Система управления заданиями Cleo

Руководство администратора

2007

1. ОБЩИЕ ПОНЯТИЯ И ТРЕБОВАНИЯ

1.1. Введение.

Система управления заданиями Cleo предназначена для управления запуском задач на многопроцессорных вычислительных установках (в том числе кластерных). Она позволяет автоматически распределять вычислительные ресурсы между задачами, управлять порядком их запуска, временем работы, получать информацию о состоянии очередей. При невозможности запуска задач немедленно, они ставятся в очередь и ожидают, пока не освободятся нужные ресурсы.

Система позволяет работать с различными типами заданий, в том числе последовательными и написанными с использованием различных версий MPI — MPICH-p4, MPICH-gm, Lam, ScaMPI, MVAPICH и другими.

В качестве вычислительной единицы используется процессор, но система оперирует и понятием вычислительного узла, который может содержать несколько процессоров. На данный момент система не различает типов процессоров (то есть в рамках одной очереди все процессоры считаются равноправными).

Система написана на языке perl5, что позволяет использовать её на различных платформах.

1.2. Системные требования

Для успешной работы Cleo необходимо наличие работоспособного интерпретатора perl версии не ниже 5.6, а также модулей IO::Socket, IO::Pipe, IO::Handle, IO::Select, IO::File, Storable, Sys::Syslog, Time::Local, POSIX, и Fcntl, соответствующих данной версии perl или более поздних версий. Также необходимо наличие сформированных системных файлов заголовков ph. Для этого от имени суперпользователя должна быть исполнена команда:

```
cd /usr/include; h2ph -r -l .
```

Для работы механизма авторизации (см. п. 2.7), требуется совместимость файловой системы /proc с Linux 2.2 и выше (точнее формата файлов /proc/PID/status и /proc/PID/cmdline).

Если предполагается запуск задач от имени различных пользователей, то запуск сервера системы должен производится с правами суперпользователя.

2. СТРУКТУРА СИСТЕМЫ

2.1. Общая структура системы

Система Cleo состоит из сервера, запускаемого с правами суперпользователя, и осуществляющего собственно управление заданиями, программы-монитора, запускаемой на всех рабочих узлах, и набора клиентских программ, осуществляющих непосредственное взаимодействие с сервером, используя протокол TCP/IP, либо через файлы состояния сервера (в последнем случае возможно лишь получение информации от сервера).

Сервер порождает набор процессов в соответствии с иерархией очередей (см. п. 2.2) и в дальнейшем каждый из этих процессов контролирует свою очередь. Головной процесс контролирует головную очередь (по умолчанию — 'main') и отвечает за взаимодействие с клиентскими программами.

В качестве клиентских программ в поставку входит, так называемый, основной клиент (программа cleo-client), который поддерживает все команды сервера по протоколу TCP/IP. С его использованием написаны скриптовые программы, осуществляющие простые операции с сервером, такие как постановка задания в очередь, снятие задания, и т.п.

Возможно написание произвольных клиентов, использующих протокол взаимодействия сервера и клиентских программ

2.2. Иерархия очередей

Система очередей способна поддерживать несколько очередей одновременно. Все они организуются в иерархию, где одни очереди включают в себя другие. Это значит, что любой процессор может использоваться одновременно несколькими очередями. Приведем небольшой пример:

<table>
<tr><td colspan="8">main</td></tr>
<tr><td colspan="4">long</td><td colspan="4">short</td></tr>
<tr><td>p1:1</td><td>p1:2</td><td>p2:1</td><td>p2:2</td><td>p3:1</td><td>p3:2</td><td>p4:1</td><td>p4:2</td></tr>
</table>

На этой картинке представлена иерархия из трех очередей над 4 вычислительными узлами, содержащими по 2 процессора. Узлы имеют имена p1, p2, p3 и p4, а их процессоры соответственно p1:1, p1:2 для узла p1 и т.д.

Очередь main включает в себя все процессоры. Такая объемлющая очередь **должна существовать всегда**, даже если ей никогда не будут пользоваться (например, если она объединяет несколько несвязанных кластеров, управление которыми осуществляется независимо). В последнем случае можно просто запретить доступ в объемлющую очередь для всех.

Очередь main в данном примере имеет две дочерних очереди — long, включающую в себя процессоры p1:1, p1:2, p2:1 и p2:2, и очередь short, включающую в себя процессоры p3:1, p3:2, p4:1 и p4:2.

Система следит за тем, чтобы каждый процессор использовался одновременно только одной задачей (если явно не оговорено обратное). Это

достигается за счет того, что задачи очередей «верхнего» уровня автоматически попадают и в дочерние очереди. Таким образом, когда задача фактически идет на счет, она присутствует и помечается как считающаяся во всех очередях, чьи процессоры она занимает.

Например, поставим в очередь short задачу на 4 процессора и затем в очередь main — задачу на 6 процессоров. Последняя задача автоматически попадет в очереди short и long. В очереди long она будет помечена как пред-запущенная, то есть для нее будут зарезервированы процессоры, но на счет она пущена не будет. В очереди short будет считаться первая задача, а вторая будет ждать окончания ее счета.

До постановки задач

Main	
long	**short**

После постановки задач

main	
задача2	
long	**short**
задача2	*задача1*
	задача2

Как только первая задача досчитается, в очереди short будет пред-запущена вторая задача. Очередь main (которой и принадлежит вторая задача) получит 8 процессоров для ее запуска. Так как заказано для нее было только 6 процессоров, то ровно 6 процессоров будет отобрано для счета. Остальные 2 процессора будут сняты с резервирования и могут быть использованы для запуска новых задач.

Каждая из очередей может быть настроена независимо. То есть в различных очередях могут быть заданы различные ограничения.

В качестве ограничений могут быть заданы количество процессоров на одну задачу, количество одновременно занятых процессоров (если одновременно запускается несколько задач), максимальное время счета задачи, максимальный приоритет задачи. Подробнее об этом см. главу «Настройка сервера через файл конфигурации и ключи».

Ограничения могут также задаваться и для отдельных пользователей, при этом они могут быть строже или мягче, чем для очереди в целом.

2.3. Приоритеты

В очереди задачи сортируются по **приоритету**. Приоритет — это целое число в диапазоне от 0 до 256. Приоритет по умолчанию задается в настройках очереди. Если он не задан, то он принимается равным 10. Задачи с более высоким приоритетом всегда ставятся выше задач с более низким приоритетом. Это значит, что задачи с более высоким приоритетом будут запущены раньше, чем остальные, даже если они были поставлены в очередь позже по времени. Приоритет можно повышать и понижать после постановки задачи в очередь. Пользователь может понизить приоритет своей задачи, а

также повысить его до величины, не превышающей установленного для него лимита.

2.4. Ограничение времени счета

Для задач можно задать ограничение времени счета. Обычно он задан по умолчанию, но можно его понизить, если это необходимо. По истечении указанного лимита, если задача еще считается, она будет принудительно снята со счета.

Планировщик, установленный по умолчанию, ориентируется на то, что задача будет считаться не более указанного лимита времени, и учитывает это при планировании пуска задач.

Приведем пример: пусть в очереди считается задача, занимающая 6 процессоров из 8 доступных, и пусть ее лимит времени не позволит ей считаться еще более чем 3 часа. За ней стоит задача, требующая 8 процессоров, и лимит времени которой составляет 10 часов. Поставим в очередь задачу на 2 процессора с лимитом 1 час:

очередь	кол-во проц.	лимит времени	статус
задача1	6	3 часа	счет
задача2	8	10 часов	ждет
задача3	2	1 час	ждет

Задача 2 не может быть пущена на счет, пока не досчитается задача 1. Однако в худшем случае это произойдет через 3 часа. Это заведомо больше, чем время счета задачи 3, для которой есть достаточное количество свободных процессоров. То есть, пустив на счет задачу 3, мы не увеличим худшее время ожидания для задачи 2. Поэтому в данной ситуации задача 3 будет пущена системой на счет в обход задачи 2.

Таким образом, может быть целесообразным задавать ограничение времени счета меньше принятого по умолчанию, но достаточным для нормального счета задачи (если это возможно оценить) с целью ускорения пуска задачи на счет.

2.5. Стратегии выбора процессоров

При старте задачи для нее выделяются процессоры. В системе есть возможность управлять стратегией выбора процессоров для задач. Есть три встроенные стратегии, перечисленные в нижеследующей таблице:

random	Процессоры выбираются случайным образом
`random_hosts`	Процессоры выбираются на случайно выбранных узлах, предпочтительно занимая все доступные процессоры на узле

`hosts_alone`	Процессоры выбираются на случайно выбранных узлах, предпочтительно занимая лишь по одному процессору на узле

Элемент случайности в выборе процессоров присутствует для достижения двух целей:

1 Б*о*льшая сбалансированность нагрузки на процессоры.
2 Предотвращение блокировок, связанных со специальным заказом конфигурации процессоров. Например, если мы выберем стратегию выбора последовательными непрерывными блоками, то вскоре может сложиться ситуация, когда доступны два слишком коротких блока, но общий объем их достаточен для запуска. Жестко удовлетворить условие выбора невозможно, и придется блокировать задачу, а с ней и остальную очередь (иначе условие может не выполняться еще очень долго).

Как видно из описания стратегий `random_hosts` и `hosts_alone`, в них заданы предпочтения для выбора процессоров. Однако, если эти предпочтения не могут быть удовлетворены, они игнорируются.

В системе также есть возможность подключения внешних модулей для задания своих стратегий.

В случае использования внешнего модуля, он должен быть описан в разделе `pe_sel_method` секции `[server]`, где также будет описано его имя. Для подключения модуля в систему достаточно просто указать его имя вместо имени встроенного метода.

Все стратегии распределения процессоров опираются на умолчание описания имен процессоров — `<имя_узла>:<идентификатор_процессора>`. Данное соглашение используется и в других целях (см. описание псевдопеременных, формирование файла конфигурации для задачи).

Технические детали задания стратегии распределения процессоров описаны в главах 10 и 11.

2.6. Политики пользователей

Для всех очередей и для каждой очереди в отдельности можно задать список пользователей, которые имеют право ставить на счет свои задачи. Все остальные пользователи не будут иметь доступа к соответствующим очередям.

Можно также просто запретить доступ к очередям списку пользователей. Тогда доступ для всех остальных будет разрешен.

Права доступа не наследуются очередями-потомками. То есть, в контексте нашего примера, если пользователю user1 разрешен доступ в очередь main, то ему не обязательно разрешен доступ в очередь short или long.

Для всех очередей и для каждой очереди в отдельности можно задать список пользователей-администраторов, которые получают возможность

произвольно задавать лимит времени работы задач, приоритеты, а также удалять любые задачи (в рамках тех очередей, для которых им даны права администраторов).

2.7. Блокировки

Системой Cleo поддерживаются следующие виды блокировок:

- блокировка очереди на запуск,
- блокировка очереди на постановку задач,
- блокировка процессора,
- отложенная блокировка процессора,
- блокировка задач,
- автоблокировка пользователей,
- автоблокировка по времени,
- автоблокировка по ограничению ресурсов.

Блокировка очереди на запуск задач означает, что задачи, стоящие в очереди не будут запущены на счёт, пока блокировка не снята. При этом уже запущенные задачи не снимаются со счёта.

Любая очередь может быть в любой момент времени заблокирована ***на добавление новых задач***. При этом на работающих и уже стоящих в очереди задачах это никак не отражается.

Указанные блокировки могут быть установлены рекурсивно, то есть не только на конкретную очередь, но и на все ее дочерние очереди. В любой момент времени эти блокировки могут быть сняты, как для отдельной очереди, так и рекурсивно.

Блокировкой процессора можно запретить исполнение задач на конкретном процессоре или узле, что может оказаться полезным для профилактики отдельных узлов, например. ***Отложенная блокировка процессора*** срабатывает только после того, как процессор заканчивает выполнение пользовательской задачи.

Отдельная задача или все задачи пользователя также могут быть заблокированы.

Кроме этих средств, существуют возможности автоматической блокировки — например блокирование новых задач указанного пользователя или всех пользователей, кроме указанных. Возможно также автоматически блокировать все задачи, которые могут не завершиться к указанному времени (это удобно, если нужно освободить процессоры для выполнения запланированных работ).

Последний приведённый вид автоблокировки — блокировка по ресурсам. Он вступает в силу, если запуск задачи пользователя нарушает наложенное ограничение по использованию ресурсов (например число одновременно занимаемых процессоров). Как только условие удовлетворяется, блокировка снимается.

2.8. Варианты запуска задач

Для запуска задач в системе Cleo предусмотрены 4 варианта:

- с внешним контролем задач,
- запуск без внешнего контроля,
- с использованием своей команды rsh на узлах (не рекомендуется).

В первом варианте запуск происходит «традиционным» способом, то есть так, как предполагает параллельная среда. Однако, все процессы задачи на узлах «берутся под контроль» мониторами системы. В случае аварийного завершения задачи или ее удаления процессы на узлах будут принудительно завершены. Этот вариант является рекомендуемым.

Второй вариант предполагает такой же способ запуска, но при этом контроль задач отключается. Данный режим не рекомендуется, так как в случае сбойного поведения задачи возможны нежелательные эффекты в виде работающих процессов завершённой задачи.

В третьем случае после запуска мониторы на узлах и сервер очередей на головной машине соответственно отслеживают вызовы команды `rsh`, сделанные задачей (такие вызовы делает, к примеру, `mpich`). Эти вызовы обрабатываются и передаются в виде запроса головному серверу. Головной сервер, в свою очередь, инициирует запуск требуемых процессов, запрашивая мониторы на узлах.

Все запущенные по такой схеме процессы задач на узлах контролируются мониторами, что обеспечивает гарантированное завершение задач.

Эмуляция rsh в данном случае ограниченная, поэтому использовать это режим не рекомендуется.

