DISPLAY с сетевым адресом X-сервера (например, export DISPLAY=129.6.100.1:0), далее в этом терминале можно запускать графические приложения на узле Супер-ЭВМ;

– X-сервер запущен в активном режиме. В этом случае X-сервер формирует окно, в котором запрашивается имя узла, далее имя пользователя и его пароль, после этого в этом окне запускается оконный менеджер.

5. ОПИСАНИЕ ОПЕРАЦИЙ

В данном разделе представлен краткий обзор программных сервисов и основных команд операционной системы Супер-ЭВМ, включая базовый набор команд ОС Linux и команд системы управления заданиями SLURM. Полное описание команд можно найти в книгах, справочных материалах и документации по ОС Linux и другим системным компонентам, установленным на Супер-ЭВМ. Для большинства команд доступны краткие описания (Manual Pages), которые можно просмотреть посредством команды вида:

`man <имя команды>`

Также для большинства команд можно получить краткую справку о назначении команды и ее параметров, вызвав команду с параметром `--help`.

Команды, обеспечивающие работу с установленным на Супер-ЭВМ дополнительным ПО, описаны более подробно.

5.1. Управление пользователями

СПО Супер-ЭВМ поддерживает две технологии ведения учетных записей пользователей:

– традиционную для ОС семейства Unux, основанную на файлах /etc/passwd и /etc/group, при которой учетные записи локализованы на узле Супер-ЭВМ или АРМ;

– централизованную, с использованием службы каталогов OpenLDAP [5], образующую Единое Пространство Пользователей (ЕПП) на всех узлах Супер-ЭВМ.

Для работы с ЕПП в состав СПО Супер-ЭВМ введен пакет zldputils, облегчающий администраторам ведение базы данных пользователей.

В состав пакета zldaputils входят утилиты добавления (zldapadd), удаления (zldapdelete), модификации (zldapreplace) записей LDAP, а также утилита создания шаблонов записей (zldapconf).

Пакет допускает возможность работы с несколькими доменами, при этом имя домена передается утилитам первым параметром.

Рассмотрим примеры использования этих утилит.

5.1.1. Работа с учетными записями

```
# zldapadd L1 user --cn user1cn --sn user1sn --uid user1 \
--homeDir /home/user1 --gidNumber 2517 --uidNumber 3628
```

Эта команда создаст пользователя с минимально достаточными атрибутами `---uid` (в системе uname), `--homeName` (в системе home), `--gidNumber`(в системе gid), `--uidNumber` (в системе uid). Атрибуты `–cn` и `–sn` являются обязательными, не существенны для ЕПП, поэтому их значение может совпадать с `–uid`. Среди остальных атрибутов следует упомянуть `–userPassword`, являющийся паролем пользователя при входе в систему. Его можно задать как при создании учетной записи, так и командой модификации учетной записи.

```
# zldapreplace L1 user user1 --userPass <[хэш-пароль>
```

Эта команда модифицирует (или задаст, если не было ранее) пароль пользователя.

Хэш-пароль можно получить с помощью команды:

```
# slappasswd -s <пароль> -h {MD5}
```

Ключем –h задается необходимый тип хеширования пароля. Допустимые значения {MD5}, {SHA}.

Эта команда модифицирует (или задаст, если не было ранее) список доступных хостов:

```
# zldapreplace L1 user user1 --host srv1 srv2 srv3
```

Эта команда удалит пользователя user1:

```
# zldapdelete L1 user user1
```

5.1.2. Работа с группами

Эта команда создаст группу group1 с gid=12345:

```
# zldapadd L1 group --cn group1 --gidNum 12345
```

Эта команда удалит группу group1:

```
# zldapdelete L1 group group1
```

Эта команда включит в группу group1 пользователей user2, user3:

```
# zldapreplace L1 group group1 --memverUid user2 user3
```

5.2. Средства для обмена информацией

5.2.1. Доступ к файлам по протоколу FTP

С помощью протокола FTP [5] пользователь может перемещать файлы между файловыми ресурсами Супер-ЭВМ и локальными или сетевыми дисками своей АРМ. Для использования FTP требуется установить соединение с узлом системы доступа ВК, доступным пользователю. При установке соединения требуется указать имя узла, имя пользователя и его пароль на Супер-ЭВМ.

5.2.2. Сетевые диски по SMB/CIFS

Сетевые диски для доступа к информации пользователя, хранящейся в системах хранения Супер-ЭВМ, по протоколу SMB/CIFS [6] организуются администратором Супер-ЭВМ по просьбе ответственного за ресурс при подаче заявки на создание ресурса. Системный администратор сообщает ответственному за ресурс сетевое имя, которое может быть использовано для настройки сетевого диска на АРМ пользователей. Подключение осуществляется через меню «Подключить сетевой диск…», доступное при нажатии правой кнопки мыши на ярлыке «Мой компьютер», как показано на рис. 2.

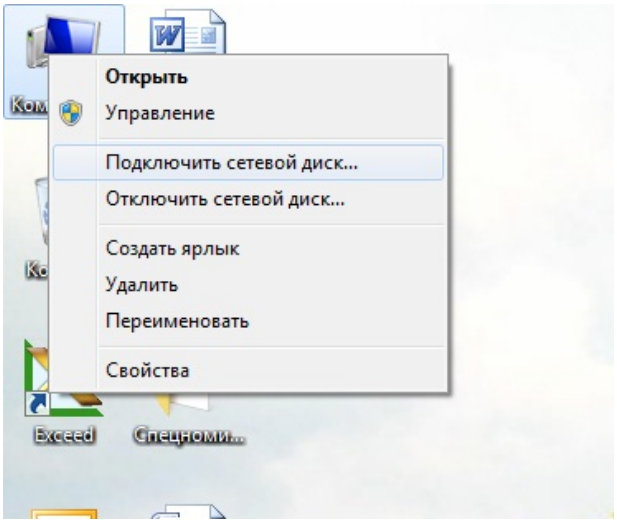

Рис. 2 – Подключение сетевого диска

В появившемся меню необходимо ввести имя (или IP адрес) сервера, с которого происходит экспорт файловой системы (сетевое имя), как показано на рис. 3.

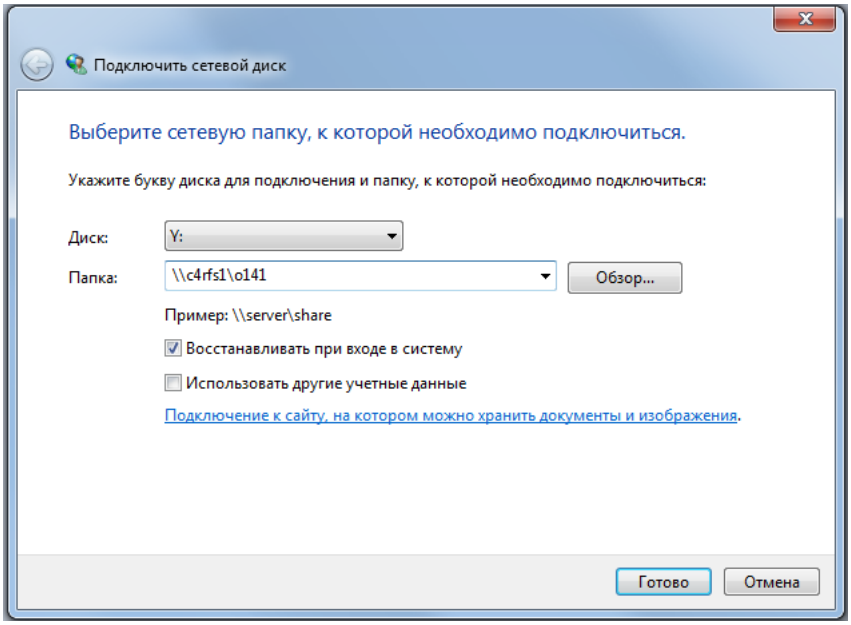

Рис. 3 – Подключение сетевого диска (продолжение)

В поле «Использовать другие учетные данные» должна присутствовать галочка.

Для сотрудников подразделений в появившемся окне авторизации ввести свои учетные данные пользователя Супер-ЭВМ (без указания домена), как представлено на рис. 4.

После выполнения настройки и подключения, созданный на Супер-ЭВМ ресурс, будет виден на АРМ как обычный сетевой диск, предоставляя при этом обычные для АРМ средства работы с содержащимися в нем файлами.

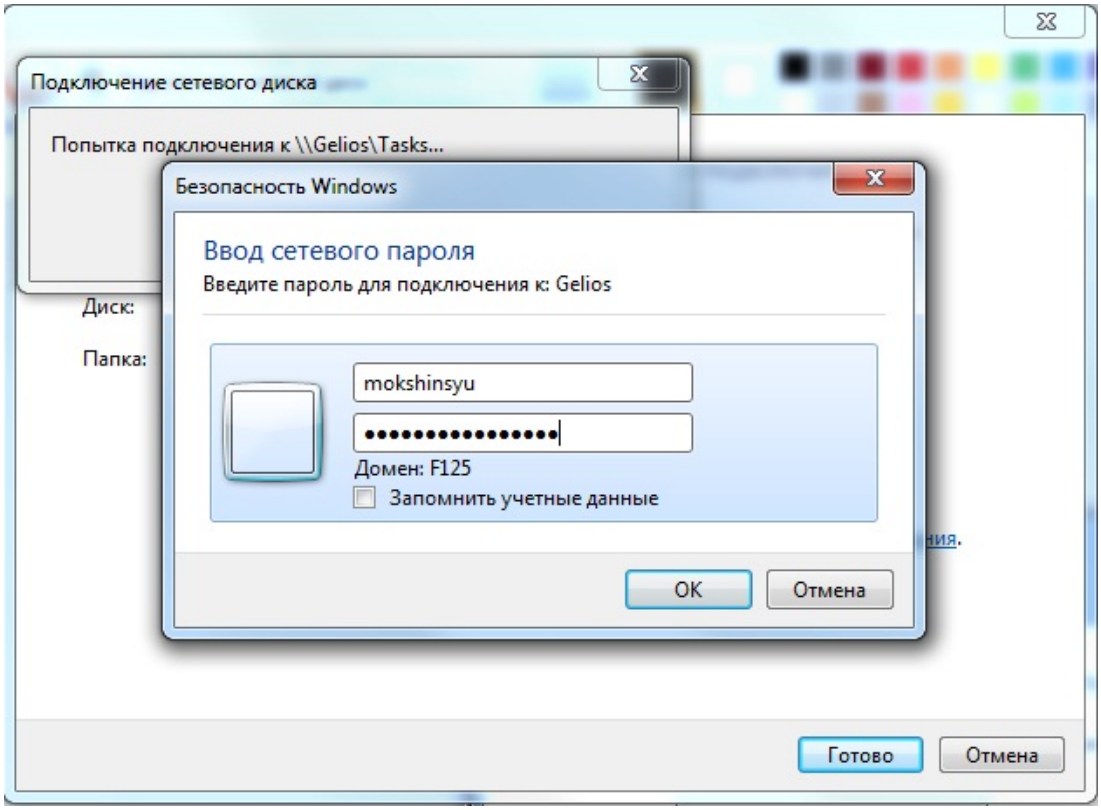

Рис. 4 – Ввод учетной записи пользователя при подключении сетевого диска

5.2.3. Электронная почта

СПО Супер-ЭВМ содержит сервис электронной почты в составе двух компонент:

- сервис передачи электронной почты PostFix [7];
- сервис получения электронной почты DoveCot [8].

Для отправки и приема почтовых сообщений пользователь может воспользоваться любым почтовым клиентом, поддерживающим протоколы SMTP и IPAM, например: Gnome Evolution, Thunderbird или Sylpfeed.

5.2.4. Сервис передачи сообщений

СПО Супер-ЭВМ содержит сервис передачи сообщений Jabber2 [9]. Для отправки и приема сообщений пользователь может воспользоваться любым XMPP-клиентом, например, Psi-Plus или Pidgin.

5.3. Информационные команды

Данная группа команд предназначена для получения различной справочной информации и управления файлами и каталогами.

5.3.1. hostname – запрос имени узла

Команда `hostname` выводит имя узла, на котором она запущена.

5.3.2. date – запрос (модификация) системных часов

Команда `date` выводит показание системных часов. Администратору данная команда позволяет установить системные часы.

5.3.3. man – справочная система Manual Pages

Команда `man` позволяет получить в формате Manual Pages описание большинства команд ОС Linux, системных вызовов и библиотечных функций. В частности, `man man` выдает описание самой команды `man`.

5.3.4. info – справочная системе GNU Info

Команда `info` позволяет получить подробное описание большинства команд Linux в формате гипертекста. В частности, `info info` выдает описание самой команды `info`.

5.4. Команды для работы с файловыми объектами

Данная группа команд используется для работы с файловыми объектами (файлами и каталогами).

5.4.1. cd, pushd, pop – установка текущего каталога

Команда `cd` предназначена для смены текущего каталога. В параметрах указывается имя нового текущего каталога.

Команда `pushd` также предназначена для смены текущего каталога. В параметрах указывается имя нового текущего каталога. В отличие от команды `cd`, команда `pushd` запоминает имя предыдущего текущего каталога в списке каталогов.

Команда `popd` предназначена для возврата предыдущего, сохраненного в списке командой `pushd` текущего каталога. Параметров нет. Работа со списком каталогов организована в режиме стека.

5.4.2. pwd – вывод имени текущего каталога

Команда `pwd` вводит имя текущего каталога.

5.4.3. mkdir – создание каталога

Команда `mkdir` позволяет создать новый каталог в файловой системе. В параметрах указывается имя создаваемого каталога и, возможно, режимы доступа к нему.

5.4.4. rmdir – удаление каталога

Команда `rmdir` позволяет удалить пустой каталог. В параметрах указывается имя удаляемого каталога.

5.4.5. ls – просмотр содержимого каталога

Команда `ls` позволяет вывести содержимое указанного в параметрах каталога. Также в параметрах указывается формат вывода.

5.4.6. rm – удаление файла

Команда `rm` позволяет удалить файл, группу файлов, содержимое каталога, включая вложенные файлы и каталоги.

5.4.7. cp – копирование файлов

Команда `cp` позволяет скопировать файл, группу файлов, содержимое каталога, включая вложенные файлы и каталоги, в другое место файловой системы.

5.4.8. mv – перемещение файлов

Команда `mv` позволяет переместить файл, группу файлов, содержимое каталога, включая вложенные файлы и каталоги, в другое место файловой системы.

5.4.9. chown – смена владельца файлов и каталогов

Команда `chown` позволяет сменить владельца и группу у файла или каталога, а также (с опцией `-R`) у каталога, включая все вложенные файлы и каталоги.

5.4.10. file – запрос информации о файле

Команда `file` позволяет вывести информацию о типе указанного в параметрах файлового объекта.

5.4.11. mc – терминальный файловый менеджер

Команда `mc` запускает файловый менеджер Midnight Commander, позволяющий выполнять операции над файлами и каталогами в диалоге в режиме терминала. Функционально и внешне Midnight Commander похож на Norton Commander и Total Commander, используемые в DOS и Windows.

Emacs – редактор и файловый менеджер Emacs [10] – многофункциональный редактор и файловый менеджер, обеспечивающий удобную работу с текстами программ, написанных на разных языках программирования, предоставляющий средства для работы с файлами и каталогами, а также обеспечивающий работу в терминальном режиме с интерпретатором команд shell.

GNU Emacs допускает работу в терминальном и графическом режимах доступа. В терминальном режиме команды вводятся с помощью сочетаний клавиш клавиатуры. В графическом режиме в интерфейсе редактора появляются меню и панель инструментов, допускающие использование манипулятора «мышь» для выполнения определенных действий.

Полное описание GNU Emacs содержится в справочной системе GNU Info.

5.4.11.1. Общее описание

Emacs предполагает работу в нескольких основных режимах:

- просмотр каталогов;
- редактирование файла;
- работа в интерпретаторе команд.

Каждый режим запускается командой `emacs`. Для ввода команд в Emacs используется два способа:

– с использованием строки в нижней части окна Emacs, появляющейся при нажатии клавиш <Alt>-x. Например, <Alt>-x dired – сначала одновременно нажимаются клавиши <Alt> и 'X', потом последовательно 'd', 'i', 'r', 'e', 'd', затем <Enter>;

– комбинациями клавиш. Символ '-' означает одновременное нажатие клавиш, <Пробел> – последовательное. Между символами '<' и '>' пишется название клавиши. Например, <Ctrl>-x r j означает одновременное нажатие клавиш <Ctrl> и 'x', затем последовательное нажатие клавиш 'r', 'j'.

Кроме того, для многих команд в Emacs предусмотрены «горячие клавиши», назначение которых хранится в настроечных файлах .emacs, в частности в личном настроечном файле пользователя ~/.emacs. Также команды доступны через меню Emacs, расположенное в верхней части окна Emacs. Команда <Ctrl>-g во многих случаях прекращает выполнение предыдущей команды.

5.4.11.2. Просмотр каталогов

Запускается командой <Alt>-x dired (F10). После ввода команды в строке ввода появляется запрос имени каталога для просмотра. В этом режиме можно перемещаться по вложенным каталогам, просматривать файлы, размещенные в каталоге, и выполнять операции над файлами. Операции набираются клавишами клавиатуры прямо в строке, содержащей имя файла (см. таблицу 2).

Таблица 2 – Команды в режиме просмотра каталогов

Команда	Описание
m	Пометить файл для групповой операции
u	Снять пометку
~	Пометить backup-файлы
d	Пометить файлы для удаления
x	Удалить помеченные файлы
D	Удалить один файл
R	Переместить помеченные файлы или переименовать файл
C	Скопировать помеченные файлы
<Enter>	Открыть файл для редактирования

5.4.11.3. Редактирование файлов

Вход в режим редактирования происходит из режима просмотра каталогов или по команде <Alt>-x find-file. В последнем случае в строке ввода запрашивается имя файла.

Основные команды режима редактирования представлены в таблице 3.

Таблица 3 – Основные команды режима редактирования

Сочетание клавиш	Описание
<Ctrl>-s	Сохранить файл
<Ctrl>-пробел	Начать выделение строк. Дальнейшее выделение производится клавишами перемещения курсора. Выделенный текст отмечается цветом фона
<Ctrl>-<del>	Удалить (переместить в буфер) выделенный текст
<Ctrl>-<ins>	Скопировать в буфер выделенный текст
<Shift>-<ins>	Вставить текст из буфера
<Ctrl>-k	Удалить конец строки (всю строку, если курсор стоит на начале строки)
<Ctrl>-x r <Space>	Отметить текущую позицию. Далее запрашивается символ – метка, назначенная позиция
<Ctrl>-x r j <метка>	Перейти к отмеченной позиции
<Alt>-<Shift>-%	Замена строки на строку
<Ctrl>-k	Закрыть буфер

Кроме того, в зависимости от типа редактируемого текста, определяемого по расширению имени файла, в меню добавляются специфичные для данного типа команды.

5.4.11.4. Shell-буфер

Вход в режим выполнения команд в интерпретаторе shell происходит по команде <Alt-x> shell (<Ctrl>-F10). В этом режиме можно выполнять любые команды интерпретатора shell, как в терминале. Работает стек команд.

5.4.11.5. Переключение между буферами

Для каждого открытого в Emacs каталога или файла, а также для shell-буфера создается отдельный буфер. Переключиться между буферами можно через меню «Buffers».

5.4.11.6. Управление окнами

Можно разбить главное окно Emacs на подокна, при этом в каждом подокне вывести свой буфер.

Команда <Ctrl>-F5 разделяет окно пополам по вертикали, <Shift>-F5 – по горизонтали. F5 убирает разделение. F6 переключает между подокнами.

5.5. Команды для работы с прикладным ПО

Эта группа команд позволяет настроить программное окружение для работы с прикладным ПО, отредактировать исходные тексты, скомпилировать и отладить программу.

5.5.1. module – настройка окружения

Правильная работа установленных на ВК сред разработки и исполнения программ во многом зависит от настройки системного окружения, в частности переменных окружения, определяющих пути доступа к компонентам сред разработки и исполнения.

Команда module облегчает формирование системного окружения текущего сеанса, требуемого для конкретной среды разработки и исполнения. В систему заложен предопределенный набор модулей, формирующих системное окружение для каждой конкретной среды. Для работы этой команды в файле ~/.bashrc необходимо присутствие строки:

```
. /opt/Modules/init/bash
```

Варианты вызова команды module представлены в таблице 4.

Таблица 4 – Варианты вызова команды module

Команда	Описание
`$ module avail`	Вывод списка доступных модулей
`$ module load <имя модуля>`	Загрузить указанный модуль
`module list`	Выдать список загруженных модулей
`module remove <имя модуля>`	Выгрузить указанный модуль

5.5.2. Семейство компиляторов GNU

Данное семейство включает в себя три компилятора (gcc для языка C, g++ для языка C++ и gfortran для языков Фортран-77/90 [11]). Каждый компилятор вызывается командой, совпадающей по имени с компилятором. Параметры командной строки задают режим работы компилятора, пути файлов с текстами и кодами программных компонент.

Пример трансляции группы файлов с текстами программы (файлы с расширениями .c и .f) и сборки кода этой программы (prog.x):

```
gcc –c file1.c
gcc –c file2.c
gfortran –c file3.f
gfortran –o prog.x file1.o file2.o file3.o
```

5.5.3. make – утилита для сборки проектов

Утилита make [12]предназначена для сборки программ, состоящих из большого числа файлов с исходными текстами. Для этого на специальном языке в файле сборки, подаваемом параметром утилите make, описываются зависимости результирующего программного кода от файлов, содержащих исходные тексты объектных файлов и библиотек. При запуске утилита make определяет время модификации всех объектов, описанных в файле сборки, и производит перекомпиляцию и сборку «устаревших» компонент. Тем самым сокращается время на сборку программы при незначительных изменениях в ее текстах. Если файл сборки при вызове утилиты не указан, то подразумевается файл с именем Makefile в текущем каталоге. Пример Makefile:

```
MPICH = /opt/mvapich/gnu
MPI   = $(MPICH)/bin
COMP  = $(MPI)/mpif77
FFLAGS=
OBJS  = tf.o
EXE   = t.x

.f.o:
          $(COMP) –c –g $(FFLAGS) $<
$(EXE): $(OBJS)
          $(COMP) –o $@ -g $(OBJS)
clean:
          rm –f *.o *~ $(EXE)
```

Полное описание make содержится в справочной системе GNU Info.

Содержимое Makefile определяет значения переменных, содержащих информацию о версии компилятора, опциях компиляции и имени исполняемого файла, получаемого в ходе выполнения make, а также некоторый набор правил компиляции исходного текста программы и линковки объектных файлов.

5.5.4. gdb – отладчик GNU

Утилита gdb [13] является символьным отладчиком для системы программирования GNU и позволяет на ИУ отлаживать последовательные и параллельные (OpenMP) программы, а также анализировать файлы, содержащие информацию о причинах аварийного завершения программ (core-файлы). Для удобства работы возможна отладка с помощью графической оболочки к gdb – DDD. Для вызова этой оболочки в командной строке следует набрать ddd (при этом на АРМ пользователя должен быть запущен X-сервер).

5.6. Параллельное программирование с использованием MPI

В СПО Супер-ЭВМ предусмотрено использование нескольких реализаций MPI [14], включая MVAPICH-2 [15], OpenMPI [16]. Для вывода списка доступных библиотек MPI необходимо выполнить команду:

```
module avail mpi
```

Для настройки окружения на работу с конкретной библиотекой MPI необходимо выполнить команду вида:

```
module load <mpimod>
```

, где <mpimod> – имя необходимого модуля, полученное с помощью команды `module avail mpi`. Например, для библиотеки MVAPICH, собранной с помощью компиляторов gcc и gfortran, соответствующий модуль имеет название mpi/mvapich/1.2/gnu.

После загрузки модуля в командной строке будет доступен ряд утилит (mpicc, mpiCC, mpif77 и mpif90), которые используются для компиляции и сборки MPI-программ.

