Система управления заданиями Cleo

Руководство пользователя

2007

Оглавление

1. ОБЩИЕ ПОНЯТИЯ И ТРЕБОВАНИЯ

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 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 (т.е. в рамках одной очереди все процессоры считаются равноправными).

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

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 `qclient3.pl`), 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:

main							
long				short			
node1:1	node1:2	node2:1	node2:2	node3:1	node3:2	node4:1	node4:2

На этой картинке представлена иерархия из трех очередей над 4 вычислительными узлами, содержащими по 2 процессора. Узлы имеют имена node1, node2, node3 и node4, а их процессоры соответственно node1:1, node1:2 для узла node1 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 node1:1, node1:2, node2:1 и node2:2, и очередь short, включающую в себя процессоры node3:1, node3:2, node4:1 и node4:2.

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 предзапущеная, 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 предзапущена 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.

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.

Например, поставим в очередь 2 задачи с приоритетом 10:

очередь	приоритет
задача1	10
задача2	10

Затем поставим задачу с приоритетом 15. Так как ее приоритет больше, она будет поставлена перед предыдущими задачами:

очередь	приоритет
задача3	15
задача1	10
задача2	10

Теперь поставим еще одну задачу с приоритетом 15. Она будет поставлена перед задачами 1 и 2 (так как ее приоритет больше), но после задачи 3, так как ее приоритет не больше, но поставлена она позже.

очередь	приоритет
задача3	15
задача4	15
задача1	10
задача2	10

О том, какими командами можно проделать эти операции, можно прочесть в главе 7 “Управление задачей в очереди”.

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. Система 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 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. Улучшение сбалансированности 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.

Все стратегии распределения процессоров опираются на умолчание описания имен процессоров – `<имя_узла>:<идентификатор_процессора>`. Данное соглашение используется и в других целях (см. описание псевдопеременных в главе 9).

Технические детали задания стратегии распределения процессоров описаны в главах 8 и 9.

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
- 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.

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. Также можно 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 использованием своей команды `rsh` на узлах
- с использованием своей команды `rsh` на головной машине
- с внешним контролем задач
- запуск без контроля

В первом и втором случаях после запуска мониторы на узлах или 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. Таким образом, можно полностью исключить использование стандартного `rsh` или `ssh`.

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

В третьем 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.

Четвёртый 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.

2.9. Внутренняя архитектура.

Система очередей представляет собой приложение клиент-сервер. Серверная программа называется `q3.pl`. 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 `qmon3.pl`. 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 процесом-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.

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

3. КОМПЛЕКТ ПОСТАВКИ

`q3.pl`	сервер системы очередей
`qsupport.pm` и `qvars.pm`	вспомогательные модули для системы
`qclient3.pl`	основной клиент системы
`qmon3.pl`	монитор для работы на вычислительных узлах
`mpirun`, `tasks`, `block_cpu`, `block_task`, `autoblock` и `change_pri`	скрипты, облегчающие работу с системой
`cleo.conf.example`	пример файла конфигурации
`qmode`	скрипт для смены режима работы системы
`cleo-stat`	программа сбора статистики по краткому логу
`Admguide.doc`	руководство администратора системы в формате MS Word
`Userguide.doc`	руководство пользователя системы в формате MS Word

4. ПОРЯДОК ОБРАБОТКИ ЗАПРОСОВ, ЗАПУСКА И ОСТАНОВА ЗАДАЧ

4.1. Порядок обработки запросов.

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

4.2. 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. Пользователю на консоль посылается сообщение, заданное параметром `exec_write` в файле конфигурации, если оно не пустое.
7. После паузы примерно в 5 секунд, исполняется команда, заданная параметром `just_exec` в файле конфигурации.

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, используемых 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)

5. ПОСТАНОВКА ЗАДАНИЙ В ОЧЕРЕДЬ

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

```
mpirun -np N [-q queue][-maxtime|-l lim][-p pri][-stdin file]
       [-stdout file] [-stderr file] prog [prog_args]
```

либо

```
mpirun -np N [-q queue][-maxtime|-l lim][-p pri][-stdin file]
       [-stdout file] [-stderr file] –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`.

`-np`	Количество процессоров
`-q`	Название очереди, в которую ставится задача
`-maxtime`	Лимит времени счета в минутах
`-l`	Лимит времени счета в секундах
`-p`	Приоритет задачи в очереди
`-stdin`	Файл для стандартного ввода
`-stdout`	Файл для стандартного вывода
`-stderr`	Файл для стандартного потока ошибок

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.

После запуска задачи ее **стандартный вывод** будет перенаправлен в файл, имя которого определяется настройками по умолчанию, либо индивидуальными настройками пользователя, 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`.

6. 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|-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[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
```

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
```

[1] Рассмотрен в главе 7 “Управление задачей в очереди”.
[2] Рассмотрен в главе 7 “Управление задачей в очереди”.
[3] Рассмотрен в главе 7 “Управление задачей в очереди”.

```
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__`.

7. УПРАВЛЕНИЕ ЗАДАЧЕЙ В ОЧЕРЕДИ

7.1. 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. 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`.

Кроме того, 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.

Удаление чужих заданий для рядового пользователя невозможно.

7.2. Смена приоритета.

Кроме удаления задачи, стоящей в очереди, можно изменить ее приоритет. Это делается с помощью команды `change_pri`:

```
change_pri [-q queue][-p port] new_pri id [id...]
```

Ниже описаны ее ключи:

`-q`	Название очереди
`-p`	Порт для обращения к серверу

`new_pri` – новое значение приоритета, `id` – идентификатор задачи (может быть перечислено несколько идентификаторов)

7.3. 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 `block_task`.

```
block_task [-h][-q queue][-U userlist][-t mask]
           [-R 'reason'][-a as_user][-b|-u] task_id
```

Описание 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` – идентификатор блокируемой или разблокируемой задачи, либо `all`.

Параметры `-u` и `-t` полностью аналогичны параметрам `-u` и `-m` команды `tasks` (см. раздел 7.1 “Удаление задач”).

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. 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.