Следует отметить, что варианты запуска могут быть индивидуальны по отношению к каждой задаче. То есть в рамках одной и той же очереди возможны разные варианта запуска. Выбор варианта осуществляется проверкой параметров `run_via_mons` и `attach_mask` при запуске задачи (см. п.13).

2.9. Внутренняя архитектура

Система очередей представляет собой приложение клиент-сервер. Серверная программа называется `cleo`. После старта она сама переходит в фоновый режим и создает дерево процессов, соответствующее дереву очередей, заданному в файле конфигурации.

Перед запуском сервера необходимо запустить на узлах процессы-мониторы `cleo-mon`. Если процесс-монитор не запущен на узле, то система считает этот узел сбойным и исключает его из рабочего поля. В дальнейшем головной процесс периодически опрашивает сбойные узлы, и если ему удаётся

соединиться с процессом-монитором, то узел помечается как работоспособный и возвращается в рабочее поле (если он не был заблокирован иначе).

Взаимодействие с сервером производится по протоколу TCP/IP через порт, указанный в конфигурации. Ниже описан основной клиент, реализующий взаимодействие с сервером.

На данный момент команды могут задаваться только с той машины, на которой запущен сервер, так как пока реализована только локальная авторизация пользователя системы. Однако, в случае необходимости, авторизация может быть отключена и все команды в этом случае могут быть выполнены удаленно. Следует отметить, что авторизация должна быть отключена как на сервере, так и в клиентских программах. Так как при отключенной авторизации невозможно проверить реальные полномочия пользователя, отключать ее не рекомендуется.

Параметры всей системы, отдельных очередей и права пользователей задаются в отдельном файле конфигурации. В любой момент конфигурация может быть перечитана из него. Однако, не все параметры могут быть изменены на ходу. Об этом см. в главе «Настройка через файл конфигурации».

Система рассчитана на многопользовательскую среду и способна обслуживать нескольких пользователей одновременно.

2.10.Поддержка параллельных сред

Система способна поддерживать несколько параллельных сред одновременно. Это значит, что отдельная очередь может быть настроена на запуск параллельных приложений одного типа (например, MPICH), а другая — другого типа (например, MVapich). Возможно также изменение параллельной среды для отдельного пользователя.

Поддержка различных параллельных сред осуществляется за счет настройки следующих параметров:

- командная строка запуска задачи
- опции формирования файла настройки задачи
- схема запуска (прямая, через мониторы на узлах, с эмуляцией rsh)

Более подробно эти вопросы освещены в главе «Примеры конфигураций».

2.11.Дополнительные возможности

Кроме возможности взаимодействия клиент-сервер, есть возможность получения данных о состоянии системы из специальных файлов — файла в котором хранится PID головного процесса (может быть использован для перезапуска системы), файлов статуса очередей, полного и сокращенного лог-файлов, а также файлов сохранения содержимого очереди. Формат этих файлов будет кратко описан ниже.

Система сделана расширяемой за счёт модулей. В данный момент реализованы следующие типы модулей:

- модули стратегий распределения процессоров,

- ExecModules — модули, вызываемые при запуске или завершении задачи,
- планировщики задач.

2.12.Запуск и останов сервера, перечитывание конфигурации

Запуск, останов и рестарт сервера очередей осуществляется с помощью скрипта `cleo.rc`, расположенного в каталоге `/etc/rc.d/init.d` или `/etc/init.d`.

Для вышеозначенных действий этот скрипт необходимо выполнить с ключами `start`, `stop` или `restart` соответственно.

Следует помнить, что изменение количества или имён процессоров в какой-либо очереди, а также имён, количества и иерархии очередей не может быть сделано без стирания сохранённого состояния системы. В противном случае возможны непредсказуемые эффекты.

2.13.Порядок обработки запросов сервером

Для выполнения любого действия с системой очередей, пользователь формирует к ней запрос, используя доступные утилиты. Далее, упомянутые утилиты взаимодействуют с сервером и возвращают пользователю результат обработки запроса.

При этом может использоваться механизм умолчаний, избавляющий пользователя от ввода лишних параметров при формировании запроса. В общем случае любой параметр запроса будет формироваться из следующих вариантов (будет использован первый не пустой вариант):

1. Параметр задан явно пользователем
2. Параметр взят из переменной окружения
3. Параметр задан в конфигурационном файле пользователя
4. Параметр задан в глобальном конфигурационном файле с указанием данного пользователя
5. Параметр задан в глобальном конфигурационном файле с указанием используемой очереди
6. Параметр задан в глобальном конфигурационном файле как значение по умолчанию
7. Параметр установлен самой системой как значение по умолчанию

Для конкретных параметров какие-то пункты могут отсутствовать. Например, число процессоров, заказываемое для задачи должно быть задано явно и никак иначе.

Переменные окружения используются клиентскими программами, и поэтому здесь описаны лишь те, которые используются дистрибутивными утилитами. Ниже приведен их список:

QS_QUEUE	очередь по умолчанию
QS_PRIORITY	приоритет при постановке в очередь
QS_TEMP	шаблон имени временного каталога для работы программы
QS_OUT	шаблон имени выходного файла
QS_REP	шаблон имени файла отчета
QS_TIMELIM	лимит времени счета в секундах

2.14.Жизненный цикл задачи в системе очередей

Получив запрос на постановку в очередь новой задачи, система проверяет, имеет ли данный пользователь право ставить задачу в данную очередь, заказывать указанное количество процессоров и лимит времени (если он указан явно), требовать заданный приоритет для задачи, не превышен ли лимит на количество задач данного пользователя в очереди и общий лимит задач в очереди.

Если все условия удовлетворены, задача ставится в очередь в соответствии с приоритетом (заданным явно или по умолчанию). Далее задача ожидает, пока она не попадет в голову очереди и для нее не освободятся нужные ресурсы. После этого задача будет запущена системой.

Задача может быть запущена и раньше, в случае если для ее запуска есть ресурсы и, кроме того, ожидаемое время окончания работающих задач больше, чем указанное максимальное время работы задачи.

Запуск задачи проходит следующим образом:

1. Формируются значения псевдопеременных для данной задачи (их список приведен ниже);
2. Формируются параметры запуска задачи (при необходимости в них подставляются значения псевдопеременных), такие как имена файла отчета и файла стандартного вывода задачи;
3. Создаются временный каталог и, при необходимости, файл с локальной конфигурацией для задачи (задается параметрами `use_file` и `file_*` в файле конфигурации);
4. Исполняется команда, заданная параметром `pre_exec` в файле конфигурации;
5. Исполняется команда, заданная параметром `exec_line` в файле конфигурации (собственно запуск задачи);
6. Выполняются подпрограммы `PreExec` во всех `ExecModules`, ассоциированные с текущим профилем;
7. Пользователю на консоль посылается сообщение, заданное параметром `exec_write` в файле конфигурации, если оно не пустое;
8. После паузы в 5 секунд, исполняется команда, заданная параметром `just_exec` в файле конфигурации;
9. Выполняются подпрограммы Post`Exec` во всех `ExecModules`, ассоциированные с текущим профилем.

После завершения задачи (нормально, аварийно, или в результате снятия ее системой) выполняются следующие действия:

1. Исполняется команда, заданная параметром `kill_script` в файле конфигурации
2. Обновляются значения псевдопеременных
3. Формируется файл отчета
4. Исполняется команда, заданная параметром `post_exec` в файле конфигурации
5. Пользователю на консоль посылается сообщение, заданное параметром `post_exec_write` в файле конфигурации, если оно не пустое
6. Информация о задаче удаляется из системы

Ниже приведен список псевдопеременных, которые используются системой:

`sexe`	короткое имя исполняемого файла
`exe`	полное имя исполняемого файла
`args`	аргументы
`path`	путь к исполняемому файлу
`task`	полное имя задачи (`$path/$sexe $args` или `$exe $args`)
`user`	имя пользователя
`dir`	рабочий каталог
`home`	домашний каталог пользователя
`pid`	pid головного процесса задачи
`id`	идентификатор задачи в очереди
`np`	число процессоров
`nodes`	список процессоров
`spaced_nodes`	список узлов
`mpi_nodes`	список пар '`узел`' '`число_задач_на_узле`'
`status`	код завершения задачи
`special`	особые отметки системы о причинах завершения задачи (пустая строка, если задача завершилась нормально)
`core`	строка '`core dumped`' или пустая строка, в зависимости от завершения задачи
`signal`	номер сигнала, приведшего к завершению задачи(0, если задача завершилась нормально)

3. ПОСТАНОВКА ЗАДАНИЙ В ОЧЕРЕДЬ

Запуск задач, скомпилированных с библиотекой MPI, осуществляется командой `mpirun` из поставки системы:

```
mpirun -np N [OPTS] prog [prog_args]
```

либо

```
mpirun -np N [OPTS] —f batch_file
```

Ключи, указанные в скобках, являются необязательными. `prog` и `prog_args` — исполняемый файл задачи и ее аргументы соответственно. Второй вариант запуска предполагает наличие файла `batch_file`, в котором перечислены программы с аргументами. В данном случае происходит последовательный запуск перечисленных программ в рамках одной задачи.

Ниже описаны значения ключей mpirun.

ключ	**аргумент**	**описание**
`-np`	`число`	Количество процессоров
`-q`	`строка`	Название очереди, в которую ставится задача
`-maxtime`	`число`	Лимит времени счета в минутах
`-l`	`число`	Лимит времени счета в секундах
`-p`	`число`	Приоритет задачи в очереди
`-stdin`	`строка`	Файл для стандартного ввода
`-stdout`	`строка`	Файл для стандартного вывода
`-stderr`	`строка`	Файл для стандартного потока ошибок
`-r`	`список`*	Список задач, которые должны **запуститься** до запуска этой задачи
`-Y`	`список*`	Список задач, которые должны завершиться **успешно** до запуска этой задачи
`-Z`	`список*`	Список задач, которые должны завершиться **неуспешно** до запуска этой задачи
`-z`	`o|or|a|and`	Политика обработки списков в ключах -r,-Y,-Z. 'o' или 'or' указывает на выполнение условия, если хотя бы одна задача из списка его выполнила. 'a' или 'and' указывает на выполнение условия только тогда, когда все задачи в списке его выполнили.

*Список указывается через запятую, пробелы не допускаются

Примеры:

```
mpirun -np 15 -q users -maxtime 10 my_test -dummy -i 50
```

Данная команда поставит задачу `'my_test'` с параметрами `'-dummy -i 50'` в очередь `users` с лимитом работы в 10 минут и запросом на 15 процессоров.

```
mpirun -np 10 my_test2 /tmp/my_test.tmp -n 15
```

Данная команда поставит задачу `'my_test2'` с параметрами `'/tmp/my_test.tmp -n 10'` в очередь по умолчанию и с лимитом работы в 10 минут и запросом на 15 процессоров.

Если ключ -maxtime (или -l) не задан, то будет взято значение из переменной окружения QS_TIMELIMIT. Если оно пустое, или данная переменная не определена, то будет использовано значение, заданное в пользовательском файле ~/.qconf параметром def_time. Если в пользовательском файле этого параметра нет, то он будет взят из глобального файла конфигурации.

Указывая ключ -maxtime, пользователь может ускорить запуск своей задачи, так как дает системе возможность планировать время счета этой задачи и, как следствие, возможность выполнить ее «досрочно», если будет такая возможность.

При постановке задачи в очередь, запоминаются все переменные окружения. Во время запуска они будут установлены в запомненные значения.

При запуске задач программой типа mpiexec, mpirun_rsh и т.д., а также при запуске команды на узле с помощью ssh переменные среды будут восстановлены только для самой этой команды, но не для запущенных ей процессов на узлах.

После запуска задачи ее **стандартный вывод** будет перенаправлен в файл, имя которого определяется настройками по умолчанию, либо индивидуальными настройками пользователя (см. раздел «Индивидуальные настройки»), либо ключом `-stdout`. Обычно этот файл создается в том же каталоге, откуда задача была запущена, и имеет имя `<имя_задачи>.out-<номер>`, где номер — это номер задачи в очереди.

По окончании работы задачи создается **файл отчета** (его имя обычно аналогично имени файла вывода задачи, но суффикс `out` меняется на `rep`), в котором указываются полное имя задачи, аргументы, время счета, имя файла вывода, код возврата и, если задача завершилась аварийно, причина останова.

Во время работы задачи специально для нее создается **временный каталог**, содержимое которого будет очищено по окончании работы. В этом

каталоге целесообразно создавать временные рабочие файлы. Узнать полный путь к нему можно во время работы программы из переменной окружения `TEMP_DIR`.

4. ПРОСМОТР СОСТОЯНИЯ ОЧЕРЕДИ

В любой момент Вы можете посмотреть состояние очереди командой `tasks:`

```
tasks [-q queue][-r][-l][-t][-f][-o][-m mask][-u userlist][-b]
                        [-d id ... id|-v]
```

Ключи, указанные в скобках, являются необязательными. Ниже описаны значения ключей.

`-q`	Название очереди
`-r`	Показывать очереди рекурсивно
`-l`	Показывать дополнительную информацию
`-t`	Показывать лимит времени по умолчанию для пользователя
`-f`	Учитывать чужие задачи
`-o`	Учитывать свои задачи
`-m mask`	Использовать маску для выборки задач (см. ниже) [1]
`-u list`	Использовать список пользователей для выборки задач[2]
`-b`	Показывать информацию о заблокированных узлах
`-v`	Показывать состояние очереди (по умолчанию)
`-d`	Удаление задачи[3]

Параметры -f, -o указывают выборку задач. По умолчанию в выборку попадают все задачи (если не указано иначе в файле конфигурации).

Указывая данные параметры можно сузить или расширить выборку.