Для выполнения программ, использующих MVAPICH, под управлением SLURM необходимо в командный файл задачи поместить команды:

```
$ export LD_LIBRARY_PATH=/opt/mpi/mvapich/gnu/lib/shared
$ srun --mpi=pmi2 <имя файла с программой>
```

Для выполнения программ, использующих MVAPICH2, под управлением SLURM необходимо в командный файл задачи поместить команды:

```
$ export LD_PRELOAD=/usr/local/lib/libpmi.so
$ srun --mpi=none <имя файла с программой>
```

Для выполнения программ, использующих OpenMPI, под управлением SLURM необходимо в командный файл задачи поставить команды:

```
$ MPI=/opt/mpi/openmpi/gnu
$ export LD_LIBRARY_PATH=$MPI/lib
$ $MPI/bin/mpirun <имя файла с программой>
```

5.6.1. Разработка параллельного приложения

Программа MPI содержит единый код, одновременно исполняемый на автономных процессах. Разделение работ по процессам выполняется посредством распределения данных и алгоритмического ветвления с использованием условных операторов языка программирования. Коды, выполняемые каждым процессом, не обязаны быть идентичными. Процессы взаимодействуют при помощи вызовов коммуникационных примитивов MPI. Каждый процесс выполняется в собственном адресном пространстве.

MPI обеспечивает пользователю надежную передачу сообщений и не предусматривает механизмов поведения при исключительных ситуациях, возникающих в работе коммуникационной среды. Программная ошибка может происходить, когда MPI процедура вызвана с неправильным аргументом. Для анализа подобных ситуаций функции передачи данных MPI возвращают код, который указывает на состояние завершения операции. По умолчанию ошибка, обнаруженная во время исполнения MPI функций, вызывает окончание параллельных вычислений. Нарушение логики обменов, например, вызов операций парных обменов MPI с неправильными значениями номеров процессов источников или получателей данных могут привести к взаимоблокировке операций.

5.6.1.1. Операции парных обменов

Передача и прием сообщений процессами является основным механизмом MPI. Основные операции парных (двухточечных) обменов это send и receive - отправить и принять сообщение, реализованы в функциях MPI_Send и MPI_Recv. MPI определяет два механизма передачи и приема сообщений: блокирующий и не блокирующий. Обращение к неблокирующим операциям позволяет повысить эффективность параллельных вычислений.

Прототипы функций передачи и приема в реализации языка С.

```
int MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest,
int tag, MPI_Comm comm)
 IN     buf         адрес начала буфера передачи
 IN     count       количество передаваемых элементов из буфера
 IN                 передачи
 IN     datatype    тип передаваемых данных
 IN     dest        ранк(номер) процесса получателя сообщения
 IN     tag         тэг сообщения (целое число)
        comm        коммуникатор

int MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source,
int tag, MPI_Comm comm, MPI_Status *status)
 OUT    buf         адрес начала буфера приема
 IN     count       количество принимаемых элементов в буфер приема
 IN     datatype    тип принимаемых данных
 IN     source      ранк(номер) процесса источника сообщения
 IN     tag         тэг сообщения (целое число)
 IN     comm        коммуникатор
 OUT    status      объект статуса
```

Буфер передачи или приема, указанный в операциях MPI_Send и MPI_Recv состоит из некоторого количества – count последовательно расположенных элементов, начинающихся с адреса buf. Тип элементов указан в параметре datatype. Длина сообщения

определяется в количестве элементов данных. Размер элемента машинно-зависим и скрыт от пользовательского уровня.

Основные типы данных MPI, которые могут быть указаны в качестве datatype, соответствуют основным типам данных базовых алгоритмических языков: C и Fortran. Для независимости кода параллельного приложения от машинного представления типа, MPI переопределяет типы данных алгоритмических языков. Типы данных MPI для языка Fortran: MPI_INTEGER, MPI_REAL, MPI_CHARACTER и т.д. Типы данных MPI для языка C: MPI_INT, MPI_DOUBLE, MPI_FLOAT, MPI_CHAR и т.д. Кроме того, MPI содержит функции-конструкторы создания производных типов для передачи неоднородных данных, например, структур. При выполнении передачи сообщений, MPI требует соответствия типов и значений тега в функциях MPI_Send и MPI_Recv, выполняющих один обмен данными. Тип данных и числовое значение тега, указанного в операции передачи, должны соответствовать типу данных и числовому значению тега, указанному в операции приема.

Определение констант, типов данных и прототипы функций содержатся в файле mpi.h для C реализации и mpif.h для Fortran реализации.

Обращения к MPI функциям возможны только внутри программной области, ограниченной следующими функциями: функцией инициализации коммуникационного пространства - MPI_Init и функцией освобождения коммуникационного пространства – MPI_Finalize.

Пример использования парных обменов. Программа выполняет передачу числовых значений элементов массива a_send из процесса с номером 0 в массив a_recv остальных процессов группы.

```
#include <mpi.h>
void main( int argc, char *argv[] ) {
     int i, proc, size;
     double a_send[10], a_recv[10];
     MPI_Status  status;
     MPI_Init( &argc, &argv);
     MPI_Comm_rank( MPI_COMM_WORLD, &proc);
     MPI_Comm_size( MPI_COMM_WORLD, &size);
     if (proc == 0) {       /* код для процесса с номером 0 */
          for(i=0; i<10; i++)
                a_send[i]=(double)i+1.2;
           for(i=1; i<size; i++)
                 MPI_Send(&a_send[0],    10,    MPI_DOUBLE,    i,    99,
MPI_COMM_WORLD);
     }
     else     /* код для остальных процессов */
            MPI_Recv(&a_recv[0],     10,     MPI_DOUBLE,     0,     99,
MPI_COMM_WORLD, &status);
```

```
     MPI_Finalize();
}
```

5.6.1.2. Коллективные операции

Коллективными называются операции, в которых участвуют все процессы группы, в противном случае возникает ситуация взаимоблокировки процессов. MPI содержит следующие функции этого типа:

– барьерная синхронизация – MPI_Barrier;

– распространение данных от одного процесса, всем членам группы – MPI_Bcast;

– сбор данных от всех членов группы на один процесс – MPI_Gather или на все процессы группы – MPI_Allgather;

– рассылка данных от одного процесса, всем членам группы – MPI_Scatter;

– операции редукции, такие как: sum, max, min и так далее, которые возвращают результат одному процессу – MPI_Reduce или всем процессам группы –MPI_Allreduce.

Пример использования коллективных операций. Программа выполняет распространение числовых значений элементов массива a_send из процесса с номером 0 на все процессы группы, включая себя.

```
#include <mpi.h>
void main( int argc, char *argv[] ) {
     int i, proc;
     double a_send[10];
     MPI_Init( &argc, &argv);
     MPI_Comm_rank( MPI_COMM_WORLD, &proc);
     if (proc == 0) {       /* код для процесса с номером 0 */
          for(i=0; i<10; i++)
               a_send[i]=(double)i+1.2;
     }
      /* код для всех процессов */
     MPI_Bcast(&a_send[0], 10, MPI_DOUBLE, 0, MPI_COMM_WORLD);
     MPI_Finalize();
}
```

Полное руководство по спецификациям функций библиотеки MPI приведено в man директориях пакетов MVAPICH и OpenMPI.

5.6.2. Компиляция и сборка параллельного приложения

Для трансляции и сборки программ используется компилятор GNU.

Свободно распространяемая система программирования GNU включает компиляторы следующих Fortran диалектов: g77 для Fortran77, gfortran для Fortran90 и компиляторы языков C и C++: gcc для ANSI C и g++ для C++.

Компиляторы gcc, g++ поддерживают параллельный интерфейс OpenMP. Модель параллельного программирования с разделением памяти в стандарте OpenMP определяется набором директив, функций и переменных среды выполнения программ. Директивы включают конструкции создания параллельной области, разделение работ итерационного и не итерационного типа, и директивы синхронизации. Управления состоянием среды выполнения параллельных многопоточных программ осуществляется через переменные окружения или с помощью функций стандарта OpenMP и включает задание количества порождаемых потоков, контроль вложенного параллелизма и динамического изменения количества порождаемых потоков для выполнения параллельных областей.

Для компиляции и сборки параллельных MPI программ пользователю предоставлены сценарии: mpicc, mpiCC, mpif77, mpif90.

Например, для получения исполняемого модуля с именем test Fortran программы test.f90 с помощью скрипта mpif90 достаточно выполнить следующую команду:

```
mpif90 –o test test.f90
```

Подробную информацию о параметрах запуска C и Fortran компиляторов можно получить, выполнив следующие команды:

```
man gcc
man g++
man g77
man gfortran
```

5.6.3. Запуск параллельного приложения

Для выполнения параллельного приложения вычислительные узлы должны иметь доступ к каталогам файловой системы, которые содержат исполняемые файлы параллельных приложений.

Запуск параллельного приложения на вычислительных узлах может осуществляться посредством команд `mpirun` или `mpirun_rsh`. Команды используют механизмы удаленного доступа `rsh` или `ssh`.

5.6.3.1. Запуск параллельного приложения посредством утилиты mpirun_rsh пакета MVAPICH

Утилита mpirun_rsh находится в каталоге размещения mvapich2. Для большинства систем вызов команды происходит следующим образом:

```
mpirun_rsh –rsh –np 8 –hostfile nodes.lst task [arg1, arg2, … argn]
```

, где:

`-rsh` указывает, что запуск MPI-приложения на удаленных вычислительных узлах осуществляется с помощью утилиты Remote Shell;

`-np 8` – общее количество запускаемых MPI процессов;

`-hostfile nodes.lst` – конфигурационный файл, содержащий список узлов, на которых будет выполнен запуск MPI программы. Пример конфигурационного файла nodes.lst:

```
ne101
ne101
ne101
ne101
ne115
ne115
ne115
ne115
```

Данный файл описывает запуск восьми MPI-процессов, по четыре на каждом вычислительном модуле. Получить полную информацию о параметрах mpirun_rsh можно выполнением команды:

```
mpirun_rsh –help
```

5.6.3.2. Запуск параллельного приложения посредством утилиты mpirun пакета OpenMPI

Механизм запуска исполняемого файла параллельной MPI-программы на удаленных вычислительных модулях с использованием утилиты mpirun, входящей в пакет OpenMPI происходит следующим образом. Непосредственно перед запуском MPI-приложения, необходимо указать с помощью какого агента будет осуществляться запуск процессов на удаленных вычислительных модулях. Для этого необходимо перейти в каталог /opt/openmpi/gcc/etc и отредактировать файл настроек параметров openmpi-mca-params.conf в соответствии с выбранным агентом. Так строка pls_rsh_agent=rsh будет в дальнейшем указывать на необходимость запуска MPI-приложений на удаленных вычислительных модулях с помощью утилиты Remote Shell. Запуск параллельного MPI-приложения осуществляется следующим образом:

```
$ mpirun –np 4 –hostfile nodes.lst task [arg1, arg2, … argn]
```

В случае, когда агент запуска указывается непосредственно в командной строке, команда запуска выглядит следующим образом:

```
$ mpirun –mca pls_rsh_agent rsh –np 4 –hostfile nodes.lst task [arg1, arg2, … argn]
```

В данном примере будет запущено четыре процесса. Файл nodes.lst содержит список вычислительных модулей, на которых будет запущено MPI-приложение. Пример файла nodes.lst:

```
ne101
ne101
ne102
ne102
```

Данный файл описывает запуск четырех MPI-процессов по два на каждом вычислительном модуле. Полный список аргументов утилиты mpirun можно получить с помощью опции `--help`.

5.6.3.3. Запуск параллельного приложения в системе SLURM

Система SLURM предназначена для запуска и управления заданиями в пакетном режиме. Для запуска задания необходимо создать файл сценария, который подается на вход команде sbatch. Файл представляет собой обычный текстовый файл сценария на языке интерпретатора shell. Первая строка должна начинаться с символов '#!', за которыми следует абсолютный путь к интерпретатору. Например:

```
#!/bin/sh
```

Требования пакетного задания (запрашиваемые ресурсы, параметры стандартного ввода-вывода и т.д.) указываются в строках, начинающихся с директивы #SBATCH, за которой следует перечисление опций. Опции соответствуют параметрам запуска команды sbatch. Строк с директивами может быть несколько, в то же время, в одной директиве можно перечислить несколько опций.

Пример файла пакетного задания:

```
#!/bin/sh
#SBATCH --nodes=10 --tasks=20 --tasks-per-node=2
#SBATCH --partition=BATCH
#SBATCH --time=10:00
#SBATCH --output=my_job-%j.out --error=my_josb-%j.err
srun hostname | sort
```

В этом примере должен быть произведен запуск двадцати команд hostname на десяти узлах с сортировкой вывода, результат должен быть записан в текущем каталоге (где была запущена команда sbatch) в файле my_job-<номер задания>.out. При

возникновении ошибки стандартный вывод ошибок должен находиться в текущем каталоге в файле my_job-<номер задания>.err.

Опции команды sbatch задаются либо непосредственно в командной строке `sbatch [опции] <имя файла сценария> [аргументы]`, либо в файле сценария директивой #SBATCH. Часто используемые опции команды sbatch представлены в таблице 5.

Таблица 5 – Часто используемые опции команды sbatch

Опция	Описание
`--comment[=]<комментарий>`	Задает произвольный комментарий, связанный с заданием
`-D, --workdir[=]<каталог>`	Задает рабочий каталог для скрипта задания. По умолчанию используется текущий
`-e, --error[=]<шаблон имени файла>`	Задает шаблон имени файла, в который будет записан стандартный вывод ошибок. Если значением является слово none, вывод ошибок отключается совсем. Если опция не указана, стандартный вывод объединяется с выводом ошибок и записывается в файл slurm-<номер задания>.out

При указании шаблона могут применяться шаблонные символы, которые начинаются с символа '%', например, slurm-%j.out. Часто используемые шаблонные символы представлены в таблице 6.

Таблица 6 – Часто используемые шаблонные символы

Символ	Описание
%j	Номер задания (jobid), присвоенный системой SLURM
%s	Шаг задания (stepid)
%J	Соединенные точкой '.' номер задания и шаг задания (jobid.stepid)
%N	Имя узла в рамках задания. В этом случае будет создан отдельный файл вывода для каждого узла
%n	Номер узла в рамках задания. 0 – номер первого узла. В этом случае будет создан отдельный файл вывода для каждого номера
%t	Идентификатор процесса в рамках задания. Будет создан отдельный файл вывода для каждого процесса
-j, --job-name[=] <имя задания>	Устанавливает имя задания, связанное с его идентификатором. По умолчанию используется имя сценария задания
-N, --nodes[=] <число узлов>	Запрашивает число узлов для задания
-n, --tasks[=] <число процессов>	Запрашивает число процессов для задания
--ntasks-per-node[=] <число процессов на узел>	Запрашивает число процессов на узле. Таким образом, например, при запросе ста процессов на семи (-N 7 --n 100) 16-процессорных узлах, шесть узлов будут загружены полностью, но на седьмом будут использованы только 4 ядра. При запросе шестнадцати процессов на каждом из семи 16-процессорных узлов (-N 7 --ntasks-per-node=16) запустится сто двенадцать процессов
-o, --output[=] <шаблон имени файла>	Устанавливает шаблон имени файла для записи стандартного вывода. Формат значений идентичный опции --error
-t, --time[=]<время>	Устанавливает длительность выполнения задания, по завершению которого задание принудительно завершается системой SLURM. Возможные форматы значения: – минуты; – минуты:секунды; – часы:минуты:секунды; – дни-часы; – дни-часы:минуты; – дни-часы:минуты:секунды

Непосредственный запуск приложения выполняется командой `srun`:

```
$ srun [опции] <имя исполняемого файла> [аргументы исполняемого файла]
```

5.7. Система управления задачами

В состав СПО Супер-ЭВМ входит система управления задачами SLURM [17] (см. `man slurm`). Данная система позволяет организовать многозадачный многопользовательский пакетный режим выполнения задач на вычислительном поле Супер-ЭВМ и предоставляет пользователю ряд утилит для формирования задач, получения справочной информации о задачах в системе и управлению задачами.

Задачей в терминологии SLURM является совокупность атрибутов, идентифицирующих задачу, требований к ресурсам Супер-ЭВМ, предоставляемых задаче, и командного файла, содержащего выполняемые в рамках задачи команды (программы).

Каждой задаче в SLURM ставится в соответствие цифровой идентификатор, используемый для запроса информации о задаче и управления задачей. Каждая задача, кроме того, имеет текстовое имя, позволяющее отличить ее от других задач. С каждой задачей при ее постановке в очередь связывается владелец (пользователь и группа пользователей), от лица которого она запущена. Эта информация используется для контроля доступа со стороны процессов задачи к объектам системы хранения.

В целях гибкости выделения ресурсов для различных классов задач все вычислительное поле может быть разбито на отдельные области (партиции). Ниже приведено краткое описание команд SLURM, предоставляемых пользователю. Полное описание можно найти в документации по SLURM, а также в справочной системе Manual Pages.

5.7.1. squeue – просмотр очереди задач

Утилита squeue предназначена для просмотра состояний задач. Утилита выдает для каждой задачи идентификатор, имя и состояние задачи, а также некоторую дополнительную информацию. В параметрах может быть указан набор атрибутов, которые должны быть выданы, и формат вывода. На рис.5 приведен пример вывода команды.

```
[user@front bin]$ squeue
  JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
     18       T10 slurm_jo     user  PD       0:00      1 (PartitionNodeLimit)
     19       T10 slurm_jo     user  PD       0:00      1 (PartitionNodeLimit)
     20       T10 slurm_jo     user   R       0:02     10 n[0105-0108,0110-0113,0115-0116]
[user@front bin]$
```

Рис.5 – Пример вывода squeue

Можно увидеть, что на партиции T10 Супер-ЭВМ в данный момент считается задача пользователя user, имеющая уникальный идентификатор 20. Задача запущена на 10 узлах вычислительного поля, перечень которых указан в столбце NODELIST. В очереди также

находятся задачи с идентификаторами 18 и 19, которые не могут быть запущены по причине PartitionNodeLimit (превышение доступного для счета количества узлов).

5.7.2. sinfo – просмотр состояния узлов вычислительного поля

Утилита sinfo предназначена для просмотра состояния партиций вычислительного поля и вычислительных узлов. На рис.6 приведен пример вывода команды.

```
[user@front bin]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
T10*         up   infinite     44  down* n[0104,0109,0114,0119,0201-0220,0301-0320]
T10*         up   infinite     13   idle n[0105-0108,0110-0113,0115-0118,0120]
T1           up   infinite      4  down* n[0104,0109,0114,0119]
T1           up   infinite     13   idle n[0105-0108,0110-0113,0115-0118,0120]
T2           up   infinite     20  down* n[0201-0220]
T3           up   infinite     20  down* n[0301-0320]
[user@front bin]$
```

Рис.6 – Пример вывода sinfo

Можно увидеть, что на партиции T10 44 узла находятся в выключенном состоянии, 13 узлов доступны для расчетов. Кроме основной партиции T10 на вычислительном поле доступны партиции T1, T2 и T3.

5.7.3. sbatch – постановка задачи в очередь

Утилита sbatch предназначена для постановки задачи в очередь на выполнение. В параметрах утилиты указывается набор атрибутов задачи и имя командного файла. Допускается указывать атрибуты задачи в самом командном файле с помощью специальной строки:

```
#SBATCH <имя атрибута> <значение>
```

Командный файл задачи может содержать как команды ОС Linux, так и команду srun из состава SLURM, предназначенную для запуска параллельной программы.

5.7.4. srun – запуск шага задачи (программы)

Утилита srun предназначена для запуска параллельной программы из командного файла задачи. Если в параметрах утилиты не указано иное, то запускаемой программе предоставляются все ресурсы, отведенные задаче. В параметрах утилиты указывается имя файла с запускаемой программой, а также может быть указана дополнительная информация о количестве запускаемых процессов и размещении их на выделенных для задачи узлах.

5.7.5. scancel – прекращение задачи

Утилита scancel предназначена для посылки процессам, запущенным в рамках одной задачи, определенного сигнала. В параметрах указывается идентификатор задачи и номер сигнала (по умолчанию – SIGTERM). В частности, утилита позволяет прекратить выполнение задачи или удалить ее из очереди.

5.7.6. scontrol show job – получение информации о задаче

Утилита scontrol предназначена для взаимодействия с основным демоном SLURM (slurmctrld) и позволяет выполнять различные действия. Однако для пользователя наибольший интерес представляет одна из ее возможностей – вывод полной информации о задаче, для чего в командной строке указывается тип выполняемого действия (show job) и идентификатор задачи.

5.7.7. sview – графический монитор системы управления задачами

Графический монитор SLURM sview позволяет получить информацию о состоянии вычислительного поля и задач, а также выполнить большинство действий по управлению задачами в графическом режиме. Запуск графического монитора возможен при наличии X-сервера на рабочей станции пользователя.

5.7.8. sacct – просмотр статистики по расчетам задач

Утилита sacct позволяет получить информацию о просчитанных на Супер-ЭВМ задачах. На рис.7 приведен пример вывода команды.

```
[user@front bin]$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
7              sppm-1-24        T10                    24     FAILED      1:0
7.batch            batch                               1     FAILED      1:0
7.0                sppm.x                              1     FAILED      1:0
8              slurm_job        T10                    10  COMPLETED      0:0
8.batch            batch                               1  COMPLETED      0:0
8.0             hostname                              10  COMPLETED      0:0
9              sppm-2-24        T10                    24  COMPLETED      0:0
9.batch            batch                               1  COMPLETED      0:0
9.0                sppm.x                              2  COMPLETED      0:0
10             sppm-2-24        T10                    24  COMPLETED      0:0
10.batch           batch                               1  COMPLETED      0:0
10.0               sppm.x                              2  COMPLETED      0:0
11             sppm-2-24        T10                    24  COMPLETED      0:0
11.batch           batch                               1  COMPLETED      0:0
11.0               sppm.x                              2  COMPLETED      0:0
12             sppm-2-24        T10                    48  COMPLETED      0:0
12.batch           batch                               1  COMPLETED      0:0
12.0               sppm.x                              2  COMPLETED      0:0
13             sppm-4-24        T10                    96  COMPLETED      0:0
13.batch           batch                               1  COMPLETED      0:0
13.0               sppm.x                              4  COMPLETED      0:0
14             sppm-8-24        T10                   192  COMPLETED      0:0
14.batch           batch                               1  COMPLETED      0:0
14.0               sppm.x                              8  COMPLETED      0:0
15              sppm-1-1        T10                    24     FAILED      1:0
15.batch           batch                               1     FAILED      1:0
15.0               sppm.x                              1     FAILED      1:0
16             slurm_job        T10        o10         10  COMPLETED      0:0
16.batch           batch                    o10         1  COMPLETED      0:0
16.0            hostname                    o10         10  COMPLETED      0:0
17             slurm_job        T10        o10         10  COMPLETED      0:0
17.batch           batch                    o10         1  COMPLETED      0:0
17.0            hostname                    o10         10  COMPLETED      0:0
18             slurm_job        T10        o10          0    PENDING      0:0
19             slurm_job        T10        o10          0    PENDING      0:0
20             slurm_job        T10        o10         10  COMPLETED      0:0
20.batch           batch                    o10         1  COMPLETED      0:0
20.0               sleep                    o10         10  COMPLETED      0:0
[user@front bin]$
```

Рис.7 – Пример вывода sacct

5.7.9. sacctmgr – управление аккаунтами Slurm

Утилита sacctmgr позволяет получить информацию о зарегистрированных в базе Slurm аккаунтах. На рис.8 приведен пример вывода команды.

```
[root@sys /]# sacctmgr list account
   Account                Descr              Org
---------- -------------------- --------------------
       o10             otdel 10               nikiet
      root default root account                 root
```

Рис.8 – Пример вывода sacctmgr

5.8. WEB-сервис

СПО Супер-ЭВМ содержит в себе web-сервер Apache [18], предоставляющий доступ к справочной информации по протоколам HTTP и HTTPS, а также обеспечивающий возможность использования web-приложений.

5.9. СУБД

СПО Супер-ЭВМ содержит два сервера баз данных: PosgreSQL [19] и MySQL (MaraiDB) [20], а также встраиваемую базу данных SQLite [21].

5.10. Системы хранения

5.10.1. Локальные системы хранения

СПО Супер-ЭВМ обеспечивает возможность работы с локальными файловыми системами ОС Linux EXT3, EXT4, XFS.

5.10.2. Параллельная файловая система Lustre

СПО Супер-ЭВМ обеспечивает возможность работы с параллельной сетевой файловой системой Lustre [22].