Например:

```
>tasks –q main
Queue: main
Running: 2; Queued: 11; Pre-runned: 11; Free: 1 of 2+14
Running:
ID   :     User: NP:      Time :      Timelimit: Task
85   :    user1:  2:  60:22:15 :Oct  7 11:26:51: prog1
113  :    user2:  4:  27:23:32 :Oct  7 05:24:18: test3
Queued:
170  :    user2:  1: 10:Oct  6 06:55:25:   0:18:00:00: !test8
172  :    user2:  1: 10:Oct  6 06:56:16:   0:18:00:00:
!B:__internal__:maximum np reached# test9
```

[1] Будет рассмотрен в главе «управление задачей в очереди»
[2] Будет рассмотрен в главе «управление задачей в очереди»
[3] Будет рассмотрен в главе «управление задачей в очереди»

В последней строке жирным выделен текст, который относится к строке выше. В силу того, что полная строка длинная, текст перенесён.

В данном примере мы видим выполняющиеся задачи prog1 пользователя user1 на 2 процессора, которая считается уже 60 часов 22 минуты, и задачу test3 пользователя user2. Первое поле — идентификатор (или номер) задачи в очереди. В очереди ожидают задачи test8 и test9 пользователя user2 на 1 процессор каждая с приоритетом 10.

Задача с номером 172 заблокирована в силу ограничения по числу занимаемых процессоров псевдопользователем __internal__.

Используя ключ `-l`, можно получить более подробную информацию:

```
>tasks -l -q main
Queue: main
Running: 2; Queued: 11; Pre-runned: 11; Free: 1 of 2+14
Running:
ID   :      User: NP:      Time :      Timelimit: Task
85   :     user1:  2:  60:20:26 :Oct  7 11:26:51: prog1
Program :/home/user1/prog1
CPUs    :node1:1,node1:2
Output  :/home/user1/prog1.out-85
Report  :/home/user1/prog1.rep-85
Workdir :/home/user1
-------------------------------------------
113  : user2:  4:  27:21:43 :Oct  7 05:24:18: test3
Program :/home/user2/test3
CPUs    :node2:1,node2:2,node3:1,node3:2
Output  :/home/user2/test3.out-113
Report  :/home/user2/test3.out-113
Workdir :/home/user2
-------------------------------------------
Queued:
170  : user2:  1: 10:Oct  6 06:55:25:   0:18:00:00: !test8
Program :/home/user2/test8
Output  :$dir/$sexe.out-$id
Report  :$dir/$sexe.rep-$id
Workdir :/home/user2
-------------------------------------------
172  : user2:  1: 10:Oct  6 06:56:16:   0:18:00:00:\
!B:__internal__:maximum np reached# test9
Program :/home/user2/test9
Blocked :__internal__:maximum np reached
Output  :$dir/$sexe.out-$id
Report  :$dir/$sexe.rep-$id
Workdir :/home/users2

-------------------------------------------------
```

Итак, мы видим, что к выдаче добавилась информация об имени файла со стандартным выводом задачи и файлом отчёта (поля `Output` и `Report`),

полное имя задачи (поле `Program`), список процессоров, занимаемых задачей (поле `CPUs`), и рабочий каталог (поле `Workdir`). Для заблокированных задач появляется поле '`Blocked`' с перечнем пар «кто заблокировал:причина». Снять блокировку может либо поставивший её, либо администратор. Автоматические блокировки ставятся от имени псевдопользователя '`__internal__`'.

5. УПРАВЛЕНИЕ ЗАДАЧЕЙ В ОЧЕРЕДИ

5.1. Удаление задач

Если по каким-то причинам Вы желаете снять свою задачу со счета или удалить ее из очереди, то можно воспользоваться программой `tasks`, описанной в предыдущей главе, с ключом `-d <id>`, где `id` — это идентификатор задачи в очереди. Допустимо передавать список идентификаторов, перечислив их через пробел.

Вместо идентификатора задачи можно указать '`all`', тогда будут удалены все задачи в очереди (если команда дается не администратором, то удалятся все задачи пользователя, давшего команду). Идентификатор (номер) указывается в первом поле (столбец ID) выдачи, получаемой при выполнении команды `tasks`.

Можно уточнить список удаляемых задач, указанием параметров `–m` и `–u`. Параметром -m можно указать маску имени задачи (аргумент этого параметра — регулярное выражение. О синтаксисе этого регулярного выражения можно прочесть, набрав команду `man perlre`). Параметр `–u` задает список пользователей (через запятую), которым должны принадлежать задачи выборки.

Например, если бы в предыдущем примере была выполнена команда

```
/home/usr2> tasks –d 2141
```

то задача compute, ожидающая запуска, была бы удалена из очереди. А выполнение команды

```
/home/usr2> tasks –d all -m 'm\S+'
```

привело бы к принудительному завершению задачи `msap`, т.к. это единственная задача, попадающая под указанную маску.

Выполнение команды

```
/home/usr2> tasks -d all
```

или команды

```
/home/usr2> tasks -d all -u usr1,usr2
```

привело бы к удалению обоих заданий.

Напоминаем, что удаление чужих заданий для рядового пользователя невозможно.

5.2. Смена приоритета

Кроме удаления задачи, стоящей в очереди, возможно изменить ее приоритет. Это делается с помощью команды `cleo-priority`:

```
cleo-priority [-q queue][-p port] new_pri id [id...]
```

Ниже описаны ее ключи:

`new_pri` — новое значение приоритета
`id` — идентификатор задачи (может быть перечислено несколько идентификаторов)
Пример:

```
change_pri 5 202 203
```

В этом примере мы задаем значение приоритета равным 5 для задач с номерами 202 и 203.

5.3. Блокировка задачи

Если пользователь или администратор не желает чтобы задача в очереди запустилась в ближайшее время, ее можно заблокировать. Предусмотрена также и обратная операция — разблокирование задачи. Блокировку и разблокировку можно проводить и для группы задач. Все эти действия осуществляются командой `cleo-blocktask`.

```
cleo-blocktask [-h][-q queue][-U userlist][-t mask]
               [-R 'reason'][-a as_user][-b|-u] task_id(s)
```

Ниже приведены описания ключей:

`-P`	Номер порта сервера
`-h`	Вывести подсказку
`-U list`	Задать список пользователей
`-t mask`	Задать маску имени задач
`-R 'reason'`	Указать причину блокировки (при разблокировании должна указываться та же причина)
`-a as_user`	(Раз)блокировать от имени as_user
`-q`	Указать очередь
`-b`	Блокировать задачу(и)
`-u`	Разблокировать задачу(и)

`task_id(s)` — идентификатор блокируемой или разблокируемой задачи, либо список идентификаторов, либо `all`.

Параметры `-U` и `-t` полностью аналогичны параметрам `-u` и `-m` команды `tasks`. (См. удаление задач)

Примеры:
`cleo-blocktask -b 335 336`
— блокировать задачи 335 и 336

`cleo-blocktask -u -q short -U user1,user3 all`
— разблокировать все задачи пользователей `usr1` и `usr2` в очереди `short`.

5.4. Установка автоблокировки

Напомним виды автоблокировок, которые могут быть установлены:

1) на запуск задачи пользователем
2) по времени

Первый вид автоблокировки — блокирование всех новых задач пользователя. Возможно также блокировать задачи всех пользователей, за исключением, возможно, указанных.

```
cleo-autoblock [-q queue][-P port][-r][-U userlist]
               [-R 'reason'][-a as_user] -b|-n [1/0]
```

Ниже приведены описания ключей:

`-q`	Указать очередь
`-P`	Номер порта сервера
`-r`	Выполнить блокировку рекурсивно для всех дочерних очередей
`-U list`	Задать список пользователей
`-R 'reason'`	Указать причину блокировки
`-b 0/1`	Блокировать новые задачи (1 - да, 0 - нет). По умолчанию используется 1
`-n 0/1`	Не блокировать новые задачи (1 - да, 0 - нет). По умолчанию используется 1
`-a user`	Произвести действие от имени пользователя user

Как видно, обязательным является задание ключа -b или -n. Эти ключи определяют, какое действие будет применяться к новым задачам. Если указан ключ '-b 1', то новые задачи указанных пользователей (или всех, если список не задан ключом -U) будут блокированы.

> Ключ '-b 0' снимает данную автоблокировку, **но не разблокируют уже заблокированные задачи.** Чтобы

разблокировать их, необходимо воспользоваться описанной выше программой `cleo-blocktask`.

Если указан ключ '`-n 1`', то задачи указанных пользователей не будут блокироваться (только с причиной, указанной в `-R` !). Таким образом можно блокировать задачи всех пользователей, кроме указанных. Ключ '`-n 0`' убирает заданную ранее автоблокировку.

Все автоблокировки (явно или неявно) используют так называемую причину блокировки. Если указано несколько причин, то задача блокируется по всем заданным причинам. Это же относится к «контрблокировкам», заданным ключом `-n`.

Например, если ключом `-b` заданы автоблокировки с причинами «dedicated» и «temporary», а для пользователя `user1` ключом `-n` указана контрблокировка с причиной «dedicated», то новые задачи пользователя `user1` будут блокироваться с причиной «temporary», а задачи всех остальных пользователей — с причинами «dedicated» и «temporary».

Примеры:

```
cleo-autoblock -q main -r -R 'dedicated' -b
```

— блокировать все новые задачи всех пользователей во всех очередях с указанием причины 'dedicated'

```
cleo-autoblock -q short -U user1,user3 -R 'dedicated' -u
```

— не блокировать задачи пользователей `user1` и `user3` в очереди `short` с указанием причины 'dedicated'.

Второй вид автоблокировки — блокирование задач по времени.

Задача данного вида автоблокировки — не допустить выполнения задач указанных пользователей к указанному времени. То есть если время ожидаемого завершения задачи превышает указанное, то задача блокируется.

Данные блокировки описываются в специальном файле — `/etc/cleo-time-restrictions`. Каждая блокировка описывается ровно в одной строке. Формат такой строки указан ниже:

```
[queue:] ena once allow hh:mm day [mth] - hh:mm day [mth] [users]
```

`queue` — опционально задаваемый параметр, указывающий очередь. Если он не указан, то блокировка применяется ко всем очередям.

`ena` — 1 или 0. Указывает активна данная блокировка или нет.

`once` — 1 или 0. Указывает на то, что блокировка должна сработать единожды.

`allow` — 1 или 0. Указывает должны ли задачи указанных пользователей блокироваться. Если указан 0, то все остальные блокировки на это время не действуют.

`hh:mm day или hh:mm day mth` — время начала блокировки. К этому времени задачи не должны считаться. В первом варианте задаётся время в часах и минутах и указывается день недели. Во втором — день месяца и месяц.

Следующий блок аналогичен вышеописанному, но задаёт время окончания блокировки.

Далее указывается опциональный список пользователей через запятую или пробел. Если список пользователей не указан, то блокировка действует на всех пользователей.

После обновления файла с описанием автоблокировок по времени, он должен быть перечитан системой. Это достигается выполнением команды `'cleo-mode update_restrict'`.

6. УПРАВЛЕНИЕ ОЧЕРЕДЬЮ И ПРОЦЕССОРАМИ

6.1. Блокировка процессоров

Если есть необходимость исключить из рабочего поля один или несколько узлов, то можно воспользоваться командой `cleo-blockcpu`:

```
cleo-blockcpu [-h][-P port][-q queue][-s][-b|-u] cpus
```

Ниже приведены описания ключей:

`-P`	Номер порта сервера
`-h`	Вывести подсказку
`-q`	Указать очередь
`-s`	Производить отложенную блокировку
`-b`	Блокировать процессор
`-u`	Разблокировать процессор

cpus — список узлов, подлежащих блокировке или разблокировке.

При проведении отложенной блокировки (ключ -s), каждый из указанных процессоров блокируется только после того, как на нём завершается задача пользователя (либо сразу, если на нём не выполняется задача в данный момент). В момент реальной блокировки исполняется скрипт, указанный в параметре `mon_delayed_block_exec` в конфигурационном файле. Этот параметр представляет собой шаблон, в котором допустимо использование следующих псевдо-переменных:

- `cpu` - имя процессора,
- `time` - текущее время (unix-time),
- `reason` - причина блокировки.

Примеры:

```
cleo-blockcpu node35
```

— блокировать узел `node35`

```
cleo-blockcpu -u node{3,4,5,10,15}
```

— разблокировать узлы `node3`, `node4`, `node5`, `node10`, `node15`. В данном примере использована возможность unix shell «раскрывать» параметры команд.

```
block_cpu -s node38
```

— блокировать узел `node38` после того, как на нём завершатся задачи.

6.2. Блокирование очереди

Администраторы системы имеют возможность блокировать очередь (или группу очередей). Это делается при помощи команды `cleo-mode`:

```
cleo-mode [-q queue][-r][-h] mode
```

Ниже приведены описания ключей:

`-h`	Вывести подсказку
`-q`	Задать очередь
`-r`	Повторить команду для всех дочерних очередей

`mode` может принимать значения

- `run` разрешить запуск задач в очереди,
- `norun` запретить запуск задач в очереди,
- `qenable` разрешить постановку новых задач в очередь,
- `qdisable` запретить постановку новых задач в очередь.

Примеры:

```
cleo-mode -q main -r norun
```

— запретить запуск новых задач во всех очередях (при этом можно ставить задачи в очередь)

```
cleo-mode -q short qdisable
```

— запретить постановку задач в очередь short (при этом уже поставленные задачи из очереди не удаляются)

6.3. Приостановка задачи

В Cleo есть возможность на время «заморозить» задачу, послав всем её процессам сигнал SIGSTOP. Если в программе не предусмотрен перехват такого сигнала (а обычно так и есть), то все процессы задачи будут приостановлены.

В дальнейшем задачу можно «разморозить». Всем её процессам будет послан сигнал SIGCONT и они продолжат работу.

Для «заморозки» и «разморозки» задачи используется команда `cleo-freeze`:

```
cleo-freeze [-q queue][-u] id [id2 ... idN]
```

Ниже приведены описания ключей:

`-q`	Задать очередь.
`-u`	Разморозить задачу.

`id` указывает идентификатор задачи. Опционально можно указать несколько идентификаторов. По умолчанию задачи замораживаются. Разморозить их можно, указывая ключ `-u`.

Примеры:

```
cleo-freeze -q main 10 11 12
```
— заморозить задачи 10, 11 и 12.

```
cleo-mode -q short -u 10
```
— разморозить задачу 10.

6.4. Дополнительные средства управления очередью

Кроме вышеописанных режимов команды `cleo-mode`, с её помощью можно задавать следующие команды:

`reload_conf`	перечитать файл конфигурации
`recreate_logs`	пересоздать файлы журналов
`update_users`	перечитать информацию о пользователях системы
`update_restrict`	перечитать информацию об автоблокировках по времени
`update_pid`	обновить файл с номером головного процесса
`reload_sced`	перезагрузить планировщик
`version`	выдать версию системы

Команда `recreate_logs` может быть полезна при автоматическом ротировании журналов пакетом logrotate или аналогичным.

Сервер Cleo может ротировать журналы самостоятельно. По умолчанию ротация журнала производится каждые 5 недель или в случае, если размер журнала превысил 1ГБ. Эти значения можно поменять в файле конфигурации с помощью параметров `max_log_days/max_log_size` и `max_short_log_days/max_short_log_size` для основного и короткого журнала соответственно. Задавая значение параметра равным 0, его действие можно отключить.

При перечитывании конфигурации обновляются не все параметры. В частности, список узлов остается прежним.

7. ОСНОВНОЙ КЛИЕНТ

Все команды серверу системы очередей могут быть даны через программу основного клиента — `cleo-client`. Ниже перечислены ее ключи:

`-v`	Просмотр состояния очереди.	
`-s`	Просмотр данных о свободных процессорах в очереди.	
`-V`	Просмотр состояния очереди (альтернативный вариант).	
`-R`	Использовать рекурсию (для `-v` и `-m`).	
`-F`	Использовать расширенную выдачу (для `-v`).	
`-T`	Выдавать техническую информацию (для `-v`).	
`-L list`	Использовать указанный список (для `-V, -d`).	
`-b id`	Блокировать задачу <`id`>.	
`-u id`	Разблокировать задачу <`id`>.	
`-B p`	Блокировать процессор <`p`>.	
`-U p`	Разблокировать процессор <`p`>.	
`-p port`	Использовать для коммуникаций порт <`port`>.	
`-d id`	Удалить задачу <`id`>.	
`-f`	Форсировать удаление.	
`-c com`	Добавить в очередь задачу <`com`> (если задаются аргументы, то необходимо заключить команду с ними в кавычки).	
`-C id`	Изменение параметров задачи с номером `id`.	
`-q abc`	Использовать очередь '`abc`'.	
`-1 file`	Использовать `file` как поток ввода.	
`-2 file`	Использовать `file` как поток вывода.	
`-3 file`	Использовать `file` как поток ошибок.	
`-n num`	Использовать <`num`> процессоров.	
`-w dir`	Использовать <`dir`> в качестве рабочего каталога.	
`-E`	Передать все переменные окружения для использования задачей.	
`-o out`	Использовать out в качестве шаблона выходного файла.	
`-r rep`	Использовать rep в качестве шаблона файла отчета.	
`-t tmp`	Использовать `tmp` в качестве шаблона временного каталога.	
`-l lim`	Ограничить время работы `lim` секундами.	
`-P pri`	Задать приоритет `pri`.	
`-M str`	Задать дополнительные параметры для команды `-m`.	
`-G cmd`	Выполнить общую команду `cmd`. Все последующие аргументы интерпретируются как пары аргумент-значение для заданной команды.	
`-m mod`	Изменить/просмотреть режим работы:	
	`run`	разрешить запуск новых задач
	`norun`	запретить запуск новых задач

	view	просмотреть текущий режим
	queue_disable	запретить постановку новых задач в очередь
	queue_enable	разрешить постановку новых задач в очередь
	reload_conf	перечитать файл конфигурации
	update_pid	обновить файл с PID головного процесса
	version	показать номер версии работающей системы

Возможные значения параметра cmd ключа -G приводятся ниже. Во всех случаях возможно применение аргумента queue, задающего имя очереди. Обязательные аргументы отмечены знаком '*'.

freeze — заморозка задачи. Аргументы:

- *id — идентификатор задачи (может быть задан несколько раз),
- val — 1 для заморозки, 0 для разморозки.

mode — смена режима работы. Аргументы: recurs — 1, если надо выполнить запрос рекурсивно для всех дочерних очередей, mode_<mod> — выбрать режим (значение этого параметра может быть любым, mod — аналогично аргументу ключа -m).

view — просмотр состояния очереди. Аргументы:

- showsub — показать также состояние дочерних очередей,
- flags — задать флаги просмотра
  - f — показывать задачи других пользователей,
  - o — показывать свои задачи,
  - p — показывать сводные данные по процессорам,
  - P — показывать краткие сводные данные по процессорам,
  - m — вывести лимит времени по умолчанию,
  - M — вывести максимальный лимит времени,
  - O — вывести данные по прочим ограничениям пользователя,
  - s — вывести режим работы системы,
  - B — вывести список заблокированных процессоров,
  - b — вывести список заблокированных задач,
  - u=u1;u2;... задать список пользователей для просмотра,
  - c — вывести список процессоров, занятых задачей,
  - '>' — вывести имя файла стандартного вывода,
  - r — вывести имя файла отчёта,
  - w — вывести имя рабочего каталога,
  - F — вывести полную командную строку задачи,
  - C — прочие данные по задаче.

`add` — добавить задачу. Аргументы:

- `path` — путь, добавляемый к переменной PATH при запуске,
- `dir` — рабочий каталог,
- *`args0 ... argsN` — аргументы для запуска задачи (`arg0` = имя исполняемого файла),
- `temp_dir` — шаблон имени рабочего каталога,
- `outfile` — шаблон имени файла стандартного вывода,
- `repfile` — шаблон имени файла отчёта,
- `one_rep` — 1, если новый файл отчёта или ст. вывода может перезаписывать старый, и 0, если нет,
- `use_empty` — пустая строка, если пакет empty не используется, или путь к команде empty, если используется,
- `empty_input` — шаблон имени канала для общения с задачей, запущенной под empty,
- `env` — запакованные[4] данные о переменных окружения,
- `profile` — имя профиля,
- `timelimit` — лимит времени работы задачи в секундах,
- `*np` — число запрашиваемых процессоров,
- `priority` — запрашиваемый приоритет.

`del` — удалить задачу. Аргументы:

- `*id` — список идентификаторов задач через запятую или слово 'all',
- `userlist` — список пользователей через запятую,
- `mask` — маска имени задачи в виде регулярного выражения perl,
- `rmask` — маска состояния задачи: r=запущенные задачи, q=стоящие в очереди (пустая строка эквивалентна 'rq'),
- `forced` — 1, если информация о задаче должна быть удалена вне зависимости от результатов удаления,
- `reason` — описание причины удаления (будет отражена в файле отчёта).

Если задан хотя бы один из аргументов `userlist`, `mask` или `rmask`, то удаление будет применено только к тем задачам из указанного списка, которые попадают под указанные ограничения. Например, если `userlist` равен '`u1`,`u2`', и `id` равно '`all`', то будут удалены все задачи пользователей `u1` и `u2`.

`debug` — выполнить произвольные команды perl внутри процесса Cleo. Аргументы:

- *`command` — текст команды,

[4]переменные объединяются в одну строку парами имя_переменной=значение через символ '\0', затем полученная строка преобразуется perl-командой pack('u',$str). Из результирующей строки удаляются все переносы строк.

- `recurse` — 1, если команда должна быть выполнена также во всех дочерних очередях.

`priority` — сменить приоритет задачи. Аргументы:
- `*id` — идентификаторы задач, перечисленные через запятую,
- `*val` — новое значение приоритета.

`chattr` — сменить произвольный атрибут задачи. Аргументы:
- `*id` — идентификаторы задач, перечисленные через запятую,
- `*attribute` — имя_атрибута и его новое значение, соединённые знаком табуляции.

`autoblock` — становить или сбросить автоблокировку. Аргументы:
- `*users` — список пользователей через запятую,
- `*val` — 1, если автоблокировка устанавливается, и 0, если сбрасывается,
- `recurse` — 1, если автоблокировка устанавливается и во всех дочерних очередях.

`block` — блокировать или разблокировать задачи. Аргументы:
- `*id` — список идентификаторов задач через запятую,
- `userlist` — список пользователей через запятую,
- `mask` — маска имени задачи в виде регулярного выражения perl,
- `username` — пользователь, от имени которого производится действие,
- `reason` — описание причины действия,
- `*val` — 1, если производится блокирование, 0, если производится разблокирование.

`block_pe` — блокировать или разблокировать процессоры. Аргументы:
- `*id` — имена процессоров и/или узлов через запятую,
- `*val` — 1, если производится блокирование, 0, если производится разблокирование,
- `reason` — описание причины действия,
- `safe` — 1, если блокировка производится в безопасном режиме.

`get_io` — получить имя канала псевдотерминала задачи. Аргумент:
- `*id` — идентификатор задачи.

Используя программу основного клиента, можно реализовать все функции работы с системой очередей. Поставляемые скрипты `mpirun, tasks` и прочие используют именно ее.

8. ТОНКАЯ НАСТРОЙКА ДЛЯ ПОЛЬЗОВАТЕЛЯ

Для каждого пользователя по умолчанию задан набор параметров и ограничений. Часть из них можно задавать при постановке задачи в очередь (например, имя очереди). Однако, для облегчения работы предусмотрена возможность настроить часть параметров индивидуально, без помощи администратора. Это делается с помощью переменных окружения и в конфигурационном файле.

Список переменных окружения приведен в главе 3.

При указании переменных среды `QS_TEMP`, `QS_OUT` и `QS_REP` можно использовать псевдопеременные, которые будут автоматически подставлены системой при запуске задачи.

Установить переменную среды можно командой export в командной строке (если Вы работаете в sh или bash). Чтобы указать, что Вы используете псевдопеременную, перед ее именем надо поставить знак '$'. Так как sh сам обрабатывает знак '$', то при задании переменной окружения перед этим знаком надо поставить обратную косую черту '\' или заключить все значение переменной в одинарные кавычки.

Например:

```
export QS_TEMP='/tmp/my-temp-$id'
export QS_OUT=\$home/out_for_\$sexe-\$id
```

Кроме переменных окружения, можно воспользоваться индивидуальным файлом настройки `.qconf`, расположенным в Вашем домашнем каталоге. В нем можно задать следующие параметры:

`post_exec`	шаблон команды, выполняющейся после завершения задачи
`post_exec_write`	шаблон строки, которая пишется на терминал после завершения задачи
`one_report`	ненулевое значение предписывает перезаписывать файлы результатов и отчетов, если они уже существуют. Иначе новые файлы создаются с постфиксом .N, где N — уникальный номер (.1, .2 и т.д.)
`tmp_dir`	шаблон имени временного каталога для работы
`repfile`	шаблон имени файла отчета
`outfile`	шаблон имени выходного файла
`def_queue`	имя очереди по умолчанию
`pe_select`	имя метода распределения процессоров
`priority`	приоритет по умолчанию
`write`	шаблон строки, которая пишется на терминал при пуске задачи
`outfile`	шаблон имени файла выдачи задачи

repfile	шаблон имени отчета задачи

Во всех параметрах, где в описании указано слово «шаблон» можно указывать псевдопеременные.

Параметры задаются в виде: <имя_параметра> = <значение>. Строки, начинающиеся со знака '#', считаются комментариями.

Пример:

```
    post_exec= echo 'End of task $sexe on $np processes (code
$status $core)'| mail $user
    def_queue= alt
    priority=  5
```

Приоритет всех параметров следующий: ключи командной строки, переменные окружения, параметры в файле настройки и, наконец, умолчания.

9. ФОРМИРОВАНИЕ ФАЙЛА КОНФИГУРАЦИИ ЗАДАЧИ

При старте задачи зачастую требуется динамически сформировать файл специального вида и передать его параллельной среде для запуска задачи (например, файл machinefile в среде mpich). Для этих целей предусмотрены специальные средства. Для такого файла могут быть заданы его имя (`use_file`), шапка (`file_head`), окончание (`file_tail`), шаблон строки (`file_line`) и параметр, указывающий на то, формировать ли строки на каждый процессор или на каждый узел (`coll_nodes`).

Кроме того, в этих параметрах можно использовать дополнительный набор псевдопеременных:

`n`	число процессов на данном узле
`node`	текущий узел
`count`	порядковый номер процесса на узле
`nid`	идентификатор процессора на узле (часть id в node:id)

В параметре `file_line` можно использовать все псевдопеременные, описанные ранее, и, кроме того, следующие:

Если параметр `coll_nodes` отличен от 0 (и не пуст), то строки будут формироваться не для каждого процессора, а для каждого узла.

Файл формируется только в том случае, если параметр `use_file` не пуст. После окончания работы задачи этот файл уничтожается.

10. НАСТРОЙКА СЕРВЕРА ЧЕРЕЗ ФАЙЛ КОНФИГУРАЦИИ И КЛЮЧИ

Сервер системы очередей зачитывает при старте свои настройки из файла. По умолчанию это `/etc/cleo.conf`. Его имя может быть указано ключом -c. Ниже перечислены ключи запуска сервера `cleo`:

`-c file`	Использовать `<file>` в качестве файла конфигурации
`-p port`	Использовать порт `<port>` для коммуникаций с клиентами
`-s file`	Использовать `<file>` как префикс имени файла для сохранения очереди (каждая очередь будет сохраняться в файле `<file>.<имя_очереди>`)
`-a file`	То же, что и предыдущий параметр, но для альтернативного файла
`-l file`	Использовать `<file>` в качестве лог-файла
`-i file`	Использовать `<file>` в качестве файла для сохранения PID головного процесса
`-x`	Отключить авторизацию клиентов (в клиентах она также должна быть отключена) (см. п. 2.7)
`-v`	Выдавать больше отладочной информации

Файл конфигурации имеет блочную структуру. Каждый блок начинается со строки вида: `[название_блока]`

Строки, начинающиеся со знака `'#'`, считаются комментариями.