5.10.3. Архивная система хранения АСХД

Архивная система хранения данных (АСХД) обладает двумя видами пользовательского интерфейса: интерфейс командной строки и web-интерфейс, реализованные в виде консольного приложения в составе ОС инструментального сервера Супер-ЭВМ и отдельного web-сервера в составе Супер-ЭВМ.

С их помощью пользователю предоставлены следующие функциональные возможности:

- архивирование данных;
- поиск архивных объектов;
- восстановление данных из архива;
- управление пользовательскими заявками;
- получение справочной информации.

Для того чтобы сохранить данные в АСХД, пользователь должен сформировать заявку на архивирование. Во время выполнения заявки АСХД помещает указанные в ней

пользовательские файлы и директории в т.н. «архивный объект», характеризующийся уникальным численным идентификатором, а также множеством системных и пользовательских атрибутов (тема, методика, заказчик, номер задачи, список доступа и т.д.), которые задаются пользователем с помощью параметров заявки.

Кроме перечисленных атрибутов архивный объект может обладать неформализованным текстовым описанием и произвольным множеством тегов, которые существенно расширяют возможности идентификации и поиска архивной информации.

Ряд параметров пользовательской заявки (тип данных, класс сервиса, тема, методика, заказчик) может принимать только предустановленные значения.

Точность соответствия между описательными атрибутами архивного объекта и его содержимым является важнейшим условием успешного нахождения данных в архиве и, как следствие, их восстановления.

5.10.3.1. Требования и ограничения

Архивная система хранения предъявляет ряд требований и ограничений:

1) описание любых архивируемых данных не должно содержать информацию, относящуюся к государственной тайне;

2) все указанные в заявке архивируемые данные должны располагаться на одном файловом ресурсе;

3) пользовательские данные, указанные в заявке на архивирование, не должны модифицироваться до момента окончания выполнения этой заявки;

4) при архивировании символьные ссылки не раскрываются и сохраняются как есть.

5) при архивировании некоторых типов данных (в частности, «Данные ММ», «Начальные данные» и «Результаты единичного расчета») АСХД требует заполнения атрибутов «методика» и «номер задачи»;

6) в АСХД пользовательские заявки обрабатываются в соответствии с установленными политиками. Приоритет отдается задаче обеспечения целостности данных, тогда как производительность выполнения заявок является второстепенной задачей;

7) в ходе своей работы АСХД контролирует значения ряда системных параметров и не допускает превышения установленных лимитов:

- суммарный объем архивируемых данных в рамках одной заявки в настоящее время ограничен 4ТБ (лимит может меняться);
- общее количество архивируемых файлов в рамках одной заявки ограничено 1млн (лимит может меняться);
- ограничено количество заявок, одновременно обслуживаемых, как отдельным агентом, медиа-сервером, так и всей системой в целом;

– ограничено время хранения в дисковом КЭШе медиа-сервера архивированных и восстановленных из архива данных;

– при необходимости в АСХД могут быть введены и другие виды лимитов.

5.10.3.2. Пользователи АСХД

АСХД обладает собственной базой данных пользователей, регистрация которых осуществляется на основании заявок на регистрацию, подаваемых руководителем отдела/лаборатории администратору АСХД в произвольной форме с перечислением ФИО и имен учетных записей.

5.10.3.3. Консольный клиент

Полный список команд консольного клиента можно получить командой:

```
asl –help
```

Консольный клиент может работать как в интерактивном, так и в пакетном режиме. Пакетный режим позволяет производить массовое архивирование данных путем запуска скриптов автоматизации.

5.10.3.3.1. Идентификация пользователя

Пользователь АСХД идентифицируется на основании unix-идентификаторов (uid и gid) запущенного процесса клиентского приложения. Статус текущего пользователя в АСХД определяется с помощью команды:

```
asl whoami
```

Зарегистрированный пользователь АСХД:

```
[ivankov@c4f1 ~]$ asl whoami
Синхронизация данных со службой каталогов
#
# Текущий пользователь ivankov:
# -------------------------------------
ФИО: Иванов Дмитрий Вадимович
uid: 1709
Является активным пользователем
В группах не состоит
Не является заказчиком
[ivanov@f1 ~]$
```

Пользователь не зарегистрирован в АСХД:

```
[user@f1 ~]$ asl whoami
Синхронизация данных со службой каталогов
[ERROR] Пользователь user НЕ зарегистрирован в ASL.
[user@f1 ~]$
```

5.10.3.3.2. Архивирование данных

Для архивирования данных пользователь указывает три обязательных параметра заявки:

– «что архивируется», т.е. путь к директории с данными либо имя текстового файла со списком архивируемых файлов и директорий;

– класс сервиса (A – архивный, B - backup);

– тип архивируемых данных (см. список типов с помощью команды `asl showdt`).

Остальные параметры заявки опциональны. Сочетание опций команды `asl archive` позволяет в полной мере описать архивируемые данные. Полный список опций можно получить командой:

```
asl archive --help
```

Синтаксис команды архивирования:

```
asl archive [--path <путь>] [--flist <имя текстового файла>]
[--class <имя класса> |-A |-B][--datatype <ид-р или код типа данных>]…
```

Некоторые опциональные параметры представлены в таблице 7.

Таблица 7 – Опции команды asl archive

Опция	Описание
`--acl[<список доступа>|<имя списка доступа>]|` `--noacl`	Со списком доступа или без
`--zadanie <номер задачи>`	Номер задачи
`-m <ид-р методики>`	Методика
`-t <код темы>`	Тема
`--customer-login <идентификатор заказчика> |` `--customer-name <ФИО вида "Вик.Вл.Петров">`	Заказчик
`--tag <тег>`	Теги архивного объекта
`--descr <описание>`	Неформализованное описание архивного объекта

Полезны опции: `--dry-run` (режим верификации заданных параметров, в котором само архивирование не производится); `--quiet` (не выводить на экран информационные сообщения); `--yes` (автоматически отвечать YES на вопросы).

Пример постановки в очередь заявки на архивирование данных, находящихся в файлах и директориях, указанных в текстовом файле flist.txt:

```
$ asl archive --flist ../flist.txt -A -d 1.1.2 -m 9 -t 20212
  -c petrov -z 12345600 --descr ‘Test 1’ --yes
Список файлов на архивирование задан в файле ../flist.txt
Сформирована заявка на СОЗДАНИЕ архива
```

```
-----------------------------------------------
Атрибуты заявки:
  Номер задания:  12345600
  Заказчик:       petrov: Петров Петр Петрович
  Методика:       Gamma (id=9)
  Тема:           НИР “Проба пера" (id=20212)
  Класс:          A
  Тип данных:     1.1.2: Данные ММ (id=4)
  Комментарий к архиву: “Test 1”
Список файлов/каталогов для архива:
    /lustre/gamma/12345601
    /lustre/gamma/12345602
    /lustre/gamma/12345603
Архивируется 39 файлов/каталогов. Общий объем: 753 MB
-------------------------------------------------
Задание принято на обработку
Задание поставлено в очередь на выполнение
tid=55555
$
```

В случае успешной постановки заявки в очередь на выполнение, на экране появляется информация об идентификаторе заявки (tid=55555).

В ходе выполнения заявки на архивирование данных в поле комментария отображается идентификатор соответствующего архивного объекта, в который будут помещены указанные в заявке пользовательские данные.

После успешного окончания выполнения заявки на архивирование данных в АСХД соответствующий архивный объект становится адресуемой информационной единицей, а указанные в заявке параметры становятся атрибутами объекта, который может быть найден в результате выполнения поискового запроса и впоследствии восстановлен.

5.10.3.3.3. Автозаполнение атрибутов

Для минимизации усилий пользователя при описании архивируемых данных применяется механизм автозаполнения, суть которого заключается в том, что значения редко изменяемых параметров заявки можно установить в нескольких местах (в файле персональных настроек, в файле-шаблоне и в переменных окружения), откуда они будут браться автоматически.

Файл конфигурации (персональных настроек) клиента с именем DEFAULTS находится в домашней директории пользователя в директории $HOME/.asl. Рекомендуется заносить в него значения наиболее постоянных атрибутов – методику, тип данных, списки доступа. Кроме того, здесь могут быть указаны следующие параметры:

```
dir_for_restore=<директория для восстановления>
template=<имя файла шаблона>
default_acl=<имя списка доступа по умолчанию>
```

Именованные списки доступа указывается в секции [acl] файла конфигурации:

```
<имя списка доступа>=<список доступа>
```

, где список доступа имеет вид:

```
"[user1[,user2…]]:[group1[,group2…]]"
```

Шаблон – текстовый файл с именем template.asl, в котором для ряда атрибутов перечислены их значения для автозаполнения. Его размещают на одном уровне иерархии с данными, на которые он оказывает действие.

В файлах шаблонов, как и в файле конфигурации, основные параметры задаются переменными с префиксом ASL_:

```
ASL_CUSTOMER=<ФИО или login заказчика>
ASL_METHOD=<идентификатор методики>
ASL_THEME=<название или номер темы>
ASL_TNUMBER=<номер задания в формате ГГЗАДАВА>
ASL_DESCRIPTION=<описание>
ASL_DATATYPE=<идентификатор типа данных>
ASL_CLASS=<класс сервиса>
```

Не рекомендуется модифицировать файл конфигурации непосредственно. Следует использовать команду клиента по работе с файлом конфигурации:

```
asl setdefaults
```

Макрос – другой удобный способ назначения атрибутов архивных объектов в файлах шаблонов. В настоящее время реализован макрос имени директории '%D%', удобный, например, для автозаполнения номера задачи по имени директории при массовом архивировании директорий, имена которых соответствуют номерам заданий:

```
ASL_TNUMBER=%D%
```

Все источники значений атрибутов имеют свой приоритет, поэтому значение какого-либо атрибута, заданное в источнике с более высоким приоритетом, переопределяет значения этого атрибута, указанного в источнике с меньшим приоритетом:

– самый низкий приоритет имеет файл конфигурации клиента, где задаются значения атрибутов, которые редко изменяются. Например, список доступа, тема, методика;

– более высокий приоритет имеют значения атрибутов, заданные в общем файле шаблонов. Он размещается на одном уровне иерархии с директориями и файлами, на которые он оказывает действие. В общем шаблоне определяются атрибуты, подходящие для большинства директорий и файлов. Например, заказчик и методика для группы каталогов задач методики;

– если среди архивируемых данных встречается директория, для которой нужно указать уникальные значения атрибутов, это можно сделать, поместив эти атрибуты в

частный шаблон внутри этой директории. Например, номер задачи, заказчик, комментарий. Приоритет частного шаблона выше приоритета общего шаблона;

– более высокий приоритет имеют значения атрибутов, заданные специальными переменными окружения с префиксом ASL_ (аналогичны описанным для шаблонов). Они переопределяют значения из шаблонов и из файла персональных настроек. Переменные окружения удобно использовать в скриптах;

– высший приоритет имеют значения атрибутов, указанные в опциях командной строки.

Возможности автоматизации задания атрибутов объекта в сочетании с интерфейсными командами консольного клиента позволяют разрабатывать скрипты и пользовательские «обертки» для автоматизации массовых операций архивирования данных.

Посмотреть значения по умолчанию, заданные в файле DEFAULTS, можно командой:

```
asl showdefaults
```

Задать новые значения – командой:

```
asl setdefaults
```

5.10.3.3.4. Поиск архивных объектов

В консольном клиенте реализована функция просмотра списка архивных объектов с возможностью фильтрации по ограниченному набору атрибутов.

Команда `asl olist` без опций выдает список всех архивных объектов текущего пользователя. Команда имеет опции фильтрации по некоторым атрибутам. Фильтрация идет по набору условий «атрибут = значение», связанных логическим «И». Опция `--all` определяет просмотр и фильтрацию среди всех зарегистрированных архивных объектов (по умолчанию фильтруются только объекты, принадлежащие текущему пользователю). Полный список опций можно получить командой:

```
asl olist --help
```

Синтаксис команды поиска:

```
asl olist [-h] [-d <тип данных>] [-z <#задания>] [-m <методика>] [-t <тема>] [-c <заказчик>] [-a | --all] [--format acl | short| medium | long] [--withtags]…
```

Основные параметры команды asl olist представлены в таблице 8.

Таблица 8 - Основные параметры команды asl olist

Параметр	Описание
-d <идентификатор/код типа данных>	Тип данных
-z <номер задания>	Номер задачи
-m <идентификатор методики>	Методика
-t <код темы>	Тема
--customer <идентификатор заказчика>	Заказчик
--format [acl | short | medium | long]	Характер выдаваемой информации
–withtags	Только объекты с тегами

Пример команды поиска объектов с конкретными методикой и темой:

```
$ asl olist -m 9 -t 87392 --format long
```

5.10.3.3.5. Восстановление данных из архива

Восстановление архивных объектов производится путем создания заявки на восстановление данных из архива, в параметрах которой пользователь указывает один или несколько идентификаторов требуемых объектов, а также место их будущего размещения (полный путь к директории) на одном из оперативных ресурсов СХ. Пользователь должен обладать правами записи в указанную директорию. При этом будет восстановлена иерархия директорий, характерная для оперативного ресурса – источника архивных данных.

Синтаксис команды восстановления данных из архива:

```
asl restore [--path <путь выгрузки>] [--yes][--descr <описание>]
<ид-р объекта> [<ид-р объекта> …]
```

Пример заявки восстановления данных архивных объектов 10001 и 10002 в директорию /lustre/gamma:

```
$ asl restore --path /lustre/gamma 10001 10002
```

Полный список опций команды восстановления из архива можно получить командой:

```
asl restore --help
```

5.10.3.4. Web-клиент

Web-клиент АСХД – универсальный, кроссплатформенный клиент, позволяющий выполнять все доступные пользователю операции: постановка заявки на архивирование,

поиск архивных объектов, постановка заявки на восстановление объектов из архива, удаление объектов, отмена пользовательской заявки.

Доступ к web-клиенту АСХД осуществляется по сетевому адресу http://asl. Для доступа необходимо указать пользовательский логин и пароль СУПЕР-ЭВМ.

5.10.3.4.1. Архивирование данных

Для архивирования данных выберите пункт меню «Создать заявку». Постановка заявки – это двухэтапный процесс. В первой форме «Параметры заявки» (см. рис. 9) необходимо задать значения описательных атрибутов, архивируемых данных, а также определить списки доступа к архивному объекту. Поля, обязательные для заполнения, будут дополнительно выделены.

Во второй форме (закладке браузера) «Файлы архива» (см. рис. 10) необходимо в рамках определенного файлового ресурса задать архивируемые файлы/директории и, при необходимости, указать исключения из них.

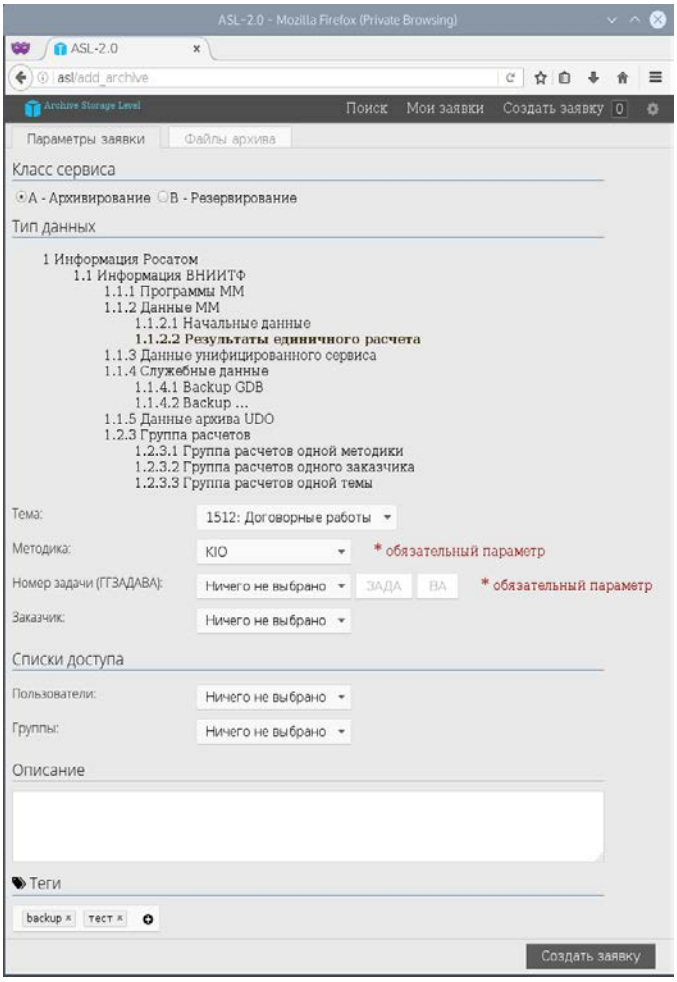

Рис. 9 – Задания атрибутов архивируемых данных

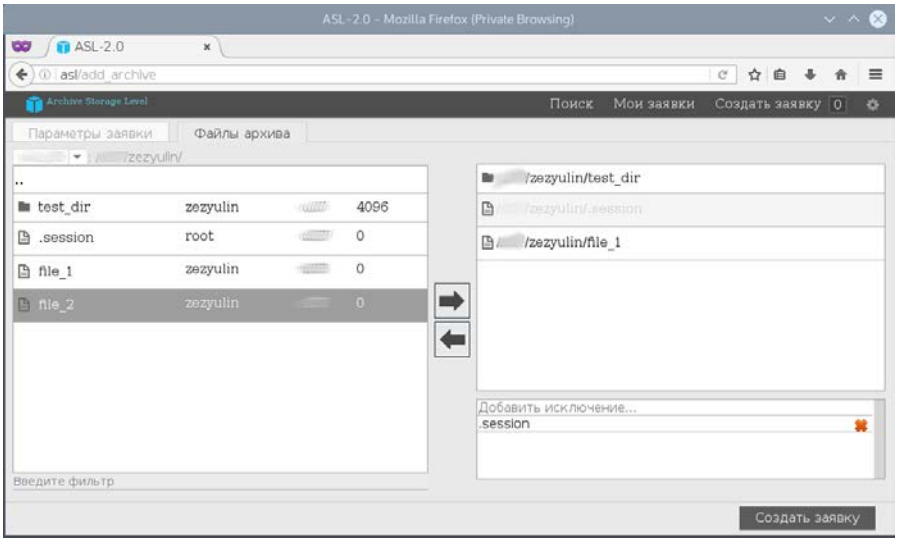

Рис. 10 – Формирования списка архивируемых данных

Чтобы инициировать выполнение заявки на архивирование, необходимо заполнить данные формы и нажать на кнопку «Создать заявку».

5.10.3.4.2. Поиск архивных данных

Для осуществления поиска архивных объектов выберите пункт меню «Поиск». Результаты поиска можно отфильтровать по следующим атрибутам: владелец, тип данных, методика, заказчик, номер задачи, дата создания заявки (см. рис. 11).

Также существует возможность использовать полнотекстовый поиск по полям «Описание» и «Теги» (см. рис. 12), достоинством которого является способность учитывать морфологию языка. Для переключения между фильтрацией и полнотекстовым поиском используйте кнопку «Поиск/Фильтр», расположенную в левой верхней части страницы. Так как в текущей реализации web-клиента полнотекстовый поиск нельзя совмещать с фильтрацией, то функции «Поиск» и «Фильтр» являются взаимоисключающими.

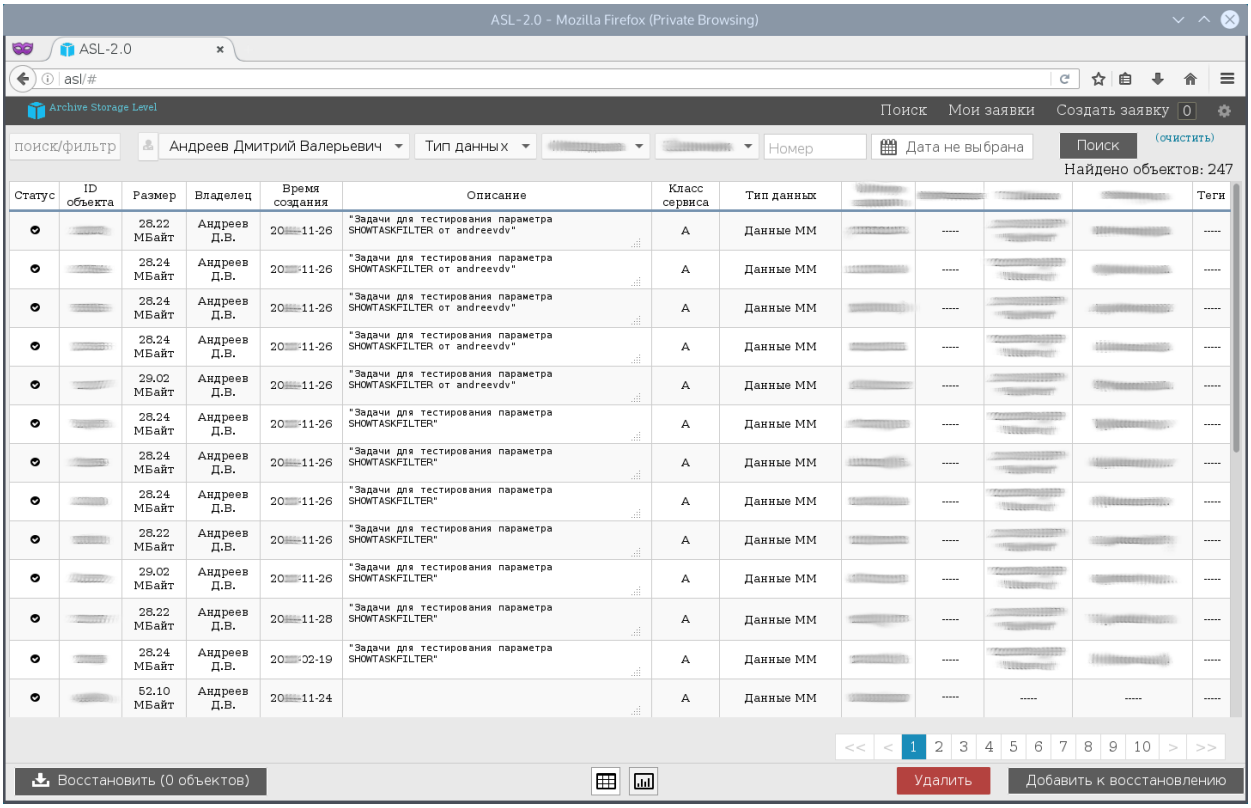

Рис. 11 – Фильтрация архивных объектов

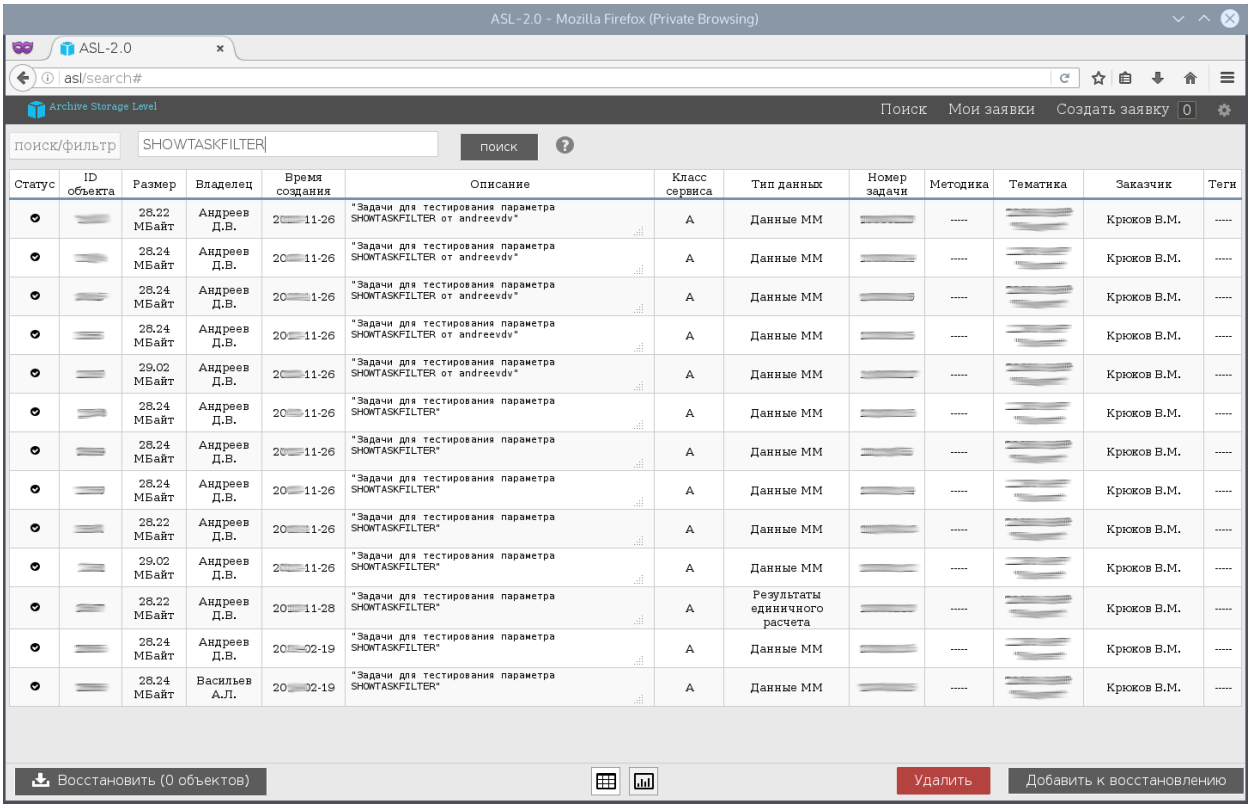

Рис. 12 – Полнотекстовый поиск

5.10.3.4.3. Поиск архивных данных

Для восстановления объектов выберите пункт меню «Поиск» (см. рис. 13). Чтобы добавить выделенные объекты к заявке на восстановление, нажмите на кнопку «Добавить к восстановлению». Чтобы выбрать место для восстановления и инициировать выполнение заявки, нажмите на кнопку «Восстановить», расположенную на нижней панели. При этом в указанном месте для восстановления будет воссоздана иерархия директорий, характерная для оперативного ресурса – источника архивных данных.