Обязательными являются блоки `server` и `clusters`. В блоке `server` задаются основные параметры сервера системы управления заданиями. В блоке `clusters` описываются очереди.

Возможно наличие блоков `groups`, `profiles`, `users`, `clusterusers` и `mod`. В них описываются соответственно группы пользователей, описания профилей пользователей, индивидуальные настройки пользователей, настройки пользователей на конкретных очередях и настройки для подключаемых модулей и планировщиков.

Все параметры задаются в виде: `имя_параметра = значение`

Список доступных параметров блока `server` приведен ниже.

Параметр	Тип	Описание
`add_pri_on_chld`	число	инкремент приоритета задачи в дочерней очереди
`adm_email`	строка	e-mail администратора (может далее подставляться как псевдопеременная adm_email)
`admins`	список	список администраторов системы

allowed_ips	список	адреса, с которых разрешено соединение с сервером
attach_mask	строка	маска для контроля запущенных задач (см. ниже)
attach_parent_mask	строка	маска для контроля запущенных задач (см. ниже)
coll_nodes	0/1	использовать одну строку на узел в опции file_line
count_first	0/1	учитывать первый узел при формировании файла, указанного в use_file (по умолчанию — 1)
compress_cmd	строка	команда для сжатия устаревших журналов, по умолчанию — 'bzip2 -c'
compress_ext	строка	расширение сжатых журналов, по умолчанию — 'bz2'
cpu_map_file	строка	файл преобразования имён процессоров для файла, указанного в use_file (формат см. далее)
debug_users	список	пользователи, которым разрешено использование команды debug
def_admview_flags	строка	флаги для просмотра по умолчанию для администраторов (см. ниже)
def_priority	число	приоритет задач по умолчанию
def_queue	строка	очередь по умолчанию
def_view_flags	строка	флаги для просмотра по умолчанию (см. ниже)
default_time	число	лимит времени счета для задач по умолчанию
exec_line	строка	строка запуска задачи по умолчанию
exec_modules	список	список ExecModules, доступных системе
exec_modules_dir	строка	путь к каталогу, содержащему все ExecModules.
exec_write	строка	строка для оповещения пользователя через программу write после запуска задачи
file_head	строка	шапка файла, указанного в use_file
file_line	строка	строка файла, указанного в use_file (на каждый узел или процессор)
file_mask	строка	регулярное выражение в формате perl, указывающее имя программы, запускаемой на узле через обычный rsh, но контроль над которой необходимо брать монитору cleo.
file_tail	строка	окончание файла, указанного в use_file
first_line	строка	первая строка файла, указанного в use_file (вместо file_line)

force_foreign_run	0/1	разрешить или запретить запуск задач из родительской очереди, если данная очередь заблокирована
gid	число	номер группы, под которой работает сервер
kill_mons_on_exit	0/1	Завершать ли мониторы на узлах при завершении работы сервера (по умолчанию - 0)
kill_script	строка	имя программы, выполняющейся после завершения или снятия работающей задачи от **root** (чистка некорректных завершений). Можно указывать параметры, в которых разрешены все псевдопеременные (см. ниже)
log_file	строка	путь к лог-файлу (единый для всех очередей)
max_count	число	внутреннее максимальное значение счётчиков сообщений
max_ext_sced_err	число	максимально допустимое число сбоев внешнего планировщика. При его превышении, планировщик отключается
max_log_days	число	максимальный возраст основного журнала в сутках, после которого он пересоздаётся, а старый журнал сжимается
max_log_size	число	максимальный размер основного журнала в байтах, после которого он пересоздаётся, а старый журнал сжимается
max_short_log_days	число	максимальный возраст короткого журнала в сутках, после которого он пересоздаётся, а старый журнал сжимается
max_short_log_size	число	максимальный размер короткого журнала в байтах, после которого он пересоздаётся, а старый журнал сжимается
max_np	число	максимальное число процессоров, которое пользователь может запросить для задачи
max_queue	число	максимальное число задач в очереди
max_sum_np	число	максимальное число процессоров, которое пользователь может занять для счета одновременно
max_tasks	число	максимальное число задач на одном процессоре (не реализовано)
max_time	число	абсолютный лимит времени счета для задач
min_np	число	минимальное число процессоров, которое пользователь может запросить для задачи

mon_back_exec	строка	программа для запуска при возвращении узла в рабочее поле (по данным монитора на этом узле)
mon_block_delay	число	время в секундах, через которое должен блокироваться узел, если был превышен тайм-аут при связи с ним
mon_fail_exec	строка	программа для запуска при сбое узла по тайм-ауту
mon_node_port	число	номер порта для связи с мониторами на узлах
mon_ping_interval	число	интервал в сек. между пингованиями узлов
mon_path_prepend	строка	строка для добавления в переменную PATH на узлах перед запуском задач (через via_mons). Строка добавляется перед старым содержимым PATH
mon_path_append	строка	строка для добавления в переменную PATH на узлах перед запуском задач (через via_mons). Строка добавляется после старого содержимого PATH
mon_port	число	номер порта для связи с мониторами на головной машине
mon_req_timeout	число	таймаут в сек. для ответа монитора на запрос
mon_rnd_ping	число	максимальное значение случайных отклонений при пинговании узлов
mon_rsh_command	строка	имя программы, используемой для эмуляции rsh на узлах
mon_run_string	строка	строка для запуска мониторов на узлах
mon_run_string	число	тайм-аут в секундах ожидания завершения команды mon_run_string
mon_timeout	число	тайм-аут в секундах ожидания ответа от монитора
norootadm	0/1	считать пользователя root администратором по умолчанию
nousers	список	список пользователей, которым запрещен доступ на счет
one_report	0/1	перезаписывать файл отчета и/или вывода, если такой уже существует (по умолчанию 0)
outfile	строка	файл вывода задачи
pe	строка	список процессоров (если на одном узле несколько процессоров, то они должны быть описаны в виде host:id1 host:id2 ... host:idN. Например, M1E:2 M1E:4)

pe_sel_die_count	число	количество неудач при запуске модуля стратегии распределения узлов перед её отключением
pe_sel_method	строка	имя новой стратегии распределения процессоров и (через пробел) имя внешней программы, ее реализующей (возможно с параметрами)
pe_select	строка	стратегия распределения процессоров по умолчанию
pid_file	строка	имя файла, в который будет записан PID головного серверного процесса
port	число	порт для соединений с сервером системы очередей
post_exec	строка	команда для выполнения после завершения задачи
post_exec_write	строка	строка для оповещения пользователя через программу write после завершения задачи
pre_exec	строка	имя программы, выполняющейся непосредственно перед запуском задачи. Можно указывать параметры, в которых разрешены все псевдопеременные (см. ниже)
priority	число	приоритет задач по умолчанию
q_fail_exec	строка	имя программы, выполняющейся после неудачного завершения задачи. Можно указывать параметры, в которых разрешены все псевдопеременные (см. ниже)
q_just_exec	строка	имя программы, выполняющейся через короткое время после запуска задачи. Можно указывать параметры, в которых разрешены все псевдопеременные (см. ниже)
q_ok_exec	строка	имя программы, выполняющейся после удачного завершения задачи. Можно указывать параметры, в которых разрешены все псевдопеременные (см. ниже)
q_pre_exec	строка	имя программы, выполняющейся непосредственно перед запуском задачи. Можно указывать параметры, в которых разрешены все псевдопеременные (см. ниже)
queue_alt_save	строка	альтернативный префикс для сохранения очереди (используется, если основной почему-либо недоступен)

queue_save	строка	префикс файла для сохранения очереди (для каждой очереди к нему будет добавляться точка и ее имя; таким образом у каждой очереди свой файл для сохранения)
repfile	строка	файл отчета задачи
root_cluster_name	строка	имя головной очереди (по умолчанию — main)
run_via_mons	0/1	запускать, используя мониторы
pseudo_rsh_port	число	номер порта для соединения эмуляторов rsh на узлах
scedule	строка	имя внешнего планировщика
scedulers	список	список подключаемых планировщиков
scedulers_dir	строка	путь к каталогу с файлами планировщиков
short_log_file	строка	путь к сокращенному лог-файлу
status_file	строка	префикс файла с текущей информацией о задачах (для каждой из очередей к нему добавляется через точку имя очереди)
time_qcheck	число	частота проверки на исчерпание лимита времени задачами (в секундах)
time_restrict_file	строка	файл с описаниями ограничений запусков задач
timeout	число	таймаут соединения с сервером (для клиентских соединений)
temp_dir	строка	каталог для временных файлов задач
use_exec_modules	список	список ExecModules, которые надо задействовать при запуске и завершении задачи
use_file	строка	имя файла, формируемого при старте задачи
use_first_line	0/1	использовать параметр first_line при формировании файла
use_monitors	0/1	использовать связь с мониторами на узлах
use_rsh_filter	0/1	использовать эмуляцию rsh
user_conf_file	строка	путь к файлу настроек пользователя от его домашнего каталога
user_fail_exec	строка	аналогично user_fail_exec, но исполняется от имени пользователя
user_just_exec	строка	аналогично user_just_exec, но исполняется от имени пользователя
user_ok_exec	строка	аналогично user_ok_exec, но исполняется от имени пользователя
user_pre_exec	строка	аналогично user_pre_exec, но исполняется от имени пользователя
users	список	список пользователей, которым разрешен запуск задач

verbose	0/1	выдавать более подробную диагностику в стартовых логах

Тип «0/1» означает, что значением данного параметра может быть «1» или «0», означающими «да» и «нет» соответственно. Тип «список» предполагает, что список значений будет записан в виде строки с использованием пробела или запятой в качестве разделителя.

В параметрах file_head, file_line, first_line, file_tail для обозначения перевода строки надо писать '\n'.

Блок clusters описывает очереди. Иерархия очередей описывается при помощи параметра parent, то есть для каждой очереди задается ее родитель. Все параметры в этом блоке задаются в виде: имя_очереди.имя_параметра = значение.

В этом разделе доступны все параметры раздела server, за исключением: allowed_ips, kill_mons_on_exit, log_file, max_count, mon_*, pid_file, port, queue_alt_save, queue_save, root_cluster_name, short_log_file, status_file, tumeout, verbose.

Дополнительные параметры описаны ниже:

parent	строка	родительская очередь
pe	список	список процессоров

В блоке groups можно определить имена для групп пользователей. Имя параметра в данной секции даёт имя группе. Пример:

```
[group]
students = petrov, ivanov, sidorov
```

Такая запись описывает группу students из 3-х пользователей petrov, ivanov, sidorov.

Имя этой группы можно использовать в любых списковых параметрах для обозначения указанных пользователей. Для этого необходимо в списке записать имя группы, предварив его знаком '+'. Допускаются вложенные группы.

Необязательный блок users описывает параметры для отдельных пользователей (не групп!) Все опции задаются в виде: имя_пользователя.имя_параметра = значение.

Блок clusterusers описывает параметры, относящиеся к отдельным пользователям в отдельных очередях. Опции задаются в виде: имя_пользователя.имя_очереди.имя_параметра = значение.

Блок profiles аналогичен блоку users, но описывает профили. Эти профили могут быть использованы пользователями в команде mpirun (параметр -as).

Опции max_queue, max_np, max_sum_np, min_np, exec_line, post_exec, post_exec_write, one_report, tmp_dir, repfile, outfile, def_queue, pe_select, priority, use_file, file_*, coll_nodes, attach_mask, attach_parent_mask, run_via_mons секции server имеют ту же семантику и в секциях profiles, users и clusterusers.

Отдельно остановимся на описании параметра attach_mask. Этот параметр описывает маску командной строки (используются регулярные выражения perl) для отслеживания запуска задачи на узлах. При использовании параметра run_via_mons он бесполезен. В противном случае, после запуска задачи, на узлах отслеживаются новые процессы пользователя с командной строкой, подходящей под маску, а также все их потомки. В случае завершения задачи эти процессы будут уничтожены.

В параметрах def_view_flags и def_admview_flags задаются флаги для просмотра состояния очереди командой tasks (точнее qclient3.pl -V). Флаги указываются одной строкой. Ниже приводится описание флагов:

f	показывать чужие задачи
o	показывать свои задачи
P	показывать количество занятых, свободных и зарезервированных процессоров
p	показывать список занятых и список свободных процессоров
m	показывать лимит времени для пользователя по умолчанию
M	показывать лимит времени для очереди по умолчанию
s	показывать блокировки очереди
b	показывать заблокированные задачи
B	показывать заблокированные процессоры
c	для каждой задачи показывать список процессоров
r	для каждой задачи показывать имя файла отчета
w	для каждой задачи показывать рабочий каталог
F	для каждой задачи показывать полную строку запуска
>	для каждой задачи показывать имя файла вывода

По умолчанию принимаются пустые значения, что эквивалентно указанию флагов o и f одновременно.

Блок mod описывает параметры, относящиеся к подключаемым модулям и планировщикам. Опции задаются в виде: имя_модуля.имя_очереди.имя_пользователя.имя_параметра = значение. имя_очереди и имя_пользователя могут быть равны '*', что означает «для всех».

11. ПОДКЛЮЧЕНИЕ ВНЕШНИХ МОДУЛЕЙ СТРАТЕГИЙ РАСПРЕДЕЛЕНИЯ ПРОЦЕССОРОВ

К системе очередей можно подключать внешние модули для собственных стратегий распределения процессоров. Модуль представляет собой программу или скрипт, который должен удовлетворять следующей спецификации:

- Запросы сервера передаются на стандартный вход модуля по одной строке на запрос.
- Все строки заканчиваются символом перевода строки.
- При старте на стандартный вход ему передается срока вида `'QSYSTEM version <версия_системы>'`. Модуль должен ответить строкой `'QS_PE_SELECT <список_методов>'`.

Список методов — это перечисленные через пробел названия методов (протоколов), которые поддерживает модуль. Система сама определяет, каким методом она будет пользоваться при общении с модулем.

На данный момент поддерживается только метод 'PLAIN'. Ниже приводится его описание:

Запрос на выделение процессоров передается сервером в виде `'N pe1 pe2 ... peM'`, где `N` — число запрашиваемых процессоров, а `pe1...peM` — имена всех свободных процессоров.

- Ответ на запрос должен содержать список (через пробел) из ВСЕХ процессоров, указанных в запросе, но первые N должны быть те, которые будут отданы задаче. Порядок остальных не имеет значения.

На ответ дается ограниченное время (в данный момент 2 секунды). Если модуль не отвечает в этот период времени, он считается неработоспособным и перезапускается. После 10 перезапусков модуль отключается от системы, и вместо него используется стандартный метод `random`.

Обращаем внимание на то, что все ответы модуля должны явно сбрасываться из буфера (функция `flush`). Модуль должен работать в бесконечном цикле, получая запросы и отвечая на них.

12. ПОДКЛЮЧЕНИЕ EXEC_MODULES

Для выполнения специфичных для параллельной среды или кластера действий при старте или завершении задачи в Cleo предусмотрено подключение внешних модулей, называемых ExecModules. Эти модули должны быть написаны на языке perl.

Модуль обязательно должен содержать переменную `$cleo`, которой присваивается версия модуля. Он также может содержать функции `pre`, `post`, `ok` и `fail`. Если какая-то из них не определена, она полагается пустой.

При запуске задачи вызывается функция `pre` модуля. В качестве аргумента ей передаётся ссылка на хэш с описанием задачи.

При завершении задачи вызывается функция `post` модуля с таким же аргументом, как и для `pre`. В случае успешного завершения задачи, независимо от вызова `pre`, вызывается функция `ok`, в случае неуспешного — `fail`. Их аргумент такой же, как и у остальных.

Пример модуля, информирующего пользователя об успешном завершении его программы по e-mail:

```
use vars qw($cleo)

sub ok(){
  my $q=$_[0];
   open MAIL "|mail -s 'task end' $q->{user}\@localhost" or
return 1;
   print MAIL "your task $q->{id} in queue $q->queue finished
successfully\n";
  close MAIL;
  return 0;
}
```

Подключить модуль можно указав имя его файла в списке в параметре `exec_modules`. Каталог, содержащий все модули, указывается в параметре `exec_modules_path`. Для того, чтобы модуль работал при запуске или завершении задач, необходимо указать имя его файла в списке в параметре `use_exec_modules` в любом разделе конфигурации (таким образом можно исполнять разные модули для разных очередей, пользователей, профилей и т.д.).

13. ПОДКЛЮЧЕНИЕ МОДУЛЕЙ ПЛАНИРОВЩИКОВ

Важной возможностью Cleo является подключение модулей планировщиков. При необходимости устанавливать собственные политики планирования задач, администратор может создать соответствующий модуль и использовать его с своей системе.

Модули должны быть написаны на языке perl. Модуль планировщика должен описывать переменную `$cleo`, содержащую номер версии модуля, а также процедуру `do_scedule`, которая и выполняет планирование задач. По окончании работы, `do_scedule` должна вернуть 0, в случае отсутствия ошибок, или 1, если таковые были обнаружены.

При вызове процедуры `do_scedule`, ей передаются следующие аргументы: ссылка на список задач, число зарезервированных процессоров (которые должны быть использованы для «предзапущенных» задач, но имена которых ещё неизвестны), и список незанятых процессоров.

Задача описывается как хэш с полями:

- `id` идентификатор задачи,
- `user` имя пользователя — владельца задачи,
- `np` заказанное число процессоров,
- `timelimit` лимит времени задачи в секундах,
- `is_own` 1, если задача принадлежит этой очереди (0 — родительской),
- `blocked` 1, если задача заблокирована (0 — если нет).

В процессе планирования модуль может вызывать предопределённые подпрограммы, описанные ниже. С их помощью модуль может получать информацию о задачах, пользовательских настройках, режиме работы и управлять запуском задач.

В процессе работы модулю запрещается пользоваться функцией `alarm`, т.к. система отслеживает время работы модуля с помощью этой функции. В случае превышения допустимого времени работы (несколько секунд), модуль отключается и вместо него используется штатный планировщик.

Если при использовании модуля были обнаружены ошибки (неверные вызовы предопределённых подпрограмм, возврат 1 процедурой `do_scedule`), то модуль также отключается.

Приведём описание подпрограмм, доступных модулю планировщика:

13.1.run

Запустить задачу.

Аргументы: ID, [PE_LIST]

ID — идентификатор задачи, PE_LIST — список процессоров, на которых задача должна быть запущена (опционально).

В настоящее время PE_LIST игнорируется.

13.2.block

Блокировать задачу.

Аргументы: ID, REASON

ID — идентификатор задачи, REASON — описание причины блокировки.

13.3.unblock

Разблокировать задачу.

Аргументы: ID, REASON

ID — идентификатор задачи, REASON — описание причины блокировки.

Если в качестве причины указано «Total», то снимаются все блокировки.

13.4.move

Переместить задачу в списке.

Аргументы: ID, ID_AFTER

Переместить задачу с идентификатором ID после задачи с идентификатором ID_AFTER. Список, переданный в качестве первого аргумента `do_scedule` **не меняется**. Если ID_AFTER равен -1, то задача помещается в начало очереди. Если ID_AFTER равен -2, то задача помещается в конец очереди.

13.5.get_task_info

Получить данные о задаче.

Аргументы: ID, NAME

ID — идентификатор задачи, NAME — название параметра, значение которого надо получить. Возможные значения NAME:

`task`	Полное имя задачи с аргументами
`args`	Аргументы задачи
`added`	Время постановки задачи в очередь (UNIX-time)
`timelimit`	Лимит времени работы задачи (в секундах)
`priority`	Приоритет задачи
`np`	Число запрошенных процессоров
`user`	Имя пользователя — владельца задачи
`dir`	Рабочий каталог
`pe_select`	Метод распределения процессоров
`queue`	Имя текущей очереди
`out`	Имя выходного файла задачи(!)
`rep`	Имя файла отчёта задачи (!)

owner	Имя очереди, которой в действительности принадлежит задача
com_line	Строка запуска задачи (!)
blocks	Список блокировок задачи (ссылка на хэш)
group	Группа, от имени которой будет запущена задача
env	Список переменных окружения задачи (строка в формате VAR1=value1;VAR2=value2...)
temp_dir	Временный каталог
occupy_full_node	1, если занимать свободные процессоры на узле
run_via_mons	1, если запускать задачу через мониторы
use_file	Имя спец-файла
one_rep	1, если пересоздавать файл отчёта
wait_cond_type	'a' или 'o' в зависимости от того, какое условие ожидания задач использовать (and/or)
wait_for_run	Список задач, которые должны быть запущены к моменту старта задачи
wait_for_ok	Список задач, которые должны быть успешно завершены к моменту старта задачи
wait_for_fail	Список задач, которые должны быть неудачно завершенных моменту старта задачи
file_mask	Маска для определения процессов задачи на узле
rsh_filter	Имя rsh-фильтра

Знаком (!) помечены параметры, которые могут содержать неразрешённые на данные момент псевдопеременные. Планировщик может обращаться к дополнительным полям задач, имена которых начинаются с двух символов подчёркивания. Эти параметры могут быть установлены с помощью подпрограммы set_task_info.

13.6.set_task_info

Задать значение дополнительного параметра задачи.

Аргументы: ID, NAME, NEW_VALUE

ID — идентификатор задачи, NAME — имя параметра, NEW_VALUE — его значение. К началу имени параметра будет автоматически добавлено два подчёркивания.

13.7.get_user_info

Получить данные о пользователе.

Аргументы: USERNAME, NAME

USERNAME — имя пользователя, NAME — название параметра. Список доступных параметров:

- add_pri_on_chld,
- attach_mask,
- coll_nodes,
- def_priority,
- def_queue,
- def_view_flags,
- default_time,
- exec_line,
- exec_module,
- exec_write,
- file_head,
- file_line,
- file_tail,
- first_line,
- gid,
- kill_script,
- max_np,
- max_queue,
- max_sum_np,
- max_tasks,
- max_time,
- min_np,
- occupy_full_node,
- one_report,
- outfile,
- pe_select,
- post_exec,
- post_exec_write,
- priority,
- q_fail_exec,
- q_just_exec,
- q_ok_exec,
- q_post_exec,
- q_pre_exec,
- repfile,
- rsh_filter,
- run_via_mons,
- temp_dir,
- use_file,
- use_first_line,
- user_conf_file,
- user_fail_exec,
- user_just_exec,
- user_kill_script,
- user_ok_exec,
- user_post_exec,
- user_pre_exec.

Их описания можно посмотреть в описании параметров сервера Cleo.

13.8.get_user_profile_info

Получить данные о пользователе с учётом профиля.

Аргументы: USERNAME, PROFILE, NAME

USERNAME — имя пользователя, NAME — название параметра, PROFILE — имя профиля.

13.9.get_pe_info

Получить данные о процессоре.

Аргументы: PE, NAME

PE — имя процессора, NAME — имя параметра.

Доступные параметры:

`blocked`	1, если процессор заблокирован, и 0, если нет
`blocked_reasons`	ссылка на массив perl с причинами блокировок процессора
`ids`	ссылка на массив perl со списком идентификаторов запущенных на процессоре задач
`own`	1, если процессор принадлежит только этой очереди, и 0, если он также принадлежит одной из дочерних
`max`	максимальной число задач, которое может быть запущено на этом процессоре

13.10.cleo_log

Записать строку в файл журнала.

Аргументы: MESG

MESG — строка для записи в файл журнала. Не должна содержать перевода строки.

13.11.list_running

Возвращает ссылку на массив всех считающихся задач. Каждая задача представлена в виде хэша с полями как в `get_task_info`. Значения полей, как правило, будут содержать уже разрешённые псевдо-переменные.

13.12.list_future

Возвращает ссылку на хэш с планируемыми временами окончания считающихся задач. Ключи хэша — идентификаторы запущенных задач. Значения — хеши с полями:

- `time` максимальное время завершения задачи (UNIX-time)
- `np` число занятых процессоров
- `npextra` число дополнительно использованных процессоров
- `user` имя пользователя — владельца задачи

`npextra` может быть отличным от нуля в том случае, если был использован параметр `occupy_full_node`. Он может быть принят во внимание ТОЛЬКО в случае, если задача принадлежит этой очереди.

В дочерних очередях параметр `np` равен количеству реально использованных процессоров.

13.13.violates

Проверяет, возможен ли запуск задачи.

Аргументы: ID, [NOACT]

ID — идентификатор задачи, NOACT — опциональный параметр.

Если запуск задачи может быть беспрепятственно произведён, функция возвращает 0.

В случае, если запуску задачи мешает какое-либо из ограничений (например `max_user_np`) или условий (например ожидание завершения другой задачи), то будет возвращено значение 1.

Если запуск не только невозможен, но и оказалось, что в результате проверки задача будет удалена из очереди, то будет возвращено значение 2.

Параметр NOACT указывает не предпринимать действий по блокировке задачи, несмотря на требования ограничений — таким образом можно изменить действие ряда ограничений в Cleo.

13.14.get_mode

Получить текущий режим работы очереди.

Возвращает целое число, биты которого соответствуют режимам работы очереди. Используя битовые операции, можно узнать активирован ли

конкретный режим. Для получения нужных битовых масок необходимо воспользоваться предопределёнными константами:

- MODE_RUN_ALLOW разрешён запуск задач
- MODE_QUEUE_ALLOW разрешена постановка задач в очередь

По умолчанию Cleo запускает задачи из родительской очереди, даже в случае, если очередь заблокирована на запуск новых задач (MODE_RUN_ALLOW=0). Это умолчание не обязательно, но желательно соблюдать при написании собственных планировщиков.