Для удаления объектов выберите пункт меню «Поиск». Чтобы удалить выделенные объекты, нажмите на кнопку «Удалить», расположенную на нижней панели.

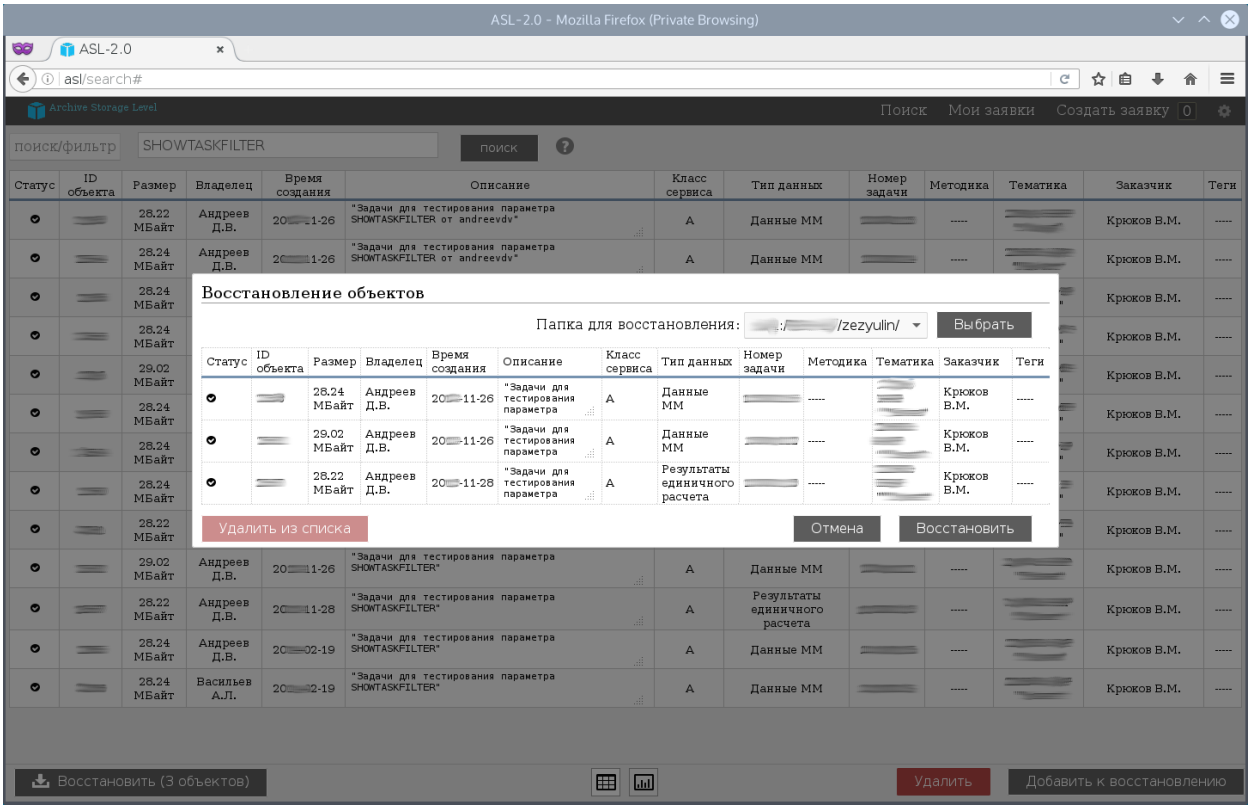

Рис. 13 – Восстановление объектов

5.10.3.4.4. Управление пользовательскими заявками

Для управления пользовательскими заявками выберите пункт меню «Мои заявки» (см. рис. 14). На данной странице будут отображены пользовательские заявки за определенный период времени. Чтобы изменить период времени, задайте необходимое значение в текстовом поле «заявки за XX дней».

Статус каждой заявки определяет цвет символа-стрелки в первом столбце таблицы: зеленый цвет – успешно завершенные заявки; красный – завершенные с ошибкой; серый – отмененные заявки и т.д. Каждая заявка сопровождается полем комментария, в котором отображаются идентификаторы архивных объектов, соответствующие успешно

выполненным заявкам на архивирование, либо причина невыполнения заявок. Чтобы отменить выполнение выделенных заявок, нажмите на кнопку «Отменить заявку».

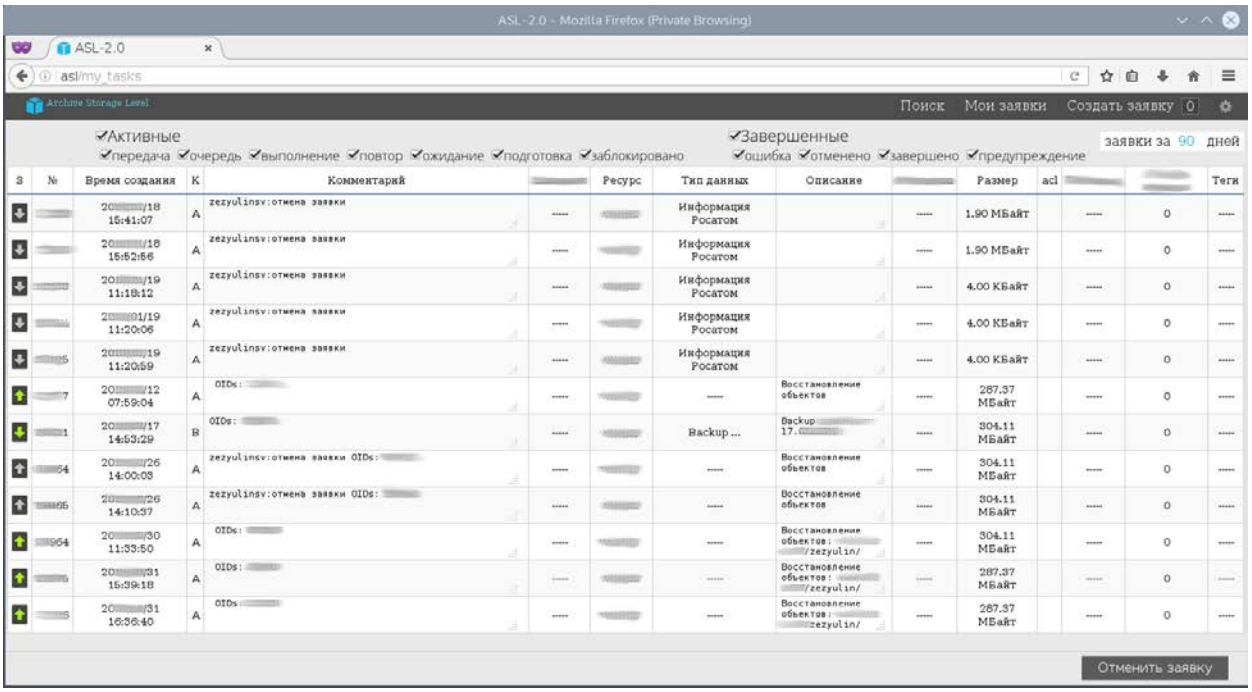

Рис. 14 – Управление пользовательскими заявками

5.10.4. Облачная система хранения СТРИЖ

Облачная система хранения СТРИЖ (ОСХД) предоставляет пользователям возможность хранить большие объемы данных с гарантированной целостностью путем репликации данных как на Супер-ЭВМ, так и ЛВС предприятия, предоставляя доступ к данным со всех устройств ЛВС.

Для пользователя доступны следующие интерфейсы взаимодействия с ОСХД:

- RESTful API (взаимодействие по протоколу HTTP);
- консольный клиент swift [23];
- FTP (предоставляется сервисом FTPcloudfs);
- web-сервис Gate (web-интерфейс собственной разработки);
- утилита синхронизации данных SynCloud (собственной разработки).

5.10.4.1. Основы аутентификации и авторизации

Для получения доступа к ресурсам ОСХД пользователю необходимо пройти процедуру аутентификации и авторизации с помощью сервиса Keystone [24]. В результате успешной авторизации пользователь получит временный билет доступа – токен.

Для получения токена пользователь должен передать сервису следующие данные:

– идентификатор учетной записи (логин);

– пароль от учетной записи;

– домен учетной записи.

В качестве базы данных пользователей домена ВК используется LDAP-каталог.

Для получения токена, привязанного к определенному проекту (scope-токена) пользователь также должен передать:

– имя проекта;

– домен проекта.

В случае успешной авторизации сервис keystone возвращает токен и URL-адрес точки доступа (endpoint) объектной системы хранения Swift. Токен имеет ограниченное время жизни – 1 час. По истечении этого времени запросы с использованием этого токена будут завершаться ошибкой. Необходимо повторить процедуру аутентификации и авторизации для получения нового токена.

Для получения токена требуется выполнить POST-запрос к сервису keystone. Данные для аутентификации и авторизации передаются в теле запроса в виде json-структуры.

В случае успеха в ответе от сервиса возвращаются:

– токен (в заголовке «X-Subject-Token»);

– адрес точки доступа (в основном теле ответа).

Пример json-структуры с данными для получения scope-токена пользователем ovchinnikovpg для персонального проекта ovchinnikovpg:

```
$ cat auth.json
{
    "auth": {
        "identity": {
            "methods": ["password"],
            "password": {
                "user": {
                    "name": "ovchinnikovpg",
                    "password": "verySecretPassword",
                    "domain": {
                        "name": "vk"
                    }
                }
            }
        },
        "scope":{
            "project": {
                "name": "ovchinnikovpg",
                "domain": {
                    "name": "vk"
```

```
                }
            }
        }
    }
}
```

Пример запроса аутентификации/авторизации:

```
$ curl -i –XPOST \
 'http://auth.query.cloud:5000/v3/auth/tokens' \
 -d @auth.json -H 'Content-Type:application/json'
HTTP/1.1 201 Created
<…>
X-Subject-Token: a82fef376d204358a536de0eac9680d3
<…>
{
    <…>
                "endpoints": [
                    {
                        "region_id": "0",
                        "url":
"http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8",
                        "region": "0",
                        "interface": "public",
                        "id": "0d0f9929b79041b893797cd93ff075ef"
                    },
    <…>
}
```

Для проверки подлинности токена требуется выполнить HEAD-запрос к сервису keystone, в заголовках которого передать два параметра «X-Auth-Token» и «X-Subject-Token», значениями которых является проверяемый токен. В случае успешной проверки возвращается код 200.

Пример запроса для проверки токена:

```
$ curl –i \
 -XHEAD 'http://auth.query.cloud:5000/v3/auth/tokens'\
 -H 'X-Auth-token:a82fef376d204358a536de0eac9680d3'\
 -H 'X-Subject-Token:a82fef376d204358a536de0eac9680d3'
HTTP/1.1 200 OK
<…>
```

Для досрочного отзыва токена требуется выполнить DELETE-запрос к сервису keystone, в заголовках которого передать два параметра «X-Auth-Token»и «X-Subject-Token», значениями которых является проверяемый токен.

Пример запроса на отзыв токена:

```
$ curl –i\
 -XDELETE 'http://auth.query.cloud:5000/v3/auth/tokens'\
 -H 'X-Auth-token:a82fef376d204358a536de0eac9680d3'\
 -H 'X-Subject-Token:a82fef376d204358a536de0eac9680d3'
```

```
HTTP/1.1 204 No Content
<…>

$ curl –i\
 -XHEAD 'http://auth.query.cloud:5000/v3/auth/tokens'\
 -H 'X-Auth-token:a82fef376d204358a536de0eac9680d3'\
 -H 'X-Subject-Token:a82fef376d204358a536de0eac9680d3'
HTTP/1.1 401 Unauthorized
<…>
```

5.10.4.2. Особенности пространства имен

Полный адрес объекта в облачной системе хранения имеет вид <account_name>/<container_name>/<object_name>.

Каждый проект (в терминологии keystone) сопоставляется с аккаунтом (в терминологии Swift) по схеме «1:1». Аккаунт следует воспринимать как область хранения. Внутри аккаунта может быть множество контейнеров. Внутри каждого контейнера может быть множество объектов. Но контейнер не обладает вложенностью, так что максимальная «глубина» иерархии внутри аккаунта равняется единице.

Имена объектов могут содержать '/'. При использовании FTP-интерфейса и web-интерфейса Gate такое именование объектов может предоставить «псевдоиерархию», что позволит проще систематизировать данные. Но следует помнить, что реальной иерархии не создается.

5.10.4.3. Особенно хранения больших объектов

Максимальный размер хранящегося в ОСХД объекта – 5ГБ. При загрузке файла, размер которого превышает указанный лимит, данный файл необходимо разбить на несколько объектов-чанков и загрузить файл-манифест, который предоставляет расположение чанков и порядок их объединения.

Существует два способа хранения больших объектов:

1) Static Large Object (SLO);

2) Dynamic Large Object (DLO).

Различия между ними заключаются в способе описания чанков в манифесте, который определяет свойства хранения. В SLO файл-манифест имеет заголовок «X-Static-Large_object» со значением «true» и содержит полный упорядоченный список чанков. Чанки описываются тремя параметрами:

1) path – путь до чанка;

2) etag – md5-сумма чанка;

3) size_bytes – размер чанка в байтах.

Пример содержания файла-манифеста SLO:

```
[
    {
        "path": "container1/object23",
        "etag": "099028f729a5847bdc2e071039c227c6",
        "size_bytes: "1468006"
    },
    {
        "path": "container234/pseudodir/object0",
        "etag": "a923ea73b97983eec477eeb293993a5b",
        "size_bytes: "268948"
    }
]
```

Чанки SLO могут располагаться в различных контейнерах и иметь различный формат имени. Изменение содержимого SLO влечет за собой изменение файла-манифеста. Манифест загружается только после загрузки всех чанков, при загрузке проверяются md5-суммы. При ошибке проверки манифест не загрузится. Порядок объединения чанков в объект определяется содержимым манифеста. Целостность SLO обеспечивает контролем md5-сумм чанков.

В DLO манифест – это файл нулевой длины, у которого есть дополнительный заголовок «X-Object-Manifest», значение которого имеет вид <container>/<prefix>. Этот заголовок указывает на контейнер, в котором находятся чанки, и префикс имени чанков. Чанки могут располагаться только в одном контейнере. Порядок объединения чанков определяется их именами – объединение производится в порядке сортировки имен. Изменение DLO не требует изменения манифеста. Манифест может быть загружен в любой момент. Целостность DLO не гарантируется.

Следующие пользовательские интерфейсы умеют автоматически работать с большими объектами:

- консольный клиент swift (SLO, DLO);
- сервис FTP FTP-cloudfs (только DLO);
- web-сервис Gate (SLO (загрузка и выгрузка), DLO (только выгрузка)).

При взаимодействии через RESful API пользователь должен сам выполнять все действия (разбитие объекта на чанки и их загрузку, создание и загрузку файла-манифеста). Утилита синхронизации SynCloud работу с большими объектами не поддерживает.

5.10.4.4. Взаимодействие с помощью RESTful API

Для взаимодействия через RESTful API необходимо пройти процедуру аутентификации/авторизации и использовать для дальнейшего взаимодействия полученный токен и адрес сетевой точки доступа.

Полное описание RESTful API (полный набор команд и их параметров) доступно в документации к ПО OpenStack Swift.

Далее приведены примеры базовых операций над сущностями в ОСХД. Примеры взаимодействия с ОСХД через RESTful API выполнены с помощью утилиты curl.

5.10.4.4.1. Базовые операции с аккаунтом

Запросы к аккаунту выполняются по адресу <endpoint_url> (адрес точки доступа).

Получение информации об аккаунте (метаданные) осуществляется HEAD-запросом. Информация об аккаунте возвращается в заголовках ответа.

```
$ curl –i\
 -XHEAD
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8'\
 -H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae'
HTTP/1.1 204 No Content
Content-Length: 0
X-Account-Object-Count: 6775
X-Account-Storage-Policy-Policy-0-Bytes-Used: 434686967262
X-Account-Storage-Policy-Policy-0-Container-Count: 4
X-Timestamp: 1495531197.05274
X-Account-Storage-Policy-Policy-0-Object-Count: 6775
X-Account-Bytes-Used: 434686967262
X-Account-Container-Count: 4
Content-Type: application/json; charset=utf-8
Accept-Ranges: bytes
x-account-project-domain-id: 5cf373b9aa76495383f8982a2f86a0ea
X-Trans-Id: tx931bc6e4d446491eaaa74-005d66619e
X-Openstack-Request-Id: tx931bc6e4d446491eaaa74-005d66619e
Date: Wed, 28 Aug 2019 11:12:30 GMT
```

Получение списка контейнеров осуществляется GET-запросом. Список контейнеров возвращается в основном теле ответа.

```
$ curl\
 -XGET
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8'\
 -H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae'
bios_update
elastic_snapshot
home2cloud.trmt
reports
```

5.10.4.4.2. Базовые операции с контейнерами

Запросы к контейнеру выполняются по адресу <endpoint_url>/<container_name> (адрес точки доступа и имя контейнера).

Создание контейнера осуществляется PUT-запросом. Результат выполнения возвращается в коде ответа:

```
$ curl -i –XPUT \
-H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae' \
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8/test_container'
HTTP/1.1 201 Created
```

Получение информации о контейнере (метаданные) осуществляется HEAD-запросом. Информация о контейнере возвращается в заголовках ответа.

```
$ curl –i\
 -XHEAD
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8/test_container'\
 -H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae'
HTTP/1.1 204 No Content
Content-Length: 0
X-Container-Object-Count: 0
X-Timestamp: 1566971895.18292
Accept-Ranges: bytes
X-Storage-Policy: Policy-0
Last-Modified: Wed, 28 Aug 2019 11:15:39 GMT
X-Container-Bytes-Used: 0
Content-Type: application/json; charset=utf-8
X-Trans-Id: txac6c5a97b4ec48569da12-005d666310
X-Openstack-Request-Id: txac6c5a97b4ec48569da12-005d666310
Date: Wed, 28 Aug 2019 11:18:40 GMT
```

Получение списка объектов в контейнере осуществляется GET-запросом. Список объектов возвращается в основном теле ответа.

```
$ curl\
 -XGET
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8/bios_update'\
 -H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae'
afulnx64.tar
firmware
firmware/X10DRC8.608
firmware/X10DRH8.618
```

Удаление контейнера осуществляется DELETE-запросом. Результат выполнения возвращается в коде ответа.

```
$ curl –i\
 -XDELETE
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8/test_container'\
 -H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae'
HTTP/1.1 204 No Content
```

5.10.4.4.3. Базовые операции с объектами

Запросы к объекту выполняются по адресу <endpoint_url>/<container_name>/<object_name> (адрес точки доступа, имя контейнера и имя объекта).

Создание (загрузка) объекта осуществляется PUT-запросом. Результат выполнения возвращается в коде ответа.

```
$ curl –i\
 -XPUT
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8/test_container/test_object'\
 -d'test_content_of'\
 -H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae'
HTTP/1.1 201 Created
```

Получение информации об объекте (метаданные) осуществляется HEAD-запросом. Информация об объекте возвращается в заголовках ответа.

```
$ curl –i\
 -XHEAD
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8/test_container/test_object'\
 -H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae'
HTTP/1.1 200 OK
Content-Length: 15
Accept-Ranges: bytes
Last-Modified: Wed, 28 Aug 2019 11:37:37 GMT
Etag: ea1885ac7373f4c545b76e54f15b4875
X-Timestamp: 1566992256.22796
Content-Type: application/x-www-form-urlencoded
X-Trans-Id: tx195496e026754ab9a61bd-005d6667ef
X-Openstack-Request-Id: tx195496e026754ab9a61bd-005d6667ef
Date: Wed, 28 Aug 2019 11:39:27 GMT
```

Получение содержимого (выгрузка) объекта осуществляется GET-запросом. Содержимое объекта возвращается в основном теле ответа.

```
$ curl\
 -XGET
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8/test_container/test_object'\
 -H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae'
test_content_of
```

Удаление объекта осуществляется DELETE-запросом. Результат выполнения возвращается в коде ответа.

```
$ curl –i\
 -XDELETE
'http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f541
8/test_container/test_object'\
 -H 'X-Auth-token:a2647f9e80e044eea6b5a7820f3692ae'
HTTP/1.1 204 No Content
```

5.10.4.5. Консольный клиент swift

5.10.4.5.1. Аутентификация и авторизация

Порядок аутентификации/авторизации с использованием консольного клиента аналогичен порядку при использовании RESTful API: сначала нужно получить токен, который используется для доступа к ресурсам ОСХД. Получение токена осуществляется командой `swift auth`, которой передаются данные для аутентификации в виде параметров:

- адрес сервиса аутентификации (keystone): параметр `--os-auth-url`;
- логин: параметр `--os-username`;
- пароль учетной записи: параметр `--os-password`;
- домен учетной записи: параметр `--os-user-domain-name`;
- имя проекта: параметр `--os-project-name`;
- домен проекта: параметр `--os-project-domain-name`.

При успешной аутентификации/авторизации программа возвращает в стандартный вывод токен и точку доступа, которые используются при дальнейших запросах в течение времени жизни токена.

Пример успешной аутентификации/авторизации пользователя ovchinnikovpg для доступа к персональному проекту ovchinnikovpg:

```
$ swift --os-auth-url http://auth.query.cloud/v3 --os-username
ovchinnikovpg --os-password VerySecretPassword --os-user-domain-name
vk --os-project-name ovchinnikovpg --os-project-domain-name vk auth
export
OS_STORAGE_URL=http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89
888bce4b90f5418
export OS_AUTH_TOKEN=5546c0c1c8424fcc89a54d5cb489ebe6
```

Пример неуспешной аутентификации:

```
$ swift --os-auth-url http://auth.query.cloud/v3 --os-username
ovchinnikovpg --os-password VeryWRONGPassword --os-user-domain-name vk
--os-project-name ovchinnikovpg --os-project-domain-name vk auth
Unauthorized. Check username/id, password, tenant name/id and
user/tenant domain name/id.
```

При запросах к ОСХД полученная информация передаются в виде параметров `--os-auth-token` (токен) и `--os-storage-url` (точка доступа).