14. ПРИМЕРЫ КОНФИГУРАЦИЙ

Приведем комментированный пример конфигурационного файла. Для удобства строки файла конфигурации мы будем приводить `непропорциональным шрифтом`, а комментарии (которые не входят в текст файла) — обычным.

```
[server]
```
Головная секция

```
verbose         = 1
```
Подробнее вести журнал

```
log_file        = /var/log/cleo.log
```
Вести общий журнал в файле /var/log/cleo.log

```
short_log_file = /var/log/cleo-short.log
```
Вести короткий журнал в файле /var/log/cleo-short.log

```
admins          = root vasya
```
Администратором системы объявляется пользователи root и vasya

```
port            = 5252
```
Открыть серверный порт (TCP/IP) 5252

```
gid             = 600
```
Запускать задачи от имени группы 600

```
max_queue       = 20
```
Ставить в очередь не более 20 задач по умолчанию

```
default_time    = 86400 # 1 day...
```
Установить максимальное время счета по умолчанию (1 день)

```
max_time        = 259200 # 3 days...
```
Установить максимальное время счета (3 дня)

Если этот или предыдущий параметр не задан, то они оба устанавливаются равными.

```
pid_file        = /var/run/cleo.pid
```
Файл с номером головного процесса сервера очередей

```
queue_save      = /var/log/cleo-save
```
Префикс файла сохранения состояния очереди (для очереди 'main', например файл сохранения будет /var/log/cleo-save.main)

```
queue_alt_save = /tmp/cleo-save
```

Альтернативный префикс файла сохранения состояния очереди

```
def_queue       = calc
```

calc объявляется очередью по умолчанию, т.е. в нее будут направляться запросы в которых очередь не указана

```
timeout         = 5
```

Таймаут соединения с клиентами устанавливается равным 5 секундам

```
one_report      = 0
```

По умолчанию не затирать файлы отчетов и стандартного вывода, если они уже существуют. Вместо этого создавать файл с таким же именем и добавленным к нему номером через точку (например out.1, а если и он существует, то out.2 и т.д.)

```
repfile         = $dir/$sexe.rep-$id
```

Шаблон имени файла отчета задачи. Например при запуске задачи /usr/bin/task1 в каталоге /tmp файл отчета будет иметь вид /tmp/task1-56 (для примера идентификатор данной задачи положен равным 56, реальный же будет выдан командой mpirun)

```
outfile         = $dir/$sexe.out-$id
```

Шаблон имени файла стандартного вывода задачи

```
post_exec       = echo 'Task done! (id=$id,code=$code
$special) | mailto -s "task done" "$user"@our.mail.server
```

Команда, которая будет выполнена после завершения задачи

```
post_exec_write = Your task '$task" finished with code
$status
```

Текст, который будет выведен пользователю на консоль (при помощи команды write) после завершения задачи

```
use_file        = $dir/machinefile-$queue_name-$id
```

При запуске задачи формировать файл с конфигурацией (по указанному шаблону)

```
coll_nodes      = 0
```

В файле конфигурации задачи не склеивать строки, относящиеся к одному узлу в одну

```
file_head       = $np\n
```

Шаблон заголовка файла конфигурации задачи

```
file_line       = $node $nid\n
```

Шаблон строки файла конфигурации задачи (для каждого процессора)

```
exec_line      = /common/mpich/bin/mpirun.ch_gm.old --gm-f
$file $task
```

Шаблон строки запуска задачи. Как видно, здесь используется сформированный файл конфигурации задачи (псевдопеременная $file)

```
kill_script    = /usr/local/sbin/gmkill-rsh.pl $pid
$spaced_nodes
```

Шаблон команды, исполняемой при завершении задачи

```
pe_sel_method   = simple /usr/local/q/qsmod -conf /etc/s.conf
```

Подключение внешнего модуля распределения процессоров с именем 'simple', который реализуется командой /usr/local/q/qsmod с параметрами '-conf /etc/s.conf'.

```
pe_select       = simple
```

В качестве метода распределения процессоров по умолчанию задается метод 'simple'

```
pe = node1:1  node1:2
pe = node2:1  node2:2
pe = node3:1  node3:2  node4:1  node4:2
```

Список процессоров

```
[groups]
```

Секция групп

```
stud      = st01,st02,st03,petrov
```

Объявление группы пользователей 'stud'

```
we        = root,qadm,vova
```

Объявление группы пользователей 'we'

```
we_n_stud = +we +stud +vova2
```

Объявление группы ,объединяющей 2 вышеописанные и содержащей, кроме того, пользователя vova2

```
[clusters]
```

Секция отдельных очередей

```
stud.parent       = main
```

Объявляется очередь 'stud' - дочерняя от 'main'

```
calc.parent       = main
```

Объявляется очередь 'calc' - дочерняя от 'main'

```
main.pe = node1:1  node1:2
main.pe = node2:1  node2:2
main.pe = node3:1  node3:2  node4:1  node4:2
```

Список процессоров очереди 'main'

```
main.users  = +we
```

Пользователи, которым разрешено ставить задачи в очередь 'main' (пользователи из группы 'we')

```
stud.pe = node1:1  node1:2
stud.pe = node2:1  node2:2
```

Список процессоров очереди 'stud'

```
stud.default_time = 300 # 5 minutes
```

Максимальное время счета задач по умолчанию (и максимальное, так как не указан параметр 'max_time', т.е. пользователь не сможет указать максимальное время больше, чем 5 минут, и оно же будет браться по умолчанию)

```
stud.users = +stud
```

Пользователи, которым разрешено ставить задачи в очередь 'stud' (пользователи из группы 'stud')

```
stud.admins = qadm
```

Администратором очереди 'stud' назначается пользователь qadm (при этом ее администраторами также считаются все администраторы **системы**)

```
calc.pe = node3:1  node3:2  node4:1  node4:2
```

Список процессоров очереди 'calc'

```
calc.nousers = root,qadm,petrov,hacker
```

Пользователи, которым запрещено ставить задачи в очередь 'calc'

```
calc.admins = qadm
```

Администратором очереди 'stud' назначается пользователь qadm

```
[users]
```

Секция пользователей

```
ivanov.max_sum_np = 20
```

Пользователю ivanov устанавливается ограничение в 20 процессоров для одновременного счета

```
sidorov.pe_select = random_alone
```

Пользователю sidorov устанавливается метод random_alone для распределения процессоров

```
vova.default_queue = main
```

Пользователю vova устанавливается очередь main по умолчанию для постановки задач

```
st01.max_np = 4
```

Пользователю st01 устанавливается ограничение в 20 процессоров для одной задачи

Как уже упоминалось выше, система способна поддерживать несколько конфигураций параллельных сред одновременно. Это осуществляется за счет настройки параметров `exec_line`, `file_*`, `use_file`, `coll_nodes` (см. главы «Настройка сервера через файл конфигурации и ключи» и «Формирование файла конфигурации задачи»).

Различные настройки сред можно задавать как для различных очередей, так и для отдельных пользователей. В ближайших версиях системы также будет поддерживаться механизм профилей, с помощью которого можно будет указывать конфигурацию при постановке задач в очередь.

Рассмотрим два практических примера: конфигурацию для MPICH и конфигурацию для последовательных задач.

14.1.Конфигурация для MPICH-p4

В основном файле конфигурации в секции server пропишем следующие строки (для удобства строки пронумерованы):

```
1 use_file   = $dir/machinefile-$queue-$id
2 coll_nodes = 0
3 file_head  = #machine file for $task\n
4 file_tail  = #end\n
5 file_line  = $node\n
6 exec_line = /usr/local/mpich/bin/mpirun
  -machinefile /tmp/machinefile-$id -np $np -nolocal $task
```

В первой строке указывается необходимость использования файла конфигурации задачи. По сути это — machinefile для MPICH.

Вторая строка указывает формировать отдельные строки для каждого процессора (даже если они расположены на одном узле).

В третьей и четвертой строках формируются заголовок и окончание конфигурационного файла. В данном случае они могли отсутствовать, так как не несут полезной информации и оформлены в виде комментариев к machinefile MPICH.

Пятая строка указывает шаблон для каждой строки machinefile, соответствующей каждому процессору. В данном случае это просто имя узла.

Наконец, шестая строка указывает шаблон для командной строки запуска задачи. Как видно, в параметре —machinefie программы mpirun задано имя файла из параметра use_file. То есть в качестве machinefile будет передан сформированный файл настройки задачи.

14.2.Конфигурация для последовательных задач

Для последовательных задач настройка проще, так как нет необходимости в формировании файла настройки задачи. Запуск может осуществляться с помощью команды rsh, но в этом случае следует помнить о том, что запуск программы может осуществляться не только из домашнего каталога пользователя. С учетом последнего требования настройка сведется к прописыванию следующих двух строк:

```
exec_line = rsh $nodes '(cd $dir; $task)'
use_file  =
```

В последней строке значение параметра пустое, так как мы не хотим формирования файла настройки задачи.

15. СБОР СТАТИСТИКИ И АУДИТ

В системе существуют средства для сбора статистики ее работы, а также возможности аудита. Статистика собирается при помощи программы `cleo-stat`, которой на вход подается файл сокращенного лога (заданный параметром short_log_file в файле конфигурации). В данном файле фиксируются все запуски программ, а также все запросы от пользователей. Программа `cleo-stat` анализирует эту информацию и генерирует отчет. В качестве ключей ей могут быть переданы следующие параметры:

`-d`	Кроме суммарного отчета, генерировать отчет по дням
`-q`	Кроме суммарных данных, генерировать данные по всем очередям
`-a`	Учитывать ещё не завершённые задачи и те задачи, которые были запущены до начала указанного периода, но завершённые после него
`дата начала`	Дата начала отчета. Формат: DD.MM[.YY[YY]], где DD — день, MM — месяц, YY — год.
`дата окончания`	Дата окончания отчета. Формат тот же, что и для даты начала отчета.

Пример:

```
> cleo-stat < /var/log/qlog-short

Time format is minutes:seconds.
Interval from Thu Jul  1 00:00:00 2004 to Fri Jul  2 00:00:00
2004

Total:        astr.time   sum.time min_np max_np  tasks

    usr1:       4256:55   83514:25     38     72      5
    usr2:       7340:55   83879:25     18     80     23
    usr3:         36:56     226:00      4     18      2
    usr4:         21:12      22:54      2      3      5
    usr5:        980:58     980:58      1      1     17
    usr6:         53:14     575:06      3     44     22
    usr7:          0:06       0:24      4      4      1
    usr8:       2903:28   12756:34      2     40     74
    usr9:          0:24       0:24      1      1      1
```

В первой колонке печатается имя пользователя, во второй — суммарное время счета его задач в минутах:секундах, в третьей — суммарное процессорное время. Далее указаны минимальное и максимальное количество процессоров, которые занимал пользователь.

Суммарное процессорное время вычисляется путем умножения времен счета отдельных задач на число процессоров, на которых они считались и сложения полученных величин.

Можно получить и более детальную информацию:

```
> cleo-stat -d -q 1.04 3.04 < /var/log/qlog-short

Time format is minutes:seconds.
Interval from Thu Apr  1 00:00:00 2004 to Sat Apr  3 00:00:00
2004

Total:       astr.time   sum.time min_np max_np  tasks

    usr1:    59:59    255:40  min:  3  max: 18
    usr2:   393:55   5013:55  min: 19  max: 54
    usr4:    16:06     16:06  min:  2  max:  2
   usr44:    64:10   1796:40  min: 28  max: 28
    usr6:     5:37     21:02  min:  2  max: 24
    usr8:    41:36    528:11  min:  2  max: 36

Queue main:

    usr2:   393:55   5013:55  min: 19  max: 54

Queue long:

    usr1:    59:59    255:40  min:  3  max: 18
    usr2:   123:55   3013:35  min: 19  max: 54
    usr4:    16:06     16:06  min:  2  max:  2
   usr44:    64:10   1796:40  min: 28  max: 28
    usr6:     5:37     21:02  min:  2  max: 24
    usr8:    41:36    528:11  min:  2  max: 36

Queue short:

    usr2:   270:00   2000:20  min:  2  max:  2

Total on each day (astronomy time):

Apr  1       usr1:    0:24
Apr  1       usr2:   17:44
Apr  1       usr4:    5:53
Apr  1      usr44:   39:37
Apr  1       usr8:    6:53
-------------------------
Apr  2       usr2:    0:34
Apr  2       usr4:    5:07
Apr  2       usr6:    5:37
Apr  2       usr8:   13:09
```

```
Queue main:
Apr  2       usr2:    6:20

Queue long:

Apr  1       usr1:    0:24
Apr  1       usr2:   11:24
Apr  1      usr44:   39:37
Apr  1       usr8:    6:53
-------------------------
Apr  2       usr2:    0:34
Apr  2       usr4:    1:42
Apr  2       usr6:    5:37
Apr  2       usr8:   11:33

Queue short:

Apr  2       usr2:  393:55
-------------------------
```

При выдаче статистики по дням печатается только суммарное время счета.

Кроме сбора статистических данных в системе есть возможность отслеживать ее состояние. В файлах, заданных параметром status_file, сохраняется состояние соответствующих очередей. Любые внешние программы могут в любой момент получить данные о состоянии очередей из этих файлов.

Примером может являться web-интерфейс мониторинга системы.

Формат упомянутых файлов схож с форматом файла конфигурации системы — он имеет такую же блочную структуру. Первым блоком идет блок global, в котором описываются параметры очереди в целом. Все параметры описываются в формате `имя_параметра = значение`. Ниже приведен список параметров в блоке global:

`free`	количество свободных процессоров
`shared_free`	количество свободных процессоров, разделяемых с дочерними очередями (устарело)
`shared_reserved`	количество зарезервированных процессоров под пред-запущенные задачи
`total_own`	количество процессоров, не разделяемых с дочерними очередями
`total_shared`	количество процессоров, разделяемых с дочерними очередями
`shared`	количество процессоров, разделяемых с дочерними очередями
`blocked`	число заблокированных процессоров

blocked_pe	список имён заблокированных процессоров через запятую
blocked_pe_reasons	список причин блокировок процессоров через запятую. Формат поля: **имя_процессора|причина_блокировки1; причина_блокировки2;... причина_блокировкиN**
own_pe	список процессоров, не разделяемых с дочерними очередями (через запятую)
shared_pe	список процессоров, разделяемых с дочерними очередями (через запятую)
own	число свободных процессоров, не разделяемых с дочерними очередями
shared	число свободных процессоров, разделяемых с дочерними очередями
tasks	количество задач в очереди, включая запущенные
running	количество запущенных задач
queued	количество задач в очереди, ожидающих запуска
foreign	количество задач, принадлежащих родительской очереди (устарело)
pending	количество задач, ожидающих разрешения ограничений
queue	число задач в очереди
mode	статус очереди — число, получаемое как сумма флагов: • 1 разрешение запуска новых задач • 2 режим авторестарта • 4 процесс активации авторестарта • 8 запрет постановки новых задач в очередь
autoblocks	

Далее следуют блоки, соответствующие задачам. Названия блоков имеют вид: `task.<номер_задания>`. Эти блоки могут отсутствовать, если в очереди отсутствуют задачи. Параметры в этих блоках (кроме параметра id) имеют вид: `имя_параметра.<номер_задания> = значение`. Список параметров приведен ниже:

id	номер задачи
np	число процессоров
user	имя пользователя-владельца задачи

state	состояние задачи — • **running** — запущена на счет • **queued** — ожидает запуска • **prerun** — ожидает освобождения процессоров в дочерних очередях • **waiting** — ожидает удовлетворения условий запуска
start	время запуска задачи (unix time[5])
owner	имя очереди, в которую изначально была поставлена задача
origid	номер задачи в родительской очереди
sexe	короткое имя задачи
exe	полное имя задачи
args	аргументы задачи
path	путь к выполняемому файлу задачи
out	полное имя выходного файла задачи
rep	полное имя файла отчета задачи
tmpdir	полное имя временного каталога для задачи
nice	значение nice для задачи
timelimit	лимит времени счета задачи (для считающихся задач — время принудительного окончания счета (unix time), а для стоящих в очереди — величина лимита в секундах)
added	время, когда задача была поставлена в очередь
ownnodes	список процессоров, выделенных для задачи и не разделяемых с дочерними очередями
sharednodes	список процессоров, выделенных для задачи и разделяемых с дочерними очередями
priority	приоритет задачи

Значения некоторых параметров могут быть пустыми — например, параметр start для задачи, ожидающей запуска или параметры, относящиеся к родительской очереди, в случае задачи поставленной именно в эту очередь. Параметры rep, out и tempdir в зависимости от того, запущена задача или нет, содержат полное имя файла/каталога или соответствующий шаблон.