Пример обращения к аккаунту с использованием параметров:

```
$ swift --os-storage-url
http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f5418
 --os-auth-token 5546c0c1c8424fcc89a54d5cb489ebe6 list
bios_update
elastic_snapshot
home2cloud.trmt
reports
```

По истечении времени жизни токена попытка обращения к ОСХД завершится ошибкой:

```
$ swift --os-storage-url
http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89888bce4b90f5418
--os-auth-token 5546c0c1c8424fcc89a54d5cb489ebe6 list
Authorization Failure. Authorization failed: Could not find token:
5546c0c1c8424fcc89a54d5cb489ebe6 (HTTP 404) (Request-ID: req-12e11fb5-
aed5-4bf7-a123-2c302a9f57e6)
```

Для удобства использования и сокращения вводимой команды все указанные параметры можно сохранить в переменных окружения. Пример набора команд для аутентификации с сохранением параметров в переменных окружения:

```
$ export OS_AUTH_URL=http://auth.query.cloud/v3
$ export OS_USERNAME=ovchinnikovpg
$ export OS_USER_DOMAIN_NAME=vk
$ export OS_PASSWORD=VerySecretPassword
$ export OS_PROJECT_NAME=ovchinnikovpg
$ export OS_PROJECT_DOMAIN_NAME=vk
$ swift auth
export
OS_STORAGE_URL=http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89
888bce4b90f5418
export OS_AUTH_TOKEN=b6902e09bf334928b17c45a52c2a52f9

$ export\
```

```
OS_STORAGE_URL=http://store.query.cloud:8080/v1/AUTH_4d13a50ae3f94cb89
888bce4b90f5418
$ export OS_AUTH_TOKEN=b6902e09bf334928b17c45a52c2a52f9
[trmt@underscape home2cloud]$ swift list
bios_update
elastic_snapshot
home2cloud.trmt
reports
```

При истечении времени жизни токена или смены проекта необходимо заново пройти процедуру аутентификации/авторизации. При использовании переменных окружения требуется изменить значения этих переменных.

5.10.4.5.2. Взаимодействие с ОСХД

Все указанные примеры выполняются с сохраненными переменными окружения. Полный список всех команд и параметров программы swift доступен с помощью команды:

```
$swift --help
```

5.10.4.5.2.1. Базовые операции с аккаунтом

Получение информации об аккаунте (метаданные):

```
$ swift stat
Account:             AUTH_4d13a50ae3f94cb89888bce4b90f5418
Containers:          4
Objects:             6775
Bytes:               434686967262
Containers       in  4
policy "policy-0":
Objects  in  policy  6775
"policy-0":
Bytes   in   policy  434686967262
"policy-0":
X-Account-Project-   5cf373b9aa76495383f8982a2f86a0ea
Domain-Id:
X-Openstack-         tx36bcc2773655491c965da-005d67976b
Request-Id:
X-Timestamp:         1495531197.05274
X-Trans-Id:          tx36bcc2773655491c965da-005d67976b
Content-Type:        application/json; charset=utf-8
Accept-Ranges:       bytes
```

Получение списка контейнеров:

```
$ swift list
bios_update
elastic_snapshot
home2cloud.trmt
reports
```

5.10.4.5.2.2. Базовые операции с контейнерами

Создание контейнера:

```
$ swift post test_container
$ swift list
bios_update
elastic_snapshot
home2cloud.trmt
reports
test_container
```

Получение информации о контейнере (метаданные):

```
$ swift stat test_container
Account:            AUTH_4d13a50ae3f94cb89888bce4b90f5418
Container:          test_container
Objects:            0
Bytes:              0
Read ACL:
Write ACL:
Sync To:
Sync Key:
Accept-Ranges:      bytes
X-Storage-Policy:   Policy-0
Last-Modified:      Thu, 29 Aug 2019 09:27:36 GMT
X-Timestamp:        1495531197.05274
X-Trans-Id:         tx36bcc2773655491c965da-005d67976b
Content-Type:       application/json; charset=utf-8
X-Openstack-        tx00b9db7e324647d692ddb-005d6798d7
Request-Id:
```

Получение списка объектов в контейнере:

```
$ swift list bios_update
afulnx64.tar
firmware
firmware/X10DRC8.608
firmware/X10DRH8.618
```

Получение содержимого (выгрузка) всех объектов в контейнере:

```
$ mkdir bios_update
$ cd bios_update/
$ ls -l
итого 0

$ swift download bios_update
firmware [auth 0.000s, headers 0.020s, total 0.020s, 0.000 MB/s]
afulnx64.tar [auth 0.000s, headers 0.051s, total 0.166s, 6.179 MB/s]
firmware/X10DRH8.618  [auth  0.000s,  headers  0.024s,  total  0.199s,
84.325 MB/s]
```

```
firmware/X10DRC8.608 [auth 0.000s, headers 0.038s, total 0.321s,
52.317 MB/s]

$ ls -l
итого 1004
-rw-rw-r-- 1 trmt trmt 1024000 авг 29 14:44 afulnx64.tar
drwxrwxr-x 2 trmt trmt    4096 авг 29 14:44 firmware
```

Если в имени объекта имеется путь, то он восстанавливается, корнем считается текущая директория.

Удаление контейнера (вместе с содержимым):

```
$ swift delete test_container
test_container

$ swift list
bios_update
elastic_snapshot
home2cloud.trmt
reports
```

5.10.4.5.2.3. Базовые операции с объектами

Создание (загрузка) объекта:

```
$ swift upload test_container home2cloud
home2cloud

$ swift list test_container
home2cloud
```

Поддерживается загрузка нескольких файлов (перечисление имен через пробел). Возможна загрузка как файлов, так и директорий. В качестве имени можно передать:

- базовое имя (home2cloud);
- относительный путь (bin/home2cloud);
- абсолютный путь (/home/trmt/bin/home2cloud).

При загрузке объекта его имя полностью совпадает с передаваемым путем загружаемого файла:

```
$ swift upload test_container /home/trmt/bin/home2cloud/home2cloud
home/trmt/bin/home2cloud/home2cloud

$ swift list test_container
home/trmt/bin/home2cloud/home2cloud
home2cloud
```

Для загрузки большого объекта следует указать размер чанка с помощью опции -S <размер_чанка_в_байтах> (--segment_size <размер_чанка_в_байтах>). По умолчанию создается DLO. Для создания SLO требуется дополнительно указать опцию `--use-slo`. Файл манифеста создается автоматически.

Получение информации об объекте (метаданные):

```
$ swift stat test_container home2cloud
Account:            AUTH_4d13a50ae3f94cb89888bce4b90f5418
Container:          test_container
Object:             home2cloud
Content-Type:       application/octet-stream
Content Length:     1374
Last Modified:      Thu, 29 Aug 2019 09:27:36 GMT
ETag:               b349dd666647d60b5549da1c9021334a
Meta Mtime:         1566802030.472712
Accept-Ranges:      bytes
X-Timestamp:        1567070855.51765
X-Trans-Id:         tx62d7e6a215e542b58682a-005d679b07
X-Openstack-        tx62d7e6a215e542b58682a-005d679b07
Request-Id:
```

Получение содержимого (выгрузка) объекта:

```
$ mkdir test_container
$ cd test_container/
$ ls -l
итого 0

$ swift download test_container home2cloud
home2cloud [auth 0.000s, headers 0.278s, total 0.278s, 0.005 MB/s]

$ ls -l
итого 4
-rw-rw-r-- 1 trmt trmt 1374 авг 26 11:47 home2cloud
```

Если имя объекта содержит псевдоиерархию, она восстанавливается на файловой системе при выгрузке объекта. Начальной точкой псевдоиерархии считается директория выгрузки объекта.

Удаление объекта:

```
$ swift delete test_container home2cloud
home2cloud

$ swift list test_container
home/trmt/bin/home2cloud/home2cloud
```

5.10.4.6. Взаимодействие с ОСХД по протоколу FTP

При взаимодействии через интерфейс FTP поддерживаются только следующие операции:

- создание/удаление контейнеров;
- загрузка/скачивание/удаление объектов;
- отображение псевдоиерархии при наличии '/' в имени объекта.

Другие операции над сущностями ОСХД не поддерживаются.

Примеры взаимодействия с ОСХД через файловый интерфейс выполнены с помощью утилиты `lftp`.

5.10.4.6.1. Аутентификация и авторизация

Параметры подключения через FTP-интерфейс имеют следующий вид: <project>.<username>@store.query.cloud

, где:

<project> – имя проекта, к которому выполняется подключение;

<user> – логин;

store.query.cloud – доменное имя ОСХД.

При подключении не указывается домен, так как сервис ftpcloudfs сконфигурирован на взаимодействие с определенным доменом.

Пример успешного подключения пользователя *ovchinnikovpg* к групповому проекту *o3*:

```
$ lftp -u o3.ovchinnikovpg store.query.cloud
Пароль: <***>
lftp o3.ovchinnikovpg@store.query.cloud:~> ls
drwxr-xr-x  964  ovchinnikovpg  ovchinnikovpg  45422806  May  21  10:55
CLOUD_DOCS
```

Пример подключения с ошибкой пользователя *ovchinnikovpg* к персональному проекту *ovchinnikovpg*:

```
$ lftp -u ovchinnikovpg.ovchinnikovpg store.query.cloud
Пароль: <***>
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:~> ls
ls:  Ошибка  регистрации:  530  Failed  to  authenticate  user
ovchinnikovpg.ovchinnikovpg:  [errno  5]  unauthorized.  check
username/id, password, tenant name/id and user/tenant domain name/id.:
```

5.10.4.6.2. Взаимодействие с ОСХД

5.10.4.6.2.1. Просмотр списка контейнеров и объектов

Для просмотра списка контейнеров и объектов используется команда `ls`. При выполнении команды в аккаунте (на верхнем уровне псевдоиерархии) будет показан список контейнеров:

```
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:~> ls
drwxr-xr-x   4 ovchinnikovpg ovchinnikovpg 34578432 Dec 19  2018
bios_update
drwxr-xr-x 206 ovchinnikovpg ovchinnikovpg 196745248756 Sep 21  2018
elastic_snapshot
drwxr-xr-x 320 ovchinnikovpg ovchinnikovpg 237542414432 Nov 01  2018
home2cloud.trmt
```

```
drwxr-xr-x 6245 ovchinnikovpg ovchinnikovpg 364725642 Jan 15  2019
reports
```

При выполнении команды в контейнере или псевдодиректории будет показан список объектов и псевдодиректорий на текущем уровне псевдоиерархии:

```
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/bios_update> ls
-rw-r--r--    1 ovchinnikovpg ovchinnikovpg   1024000 May 31 16:13
afulnx64.tar
drwxr-xr-x    1 ovchinnikovpg ovchinnikovpg         0 Aug 28 10:54
firmware
```

5.10.4.6.2.2. Навигация

Для перехода между уровнями псевдоиерархии используется команда `cd`:

```
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/> cd bios_update/
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/bios_update>
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/bios_update> cd ..
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/>
```

Переход может быть осуществлен по абсолютному пути:

```
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_1> cd /
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/>
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/> cd
/test_container/test_pseudo_0/
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0>
```

5.10.4.6.2.3. Создание контейнеров и псевдодиректорий

Для создания контейнеров и псевдодиректорий используется команда `mkdir`.

При выполнении команды в аккаунте (на верхнем уровне псевдоиерархии) будет создан контейнер:

```
lftp      ovchinnikovpg.ovchinnikovpg@store.query.cloud:/>      mkdir
test_container
mkdir ok, `test_container' создан
```

При выполнении команды `mkdir` в контейнере или псевдодиректории будет создана псевдодиректория:

```
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/> cd
test_container
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container>
mkdir test_pseudo_0
```

```
mkdir ok, `test_pseudo_0' создан
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container> ls
drwxr-xr-x   1 ovchinnikovpg ovchinnikovpg        0 Aug 28 11:28
test_pseudo_0
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container> cd
test_pseudo_0/
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> mkdir test_pseudo_00
mkdir ok, `test_pseudo_00' создан
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> ls
drwxr-xr-x   1 ovchinnikovpg ovchinnikovpg        0 Aug 28 11:29
test_pseudo_00
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> cd test_pseudo_00/
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0/test_pseudo_00>
```

Если в создаваемом имени встречается символ '/', то это имя интерпретируется как путь. При создании пути с несколькими уровнями вложенности рекомендуется использовать опцию `-p`:

```
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/> cd
test_container/
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container>
mkdir -p test_pseudo_1/test_pseudo_10
mkdir ok, `test_pseudo_1/test_pseudo_10' создан
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container> ls
drwxr-xr-x   1 ovchinnikovpg ovchinnikovpg        0 Aug 28 11:30
test_pseudo_0
drwxr-xr-x   1 ovchinnikovpg ovchinnikovpg        0 Aug 28 11:30
test_pseudo_1
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container> cd
test_pseudo_1
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_1> ls
drwxr-xr-x   1 ovchinnikovpg ovchinnikovpg        0 Aug 28 11:30
test_pseudo_10
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_1>
```

5.10.4.6.2.4. Загрузка объектов в облако

Загрузка объектов в облако выполняется с помощью команды put. Файл на локальной файловой системе может быть выбран как по относительному пути:

```
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> put home2cloud
1374 байта перемещены
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> ls
-rw-r--r--   1 ovchinnikovpg ovchinnikovpg     1374 Aug 28 11:48
home2cloud
drwxr-xr-x   1 ovchinnikovpg ovchinnikovpg        0 Aug 28 11:44
test_pseudo_00
```

Так и по абсолютному:

```
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> put /home/trmt/ninja-enter
1382 байта перемещены
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> ls
-rw-r--r--   1 ovchinnikovpg ovchinnikovpg     1374 Aug 28 11:48
home2cloud
-rw-r--r--   1 ovchinnikovpg ovchinnikovpg     1382 Aug 28 11:49
ninja-enter
drwxr-xr-x   1 ovchinnikovpg ovchinnikovpg        0 Aug 28 11:44
test_pseudo_00
```

Команда put не может быть применена на директории или с wildcard:

```
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_1/test_pseudo_10> put /home/trmt/bin/
put: /home/trmt/bin/: Это каталог
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_1/test_pseudo_10> put /home/trmt/bin/*
put: /home/trmt/bin/*: Нет такого файла или каталога
```

С помощью опции -o указывается имя объекта назначения. Имя может быть простым или содержать '/'. В случае, если имя содержит '/', создается псевдоиерархия. Команда put не может создавать контейнеры.

```
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/> put
/home/trmt/bin/inotify_strace.sh -o /bin/test/inotify_strace.sh
461 байт перемещен
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/>
```

Загрузка больших объектов в облако осуществляется с помощью DLO в автоматическом режиме.

Для загрузки директории в облако используется команда `mirror` с опцией `-R`. При загрузке директории в аккаунт (на верхнем уровне псевдоиерархии) загружаемая директория становится контейнером:

```
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/> mirror -R /home/trmt/bin/
Всего: 4 каталога, 21 файл, 0 ссылок
Созданы: 21 файл, 0 ссылок
8245520 байтов перемещено за 4 секунды (1.90Мб/с)
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/> ls
drwxr-xr-x   1 ovchinnikovpg ovchinnikovpg        0 Aug 28 12:00 bin
drwxr-xr-x     4 ovchinnikovpg ovchinnikovpg 34578432 Dec 19   2018 bios_update
drwxr-xr-x 6245 ovchinnikovpg ovchinnikovpg 364725642 Jan 15   2019 reports
drwxr-xr-x     6 ovchinnikovpg ovchinnikovpg       2756 Aug 28 11:28 test_container
```

При загрузке директории в контейнер или псевдодиректорию загружаемая директория продолжает псевдоиерархию:

```
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pseudo_1> mirror -R /home/trmt/bin/
Всего: 4 каталога, 21 файл, 0 ссылок
Созданы: 21 файл, 0 ссылок
8245520 байтов перемещено за 4 секунды (1.98Мб/с)
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pseudo_1> ls
drwxr-xr-x   1 ovchinnikovpg ovchinnikovpg        0 Aug 28 12:01 bin
drwxr-xr-x     1 ovchinnikovpg ovchinnikovpg           0 Aug 28 11:45 test_pseudo_10
```

5.10.4.6.2.5. Выгрузка объектов из облака

Для выгрузки объектов из облака используется команда `get`:

```
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pseudo_1/bin> ls
drwxr-xr-x     1 ovchinnikovpg ovchinnikovpg           0 Aug 28 12:14 home2cloud
-rw-r--r--     1 ovchinnikovpg ovchinnikovpg        7920 Aug 28 12:01 mkdirmany775
```

```
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_1/bin> get mkdirmany775 -o /tmp/
7920 байтов перемещено
```

Опция `-o` позволяет указать директорию назначения на локальной файловой системе.

С помощью команды `put` нельзя выгрузить контейнер или псевдодиректорию.

Для выгрузки контейнера или псевдодиректории используется команда `mirror`:

```
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_1/bin> mirror home2cloud /tmp/
Всего: 1 каталог, 3 файла, 0 ссылок
Созданы: 3 файла, 0 ссылок
3723 байта перемещены
```

5.10.4.6.2.6. Удаление контейнеров, псевдодиректорий объектов

Для удаления контейнеров, псевдодиректорий и объектов используется команда `rm`:

```
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> ls
-rw-r--r--    1 ovchinnikovpg ovchinnikovpg       1374 Aug 28 11:48
home2cloud
-rw-r--r--    1 ovchinnikovpg ovchinnikovpg       1382 Aug 28 11:49
ninja-enter
drwxr-xr-x    1 ovchinnikovpg ovchinnikovpg          0 Aug 28 11:44
test_pseudo_00
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> rm home2cloud
rm ok, `home2cloud' удален
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> ls
-rw-r--r--    1 ovchinnikovpg ovchinnikovpg       1382 Aug 28 11:49
ninja-enter
drwxr-xr-x    1 ovchinnikovpg ovchinnikovpg          0 Aug 28 11:44
test_pseudo_00
```

Для удаления контейнера или директории псевдоиерархии необходимо использовать опцию `-r`. Эта опция также удаляет все содержимое контейнера или псевдоиерархии:

```
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_0> cd ..
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container> ls
drwxr-xr-x    1 ovchinnikovpg ovchinnikovpg           0 Aug 28 11:44
test_pseudo_0
drwxr-xr-x    1 ovchinnikovpg ovchinnikovpg           0 Aug 28 11:30
test_pseudo_1
lftp ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container> rm
-r test_pseudo_0
rm ok, 3 файла удалено
```

Для удаления контейнеров и псевдодиректорий также используется команда `rmdir`:

```
lftp
ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container/test_pse
udo_1> rmdir test_pseudo_10/
rmdir ok, `test_pseudo_10/' удален
```

Данная команда не может удалить непустую псевдодиректорию или контейнер:

```
lftp    ovchinnikovpg.ovchinnikovpg@store.query.cloud:/test_container>
rmdir test_pseudo_0
rmdir: Ошибка доступа: 550 Directory not empty. (test_pseudo_0)
```

5.10.4.7. Web-сервис Gate

Для начала работы требуется аутентификация, которая выполняется на соответствующей странице (рис. 15).

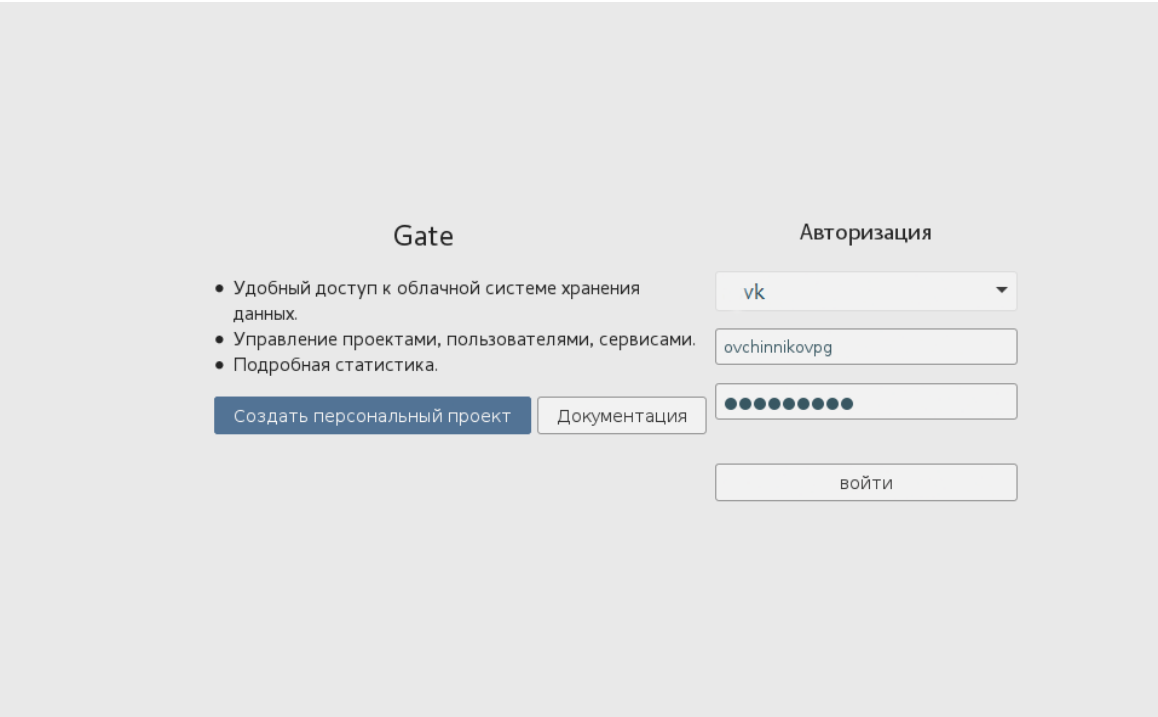

Рис. 15 – Страница аутентификации

После аутентификации требуется выбрать из списка доступных проектов проект для взаимодействия (рис. 16).

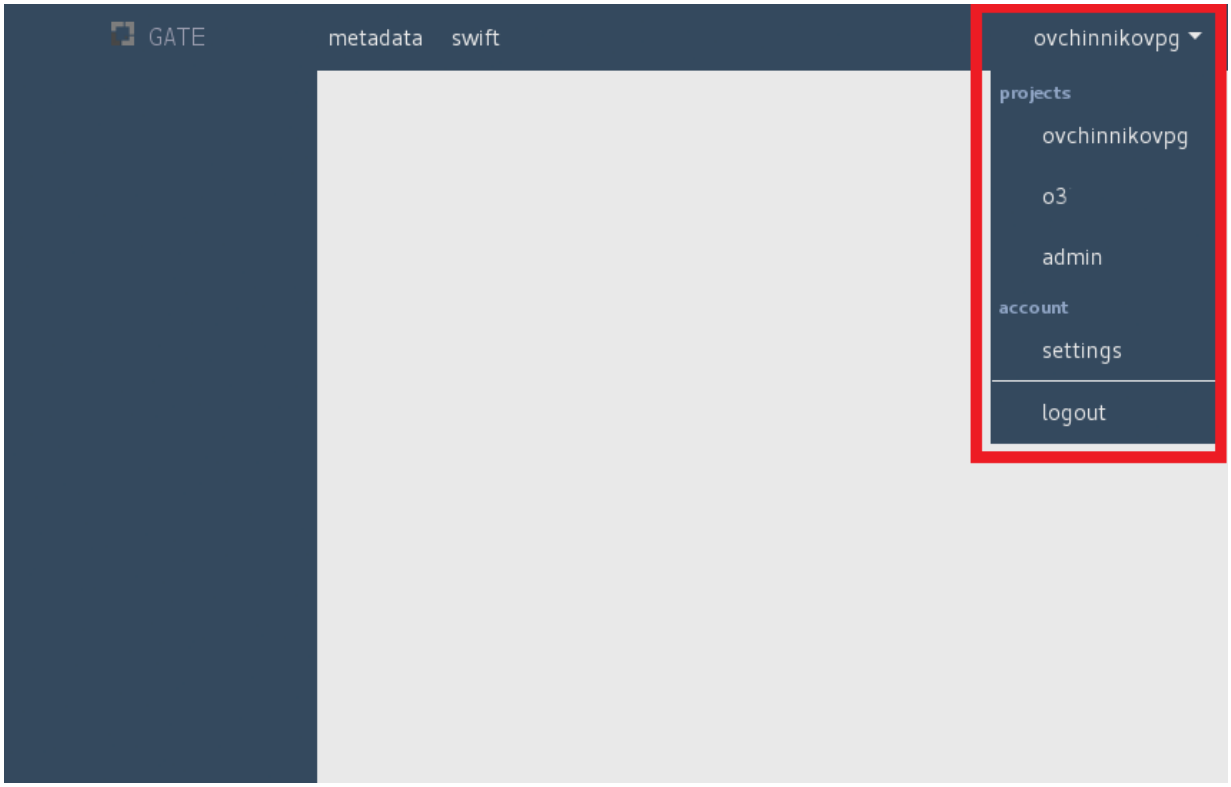

Рис. 16 – Выбор проекта

Для начала работы с ресурсами ОСХД выбранного проекта требуется перейти во вкладку «swift» (рис. 17).

Рис. 17 – Переход к ресурсам проекта

На верхнем уровне иерархии (аккаунте) отображается список контейнеров (рис. 18). Для получения информации о контейнере (метаинформации) требуется кликнуть один раз в любом месте строки этого контейнера, кроме имени контейнера (рис. 19).

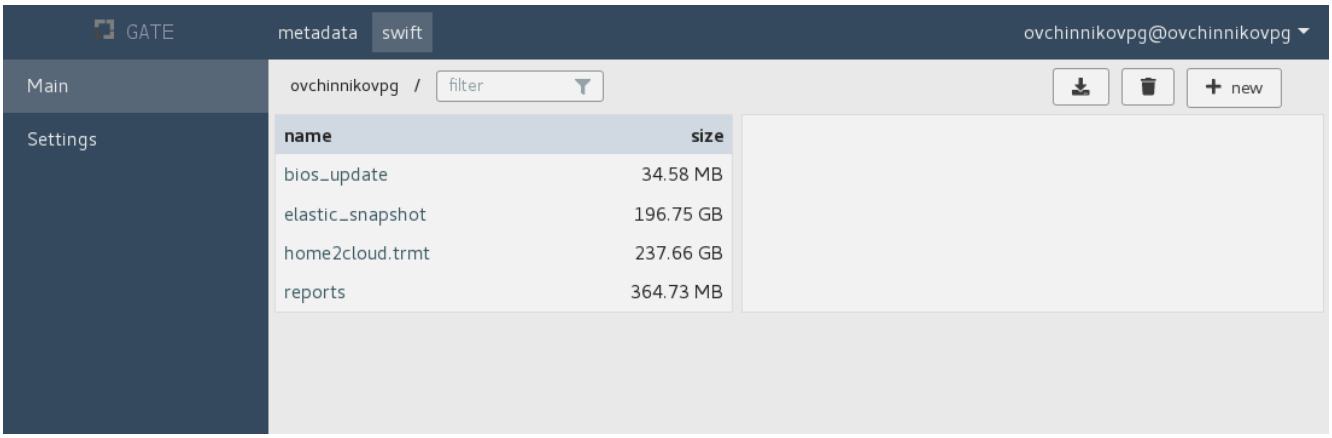

Рис. 18 – Общий вид ресурсов проекта

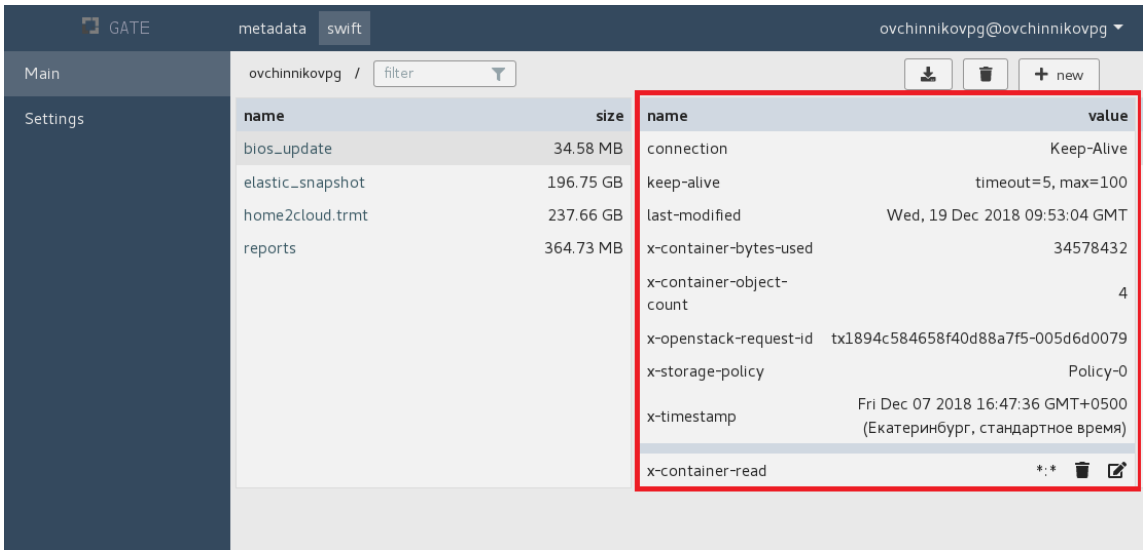

Рис. 19 – Получение метаинформации о контейнере

Для перехода внутрь контейнера требуется кликнуть на имя контейнера. При этом изменяется строка текущего положения в иерархии, показывая уровень погружения (рис. 20).

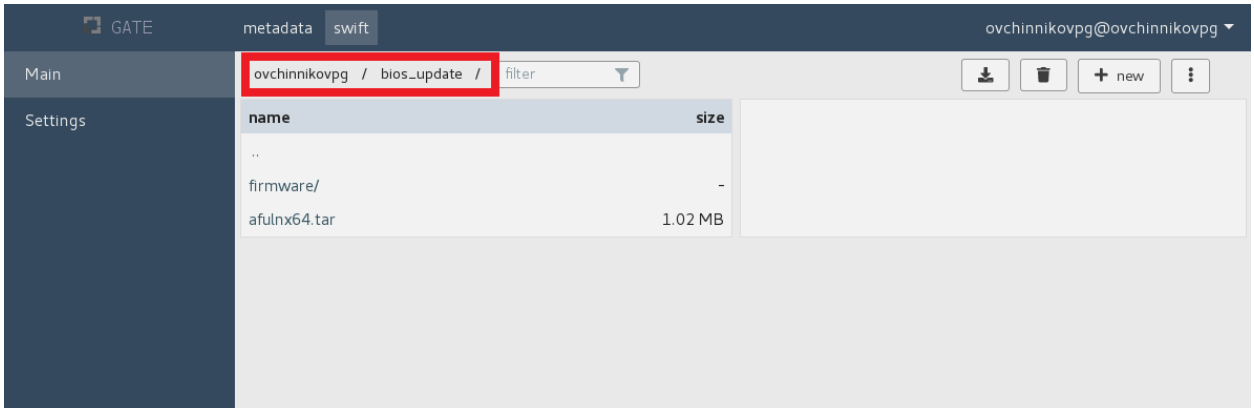

Рис. 20 – Просмотр содержимого контейнера

По умолчанию включен режим просмотра содержимого с псевдоиерархией. Для перехода внутрь псевдодиректории требуется кликнуть на ее имя. При этом также изменяется строка текущего положения в иерархии, показывая уровень погружения (рис. 21).

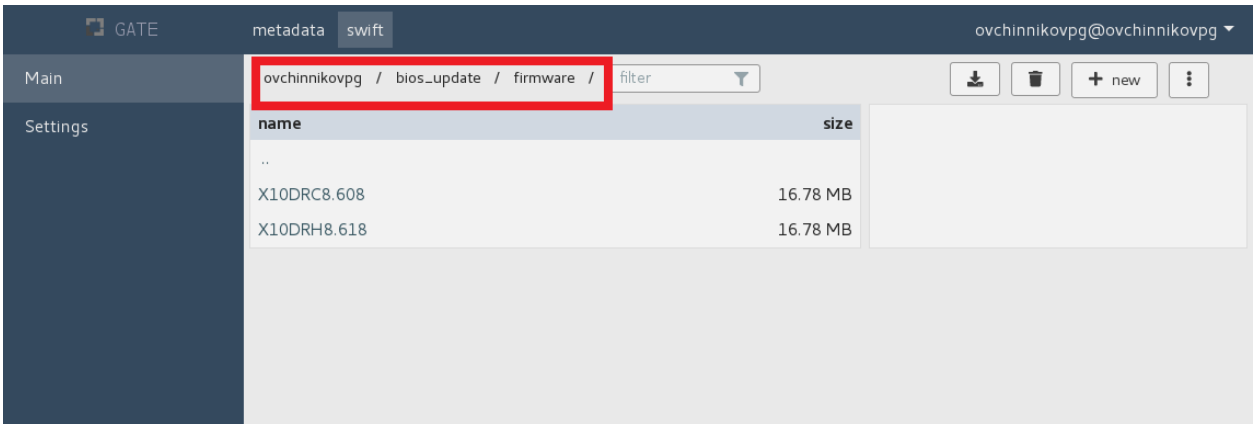

Рис. 21 – Просмотр содержимого псевдодиректории

Чтобы отключить режим псевдоиерархии, требуется в выпадающем меню настроек отображения выбрать пункт «switch pseudo-hierarchy». После этого внутри контейнера будут отображаться все объекты с реальными именами (рис. 22).

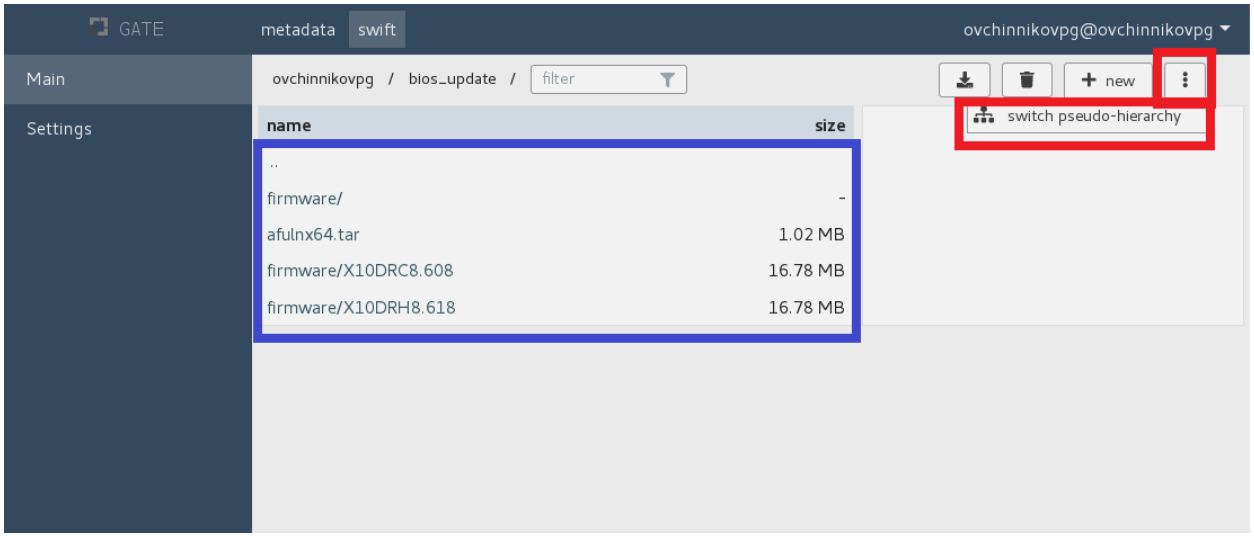

Рис. 22 – Переключение режима отображения объектов

Для создания контейнера требуется в корне иерархии (аккаунте) выбрать из выпадающего меню «+ new» пункт «+ container» (рис. 23).

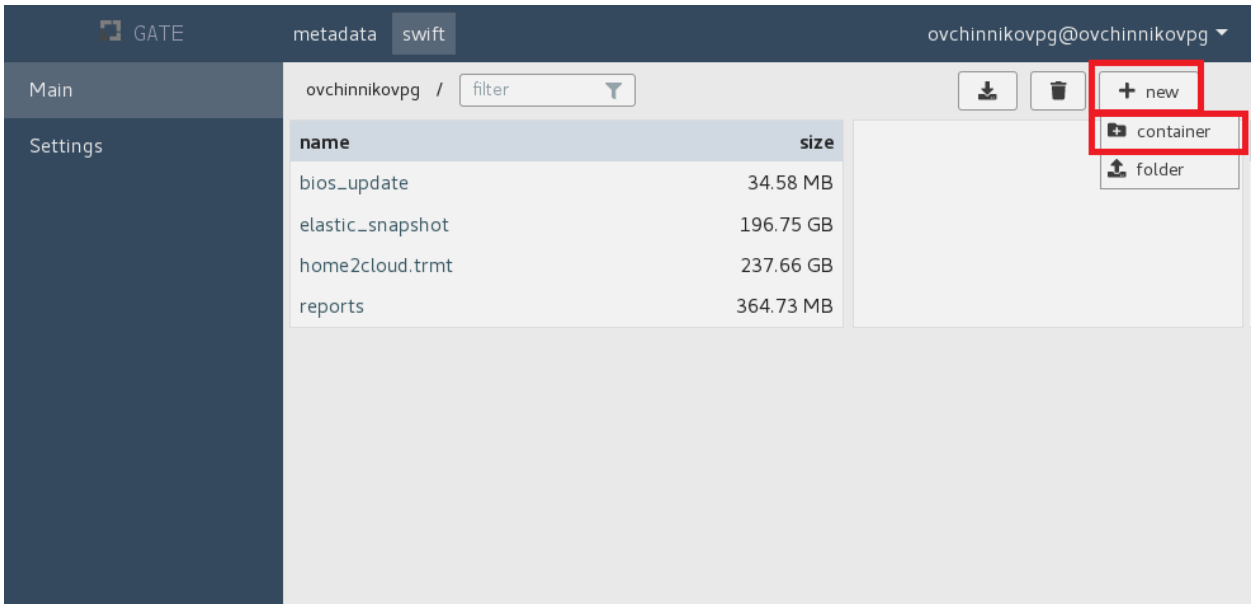

Рис. 23 – Меню создания контейнера

После этого появится диалоговое окно (рис. 24), в котором требуется ввести имя контейнера и опционально выбрать политику хранения для этого контейнера. По нажатию на кнопку «ОК» создастся контейнер.

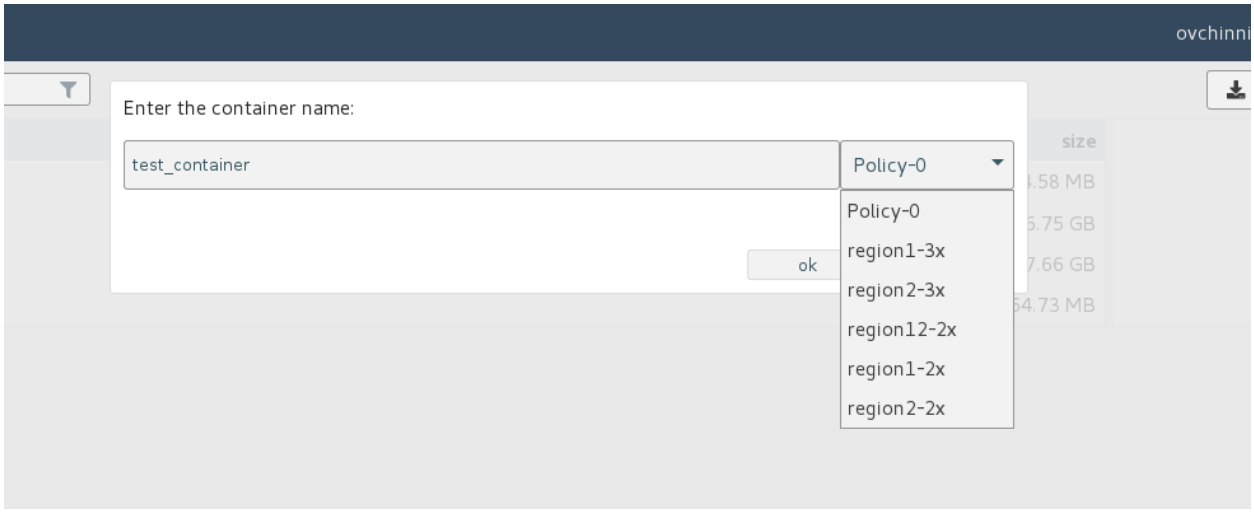

Рис. 24 – Создание контейнера

Также контейнер может быть создан при загрузке директории в корень иерархии (аккаунт). В таком случае контейнер будет носить имя загружаемой директории. (ВНИМАНИЕ: функционал загрузки директории работает в браузерах Firefox версии 53 и новее). Для загрузки директории в выпадающем меню «+ new» требуется выбрать пункт «directory». Появится окно файлового менеджера, в котором требуется выбрать загружаемую директорию (рис. 25).

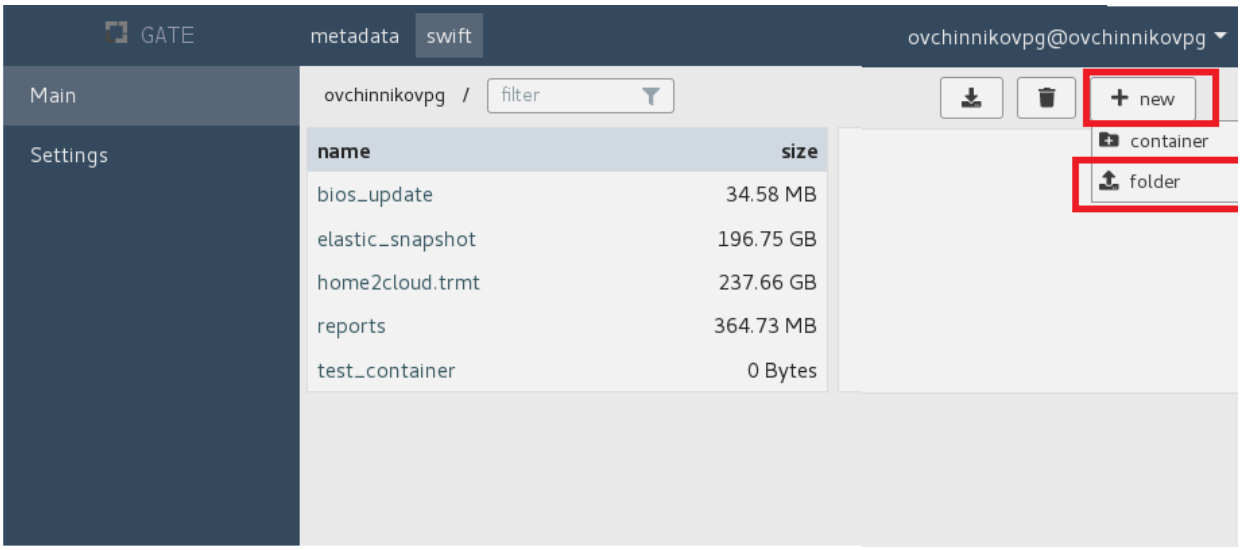

Рис. 25 – Меню загрузки директории

При успешной загрузке директории в корне появится одноименный контейнер, содержимое которого будет повторять содержимое загружаемой директории (рис. 26).

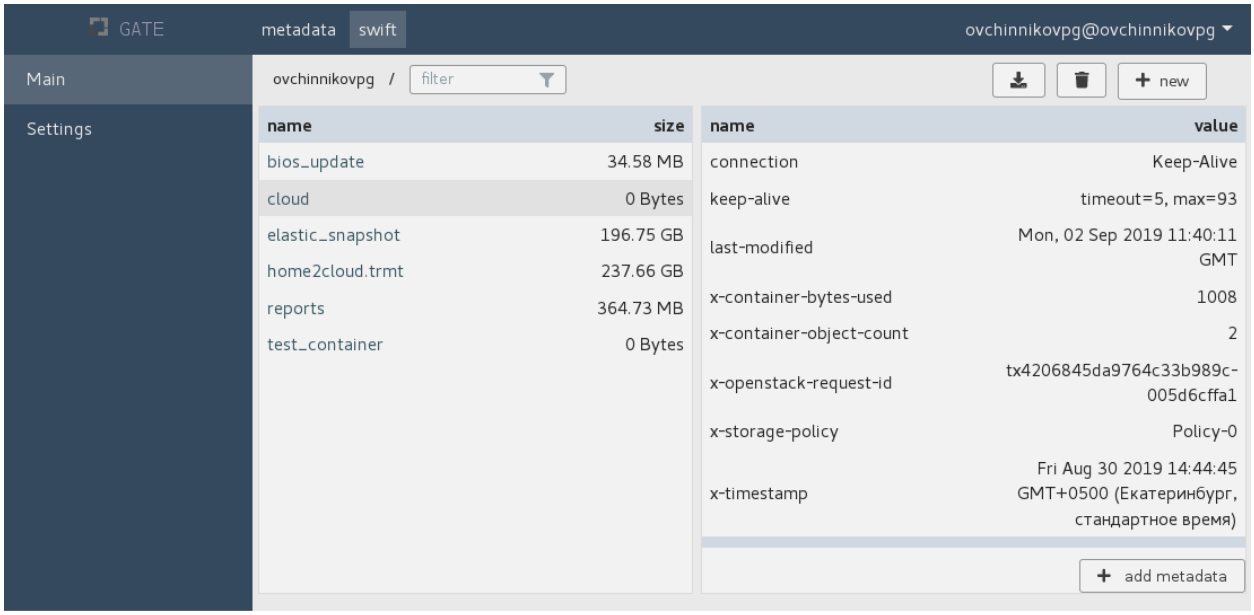

Рис. 26 – Метаинформация загруженной директории

Для создания псевдодиректории требуется внутри контейнера или псевдодиректории в выпадающем меню «+ new» выбрать пункт «+ folder» (рис. 27).

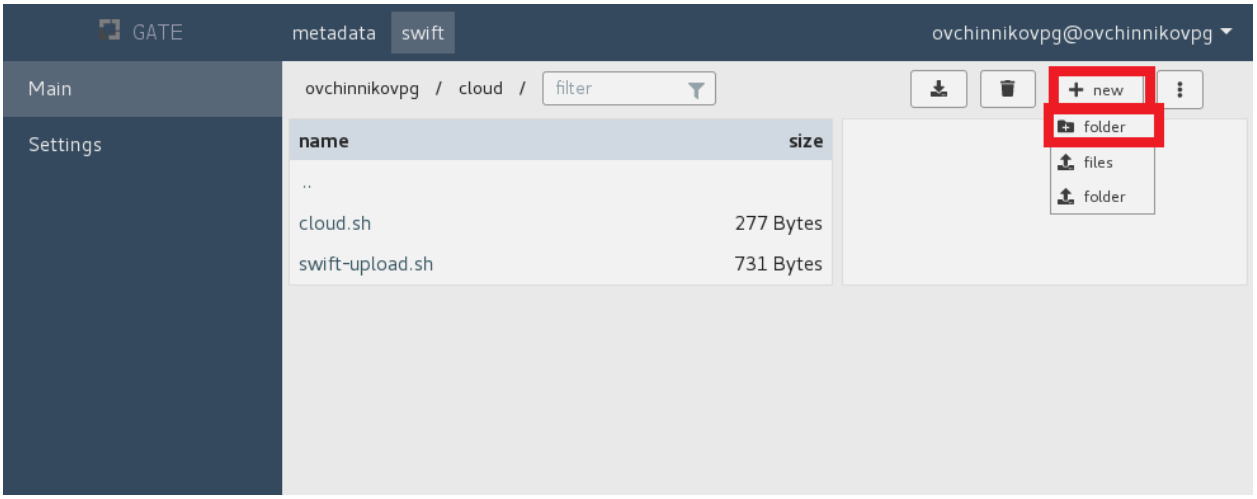

Рис. 27 – Меню создания псевдодиректории

Появится диалоговое окно с требованием ввода имени псевдодиректории (рис. 28). При нажатии на кнопку «ОК» создастся псевдодиректория с указанным именем.