Ниже приведен пример содержимого файла состояния очереди:

```
[global]
free = 32
shared_free = 0
shared_reserved = 0
own = 32
shared = 0
tasks = 2
running = 1
queued= 1
foreign= 0
```

[5] Количество секунд с 1 января 1970 года

```
pending= 0
queue= 1
[task.2259]
id=2259
np.2259 = 1
user.2259 = usr2
state.2259 = running
start.2259 = 987413899
owner.2259 = long
origid.2259 =
sexe.2259 = chain_fcc.exe
exe.2259 = /usr/home/usr2/chain_fcc.exe
args.2259 =
path.2259 = /usr/home/usr2
out.2259 = /usr/home/usr2/chain_fcc.exe.out-2259
rep.2259 = /usr/home/usr2/chain_fcc.exe.rep-2259
tmpdir.2259 = /tmp
realowner.2259 = long
nice.2259 = 0
timelimit.2259 = 987518299
added.2259 = 987413899
ownnodes.2259 = p2:2
sharednodes.2259 =
[task.2268]
id=2268
state.2268 = queued
np.2268 = 4
owner.2268 = long
user.2268 = usr4
origid.2268 =
sexe.2268 = msap
exe.2268 = /usr/home/usr4/work/Test/msap
args.2268 =
path.2268 = /usr/home/usr4/work/Test
out.2268 = $sexe.out
rep.2268 = $sexe.rep
tmpdir.2268 = /tmp
added.2268 = 987495736
realowner.2268 = long
nice.2268 = 0
timelimit.2268 = 259200
priority.2268 = 10
ownnodes.2268 =
sharednodes.2268 =
```

16. ПРОТОКОЛ ВЗАИМОДЕЙСТВИЯ СЕРВЕРА С КЛИЕНТАМИ

16.1.Общая схема протокола

Любое приложение может взаимодействовать с сервером системы очередей, используя протокол TCP/IP. Для этого приложение должно установить соединение с сервером по адресу localhost:PORT, где PORT — номер порта, указанный в параметре `port` конфигурационного файла, либо указанный серверу ключом -p.

Все строки при обменах заканчиваются последовательностью '\n' (в терминах языка Си). Все ответы сервера начинаются с символа '+' либо '—'. Первое означает, что текущая фаза взаимодействия успешна. Символ '—' означает неудачу текущей фазы. Текст непосредственно за символом '—' описывает причину неудачи.

Далее везде предполагается успешное течение взаимодействия.

После установления соединения клиентское приложение посылает серверу одну из команд, описанных в разделе 14.2.

В ответ сервер посылает строку вида:

```
'+auth:auth_text'
```

(здесь и далее обрамляющие кавычки не посылаются сервером либо клиентом).

В ответ клиентское приложение меняет свой параметр argv[0] (в терминах языка Си) на строку, которую сервер указал в качестве auth_text и после этого отвечает серверу строкой

```
'+ok'
```

Далее сервер обрабатывает команду клиента и возвращает ответ на нее.

В случае, если сервер был запущен с ключом -x, смена клиентом параметра argv[0] не обязательна. В этом случае клиент может быть запущен не на том же компьютере, что и сервер. Однако, при этом сервер всегда будет успешно проводить авторизацию, независимо от того какой именно пользователь запускает клиентское приложение. Таким образом появляется угроза несанкционированного доступа к правам администратора.

16.2.Список команд сервера системы очередей

Все команды имеют следующую структуру:

```
command:user:pid:+
{parameter: value}
end
```

`command` — это имя команды, `user` — имя пользователя (не UID!), `pid` — pid программы-клиента. UID указанного пользователя должен совпадать с текущим UID программы-клиента. Символ `'+'` в конце обязателен.

В фигурных скобках указан шаблон строки-параметра команды. Таких строк может быть 0 и более. Завершающая строка (как видно из структуры) — строка с единственным словом `'end'`.

В каждой строке-параметре команды parameter означает имя `параметра` команды, а `value` — его значение. `value` не может содержать символов `'\n'` и `'\0'`.

Список команд сервера приведён в главе «Основной клиент» в описании ключа `-G`. Ниже приведены варианты ответов сервера на команды.

При успешном выполнении команды `add` сервер возвращает строку:

```
+Successfully added to queue (ID=...)
```

Где вместо многоточия будет указан идентификатор задачи.

При успешном выполнении команды del сервер возвращает строку:

```
+Deleted
```

При успешном выполнении команды `block`, `block_pe` или `priority` сервер возвращает строку:

```
+ok
```

При успешном выполнении команд `view` и `stat` сервер возвращает строку

```
+ok
```

За которой следуют строки соответствующего ответа на запрос. Мы не будем приводить полный формат ответов сервера для этих команд, так как они различны для обычного, полного, технического вариантов и их комбинаций. Эти форматы можно увидеть в выводе команды `cleo-client` с соответствующими ключами — она не изменяет ответ сервера и просто его печатает.

Кроме того, настоятельно рекомендуется получать данную информацию не от сервера, а из его файлов состояния очередей.

16.3.Режимы команды mode

параметр	значение	Числ. знач.

mode_version	выдает версию сервера	0
mode_update_pid	обновляет файл с PID головного процесса	0
mode_reload_conf	перечитывает конфигурацию сервера	0
mode_update_users	обновляет настройки пользователей	0
mode_norun	запрещает запуск новых заданий в данной очереди (возможно рекурсивно).	1
mode_run	разрешает запуск новых заданий в данной очереди (возможно рекурсивно).	1
mode_qdisable	запрещает постановку новых заданий в данную очередь (возможно рекурсивно).	8
mode_qenable	разрешает постановку новых заданий в данную очередь (возможно рекурсивно).	8
mode_nosane	отмена режима автоматической перезагрузки	2
mode_view	просмотр текущего режима	0

17. ФОРМАТ ФАЙЛОВ ЖУРНАЛА

При работе система ведет два файла журнала — короткий и полный. Формат короткого журнала (задается параметром `short_log_file` в файле конфигурации) фиксирован. При развитии системы его формат может меняться только путем добавления новых событий, но не изменением формата старых.

Именно этот файл используется для сбора статистики.

Второй файл (задается параметром `log_file` в файле конфигурации) служит для целей отладки и исправления нештатных ситуаций. Его формат меняется от версии к версии. Ниже будет приведен его сокращенный формат, который не меняется либо меняется незначительно. Строки, не указанные здесь можно опускать при чтении файла журнала.

Общий формат файла:

```
[дата] имя_очереди :УРОВЕНЬ сообщение.
```

[дата] приводится в формате команды date (Mon Jan 1 01:02:03 0001). УРОВЕНЬ — уровень важности сообщения, один из INFO, WARN, ERROR, DEBUG, DEBUG2. В коротком журнале это поле отсутствует.

Пример:

```
[Fri Jun  1 12:51:05 2001] main    :INFO VIEW request danila; long
```

17.1.Формат короткого файла журнала

Ниже приводится список сообщений в коротком файле журнала:

Строка (строки в журнале)	Событие в системе
`VIEW request user; queue`	Запрос пользователя user на просмотр состояния очереди queue
`ADD request user; queue; NP; task_with_args`	Запрос пользователя user на постановку в очередь queue задания с аргументами task_with_args и заказом для него NP процессоров
`ADDED N; parent; parent_id; user; NP; task_with_args`	Задача из родительской очереди parent с идентификатором parent_id (если есть) пользователя user с заказом NP процессоров была поставлена в очередь с номером N.
`DEL request user; queue; N`	Запрос пользователя user на удаление задачи с номером N из очереди queue

PRI request user; queue; N; P	Запрос пользователя user на измерение приоритета задачи с номером N в очереди queue на P
MODE request user; queue; SET; CLEAN	Запрос пользователя user на изменение (просмотр) режима работы сервера. SET и CLEAN являются суммами флагов, которые требуется установить и сбросить соответственно. Значения флагов приведены в графе 'числ. знач.' в главе «режимы команды mode»
BLOCK request user; queue; N; block	Запрос пользователя user на блокировку/ разблокировку задачи с номером N в очереди queue. Если параметр block равен 1, то запрос на блокировку, если 0 — то на разблокировку
BLOCK_PE request user; pe; block	Запрос пользователя user на блокировку/ разблокировку процессора pe. Если параметр block равен 1, то запрос на блокировку, если 0 — то на разблокировку
STAT request user; queue	Запрос пользователя user на получение состояния очереди queue
Server killed by signal SIGNAL	Завершение работы очереди по сигналу SIGNAL. Вместо SIGNAL может быть [master died], что означает завершение работы вследствие завершения родительской очереди.
RUN id; user; NP; task_with_args	Запуск задачи с номером id пользователя user на NP процессорах. task_with_args — сама задача с аргументами
END_TASK0 id	Завершение задачи с номером id
END_TASK id; user; status; signal; hours:min:sec	Завершение задачи с номером id пользователя user с кодом завершения status по сигналу signal. Время работы задачи — в последнем аргументе
END_TASK_NODES id; node1:1,node1:2,…, nodeN:X	Завершение задачи с номером id. Далее перечисляются все процессоры, на которых она считалась

При завершении задачи в журнал попадают все три строки `END_TASK*`.

17.2.Формат полного файла журнала

Как уже говорилось, в полный файл журнала попадает в основном отладочная информация, которая может быть полезной при разрешении нештатных ситуаций.

Ниже приводятся список возможных сообщений в журнале. Строки, не описанные здесь следует игнорировать при чтении журнала.

Строка (строки в журнале)	Событие в системе
`Start new server. Port PPPP, N processors`	Старт новой очереди (не только сервера) PPPP — номер серверного порта (актуален только для головной очереди), N — число обслуживаемых процессоров
`PE_LIST [node1:1, …,nodeX:Y]`	Список процессоров, обслуживаемых очередью
`DUMP`	Сохранение текущего состояния очереди в файле состояния
`DUMP DONE`	Завершение сохранения текущего состояния
`GOT: …`	Параметр полученной команды
`SEND TO CHILDS`	Рассылка запросов к дочерним очередям
`MESSAGES FROM CHILDS`	Приход сообщений от дочерних очередей
`Message from queue1 to queue2 (…)`	Получено сообщение от очереди `queue1` к `queue2` (возможно транзитное) За этой строкой могут появиться строки без ведущей даты, если они являются частью сообщения.
`Forwarding to 'queue'`	Транзит сообщения к очереди `queue`
`SCEDULE`	Запуск планировщика задач
`Can_run: id=XXX own=XX shared=X req=X res=X`	Задача выбрана планировщиком для запуска. `id` — идентификатор, `own` — сколько «своих» процессоров свободно, `shared` — сколько «разделяемых» процессоров свободно, `req` — сколько процессоров запрошено, `res` — сколько процессоров «зарезервировано»
`FREE: …`	список свободных процессоров
`SHUFFLE USE …`	Запрос к модулю стратегии распределения процессоров.
`USED:`	Список использованных процессоров
`Try to exec '…', using outfile='…'`	Попытка выполнить строку, как старт задания. Указан шаблон файла стандартного вывода. Далее идет строка, в которой дата заменена на многоточия. В ней сообщается шаблон имен файла отчета и временного каталог для задачи

`REAL outfile='…' repfile='…' tmp='…'`	Имена файлов стандартного вывода и отчета, а также временный каталог для последней запускаемой задачи (с подставленными псевдопеременными)
`Exec: '…'`	Запуск командной строки как головного процесса новой задачи
`Error while read_block. Retry...`	Ошибка асинхронного чтения. Попытка чтения вновь.
`REQUESTING TASK on 'queue'`	Запрос на постановку новой задачи в дочернюю очередь queue
`SENDING ADD: …`	посылка параметра к запросу на постановку новой задачи
`_Got …`	Получен запрос от родительской очереди
`_Send …`	Посылка ответа родительской очереди
`PARSE: '…' -> '…'`	Подстановка псевдопеременных: слева от '->' указана исходная строка, справа — результирующая
`Warning! Strange message from child... (…) Skipping.`	Предупреждение о некорректном ответе дочерней очереди
`Del_task: '…'`	Удаление задачи с указанным идентификатором
`Del_task2: …`	Удаление задачи с указанным PID
`_Dead... XXX res=YYY`	Завершение задачи с идентификатором XXX. Зарезервировано процессоров — YYY

Оглавление

17.08.2007