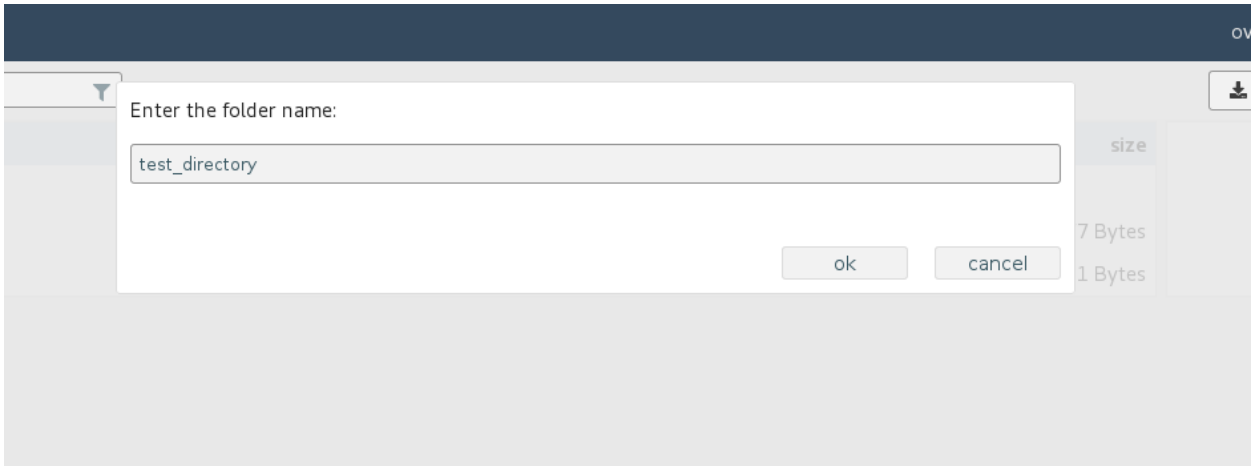

Рис. 28 – Диалоговое окно создания псевдодиректории

Псевдодиректория также создается при загрузке директории в контейнер или псевдодиректорию. Для загрузки директории требуется внутри контейнера или псевдодиректории в выпадающем меню «+ new» выбрать пункт «folder» (рис. 29).

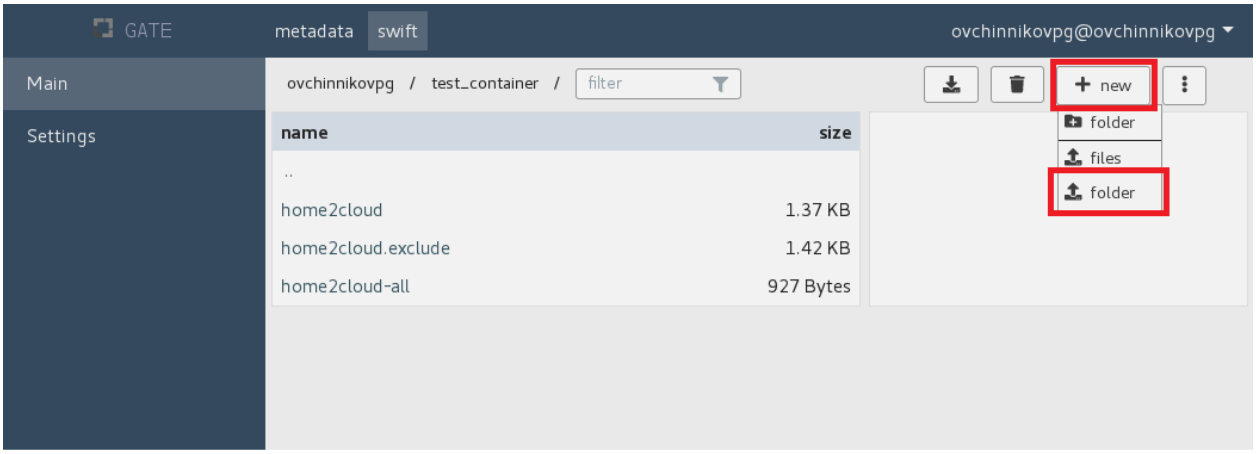

Рис. 29 – Меню загрузки директории

Дальнейшие действия аналогичны загрузке директории в качестве контейнера.

Для загрузки объектов требуется внутри контейнера или псевдодиректории в выпадающем меню «+ new» выбрать пункт «files» (рис. 30).

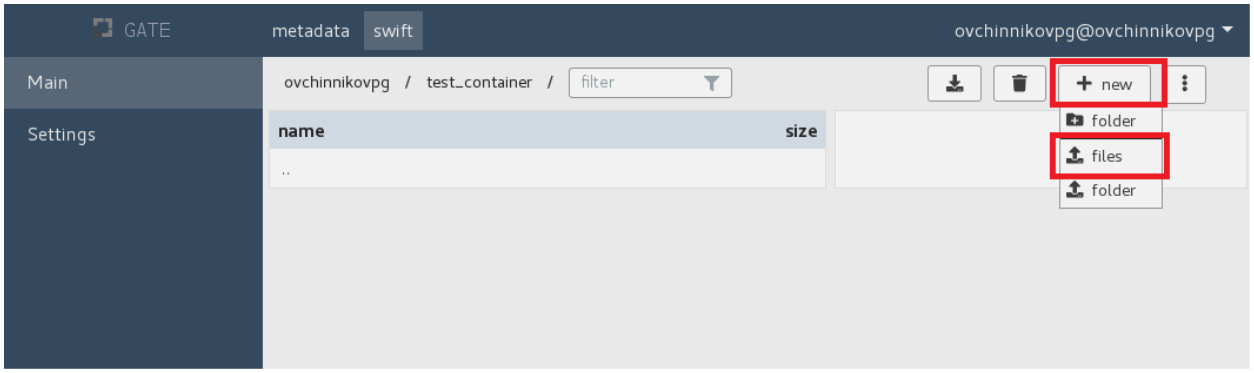

Рис. 30 – Меню загрузки файлов

Появится окно файлового менеджера, в котором требуется выбрать файл (один или несколько) для загрузки. При успешной загрузке объекты отобразятся на текущем уровне псевдоиерархии.

Загрузка больших объектов осуществляется с помощью SLO в автоматическом режиме.

Чтобы скачать контейнеры, псевдодиректории, файлы требуется выделить имена и нажать кнопку скачивания (рис. 31).

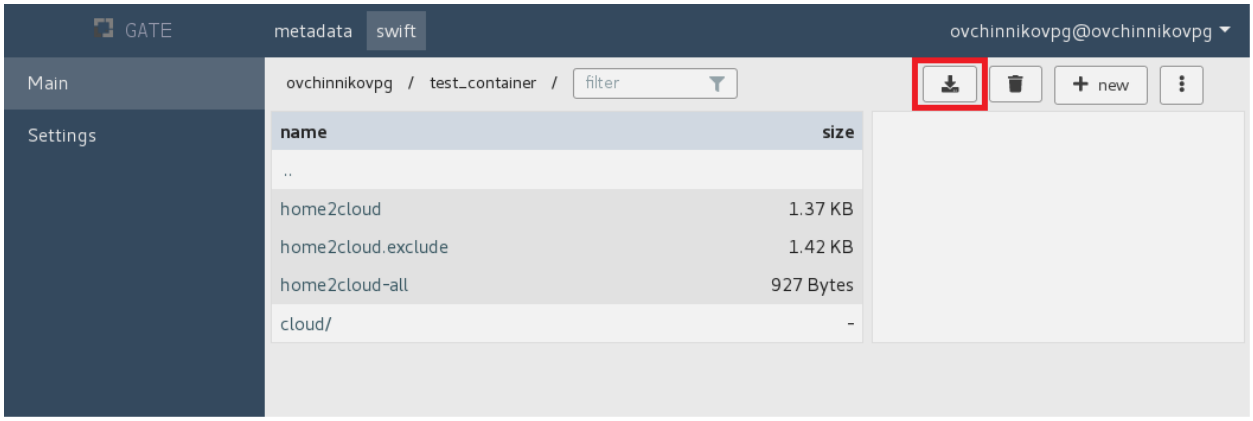

Рис. 31 – Скачивание объектов

Появится стандартное окно файлового менеджера, в котором указывается место назначения для скачиваемых объектов. При скачивании контейнера или псевдодиректории содержимое скачивается в виде zip-архива.

Скачивание больших объектов осуществляется в автоматическом режиме. Скачать можно как SLO, так и DLO.

Для удаления контейнеров, псевдодиректорий или объектов требуется выделить удаляемые имена и нажать кнопку удаления (рис. 32).

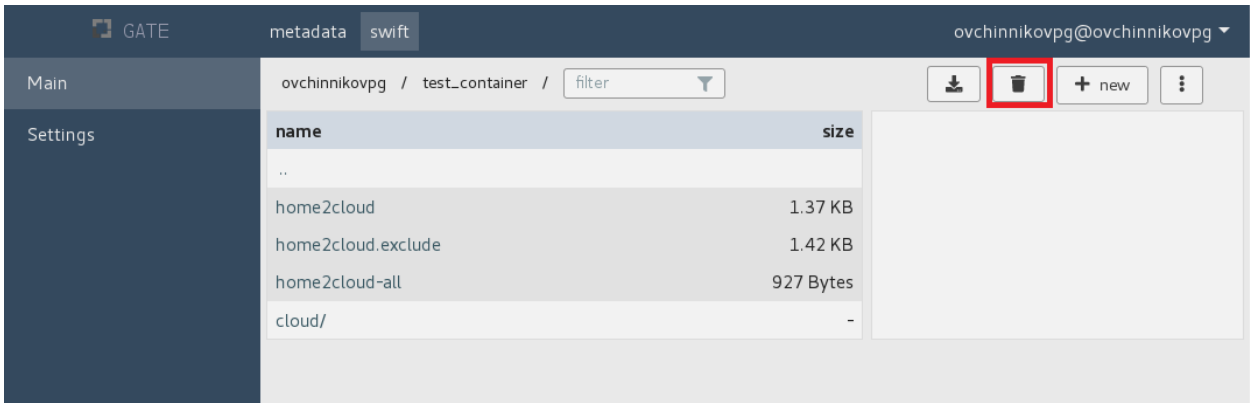

Рис. 32 – Удаление объектов

Появится окно подтверждения удаления. Контейнеры и псевдодиректории удаляются со всем содержимым.

Для получения информации об аккаунте (проекте) требуется перейти во вкладку «Settings» (вертикальное меню слева). Информация о проекте содержит, в частности, описание проекта, участников проекта, использованный объем и его ограничение, количество контейнеров и объектов (рис. 33).

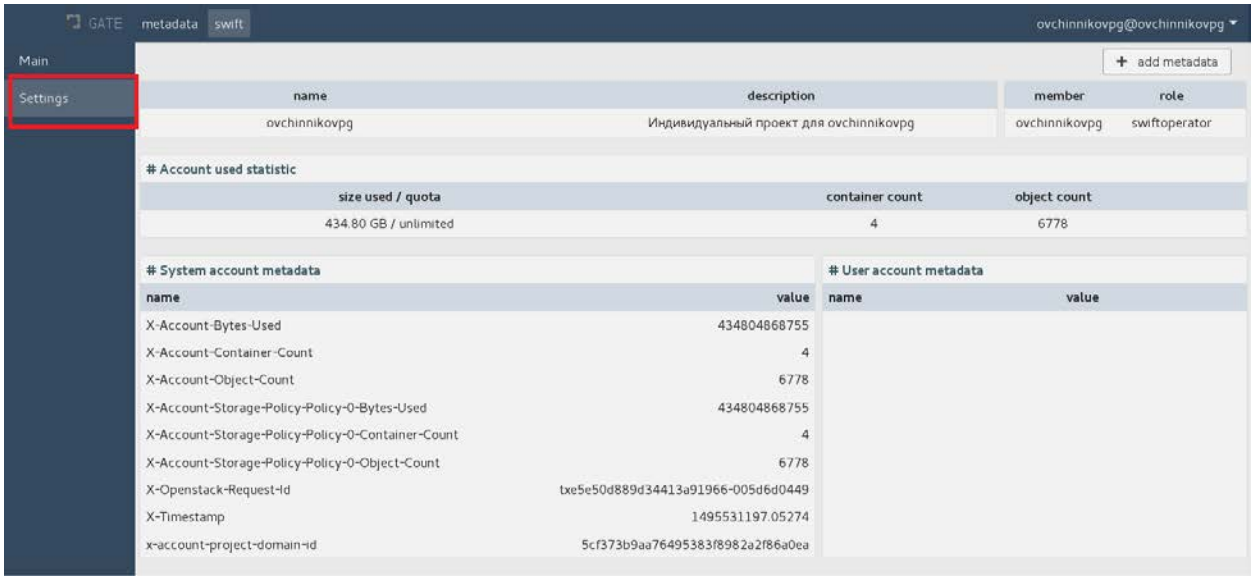

Рис. 33 – Информация о проекте

Для получения информации об учетной записи требуется в выпадающем меню учетной записи выбрать пункт «Settings» (рис. 34).

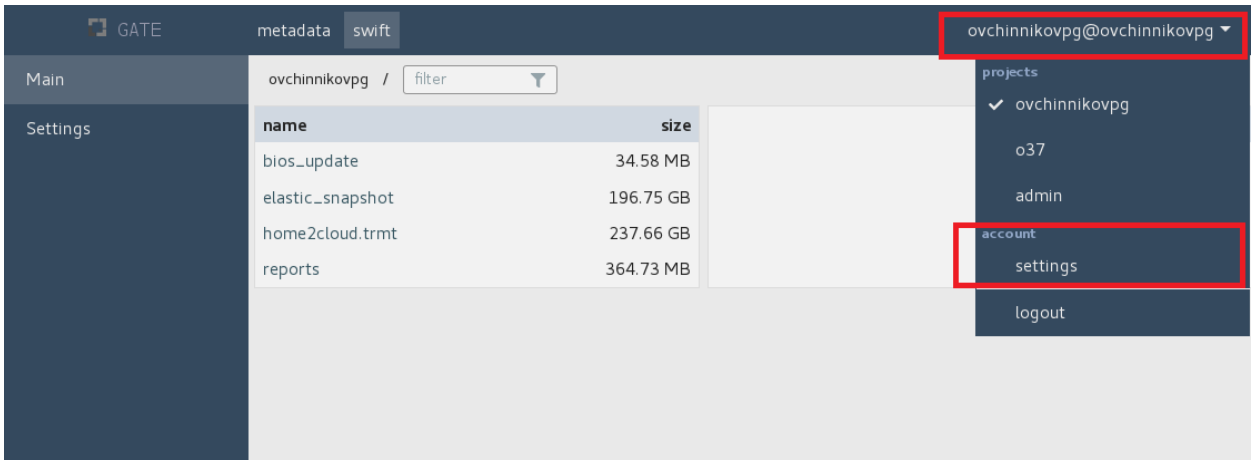

Рис. 34 – Меню учетной записи

В информации об учетной записи содержатся сведения о пользователе, проектах, к которым пользователь имеет доступ, активных токенах для доступа к проектам (с возможностью скопировать полный токен), а также информация об используемых дополнениях web-интерфейса (рис. 35).

Рис. 35 – Информация об учетной записи

5.10.4.8. Утилита синхронизации SynCloud

SynCloud – высокоуровневое межплатформенное клиентское приложение, выполняющее функции синхронизации между определенной директорией на файловой системе и контейнером в ОСХД. SynCloud представляет собой много процессное приложение, написанное на языке python с встроенным web-сервером (используется фреймворк bottle).

5.10.4.8.1. Установка, запуск и завершение работы

SynCloud распространяется в виде архива, в котором содержатся все необходимые модули для работы приложения.

Корректная работа SynCloud обеспечивается в следующих версиях Python (соответственно, любая из этих версий должна быть установлена в операционной системе): 2.6, 2.7, 3.4 и новее.

SynCloud не поддерживает версию Python 3.0.

Для запуска приложения требуется запустить в интерпретаторе Python файл <путь_до_директории_установки>/syncloud.py. При этом запустится приложение с web-сервером, а также откроется web-браузер, используемый в системе по умолчанию, в котором откроется страница с web-интерфейсом приложения. Адрес и порт web-интерфейса сохраняется в файл $HOME/.syncloud/<cloud_username>/web_url.txt (для Windows: %HOMEPATH%\.syncloud\<cloud_username>\web_url.txt).

Все конфигурационные файлы приложения располагаются в директории $HOME/.syncloud (для Windows: %HOMEPATH%\.syncloud). При отсутствии этой директории она создается при запуске приложения.

Для просмотра всех параметров запуска приложения используется опция -h:

```
$ python ./syncloud.py -h
usage: syncloud.py [-h] [--debug] [--bottle_output]
Синхронизатор локальных каталогов с
облачной системой хранения
optional arguments:
  -h, --help       show this help message and exit
  --debug          включить режим отладки
  --bottle_output  включить вывод сообщений
                   web-фреймворка Bottle в stdout и stderr
```

Для завершения работы программы требуется послать сигнал завершения основному процессу. Если программа запущена не в фоновом режиме, то сигнал можно послать нажатием комбинации клавиш <Ctrl>-C.

5.10.4.8.2. Правила синхронизации

Правила синхронизации – это однозначное сопоставление локальной директории и контейнера в ОСХД с набором параметров для осуществления синхронизации между ними. Для правила синхронизации задаются следующие параметры:

- путь до локальной директории;
- имя контейнера в ОСХД;
- направление синхронизации;
- интервал синхронизации;
- режим синхронизации («зеркало», «фильтр»).

5.10.4.8.2.1. Направление синхронизации

В программе SynCloud предусмотрена только односторонняя синхронизация между источником и приемником. Поддерживаются два направления:

1) источник – локальная директория, приемник – контейнер в ОСХД;

2) источник – контейнер в ОСХД, приемник – локальная директория.

5.10.4.8.2.2. Интервал синхронизации

SynCloud может выполнять синхронизацию:

1) однократно (по требованию);

2) регулярно (по расписанию).

Регулярный запуск синхронизации задается в виде временного интервала. Время следующей синхронизации рассчитывается как «время последней синхронизации (в т.ч. по требованию) + интервал синхронизации»;

5.10.4.8.2.3. Режимы синхронизации

Режим синхронизации определяет набор объектов, которые будут получены на приемнике при завершении процесса синхронизации с источником. Режимов два:

1) «зеркало»;

2) «фильтр».

В режиме «зеркало» устанавливается полное соответствие между источником и приемником. Объекты, которые присутствуют на приемнике и отсутствуют на источнике, удаляются с приемника.

В режиме «фильтр» можно ограничить имена объектов, подлежащих синхронизации. Поддерживается как «белый список» («white list» – синхронизации подлежат объекты, чьи имена выполняют условия фильтров), так и «черный список» («black list» - синхронизации подлежат объекты, чьи имена не выполняют условия фильтров). При отсутствии выражений в фильтре синхронизации подлежат все объекты источника. Удаление объектов на приемнике не происходит.

Фильтры поддерживают файловые шаблоны (wildcard).

5.10.4.8.3. Взаимодействие с помощью web-интерфейса

Основным инструментом управления приложением SynCloud является web-интерфейс.

Перед началом работы с приложением необходимо пройти процедуру аутентификации. На начальной странице после запуска приложения будет предложена форма аутентификации, в которую требуется ввести домен, логин и пароль (рис. 36).

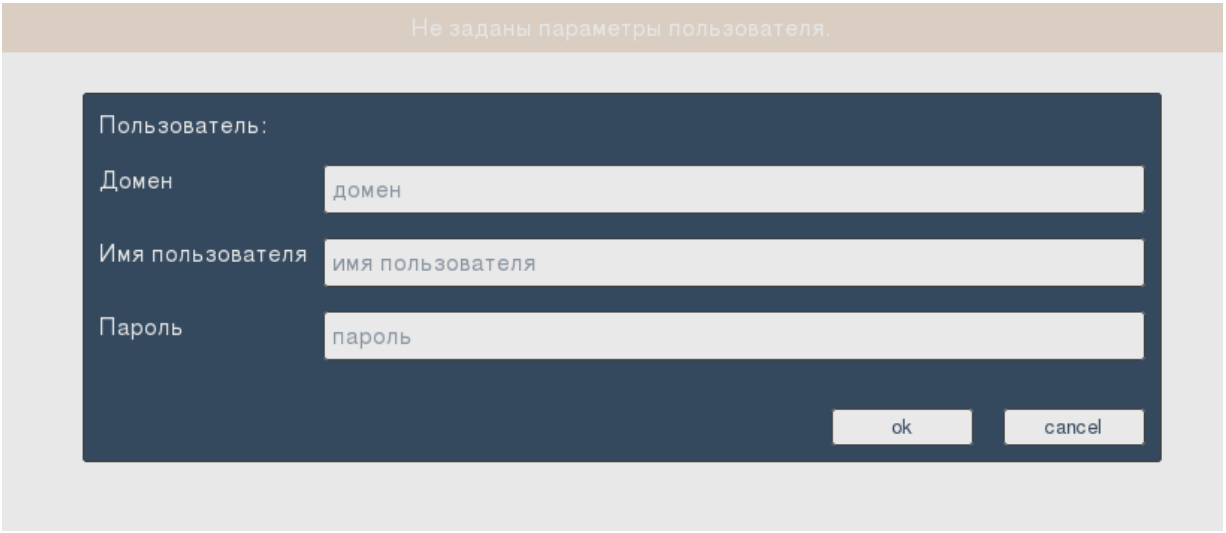

Рис. 36 – Форма аутентификации

При ошибке аутентификации появится предупреждение (рис. 37).

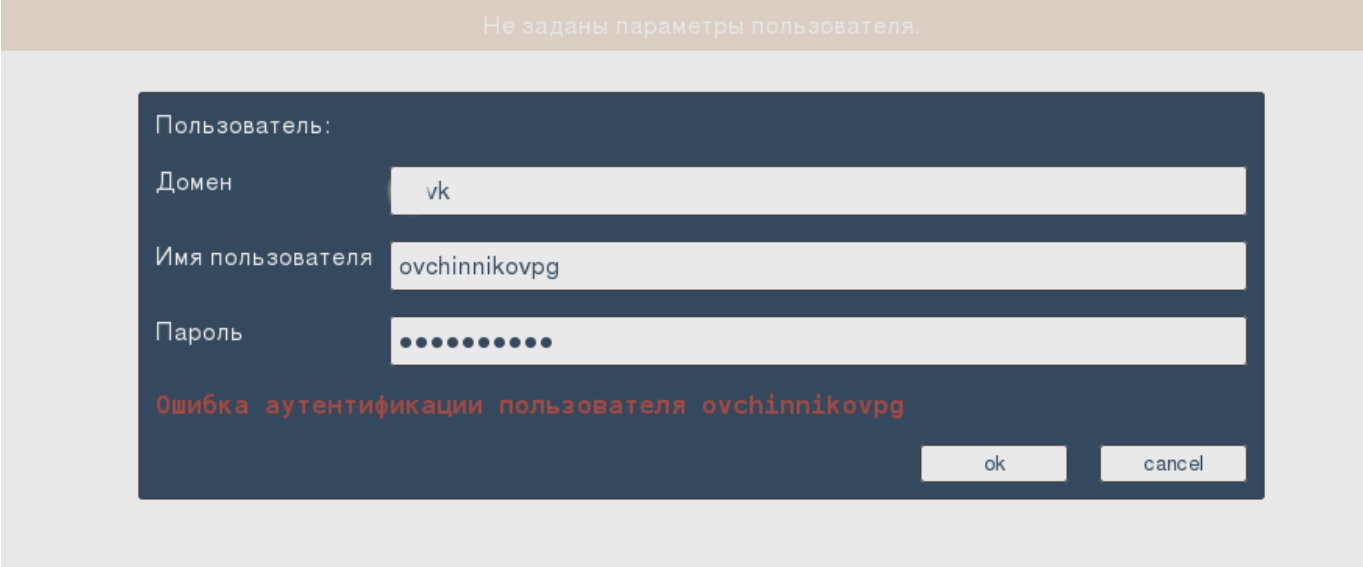

Рис. 37 – Сообщение об ошибке аутентификации

При успешной аутентификации на странице отобразится основной интерфейс управления (рис. 38).

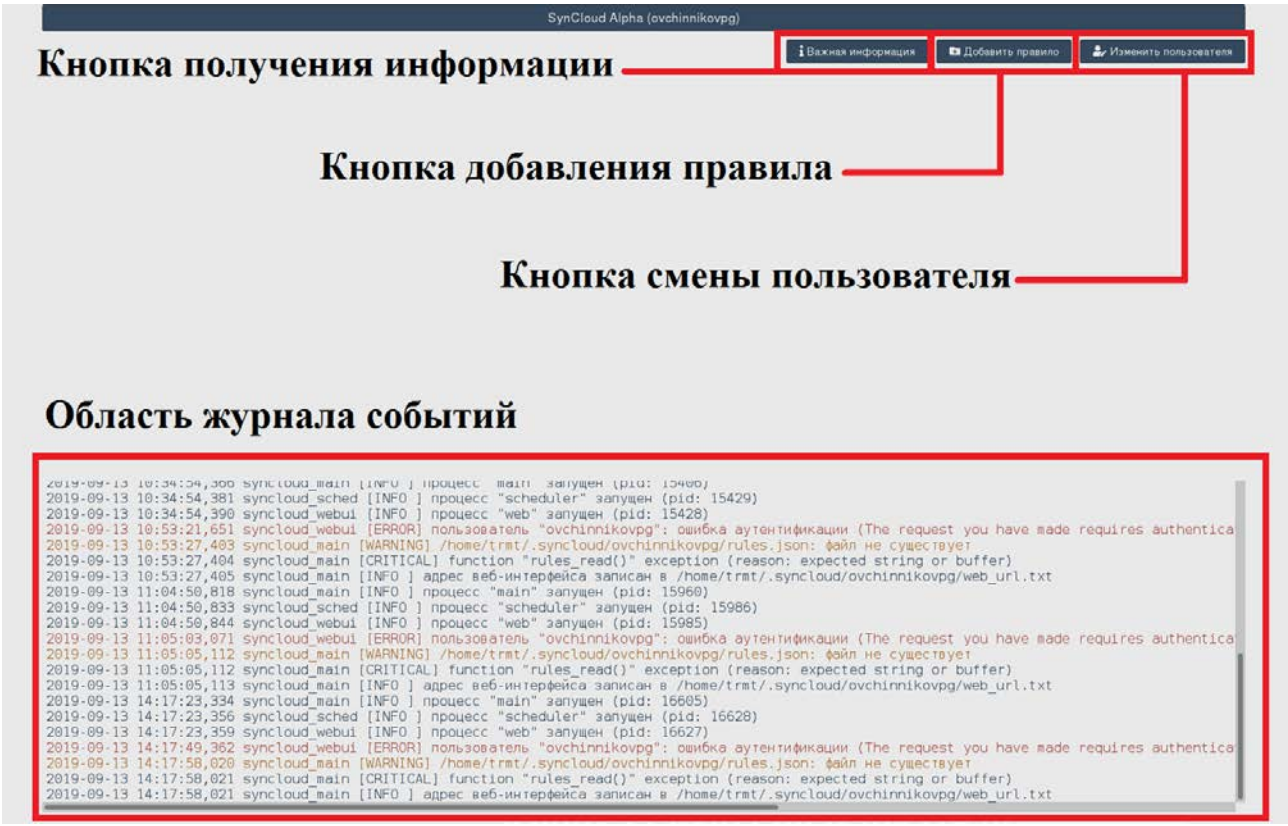

Рис. 38 – Вид основного интерфейса управления

Для смены пользователя требуется нажать кнопку «Изменить пользователя» и в появившемся диалоге аутентификации ввести новые данные.

Нажатие кнопки «Важная информация» вызывает окно (рис. 39), содержащее особенности функционирования программы, на которые стоит обратить внимание.

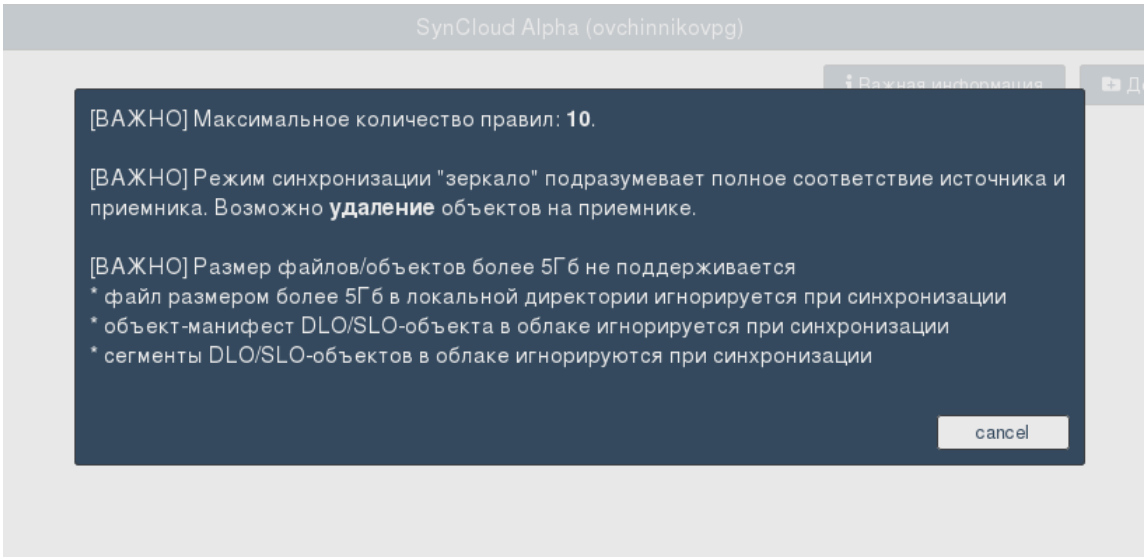

Рис. 39 – Информационное окно

Для создания нового правила синхронизации требуется нажать кнопку «Добавить правило». Появится форма редактирования правила (рис. 40), в котором задаются параметры правила синхронизации.

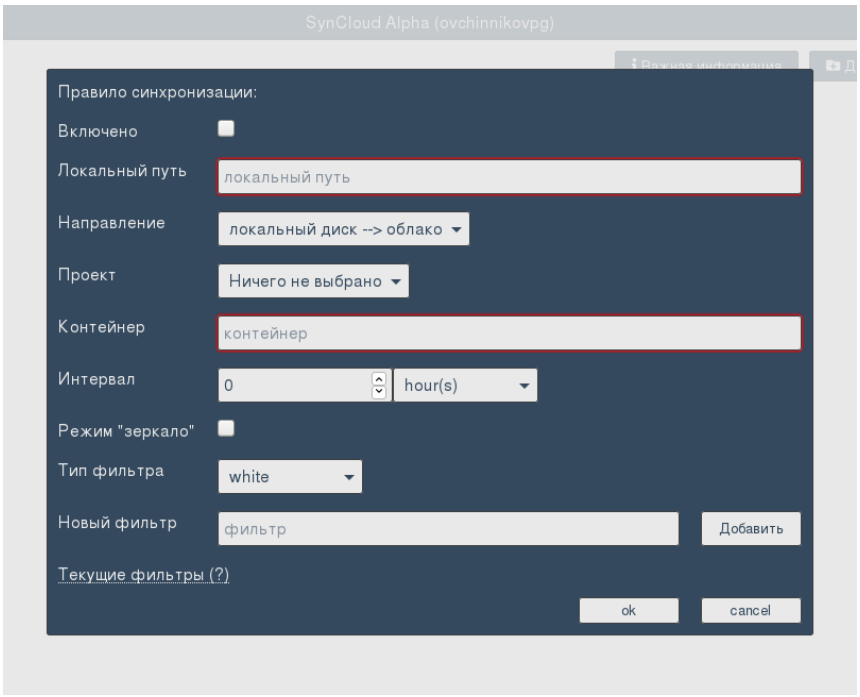

Рис. 40 – Форма редактирования правила по умолчанию

После редактирования всех параметров при нажатии на кнопку «ok» правило добавится в список и отобразится в основном интерфейсе (рис. 41). Сохраненные правила пользователя хранятся в директории $HOME/.syncloud/<cloud_username>/ (для Windows:

%HOMEPATH%\.syncloud\<cloud_username>\) в файле rules.json в формате JSON-структуры.

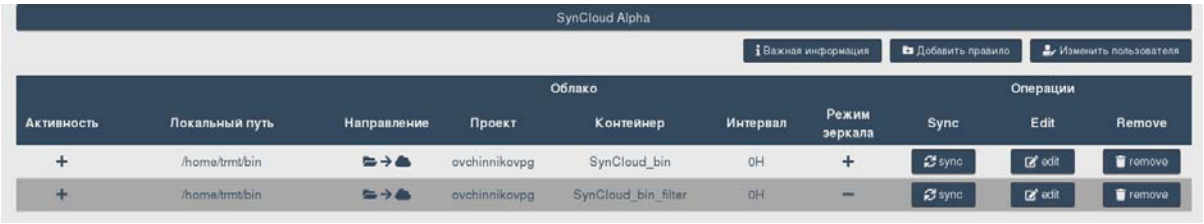

Рис. 41 – Правила синхронизации в основном интерфейсе

Для редактирования правила требуется нажать кнопку «Edit». Для удаления правила – кнопку «Remove».

Для синхронизации по требованию требуется нажать кнопку «Sync». Информация о начале процесса синхронизации отобразится в области журнала событий. Кроме того, кнопка «Sync» станет недоступной, пока процесс синхронизации не завершится (рис. 42).

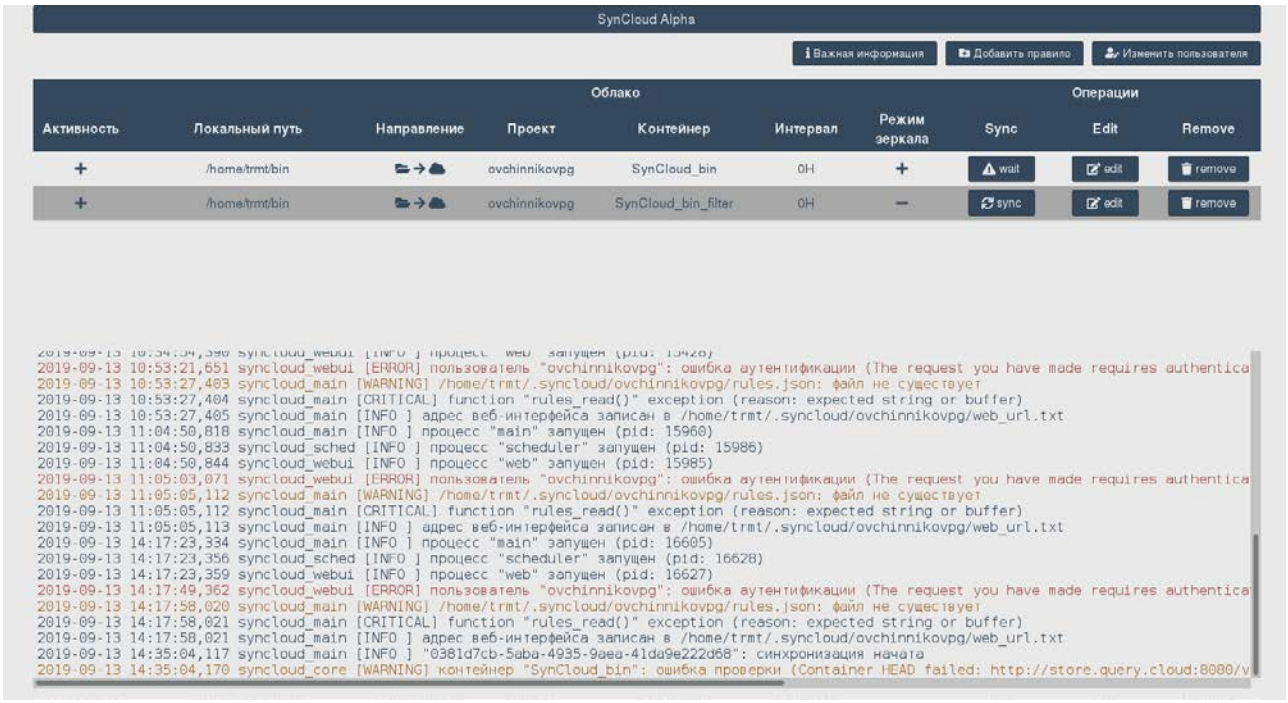

Рис. 42 – Процесс синхронизации правила

5.10.4.8.4. Примеры правил синхронизации

Для примера созданы два правила синхронизации с направлением из локальной файловой системы в ОСХД. Источником для обоих правил служит директория /home/trmt/bin/. Одно правило синхронизации использует режим «зеркало», второе – режим «фильтр» с указанием белого списка имен.

Структура директории-источника выглядит следующим образом:

```
$ tree
```

```
.
|-- cloud
|   |-- cloud.sh
|   `-- swift-upload.sh
|-- cp
|-- home2cloud
|   |-- home2cloud
|   |-- home2cloud-all
|   `-- home2cloud.exclude
|-- inotify_log.sh
|-- inotify_strace.sh
|-- iozf.sh
|-- lnav
|-- manydirs.sh
|-- mkdirmany775
|-- periodic.sh
|-- qmmp_kill.sh
|-- random_file.sh
|-- remat.sh
|-- ssh_tunnel.sh
|-- tar_multivol_mc.sh
`-- tar_namer.sh

2 directories, 19 files
```

Пример правила синхронизации в режиме зеркало приведен на рис. 43.

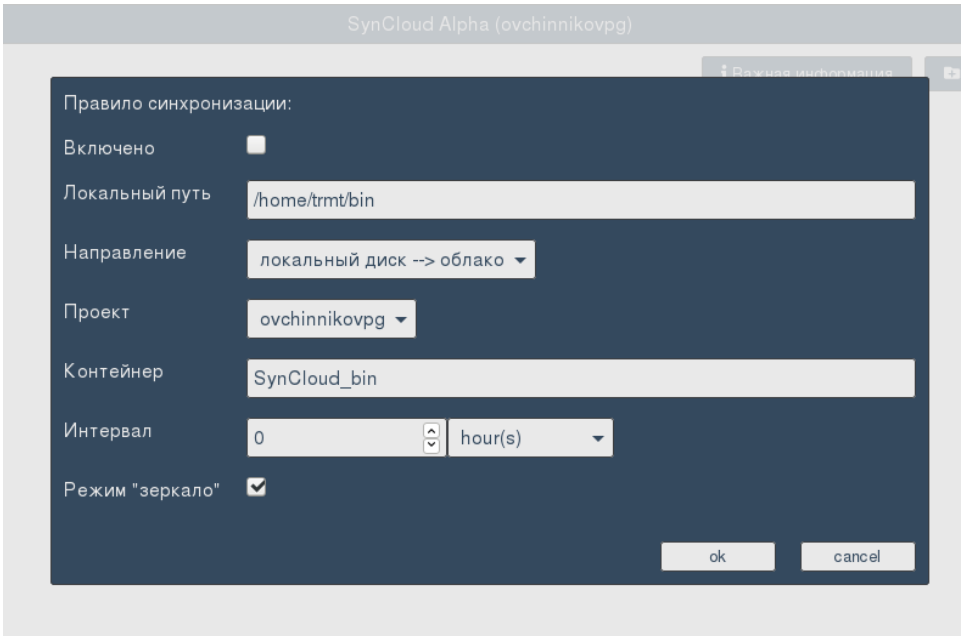

Рис. 43 – Правило синхронизации в режиме зеркало

Результат синхронизации директории в режиме зеркало приведен на рис. 44. В соответствии с существующей иерархией в источнике была сформирована псевдоиерархия в контейнере-приемнике. Все файлы сохранены в виде объектов внутри контейнера с сохранением относительных путей.

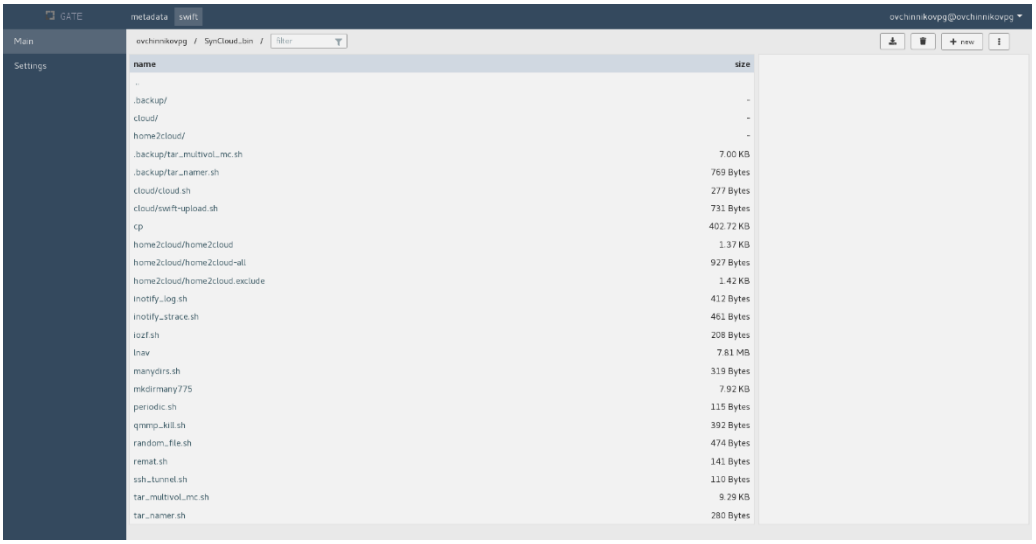

Рис. 44 – Результат синхронизации в режиме «зеркало»

Пример правила синхронизации с использованием фильтра приведен на рис. 45. Правилом ограничиваются имена файлов, подлежащих синхронизации: синхронизируются только файлы, имена которых совпадают с выражениями «inotify_*» и «home2cloud/*».

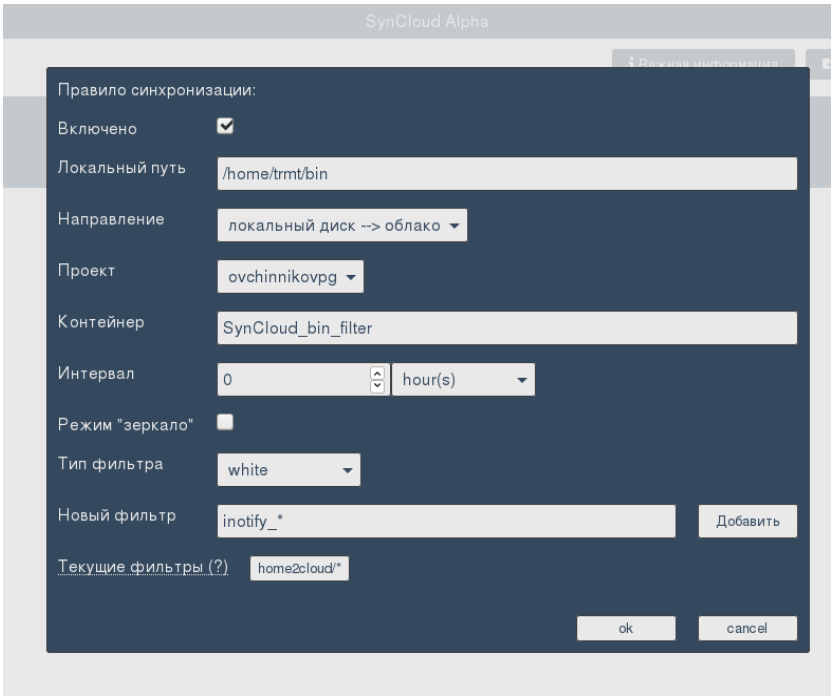

Рис. 45 – Пример правила синхронизации с использованием фильтра

Результат синхронизации приведен на рис. 46. Для файлов, имена которых подошли под фильтр, была сформирована псевдоиерархия в контейнере-приемнике, соответствующая иерархии в источнике. Файлы, имена которых не удовлетворили условиям фильтров, синхронизированы не были.

Рис.46 – Результат синхронизации в режиме «фильтр»

5.10.5. Subversion

Subversion (SVN) [25]– это сервис управления версиями проектов, который помогает пользователям (в т.ч. группам пользователей) разрабатывать программные и иные виды проектов. SVN позволяет управлять файлами и каталогами, а также сделанными в них изменениями.

Перед началом работы с сервисом пользователю необходимо установить на АРМ программный пакет Subversion. Также на сайте сервиса потребуется завести проект.

После этого с помощью набора команд (начинающихся с «svn») пользователь может загружать в хранилище и получать из него файлы и каталоги любой версии своего проекта.

Далее указан простейший набор команд для работы с сервисом SVN.

Инициализация проекта в хранилище, создание его первой копии:

```
$ svn import ProjectName http://svn/repos/ProjectName -m «init Project»
```

Создание рабочей копии на АРМ для разработки проекта:

```
$ svn checkout http://svn/repos/ProjectName
```

Получение обновлений проекта из хранилища:

```
$ svn update
```

Фиксация изменений в хранилище:

```
$ svn commit -m «new in Project»
```

Добавление файла или каталога под управление svn:

```
$ svn add file_or_dir_name
```

Удаление файла или каталога:

```
$ svn delete file_or_dir_name
```

Копирование файла или каталога:

```
$ svn copy foo bar
```

Перемещение файла или каталога:

```
$ svn move foo bar
```

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

GDB	—	Свободно распространяемый отладчик программ
GNU	—	Название сообщества разработчиков свободно распространяемого ПО
IB	—	InfiniBand
IO	—	Input-Output
MIMD	—	Multiple Instruction Multiple Data
MPE	—	Multi-Processing Environment
MPI	—	Messages Passing Interface
MPMD	—	Multiple Program Multiple Data
MPPs	—	Massively Parallel Processing
NFS	—	Network File System
OpenMP	—	Стандарт библиотеки передачи сообщений для использования в параллельных программах на многопроцессорных ЭВМ с общей памятью
SMP	—	Symmetric Multiprocessors
TFLOPS	—	10**(12) FLOPS
АСХД	—	Архивная система хранения данных
АУ	—	Административный узел
ВУ	—	Вычислительный узел
ЕПП	—	Единое пространство пользователей
ИУ	—	Инструментальный узел
ОС	—	Операционная система
ОСХД	—	Облачная система хранения СТРИЖ
ПО	—	Программное обеспечение
СПО	—	Система пакетной обработки
СУ	—	Сервисный узел
СУБД	—	Система управления базами данных
СУЗ	—	Система управления заданиями
СХ	—	Система хранения
ФС	—	Файловые сервера

ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1) Scintific Linux. http://scientificlinux.org/.

2) SSH.COM. http://www. ssh.com/.

3) Графический интерфейс в Linux. http://heap.altlinux.org/alt-docs/text books/linux intro/x11.html.

4) OpenLDAP.Community developedLDAP software. http://openldap.org/.

5) RFC 959 – File Transfer Protocol. http://rfc-editor.org/in-notes/rfc959.txt.

6) All about CIFS. http://cifs.com/.

7) The Postfix Home Page. http://www.postfix.org/.

8) DoveCot. http://www.dovecot.org/.

9) Jabberd. http://jabberd2.org/.

10) GNU Emacs. http://gnu.org/software/emacs/.

11) GCC, the GNU Compiler Collection. http://gcc.gnu.org/.

12) GNU Operating System. GNU Make. http://www.gnu.org/software/make.

13) GDB. The GNU Project Debugger. http://www.gnu.org/software/gdb/.

14) Message Passing Interface Forum, «MPI: A Message Passing Interface». // In Proceedings of Supercomputing’93. IEEE Computer Society Press, November 1993, pp. 878–883.

15) MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP and RoCE. http://mvapich.cse.ohio-state.edu/.

16) OpenMPI: Open Source High Performance Computing. http://www.open-mpi.org/.

17) SLURM workload manager. http://slurm.schedmd.com/.

18) Документация по HTTP серверу Apache версии 2.2. http://httpd.apache.org/docs/2.2/index.html.

19) PostgreSQL: The World’s Most Advanced Open Source Relational Database. http://www.postgresql.org/.

20) MariaDB Server: The open source relational database. http://mariadb.org/.

21) SQLite Home Page. http://www.sqlite.org/index.html.

22) Lustre. http://www.lustre.org.

23) Openstack. Welcome to Swift’s documentation. http://docs.openstarck.org/swift/latest/.

24) Openstack. Kestone, the OpenStack Identity Service. http://docs.openstarck.org/keystone/latest/.

25) Управление версиями в Subversion (для Subversion 1.4). http://svnbook.red-bean.com/index.ru.html.

Лист регистрации изменений

Номера листов (страниц)					Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подп.	Дата
Изм.	изменен ных	замененных	новых	анулированных					